ALTCRAFT PLATFORM
User Guide and API Reference

Contents

Sections and articles in this PDF.

User documentation

where to start.mdx

Getting Started ..... where-to-start.mdx

FAQ.mdx

FAQ ..... FAQ.mdx

glossary.mdx

Altcraft glossary ..... glossary.mdx

archive.mdx

Documentation archive ..... archive.mdx

profiles and databases

Profiles and databases ..... profiles-and-databases/index.mdx
Subscription resources profiles-and-databases/66427.mdx
Managing databases profiles-and-databases/66420.mdx
Subscriber profile profiles-and-databases/9405665.mdx
Profiles import and data update ..... profiles-and-databases/profiles-import-and-data-update.mdx
Common Errors When Importing Profiles ..... profiles-and-databases/common-errors-when-importing.mdx
Scheduled customer data import profiles-and-databases/9404661.mdx
Managing Data Tables ..... profiles-and-databases/data-tables.mdx
Automatic data collection profiles-and-databases/66433.mdx
Bulk customers profiles update profiles-and-databases/9406694.mdx
Double opt-in subscription profiles-and-databases/9404703.mdx
Suppression lists profiles-and-databases/66534.mdx
Profile relations profiles-and-databases/20480122.mdx
Profile history export profiles-and-databases/49710319.mdx
Profile export profiles-and-databases/76808636.mdx
Creating a static segment based on import results profiles-and-databases/37847201.mdx
How to open a CSV file ..... profiles-and-databases/csv-in-excel.mdx
Matching ..... profiles-and-databases/matching.mdx
Types of fields in the database ..... profiles-and-databases/types-of-db-fields.mdx
Global control groups ..... profiles-and-databases/gcg.mdx
Subscription Manager ..... profiles-and-databases/subscription-manager.mdx

communication channels

Communication channels ..... communication-channels/index.mdx
WhatsApp ..... communication-channels/whatsapp.mdx
Viber*™ ..... communication-channels/viber.mdx
Notify ..... communication-channels/notify.mdx
Communication Channels Workflow ..... communication-channels/scheme.mdx
Email: sending domain configuration communication-channels/email/9404444.mdx
Email: setting up and using postmasters ..... communication-channels/email/postmasters.mdx
How email tracking works ..... communication-channels/email/email-tracking.mdx
Quick Start ..... communication-channels/email/first-mailing/quick-start.mdx
Email ..... communication-channels/email/first-mailing/email.mdx
Push Channel ..... communication-channels/push/index.mdx
Mobile Push ..... communication-channels/push/mobile-push/index.mdx
First Mobile Push Mailing ..... communication-channels/push/mobile-push/mobile-push-first-mailing.mdx
Mobile Push Providers ..... communication-channels/push/mobile-push/providers/index.mdx
Firebase Cloud Messaging ..... communication-channels/push/mobile-push/providers/fcm.mdx
Apple Push Notification Service ..... communication-channels/push/mobile-push/providers/apns.mdx
Huawei Mobile Services ..... communication-channels/push/mobile-push/providers/huawei.mdx
RuStore ..... communication-channels/push/mobile-push/providers/rustore.mdx
Yandex.AppMetrica ..... communication-channels/push/mobile-push/providers/appmetrica.mdx
Integrate your app with Altcraft ..... communication-channels/push/mobile-push/mobile-push-app-integration/index.mdx
Deprecated: Setup and Connection ..... communication-channels/push/mobile-push/deprecated/setting-up.mdx
Processing and adding a subscription ..... communication-channels/push/mobile-push/deprecated/adding-subscription.mdx
Event registration ..... communication-channels/push/mobile-push/deprecated/events.mdx

Providers: push message structure ..... communication-channels/push/mobile-push/deprecated/provider-structure.mdx
Web Push ..... communication-channels/push/web-push/index.mdx
First Web Push Mailing ..... communication-channels/push/web-push/web-push-first-mailing.mdx
Resource and Website Setup ..... communication-channels/push/web-push/configuration.mdx
Transferring Data to the Platform ..... communication-channels/push/web-push/data-import.mdx
Web Push SDK Methods ..... communication-channels/push/web-push/methods.mdx
PWA and Push Notifications ..... communication-channels/push/web-push/pwa-push.mdx
Web Push Providers ..... communication-channels/push/web-push/providers/index.mdx
Firebase Cloud messaging ..... communication-channels/push/web-push/providers/firebase-cloud-messaging.mdx
Apple Safari ..... communication-channels/push/web-push/providers/apple-safari.mdx
Mozilla Services ..... communication-channels/push/web-push/providers/mozilla-services.mdx
Migrating push subscriptions from third-party services communication-channels/push/web-push/migration/37847852.mdx
How to transfer push subscriptions configured for Safari? communication-channels/push/web-push/migration/37847870.mdx
Migration from OneSignal communication-channels/push/web-push/migration/37847889.mdx
SMS channel ..... communication-channels/sms/index.mdx
SMS ..... communication-channels/sms/sms-first-mailing.mdx
Telegram ..... communication-channels/telegram/index.mdx
Telegram Bot ..... communication-channels/telegram/telegram-bot.mdx
Telegram Group ..... communication-channels/telegram/telegram-group.mdx
Max ..... communication-channels/max/index.mdx
MAX Bot ..... communication-channels/max/max-bot.mdx
MAX Group ..... communication-channels/max/max-group.mdx

segmentation

Segmentation ..... segmentation/index.mdx
Static Segments segmentation/9407279.mdx
Dynamic Segments ..... segmentation/dynamic-segments.mdx
Updatable Segments ..... segmentation/updatable-segments.mdx
Best Send Time (BST) segmentation/89325640.mdx
Logical operators "AND" and "OR" ..... segmentation/operators-and-or.mdx
Recommendations for working with segments ..... segmentation/segmentation-recommendations.mdx
Segmentation Conditions ..... segmentation/conditions/index.mdx
Segmentation by Profile data ..... segmentation/conditions/profile-data.mdx
Segmentation by Interactions with Entities ..... segmentation/conditions/interactions.mdx
Segmentation by Activity of the channel ..... segmentation/conditions/activity.mdx
Segmentation by external data ..... segmentation/conditions/external-data/index.mdx
Segmentation by external data segmentation/conditions/external-data/66468.mdx
Segmentation by external SQL tables segmentation/conditions/external-data/8650757.mdx
Recommendations for segmentation by external data ..... segmentation/conditions/external-data/external-data-recommendations.mdx

message templates

Message templates ..... message-templates/index.mdx
Managing message templates message-templates/66476.mdx
Template fragments message-templates/66494.mdx
Image gallery message-templates/66507.mdx
Content personalization message-templates/66490.mdx
Creating tables based on array elements message-templates/37847460.mdx
Block editor for email template ..... message-templates/block-editor.mdx
Web version of an email ..... message-templates/webversion.mdx
AMP Email message-templates/89326164.mdx
Importing and exporting a message template message-templates/56258129.mdx
Importing a template from a third-party service message-templates/76808233.mdx
Exporting a template from Pixcraft ..... message-templates/export-from-pixcraft.mdx
Working with message templates ..... message-templates/managing-templates/index.mdx
Working in the editor ..... message-templates/managing-templates/template-editor.mdx
Email template ..... message-templates/managing-templates/email.mdx
SMS template ..... message-templates/managing-templates/sms.mdx
Push template ..... message-templates/managing-templates/push.mdx
MAX template ..... message-templates/managing-templates/max.mdx
Telegram template ..... message-templates/managing-templates/telegram.mdx
WhatsApp template ..... message-templates/managing-templates/whatsapp.mdx
Viber template ..... message-templates/managing-templates/viber.mdx

Notify template ..... message-templates/managing-templates/notify.mdx
Visual editor for email-template ..... message-templates/visual-editor/index.mdx
Visual editor interface message-templates/visual-editor/37848561.mdx
Adding blocks message-templates/visual-editor/37848658.mdx
Elements and their settings message-templates/visual-editor/37848574.mdx
Custom blocks message-templates/visual-editor/37849457.mdx
Style manager message-templates/visual-editor/37848592.mdx
Layer manager message-templates/visual-editor/37849442.mdx
Altcraft Variables and Functions ..... message-templates/altcraft-variables-and-functions/index.mdx
Logical expressions in messages message-templates/altcraft-variables-and-functions/66511.mdx
Loops in messages message-templates/altcraft-variables-and-functions/2196195.mdx
Market variables in templates message-templates/altcraft-variables-and-functions/56257541.mdx
Using the JSONPath functionality ..... message-templates/altcraft-variables-and-functions/jsonpath.mdx
Dynamic HTML content message-templates/dynamic-content/66496.mdx
Dynamic JSON content message-templates/dynamic-content/66498.mdx
Content from SQL database in templates message-templates/dynamic-content/66501.mdx
Dynamic API content message-templates/dynamic-content/66503.mdx

campaigns

Mailings ..... campaigns/index.mdx
Broadcast mailings campaigns/66526.mdx
Trigger mailing campaigns/66530.mdx
Regular mailing campaigns/9405823.mdx
Multivariate testing campaigns/9405875.mdx
Placement mailing campaigns/9406025.mdx
Mailing testing campaigns/66542.mdx
Mailing schedule campaigns/66538.mdx
Mailings calendar campaigns/66540.mdx
Mailing log — errors and troubleshooting ..... campaigns/campaign-log-errors.mdx
Managing the Sender Queue ..... campaigns/managing-a-queue.mdx

automation scenarios

Automation scenarios ..... automation-scenarios/index.mdx
Managing scenarios automation-scenarios/9406861.mdx
Nodes of the scenario automation-scenarios/9407058.mdx
Classic marketing scenarios automation-scenarios/22708542.mdx
Step-by-step welcome scenario guide automation-scenarios/25985320.mdx
Scenario for automatic notification of the manager automation-scenarios/28934313.mdx
Abandoned cart scenario automation-scenarios/56254697.mdx
Cycle Handling in Automation Scenarios ..... automation-scenarios/cycle-processing.mdx
High-priority scenarios ..... automation-scenarios/high-priority-scenarios.mdx

super campaigns

Campaigns ..... super-campaigns/index.mdx
Working with Campaigns ..... super-campaigns/where-to-start.mdx
Local control groups (LCG) ..... super-campaigns/lcg.mdx
Stratification violation error when limit is reached ..... super-campaigns/stratificcation_cases.mdx
Audience expansion in campaigns ..... super-campaigns/audience-expansion.mdx
Audience building ..... super-campaigns/audience-distribution.mdx

market

Market ..... market/index.mdx
Market settings market/49710292.mdx
Orders market/49710295.mdx
Market variables in message template market/56252235.mdx
Guide: how to send an order confirmation email market/56257397.mdx
Products ..... market/products/index.mdx
How to create a product manually market/products/56249576.mdx
How to import a product from a file market/products/56249588.mdx
Scheduled product import ..... market/products/feed-import.mdx
Product and SKU Segments ..... market/products/segments.mdx
Preparing the YML file ..... market/products/yml.mdx

loyalty programs

Loyalty programs ..... loyalty-programs/index.mdx
Loyalty programs ..... loyalty-programs/loyalties.mdx
Loyalty integration with external systems ..... loyalty-programs/loyalty-market-integraton.mdx
Creating a loyalty program from scratch ..... loyalty-programs/loyalty-where-to-start.mdx
Basic loyalty program use cases ..... loyalty-programs/loyalty-cases.mdx
Order Segments ..... loyalty-programs/order-segments.mdx
Promotion codes loyalty-programs/9406715.mdx

reports and analytics

Reports and analytics ..... reports-and-analytics/index.mdx
Channel report ..... reports-and-analytics/channels-report.mdx
Traffic report ..... reports-and-analytics/traffic-report.mdx
Cohorts report ..... reports-and-analytics/cohorts-report.mdx
Lifetime report ..... reports-and-analytics/lifetime-report.mdx
Funnels report ..... reports-and-analytics/funnels-report.mdx
Goals report ..... reports-and-analytics/goals-report.mdx
Audience growth report ..... reports-and-analytics/audience-growth-report.mdx
Click map report ..... reports-and-analytics/click-map-report.mdx
Loyalty programs report ..... reports-and-analytics/loyalty.mdx
Bounces report ..... reports-and-analytics/bounces.mdx
Undeliveries report ..... reports-and-analytics/undeliveries.mdx
Report on global control groups ..... reports-and-analytics/gcg.mdx
Summary report ..... reports-and-analytics/summary-report/index.mdx
Summary report metrics reports-and-analytics/summary-report/33063594.mdx

weblayers

Data analytics weblayers/forms/37848282.mdx
Binding data channel and forms weblayers/forms/37849222.mdx
NPS testing ..... weblayers/forms/nps.mdx
Create a form ..... weblayers/forms/form-creating/index.mdx
General settings ..... weblayers/forms/form-creating/general-settings.mdx
Form customization with custom code ..... weblayers/forms/form-creating/form-customization-code.mdx
Form constructor weblayers/forms/form-creating/37848206.mdx
Appearance weblayers/forms/form-creating/37848222.mdx
Actions and form publication weblayers/forms/form-creating/37848225.mdx
Conditional logic in forms and surveys weblayers/forms/form-creating/37849748.mdx
Pixels ..... weblayers/pixels/index.mdx
Goal customer actions and scoring weblayers/pixels/9406180.mdx
Pop-ups ..... weblayers/popups/index.mdx
Creating and publishing a pop-up weblayers/popups/76809121.mdx
Setting up a popup in the code editor weblayers/popups/109903904.mdx
Managing pop-ups manually via script weblayers/popups/109903905.mdx
Popup analytics weblayers/popups/109903934.mdx
Guide: pop-up for push subscriptions ..... weblayers/popups/popup-to-push.mdx
Case: Creating a pop-up with the "Wheel of Fortune" widget ..... weblayers/popups/wheel-of-fortune.mdx
Basic cases of placing a popup via the Tag Manager ..... weblayers/popups/popup-with-tagmanager.mdx
Tag Manager ..... weblayers/tag-manager/index.mdx
Configuring and installing Tag Manager weblayers/tag-manager/95715878.mdx
Trigger types weblayers/tag-manager/95715843.mdx
Variable types weblayers/tag-manager/95715946.mdx
Linking a pixel and the Tag manager weblayers/tag-manager/109903922.mdx

settings

Settings ..... settings/index.mdx
Account settings settings/9407614.mdx
Custom links settings/9407616.mdx
Virtual senders settings/9405938.mdx
Sending policies settings/9405554.mdx
Audit journal settings/3113292.mdx
Tags FAQ settings/89326254.mdx
Attribute settings ..... settings/attributes.mdx
Connections ..... settings/connections/index.mdx
Connection to Facebook Ads ..... settings/connections/fb-connection.mdx

Connection to Google Ads ..... settings/connections/google-ads.mdx
Connecting to Yandex.Audience™ ..... settings/connections/ya-audiences.mdx
Connection to 360dialog settings/connections/360.mdx
Connection to Edna ..... settings/connections/edna.mdx
Connection to Devino Telecom ..... settings/connections/devino.mdx
Connection to SMSTraffic ..... settings/connections/smstraffic.mdx
Connection to VK Ads™ ..... settings/connections/vk-connection.mdx
Connection to MTS OmniChannel ..... settings/connections/mts.mdx
Connection via OAuth 2.0 ..... settings/connections/oauth20.mdx
Basic Authentication connection ..... settings/connections/basic.mdx
Token Authentication connection ..... settings/connections/token.mdx
Custom Authentication Connection ..... settings/connections/custom.mdx
Connecting to MAX ..... settings/connections/max.mdx
Connection to Notify ..... settings/connections/notify.mdx
Connection to Rapporto ..... settings/connections/rapporto.mdx
Users, groups and access ..... settings/users-groups-access/index.mdx
Password and login security ..... settings/users-groups-access/login-security.mdx
Two-Factor Authentication (2FA) settings/users-groups-access/56255904.mdx

integrations

Facebook Ads Manager ..... integrations/facebook-ads-manager.mdx
Google Ads Audiences ..... integrations/google-audience.mdx
WhatsApp ..... integrations/whatsapp.mdx
Static segment synchronization integrations/9407286.mdx
Viber ..... integrations/viber.mdx
VK Ads ..... integrations/vk.mdx
Notify ..... integrations/notify.mdx
MAX ..... integrations/max.mdx
Yandex AppMetrica ..... integrations/yandex/yandex-app-metrica.mdx
Yandex.Audience™ ..... integrations/yandex/yandex-audience.mdx
Lpgenerator™ ..... integrations/action-hooks/lpgenerator.mdx
Tilda™ ..... integrations/action-hooks/tilda.mdx
Altcraft Action Hooks integrations/action-hooks/37849833.mdx
Action hooks event types integrations/action-hooks/37849805.mdx
Action Hook Message Structure integrations/action-hooks/37849819.mdx
JSON batch request (HTTP POST action hook) integrations/action-hooks/37849807.mdx
Message to RabbitMQ broker integrations/action-hooks/37849816.mdx
Message to RabbitMQ exchanger integrations/action-hooks/37849810.mdx
Message to Kafka broker integrations/action-hooks/37849813.mdx
Test event integrations/action-hooks/37849821.mdx
Connecting Altcraft to Albato integrations/albato-integrations/76808402.mdx
Launching the welcome scenario using Albato integrations/albato-integrations/768084024.mdx
Transmitting event data integrations/albato-integrations/768084025.mdx
Setting up a trigger mailing integrations/albato-integrations/768084026.mdx
Event registration integrations/albato-integrations/768084027.mdx
Integration scope integrations/additional-info/45514832.mdx
Data Transmitted During Synchronization ..... integrations/additional-info/transmitted-data.mdx

api

API requests: where to start ..... api/index.mdx
Import or update a profile ..... api/api-import.mdx
Trigger mailing launch ..... api/api-trigger.mdx
Engage profile in scenario ..... api/api-scenario.mdx

changelog

Changelog ..... changelog/index.mdx
v2026.3.78 ..... changelog/v2026378.mdx
v2026.2.77 ..... changelog/v2026277.mdx
v2026.1.76 ..... changelog/v2026176.mdx
v2025.4.75 ..... changelog/v2025475.mdx
v2025.4.74 ..... changelog/v2025474.mdx
v2025.3.73 ..... changelog/v2025373.mdx
v2025.2.72 ..... changelog/v2025272.mdx

v2025.1.71 changelog/v2025171.mdx
v2024.4.70 changelog/v2024470.mdx
v2024.3.69 changelog/v2024369.mdx
v2024.2.68.2 changelog/v20242682.mdx
v2024.1.68 changelog/v2024168.mdx

API / Developer Guide

API interaction api-interaction.mdx
Matching 37848489.mdx
Importing the API collection in Postman api-collection.mdx
List of API endpoints api-endpoints-list.mdx
Import profile profiles/import.mdx
Update profile profiles/update.mdx
Import multiple profiles profiles/2195504.mdx
Update multiple profiles profiles/2195501.mdx
Add multiple profiles profiles/2195496.mdx
Add profile to database profiles/2195477.mdx
Import profile to RabbitMQ profiles/2195492.mdx
Get profile data profiles/2195482.mdx
Uploading profiles to a file profiles/3115796.mdx
Delete profile profiles/28936545.mdx
Subscription fields functional update profiles/3113721.mdx
Database fields functional update profiles/3113720.mdx
Merging multiple profiles profiles/41484440.mdx
Unsubscribe profile from resource profiles/41484441.mdx
Profile splitting profiles/41484445.mdx
Get data for multiple profiles profiles/37847989.mdx
Add or edit subscription profiles/subscriptions/2195519.mdx
Get all profile subscriptions profiles/subscriptions/2195522.mdx
Get all subscriptions from multiple profiles profiles/subscriptions/2195524.mdx
Get profile subscription profiles/subscriptions/2195527.mdx
Delete profile subscription profiles/subscriptions/2195529.mdx
Restore deleted subscription profiles/subscriptions/2195531.mdx
Suspend all subscriptions profiles/subscriptions/2195533.mdx
Unsuspend all suspended subscriptions profiles/subscriptions/2195539.mdx
Get profile action history profiles/action-history/2195546.mdx
Get multiple profiles action history profiles/action-history/2195548.mdx
Profile relations profiles/profile-relations/index.mdx
Attach relation profiles/profile-relations/9406148.mdx
Detach relation profiles/profile-relations/9406152.mdx
Modify relation properties profiles/profile-relations/9406155.mdx
Overwrite relation properties profiles/profile-relations/9406157.mdx
Get profile relations info profiles/profile-relations/9406159.mdx
Get profile relations info profiles/profile-relations/get-relations-list.mdx
Get database statistics databases/get-counters.mdx
Update statistics on database databases/refresh-counters.mdx
Get database list databases/2195563.mdx
Get database information databases/2195565.mdx
Get database fields databases/2195567.mdx

Database wipe databases/2195815.mdx
Get resource statistics resources/get-counters.mdx
Update statistics on resource resources/refresh-counters.mdx
Get resources list resources/2195587.mdx
Get resource information resources/2195590.mdx
Get resource subscription fields resources/2195591.mdx
Create segment segments/2195594.mdx
Get statistics on resource segments/get-counters.mdx
Update statistics on segment segments/refresh-counters.mdx
Add or remove profile segments/9407390.mdx
Get profile data in a static segment segments/9407394.mdx
Update segment segments/2195596.mdx
Get segment information segments/2195598.mdx
Get segments list segments/2195600.mdx
Delete segment segments/2195602.mdx
Create suppression list suppression-lists/2195613.mdx
Update suppression list suppression-lists/2195615.mdx
Get suppression list info suppression-lists/2195617.mdx
Get the list of suppression lists suppression-lists/2195619.mdx
Delete suppression list suppression-lists/2195621.mdx
Upload suppression list data to file suppression-lists/76809125.mdx
Suppression list actions suppression-lists/suppression-list-actions/index.mdx
Check if email is suppressed suppression-lists/suppression-list-actions/2195633.mdx
Add email to suppression list suppression-lists/suppression-list-actions/2195625.mdx
Add multiple emails to suppression list suppression-lists/suppression-list-actions/2195627.mdx
Remove email from suppression list suppression-lists/suppression-list-actions/2195637.mdx
Remove all emails from suppression list suppression-lists/suppression-list-actions/2195641.mdx
Check if domain is suppressed suppression-lists/suppression-list-actions/2195635.mdx
Add domain to suppression list suppression-lists/suppression-list-actions/2195629.mdx
Add multiple domains to suppression list suppression-lists/suppression-list-actions/2195631.mdx
Remove domain from suppression list suppression-lists/suppression-list-actions/2195639.mdx
Remove all domains from suppression list suppression-lists/suppression-list-actions/2195643.mdx
Check if phone number is suppressed suppression-lists/suppression-list-actions/76809084.mdx
Add phone number to suppression list suppression-lists/suppression-list-actions/76809087.mdx
Add multiple phones to suppression list suppression-lists/suppression-list-actions/76809091.mdx
Remove phone number from suppression list suppression-lists/suppression-list-actions/76809096.mdx
Remove all phone numbers from suppression list suppression-lists/suppression-list-actions/76809100.mdx
Get templates list templates/2195606.mdx
Get template info templates/2195608.mdx
Delete template templates/56255090.mdx
Add template templates/56255095.mdx
Update template templates/56255277.mdx
Channel object templates/Channel-object_56255164.mdx
Get campaign information super-campaigns/get-scampaigns-info.mdx
Get campaign list super-campaigns/get-scampaigns-list.mdx
Activate campaign super-campaigns/scampaigns-activation.mdx
Complete campaign super-campaigns/scampaigns-complete.mdx
Deactivate campaign super-campaigns/scampaigns-deactivation.mdx

Get campaign status ..... super-campaigns/scampaigns-status.mdx

Activate mailing campaigns/2195660.mdx

Deactivate mailing campaigns/2195662.mdx

Get mailing list campaigns/2195648.mdx

Get mailing information campaigns/2195650.mdx

Get mailing log campaigns/2195652.mdx

Clone mailing campaigns/2195654.mdx

Delete mailing campaigns/2195656.mdx

Get mailing status campaigns/2195664.mdx

Get broadcasts list campaigns/broadcast-campaigns/2195885.mdx

Get broadcast information campaigns/broadcast-campaigns/2195887.mdx

Create broadcast mailing campaigns/broadcast-campaigns/2195891.mdx

Update broadcast mailing campaigns/broadcast-campaigns/2195895.mdx

Launch a broadcast mailing campaigns/broadcast-campaigns/3116196.mdx

Get regular mailings list campaigns/regular-campaigns/2195899.mdx

Get regular mailing information campaigns/regular-campaigns/2195901.mdx

Create regular mailing campaigns/regular-campaigns/2195903.mdx

Update regular mailing campaigns/regular-campaigns/2195905.mdx

Launch a regular mailing campaigns/regular-campaigns/3116201.mdx

Get trigger mailings list campaigns/trigger-campaigns/2195915.mdx

Get trigger mailing information campaigns/trigger-campaigns/2195917.mdx

Create trigger mailing campaigns/trigger-campaigns/2195922.mdx

Update trigger mailing campaigns/trigger-campaigns/2195925.mdx

Trigger launch (API call) campaigns/trigger-campaigns/2196177.mdx

Profile import + trigger mailing launch campaigns/trigger-campaigns/22708362.mdx

Task for bulk trigger launch campaigns/trigger-campaigns/33063451.mdx

Task for bulk profiles import + trigger launch campaigns/trigger-campaigns/33063474.mdx

Bulk trigger launch campaigns/trigger-campaigns/37847471.mdx

Bulk profiles import + trigger mailing launch campaigns/trigger-campaigns/37847473.mdx

Clone a trigger mailing campaigns/trigger-campaigns/56258725.mdx

Data array ..... campaigns/trigger-campaigns/Data-array_41484531.mdx

Engage profile in scenario automation-scenarios/2195668.mdx

Import and engage profile in scenario automation-scenarios/28934227.mdx

Batch import and engage profiles in a scenario automation-scenarios/33063492.mdx

Task for batch import and engaging profiles in the scenario automation-scenarios/33063493.mdx

Get scenarios list automation-scenarios/95715617.mdx

Activate scenario ..... automation-scenarios/activate-workflow.mdx

Deactivate scenario ..... automation-scenarios/deactivate-workflow.mdx

Get scenario information ..... automation-scenarios/workflow-get.mdx

Change scenario priority ..... automation-scenarios/workflow-priority.mdx

Get profile tier in a loyalty program ..... loyalty/tiers.mdx

Export points transactions ..... loyalty/transactions-export.mdx

Expiring points for a period ..... loyalty/expiring-points.mdx

Get profile account transactions ..... loyalty/transactions-list.mdx

Get trigger promotions list ..... loyalty/trigger-promotions.mdx

Accrue points to a member ..... loyalty/accrue-points.mdx

Redeem member points ..... loyalty/redeem-points.mdx

Commit temporary transaction ..... loyalty/commit-transaction.mdx

Preliminary Order Calculation ..... loyalty/preorder.mdx

Order Confirmation ..... loyalty/preorder-commit.mdx

Roll back temporary transaction ..... loyalty/rollback-transaction.mdx

Cancel points transaction ..... loyalty/cancel-transation.mdx

Get points account balance ..... loyalty/balance.mdx

Register member in a loyalty program ..... loyalty/register-member.mdx

Batch adding participants to loyalty program ..... loyalty/register-member-batch.mdx

Task for batch add participants to loyalty program ..... loyalty/register-member-batch-task.mdx

Remove member from loyalty program ..... loyalty/delete-member.mdx

Get form information ..... forms/get.mdx

Get form list ..... forms/list.mdx

Export form fill data by user ..... forms/fill.mdx

Export form fill data ..... forms/fills.mdx

Publish form ..... forms/activate.mdx

Unpublish form ..... forms/deactivate.mdx

Delete form ..... forms/delete.mdx

Import promo codes promocodes/2195573.mdx

Get promo code information promocodes/2195575.mdx

Activate promo code promocodes/2195577.mdx

Update promo code promocodes/2195579.mdx

Attach promo code promocodes/2195581.mdx

Detach promo code promocodes/2195583.mdx

Get all promo codes promocodes/3113099.mdx

Goals and goal values registration goals/2195843.mdx

Processing and adding a subscription application-push-notifications/2196103.mdx

Add app push events application-push-notifications/2196102.mdx

Order data object market/market-objects/56249763.mdx

Product data object ..... market/market-objects/Product-data-object_56249759.mdx

SKU data object ..... market/market-objects/SKU-data-object_56249769.mdx

Categories array ..... market/market-objects/Categories-array_56249780.mdx

Custom fields array ..... market/market-objects/Custom-fields-array_56249775.mdx

Import order and item statuses market/orders/56249746.mdx

Get orders list market/orders/56249749.mdx

Delete order market/orders/76808865.mdx

Get order status ..... market/orders/get-status.mdx

Update order line status ..... market/orders/update-order-line-status.mdx

Import products, SKUs and categories market/products-and-sku/56249718.mdx

Import SKUs and categories market/products-and-sku/56249733.mdx

Get products list market/products-and-sku/56249723.mdx

Get SKUs list market/products-and-sku/56249724.mdx

Delete products ..... market/products-and-sku/delete-product.mdx

Delete SKU ..... market/products-and-sku/delete-sku.mdx

Get summary report analytic-reports/2195743.mdx

Get soft bounces report analytic-reports/2195748.mdx

Get undeliveries report analytic-reports/2195750.mdx

Get senders list senders/2195754.mdx

Get virtual senders list senders/virtual-senders/2195762.mdx

Get virtual sender information senders/virtual-senders/2195765.mdx

Clone virtual sender senders/virtual-senders/2195767.mdx
Create virtual sender senders/virtual-senders/2195769.mdx
Update virtual sender senders/virtual-senders/2195771.mdx
Delete virtual sender senders/virtual-senders/2195773.mdx
Add segmentation query ..... datatables/query_segments/add.mdx
Update segmentation query ..... datatables/query_segments/update.mdx
Get segmentation query information ..... datatables/query_segments/get.mdx
Get segmentation queries list ..... datatables/query_segments/list.mdx
Delete segmentation query ..... datatables/query_segments/delete.mdx
Add template query ..... datatables/query_templates/add.mdx
Update template query ..... datatables/query_templates/update.mdx
Get template query information ..... datatables/query_templates/get.mdx
Get template queries list ..... datatables/query_templates/list.mdx
Delete template query ..... datatables/query_templates/delete.mdx
AKMTA object ..... objects/AKMTA-object_2195811.mdx
Content object ..... objects/Content-object_2195781.mdx
Email rule object ..... objects/Email-rule-object_2195791.mdx
File object ..... objects/File-object_2195839.mdx
Profile data object ..... objects/Profile-data-object_3116008.mdx
Sender object ..... objects/Sender-object_2195784.mdx
Sender types ..... objects/Sender-types_2195789.mdx
SMS rule object ..... objects/SMS-rule-object_2195796.mdx
Start schedule object ..... objects/Start-schedule-object_2195802.mdx
Subscription object ..... objects/Subscription-status_2195807.mdx
Trigger types ..... objects/Trigger-types_2195809.mdx
Upload file miscellaneous/2195777.mdx
Get message web version miscellaneous/2196089.mdx
Push providers miscellaneous/37848490.mdx
Deduplication of requests miscellaneous/41484312.mdx
How to send API request with RabbitMQ miscellaneous/44662785.mdx
List of gender identifications miscellaneous/9405775.mdx
Obtain valid values for fields: browsers, devices, tz, oses, languages ..... miscellaneous/dictionary.mdx
mSDK ..... sdk/mobile/index.mdx
Managing JWT and Role Token ..... sdk/mobile/jwt-r-token.mdx
Quick Start ..... sdk/mobile/android/quick-start.mdx
SDK Functionality ..... sdk/mobile/android/functionality.mdx
SDK Configuration ..... sdk/mobile/android/setup.mdx
Public SDK API ..... sdk/mobile/android/api.mdx
Firebase Cloud Messaging ..... sdk/mobile/android/providers/fcm.mdx
Huawei Mobile Services ..... sdk/mobile/android/providers/hms.mdx
RuStore ..... sdk/mobile/android/providers/rustore.mdx
Quick Start ..... sdk/mobile/ios/quick-start.mdx
SDK Functionality ..... sdk/mobile/ios/functionality.mdx
SDK Configuration ..... sdk/mobile/ios/setup.mdx
Public SDK API ..... sdk/mobile/ios/api.mdx
Apple Push Notification Service ..... sdk/mobile/ios/providers/apns.mdx
Firebase Cloud Messaging ..... sdk/mobile/ios/providers/fcm.mdx
Huawei Mobile Services ..... sdk/mobile/ios/providers/hms.mdx

Quick Start ..... sdk/mobile/react-native/quick-start.mdx
SDK Configuration ..... sdk/mobile/react-native/setup.mdx
SDK Functionality ..... sdk/mobile/react-native/functionality.mdx
Public SDK API ..... sdk/mobile/react-native/api.mdx
Provider setup ..... sdk/mobile/react-native/providers.mdx
Quick Start ..... sdk/mobile/flutter/quick-start.mdx
SDK Configuration ..... sdk/mobile/flutter/setup.mdx
SDK Functionality ..... sdk/mobile/flutter/functionality.mdx
SDK Public API ..... sdk/mobile/flutter/api.mdx
Provider Setup ..... sdk/mobile/flutter/providers.mdx

Getting Started

Altcraft Platform is an omnichannel CDP platform designed for storing customer data and managing communications across multiple channels such as email, SMS, push notifications, messengers, and more.

The platform includes a wide range of features and tools, so during the first login it may not be immediately clear where to start and how different interface modules are connected. This article helps you understand the core logic of Altcraft Platform and points you to the documentation sections worth exploring next.

If you are just starting to work with the platform, we also recommend reviewing the FAQ and Glossary sections.

If you prefer learning through practical examples, this article also includes videos demonstrating basic operations: preparing a profile database and launching the first email mailing.

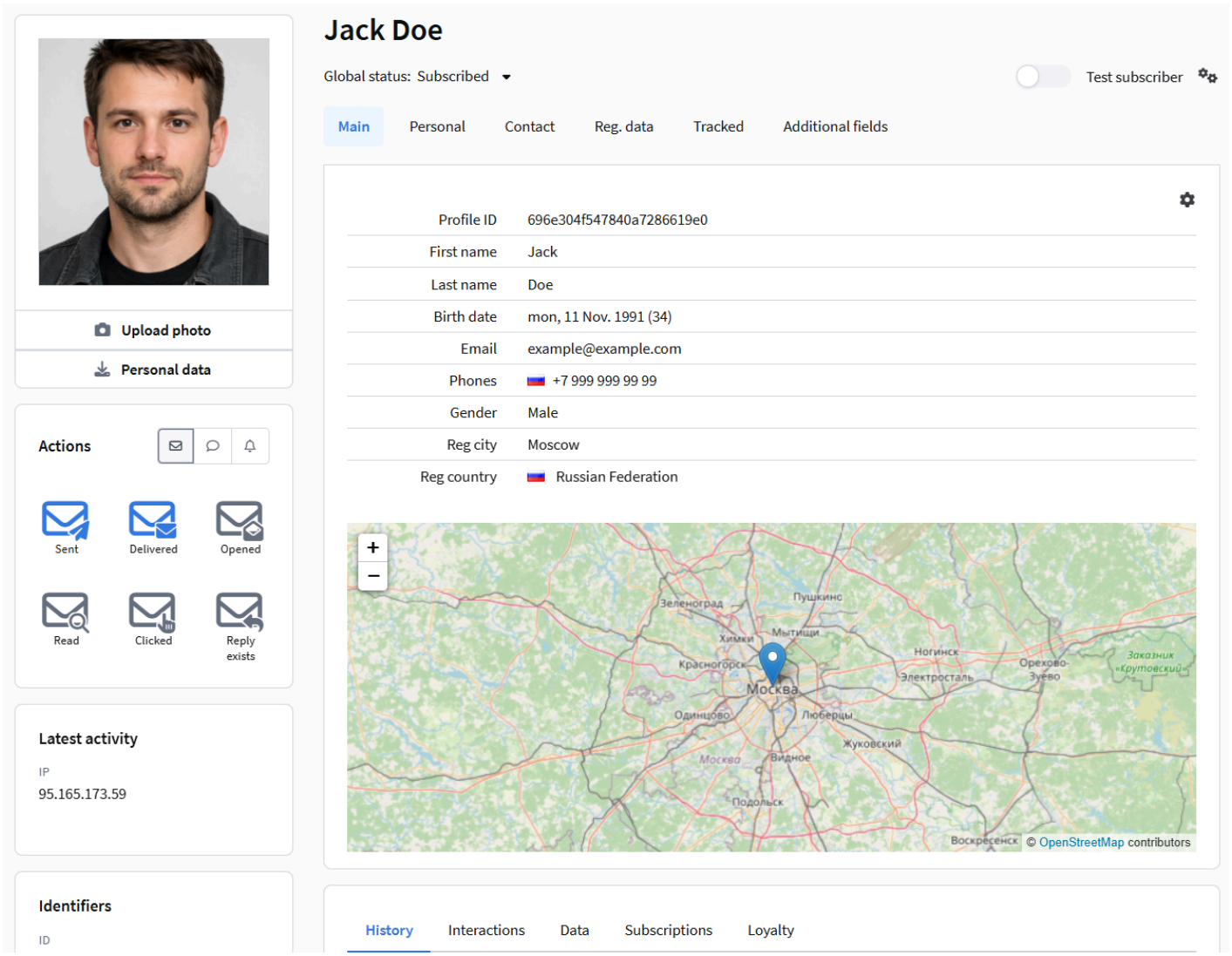
Profile Databases and Subscriber Profile

Altcraft Platform is built around data. All communications, automation, and analytics are based on customer profiles and their actions. All customer data in Altcraft is stored in **profile databases**.

A **profile database** defines the data structure: which fields are used, what data types are allowed, and which resources (subscription sources) can be associated with profiles. In most cases, a single profile database corresponds to a specific product, service, or business unit. More details on data structure configuration are available in this article.

A **subscriber profile** is a unified digital customer profile. It combines contact details, personal information, action history, channel subscriptions, and participation in marketing activities. In the documentation, you may encounter the terms “customer”, “subscriber”, or “user”, but in all cases they refer to the same object — the subscriber profile.

For example, a profile may initially contain only an email address, later include an SMS subscription, and then record activity from a mobile application. All this information is stored within a single subscriber profile.



A detailed description of the profile structure and its tabs is available in this article.

Resources and Subscriptions

Altcraft uses **subscription resources** to manage communications.

A **resource** is a logical content source within which a customer receives messages. It can be viewed as a context or interaction point: a website, mobile application, service, brand, or a specific company project.

![img](/img/attachments/user-guide-where-to-start/en/resourse-subs.png)

Resources allow you to separate customer subscriptions and consents. The same profile can be subscribed to multiple resources and have different subscription statuses in each of them.

A profile subscription links a customer to a specific resource and communication channel. It is the subscription that determines whether messages can be sent to particular contact details.

More information about configuring resources and subscriptions is available in this article.

How profile databases, resources, and subscriptions are created

Below is a basic process of preparing data for communications: creating a profile database, creating a resource, and adding profiles with subscriptions. These steps form the foundation for further segmentation, mailings, and automation scenarios.

Data and Action History

Altcraft Platform tracks all key customer actions and stores them in the subscriber profile history. This includes message sends and opens, link clicks, form submissions, website or app events, as well as order and promo code data.

![img](/img/attachments/user-guide-where-to-start/en/profile_history_events.png)

Action history is used for segmentation, automation, and analytics. For example, you can send a message only to customers who opened emails in the last seven days or made a purchase.

Audience Segmentation

Segmentation allows you to group profiles based on defined conditions.

![img](/img/attachments/user-guide-where-to-start/en/segment_cond.png)

Altcraft supports static, dynamic, and updatable segments. These segment types differ in how the audience is formed and how data is refreshed.

Static segments are used for working with fixed lists of profiles. Dynamic segments are recalculated at the time of message sending and always use up-to-date data. Updatable segments are recalculated on a defined schedule and store the result until the next update.

For example, you can create a segment of customers who placed an order in the last week or a segment of active subscribers who opened emails during the last month.

![img](/img/attachments/user-guide-where-to-start/en/segment_list.png)

More details about segment types and selection conditions are available in this article.

Communications

Communications in Altcraft Platform refer to sending messages to subscriber profiles through various communication channels. The platform supports email, SMS, Web Push, Mobile Push, messengers, and other channels that can be used individually or as part of omnichannel scenarios.

Each channel has its own configuration and delivery specifics, but the overall communication logic in the platform is unified. Messages are sent based on profile data and always take into account contact details, subscriptions, and consents.

Altcraft supports several main types of mailings. Broadcast mailings are used for one-time sends, such as news or promotions. Regular mailings are suitable for repeated communications based on a schedule. Trigger mailings automatically send messages in response to customer actions or specific events.

![img](/img/attachments/user-guide-where-to-start/en/mailing_type.png)

Communications can be launched for the entire profile database or for selected audience segments. Mailings are often part of more complex processes — automation scenarios or campaigns.

How to launch the first email mailing

> **INFO**
>
> To send email messages, a configured “DKIM” is required for the sender domain. It is not issued automatically. Please contact technical support via chat to set it up.

Below is an example of the full process of creating an email mailing: from preparing a template to launching the send. The video shows how working with profiles, subscriptions, and templates comes together in a real communication process.

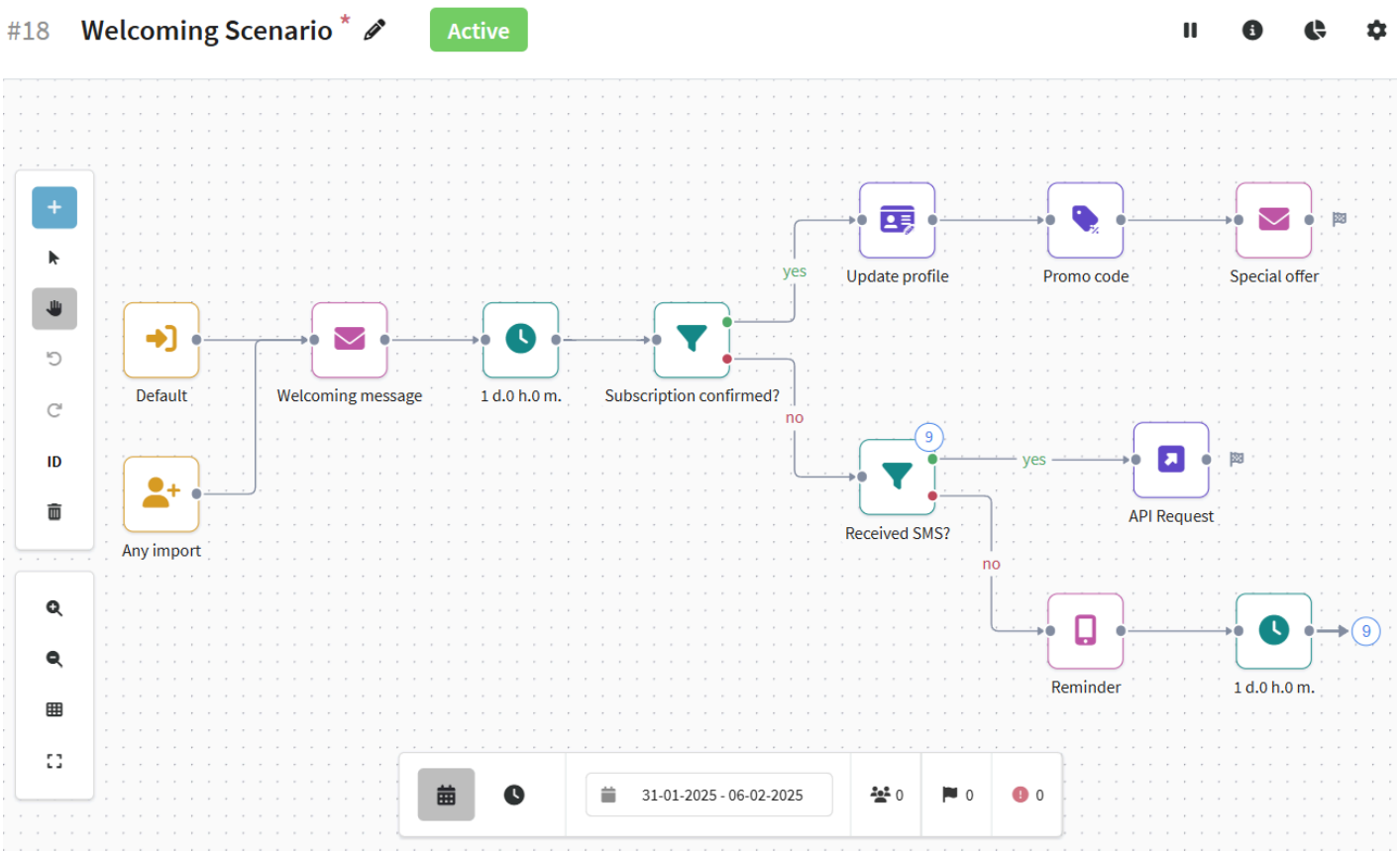
> **INFO: Notice:**
>
> In the interface shown in the video, the term “campaign” may be used to refer to a mailing.
> **In the current version of the platform, the terminology has changed:**
>
> **Mailings** are a separate tool for sending messages (Broadcast, Regular, Triggered).
>
> **Campaigns** are a different tool designed for building complex marketing processes and combining communications, scenarios, and customer interaction logic.

For your first hands-on experience with message sending, we recommend starting with step-by-step channel guides, for example Your First Email mailing. A full list of tutorials is available in the Communication Channels section.

Automation Scenarios

Automation scenarios allow you to build communication flows based on customer actions, events, and conditions.

Scenarios are used for onboarding, welcome journeys, reactivation, reminders, and other common marketing use cases. Profiles can enter scenarios either from segments or directly from the database if they meet the entry conditions.



An overview of the automation tool is available in the Automation Scenarios section, and examples of common scenarios can be found in Classic Marketing Automation Scenarios.

Other Platform Capabilities

Altcraft Platform is not limited to data storage, segmentation, and communications. The platform includes additional modules that support comprehensive customer lifecycle management.

Marketing activities can be managed using mailings. Learn more in the mailings section.

Working with product and order data is described in the Market section.

The loyalty module allows you to create reward programs and use promo codes. More details are available in the Loyalty section.

The web layer is used for data collection and subscription management. Learn more in the Web Layer section.

Analytics is covered in the Reports and Analytics section.

Integrations are described in the Integrations section, and technical capabilities are covered in the Developer Documentation and API: Getting Started.

FAQ

Managing profiles subscriptions

Are people who unsubscribe not by the link in the letter, but in the mail service interface, marked as unsubscribing?

Yes, if mail service supports special header, which is transferred into the letter.
Global unsubscription is performed, if resource was not chosen before mailing.
Resource unsubscription is performed, if resource was chosen before mailing.

Why the profile status is changed when unsubscription from non-resource mailing is performed?

If the mailing is performed without selecting resource, global profile status is used.

If the user will subscribe again, will he restore from the suppression list?

No, if the mailing was performed without resource, or with the same resource. Use API to restore him.
In future, we recommend to use several resource subscriptions.

What is resource identifier?

Resource is the mailing source by one or several channels. For example, one website - one resource.
Resource identifier is a number, which is displayed next to its name. It is used in API queries.

What is the difference between suppression list and unsubscription?

Suppression list and unsubscription are different things. Suppression list is commonly used for temporary exception from the mailings, like the situation when content will have negative reaction from the subscribers.

How can we manage subscription confirmation for users, who can see only text version?

They can also have the link. You can set its name like `<name="Name">`. Part in angle brackets will not be displayed while mailing.

What essences can be marked with tag in the platform?

You can mark databases, segments, channels, templates, scheduled imports, suppression lists, mailings, campaigns, workflows, promocodes, summary reports.

Why does the number of unsubscribe link clicks not match the number of unsubscribes?

A user can unsubscribe not only via the link in the email, but also through the service interface (e.g., by clicking a button). These actions are not always tracked as link clicks. As a result, the total number of clicks may differ from the number of actual unsubscribes.

Managing databases

How can developer configure data exchange with the website?

Developer must be provided with API token and link to documentation page — see User API.

Can we upload database without email addresses? For example, for SMS mailing.

Yes, you can. Select coincidence by profile email address. This field will be filled with data like “ID@localhost”.
Profile identifier will be generated by the platform.

How can we spot subscriber territory or country?

The platform can spot geolocation by IP address.

What field type can be used for currency?

Unfortunately, there are no fractional fields in the database. Use integer type.

How can we transfer resource subscriptions during profile import?

You can include subscriptions array into the `data` object during subscriptions import like:

```
subscriptions": [
      {
        "resource_id": 46,
        "channel": "email",
        "email_data": {
          "email": "example_email@example.org",
        },
        "priority": 0,
        "status": "subscriber"
      }, ...
```

Or simple option:

```
subscriptions": [
      {
        "resource_id": 46,
        "channel": "email",
        "email": "example_email@example.org",
        "priority": 0,
        "status": "subscriber"
      },
```

Can we transfer several profiles in one API query for profile import?

For several profiles import, use Massive profiles import.

How can we clear the database?

Use API query to clear the database.

Also, you can clear the database using query to MongoDB.

Can we spot devices, OS and browsers used by customers?

Information is collected in the User Agent field in customer profile.

How can we create the field, which displays customer's interests in different categories?

You can create "Tags" type field in the database. Tags can be added and removed manually, using APIs and workflows.

How can we clone the database with all its profiles and subscriptions?

When cloning the database, only its structure is copied. For copying profiles, press "Copy profiles" button, and after that you will be suggested to select the segment.

If i upload customers from the database, will email addresses be uploaded from the profile or from the profile subscriptions?

You can export only encrypted subscriptions. By default, all profile subscriptions will be exported. In the export menu, you can configure, what subscriptions will be considered.

Why does the report metric not match the number of profiles in the segment?

Reports are based on events, not individual profiles. A single profile can generate multiple identical events — for example, clicking a link twice or opening an email multiple times. In such cases, the event count exceeds the number of unique profiles in the segment.

Why doesn't the static segment recalculate automatically?

A static segment does not update automatically — it is a fixed list of profiles formed based on the specified conditions at the time of creation. If the data in the database has changed, the static segment must be recalculated manually. To do this, click the **"Recount** button at the top of the segment editor.

When recounting, only the current filters will be used. Profiles added or removed manually will not be preserved.

Managing mailings

What is mailings queue used for?

It displays present mailing status for certain mailing.

How can we send mailing to subscribers, who did not receive it?

You can create the segment and select customers, who did not receive the message.

Is there a global limit of mailing amount to one customer in a day or a week? Can it be ignored for some messages? How can we configure it?

Yes, it is called Sending policy. It can be set for a resource or for the account. If necessary, it can be changed in some mailings.

How can we send the mailing partially?

You can set limits for sending in the mailing.

I open the letter and put it into spam, why the complain is not displayed in the report?

Perhaps, FBL/CFL was not configured. Contact your system administrator.

Can we have an example of API query for trigger sending to several subscribers?

```
{
  "token": "e00c3da0a53e4e54a795da58701f4d70",
  "id": 5511,
  "custom_request": {
    "Custom_ID": {
      "$in": [
        "IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"
      ]
    }
  }
}
```

Can we use blind carbon copy during sending mailings?

Yes, Altcraft Platform supports BCC in messages. To use it, see mailing Smart settings and enter address for BCC in **Send hidden copy to these addresses** block.

Managing message templates

How can we use data tables?

See Using content from SQL database in the messages.

Managing scenarios

How does the profile move through the scenario if the mailing element failed to send a message?

If sending a message ended in an error (e.g. due to a technical reason), the profile automatically moves to the next element according to the scenario.

How does the profile move through the scenario if there is an error in the element of an API call, profile update or condition?

If an error occurs in the element and the scenario does not provide a separate exit for error handling, the profile will be excluded from the scenario. This is done to prevent incorrect execution.

The HTTP element supports error handling — if a request fails, it can retry or send the profile along an alternative path.

What happens if you delete the element where the profiles reside?

If an element is deleted during scenario execution and it contains active profiles, such profiles will be immediately excluded from the scenario.

How to see the apicontent of a profile?

To view the apicontent of a profile, hover the cursor over the desired element in the scenario. An icon with a number above it will be displayed - this is the number of profiles in the element. Click on this icon to open the list of profiles. In the opened window you can see the data of each profile, including the apicontent block, if API requests were used in the scenario.

Recommendations for managing scenarios

- Try to avoid conditions with negation (e.g. “is not equal”) - it complicates the logic and reduces the visibility of the scenario.
- It is recommended to filter out most of the profiles already in the start element to minimize unnecessary load on the scenario.
- Use static segments.

Other

Not all functions are available in the interface.

Contact platform administrator for turning Smart settings on.

What coding is used for files import?

UTF-8 coding is used.

How can we download the report?

Select Tasks in the main menu.

Is it possible to automatically update profile fields during event transfer into the platform?

Yes, you can use workflows for this.

Why are there no complains in the report?

FBL must be configured for complaints display in the report.

> **NOTE**
>
> Complaints are processed with automatic subscription change.

What is pixel? What is it used for?

Pixel is a script, that send query to the tracking server, when subscriber performs any actions on the website or in the application.

Pixel can transfer information related to this action, like website address, which is subscriber source, achieved goal and others. This information is written into every pixel code and it is transferred in profile history. Achieved goals and action price can be used for segmentation, including automatic segmentation in the workflows.

Also, pixel information is used in “Pixels” and “Goals” reports.

Is it possible to make segment by User-Agent parameter?

Yes, it is divided by Browser/Platform in the segments.

Promocodes import and generation buttons are not available.

You need to save loyalty program first.

The file name in the attachment in emails has incorrect name.

The problem can be resolved by the following methods:

1. Reinstall Outlook from the Standard package.
2. Configure the Outlook client: use **UTF-8** for the **Outgoing message encoding parameter**.

- In Outlook select **File** -> **Option** -> **Advanced**.
- In the **International options** section, set the **Preferred encoding for outgoing messages** option to **UTF-8**.

3. Avoid using Russian characters in the file name.
4. Consider using a different version of Outlook.

How are email opens and reads tracked?

At the end of each email, two invisible 1×1 pixels are automatically added: an open pixel and a read pixel.

Open pixel — when this pixel loads, the platform records an email open event.

Read pixel — placed after the main email content. When this pixel is requested, the server keeps the HTTP connection open for up to 12 seconds and measures how long the mail client holds the connection. Based on the connection duration, the event type is determined:

- **Glanced:** 0–4 seconds
- **Skimmed:** 4–8 seconds
- **Read:** 8+ seconds

If the connection is closed before 12 seconds (for example, the recipient closed the email), the actual viewing time is recorded and the corresponding event type is registered.

NOTE: Email tracking limitations

Open, glance, skim, and read metrics depend on the loading of tracking pixels in the email. Some mail clients and proxy servers may block or proxy images, which affects the accuracy of these metrics:

- **Image blocking** (Outlook, Thunderbird by default) — pixels do not load, opens are not recorded. Solution: add the tracking domain to the exceptions list.
- **Proxy servers** (Gmail Image Proxy, Apple Mail Privacy Protection, corporate proxies) — load pixels through a proxy, which records the open but does not allow measuring viewing time (glanced/skimmed/read).
- **Gmail prefetching** — may load images before the email is actually opened (preview pane).

For accurate engagement assessment, it is recommended to use click metrics (CTR), which do not depend on tracking pixels.

How is a profile’s geolocation tracked?

A profile’s geolocation is determined by its IP address at the moment the email is opened or an action is performed.

Altcraft glossary

Email Marketing Terms

API (Application Programming Interface) — the language of communication of one computer program with another. By delivering requests and returning responses, two computer programs talk to each other in order to execute some common task together.

BCC (Blind Carbon Copy) — hidden recipients of the letter, whose addresses are not shown to other recipients.

Caller ID— the identifier of the subscriber-sender of SMS messages. It can contain Latin letters and numbers without spaces.

Double Opt-In — the user indicated his email at the moment of subscribing and confirmed it by clicking on the special link in the confirmation letter. Advantage: a more interested and active audience gets into the database. Disadvantage: conversion reduces, because a part of the audience does not confirm the subscription.

DKIM (Domain Keys Identified Mail) — an email authentication technique that adds a digital signature to the email. It allows the receiver to check that an email was indeed sent and authorized by the owner of that domain.

OR (Open Rate) — the percentage of subscribers who open a specific email out of your total number of subscribers.

CTR (Click-Through Rate) — the ratio of users who click on a specific link to total nubmer of delivered messages.

CTOR (Click to Open Rate) — the ratio of users who click on a specific link to the number of total users who open the letter.

SB (Soft Bounce) — the letter returns because the email server is temporarily unable to deliver your message. There are two possible reasons: the recipient's email box is full or recipient's email server is temporarily inaccessible.

SPF (Sender Policy Framework) — an email-authentication technique that allows you to check if the sender's domain is forged. It is used to prevent spammers from sending messages on behalf of your domain. You need to add a special TXT record to the sending domain (sender domain, from domain), which determines the policy of allowing sending this domain from various hosts.

SQL (Structured Query Language) — a special language used for managing databases. With the help of SQL, you can set the structure of the database, organize the selection of data from it according to some specific criteria, delete data.

HB (Hard Bounce) — email addresses that do not accept emails due to invalid emails or an unexpected error during sending.

Pop-up — a small window that appears on top of the main content of a web page.

SMS gateway — an infrastructure component that enables the delivery of SMS messages from the Altcraft platform to mobile network operators (or intermediaries). It handles routing, formatting, message queues, delivery reports, and error processing.

Event capture — a mechanism for recording and transmitting user events (e.g., page views, cart additions, link clicks) to the Altcraft platform. These events are used for segmentation, personalization, trigger-based communications, and analytics.

Web version of email — an online page that displays the content of an email (HTML) identical to the sent message, but accessible via a web browser.

Altcraft Platform internal terms

Subscriber profile — contains information about mailing receivers - customers, partners.

Profiles database — defines structure of profile data fields and subscriptions.

Static segment — fixed profiles list, formed by several conditions, or by other ways (manually, by scenario or API). Static segment can be counted during selection from the database, but it will not be recounted automatically.

Dynamic segment — set of rules, that will be used for selection in the moment of using the segment — like mailing launch.

Profile subscription — a record that combines contact data for a specific communication channel and information about registration.

Resource — objectfor organizing customer subscriptions to certain content via certain communication channels.

Mailing — creates message for certain resource subscribers, based on selected template, and adds customer information from the database, dynamic content, parts and fragments.

Broadcast mailing — tool used when you need to communicate to a specific audience one single time.

Regular mailing — a tool for organizing **massive multiple** mailings to your subscribers.

Trigger mailing — tool for organizing transactional messages.

Multivariate testing — mailing that can have from 2 to 8 alternatives with different content and settings. You can compare efficiency of this messages and select the best one.

Mailing log — tool for tracking all stages and errors during mailing sending.

Message template — a set of content that you send to your customers within one or more marketing mailings.

Scenario — sequences of actions used for marketing workflow automation.

Channels report — a report that provides data on all events tracked in mailings: sends, deliveries, opens, clicks, and more.

Traffic report — a report that provides data on profile traffic from sources such as channels, forms, pixels, and pop-ups.

Cohorts report — report by groups of subscriber profiles from specific databases imported at the same time period.

Lifetime report — a tool to analyze customer cohorts lifetime.

Conversion funnel report — you can analyze your marketing strategy conversion or count it for certain action using this report.

Goals report — this report shows analytics about website visits and performing target actions like: income form deal, partner commission etc.

Audience growth report — report that shows subscriber profiles quantity changes.

Click map report — report that allows you to compare different message versions and different links values

User panel — interface for user to work with Altcraft Platform.

Administrator panel — service interface used by Altcraft Platform administrator.

Tag — mark for databases, segments, message templates, database imports, forms, suppression lists, products and orders, used for navigation in platform; used to navigate and improve the search for these entities in the platform.

Connection — a mechanism type which let you authorize in external systems to expand Altcraft Platform capabilities with the functionality of these systems. You usually need an account in an external system with API access rights to create such a connection.

Sending policy — settings which regulate number of messages per client. Such policy can be applied to an account, a resource or a mailing.

Suppression list — ɜa list of email addresses, email domains and phone numbers, which you do not want to send messages to. Suppression list filters an email address or a phone number at moment of mailing launch.

Tracking domain — a domain for tracking subscriber activity. It allows you to find out which links the recipient of the message has clicked on.

Pixel — HTML or JavaScript code placed on website, used for customers behavior tracking.

Watchers database — profiles database, that can be used to control mailings sending process and result. These profiles will receive same messages as your customers.

Market — functionality that allows you to import product and order data from various marketplaces into the platform and use this data to communicate with customers.

Form — a service that allows you to receive feedback from your customers. So you can find out the missing information directly by creating questionnaires or surveys.

Tag manager — a tool that allows you to manage HTML tags and JavaScript code without any need for developers. It is opportunity for you to create, add, and update code snippets on your site to track user activity and feed data into the platform.

mSDK — a mobile Software Development Kit (SDK) for integrating push notifications into mobile applications. It helps developers easily connect the Altcraft notification system to their Android and iOS apps via Firebase and APNs platforms.

Best Send Time Module — a feature that automatically calculates the optimal time to send a message to each user based on their behavior and historical data (activity, opens, clicks). It helps increase the effectiveness of mailings by minimizing idle time and improving engagement.

Subscription manager — a page that allows subscribers to manage their profile data and subscriptions to a resource: pause, resume, or suspend them for a specified period (14, 30, 60, or 90 days). If your company already uses its own subscription management system, it can be integrated with the Altcraft Platform.

Documentation archive

With the release of new versions of Altcraft Platform, the documentation presented on the website is updated in accordance with platform changes. For user convenience, documentation for outdated versions is stored for six months after the release of the next version. Below are links to the administrator documentation, user documentation, and API reference combined into a single PDF file.

- **Version 77** — PDF
- **Version 76** — PDF
- **Version 75** — PDF
- **Version 74** — PDF
- **Version 73** — PDF
- **Version 72** — PDF
- **Version 71** — PDF

INFO: Important

The documentation versions listed above may contain factual inaccuracies. No changes are made to archived versions; they remain in the state in which they were archived.

Profiles and databases

How to start

Create a database

![img] (/img/attachments/3114123/9408120.png)	Start from creating a database, where you will store digital profiles of your customers. Define the structure for your base, using standard data fields and custom data fields to implement **Creating a database is your first step in using Altcraft Platform.**

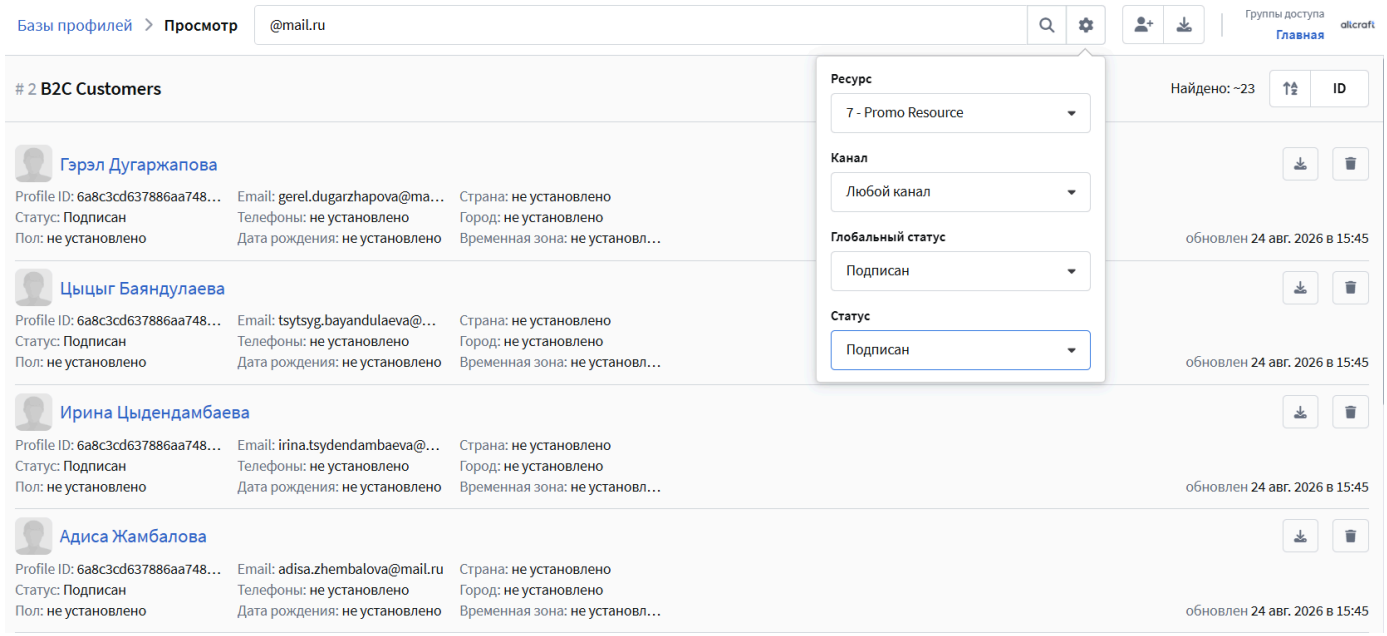
Create subscriptions resource

![img] (/img/attachments/3114123/9408121.png)	Communications with customers in Altcraft is designed for interaction with several communication channels, including different contacts in one channel. Subscriptions of profiles for different types of mailing (advertising, inbound, service, partners) are grouped into Resources. A resource is a content publisher for which you can separately configure sender data, tracking, sending policies, and also communications channels used. **Create at least one resource to start communication with your customers**

TIP

To **find** a specific customer profile in your databases, use the **Search** tool in the **Data** section. Enter the customer's data — the system will automatically determine the search mode.

You can set search parameters: **Resource**, **Channel**, **Global status**, and subscription **Status**.



Contents

Subscription resources

Resource in Altcraft is a tool for organizing customer subscriptions to certain content via certain communication channels. Subscriptions to different resources are not connected to one another. Unsubscribing to one resource does not affect other subscriptions and **global profile status**, thus a subscriber can choose which content to receive - and you get more opportunities to retain your customer.

One or several communication channels can be configured in a resource for forming subscriptions. A resource can be:

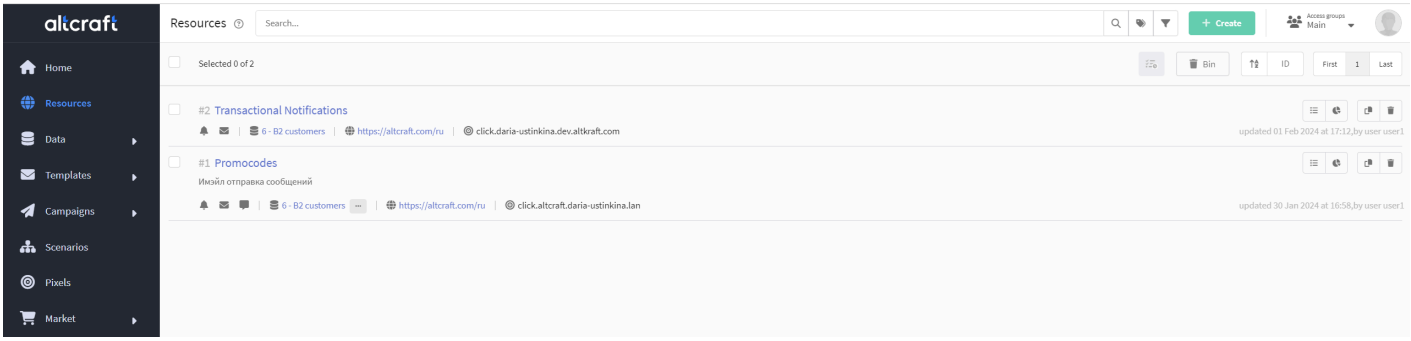
- Website or landing page
- Mobile or desktop application
- Registration form on the partner's website
- Certain type of mailing
- Special offer or product

For example, if a subscriber is not satisfied with advertisement mailings, but he is interested in information mailings, using two resources, the chance of saving potential customers and making a deal is much higher.

Different resources can use different tracking and sender domains - and different **From names** as well. In channel reports, general statistics intersections are available for resources, which makes effective end-to-end analysis of different marketing strategies possible.

Managing resources

Choose **Resources** in the Main menu to start managing resources.



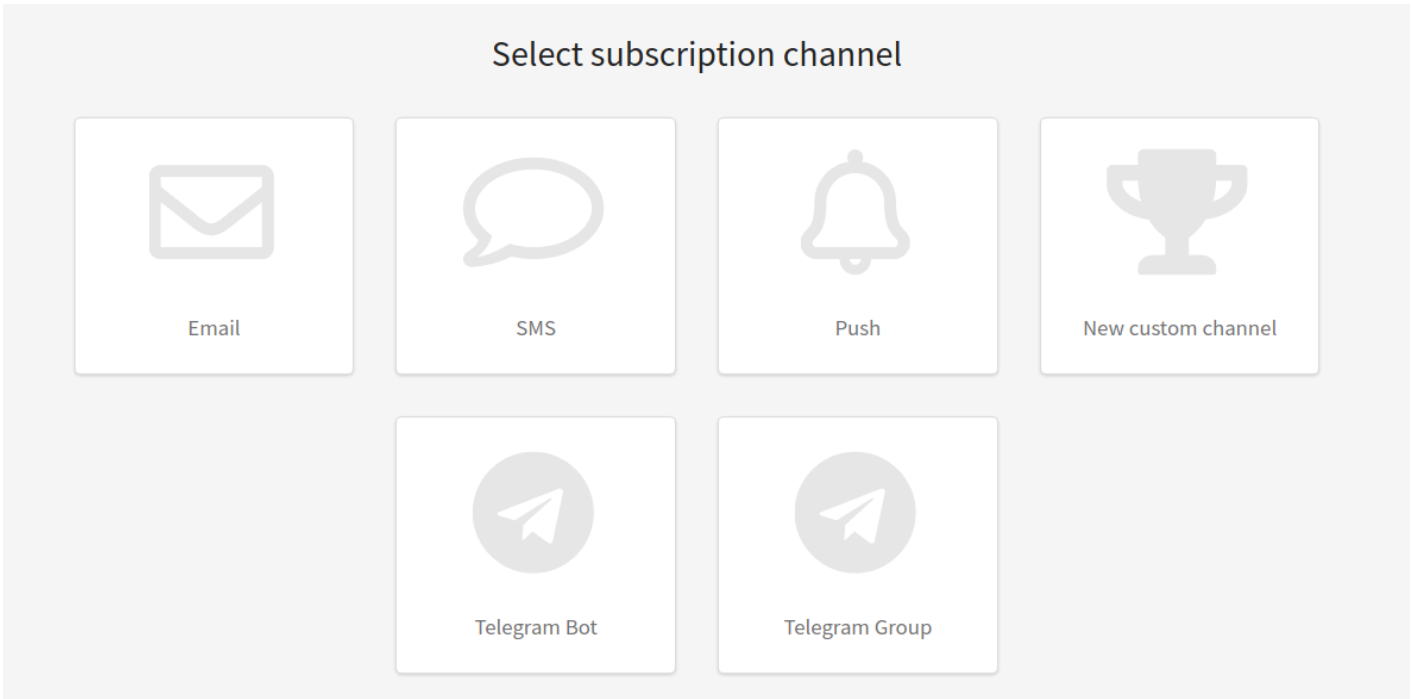
Click on the resource name to edit it. Use the action menu on the right side to view the resource's subscribers

, clone the resource

, or delete it

. To add a new resource, click **+ Create** at the top of the page.

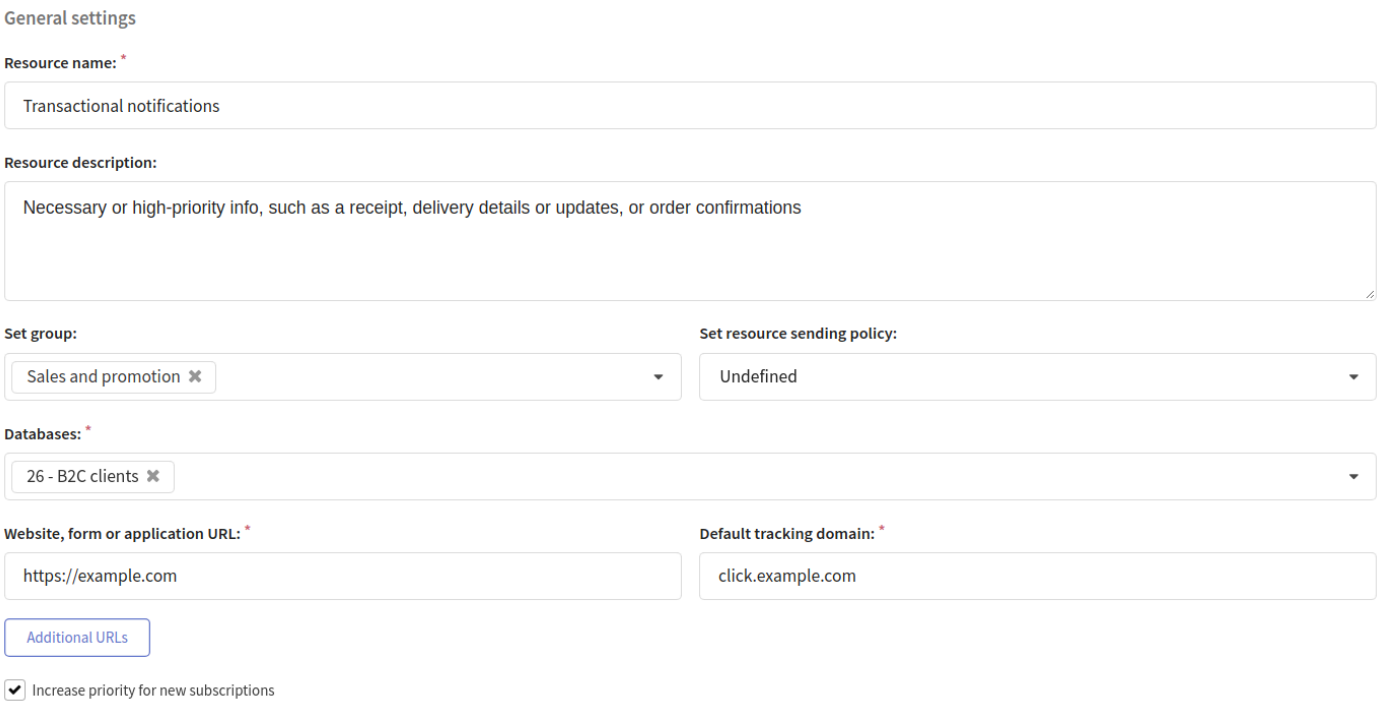
Next, select a primary subscription channel. Note that you can add another one at any time:



General settings

For a new resource, enter a **Name** and **Description** (optionally). For every resource you can configure:

- **Access group** — place resource in account **objects group**.
- **Resource sending policy** — limitcommunication with one subscriber within the resource.
- **Databases** — selectdatabases whose profiles can subscribe to the resource.
- **Website, form or application URL** — URL of your resource or its opt-in page. Additional URLs can be added.
- **Default tracking domain** — domain for tracking subscriber actions within the sent messages. In mailing **Smart settings** you can override or add tracking domains.



TIP: Increase priority for new subscriptions

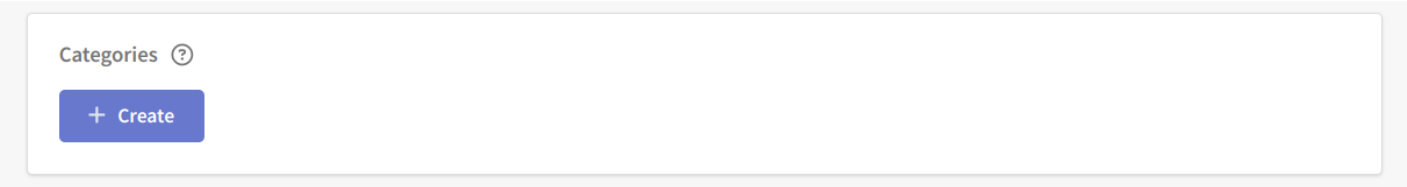
A user can subscribe to one resource several times by entering different contact details (email or phone number). If you activate the "**Increase priority for new subscriptions**" option, the contact information that the user provided during the last subscription will become a priority. It is to this email or number that messages will be sent in subsequent mailings.

Please note that the "Increase priority for new subscriptions" option applies only if the priority is not explicitly specified when adding a new subscription. If the priority of the new subscription is explicitly specified (for example, in an API request), then the priorities of existing subscriptions remain unchanged.

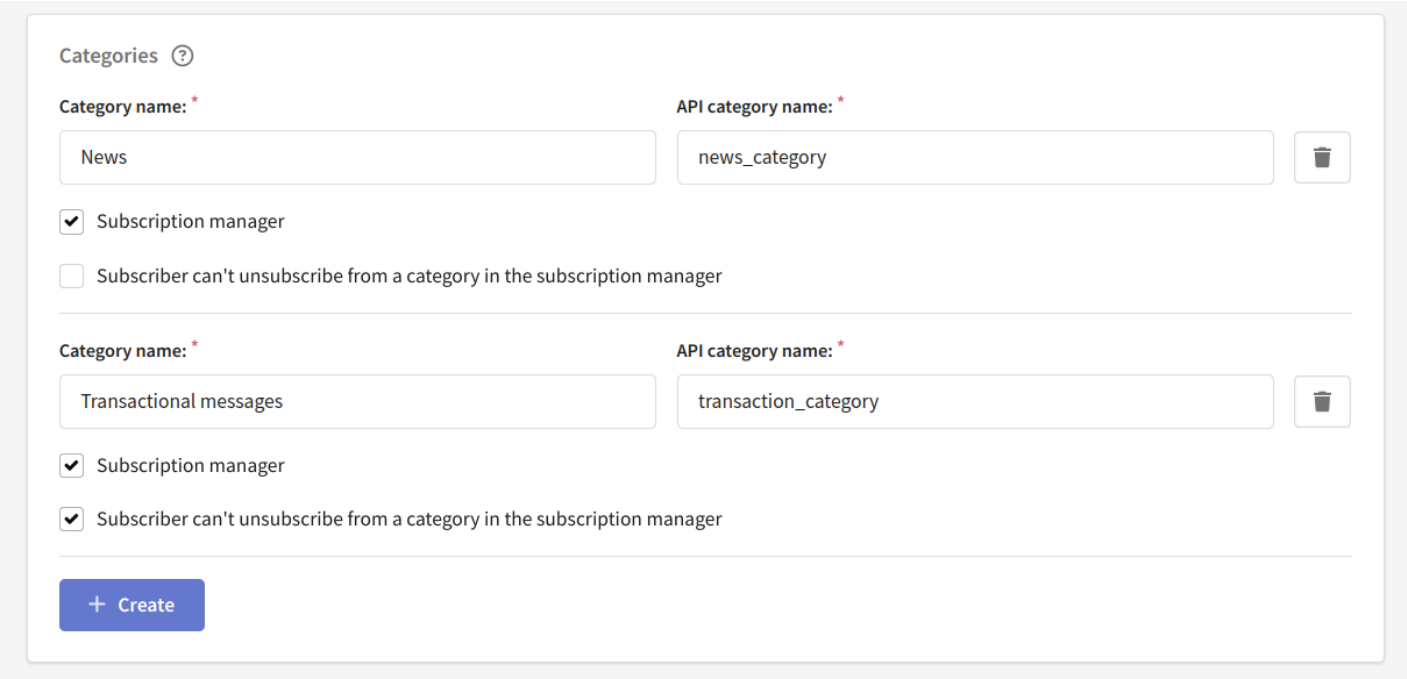
Resource categories

Categories enable you to specify the type of content a profile is subscribed to. For instance, a resource like "Mobile App" can include categories such as "News", "Promotional Offers", and "Transactional messages". If a client unsubscribes from one category within a resource, they will still receive messages from other categories. Unlike subscribing to a resource, subscribing to a category has just two statuses: "Subscribed" and "Unsubscribed". You can manually modify this status in customer profiles or via API requests.

To add a new category to a resource, click the **+ Create** button:



Specify a custom **category name** and **name** that will be used in **API requests**:



Additionally, you have the option to add a category to the subscription manager, This way, you allow subscribers to manage their category subscriptions by their own. If receiving messages within this category is required, activate the option “Subscriber can’t unsubscribe from a category in the subscription manager”.

The number of categories is not limited. To delete a category, click the button

.

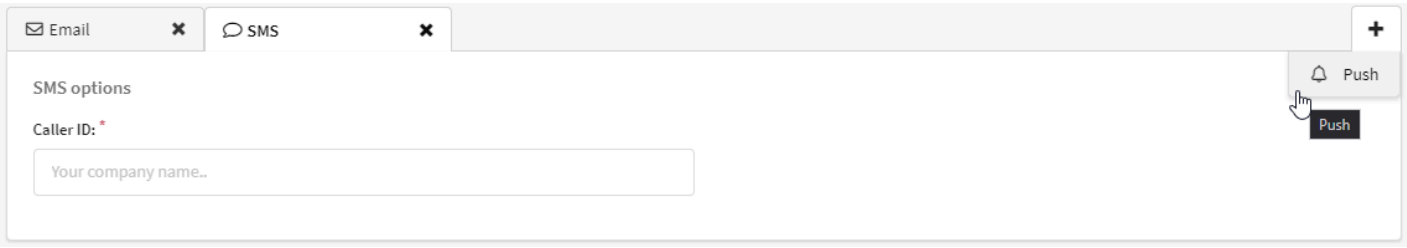
Please note that after saving the resource, you will be able to edit the category name, but not the API name.

Communication channels settings

INFO

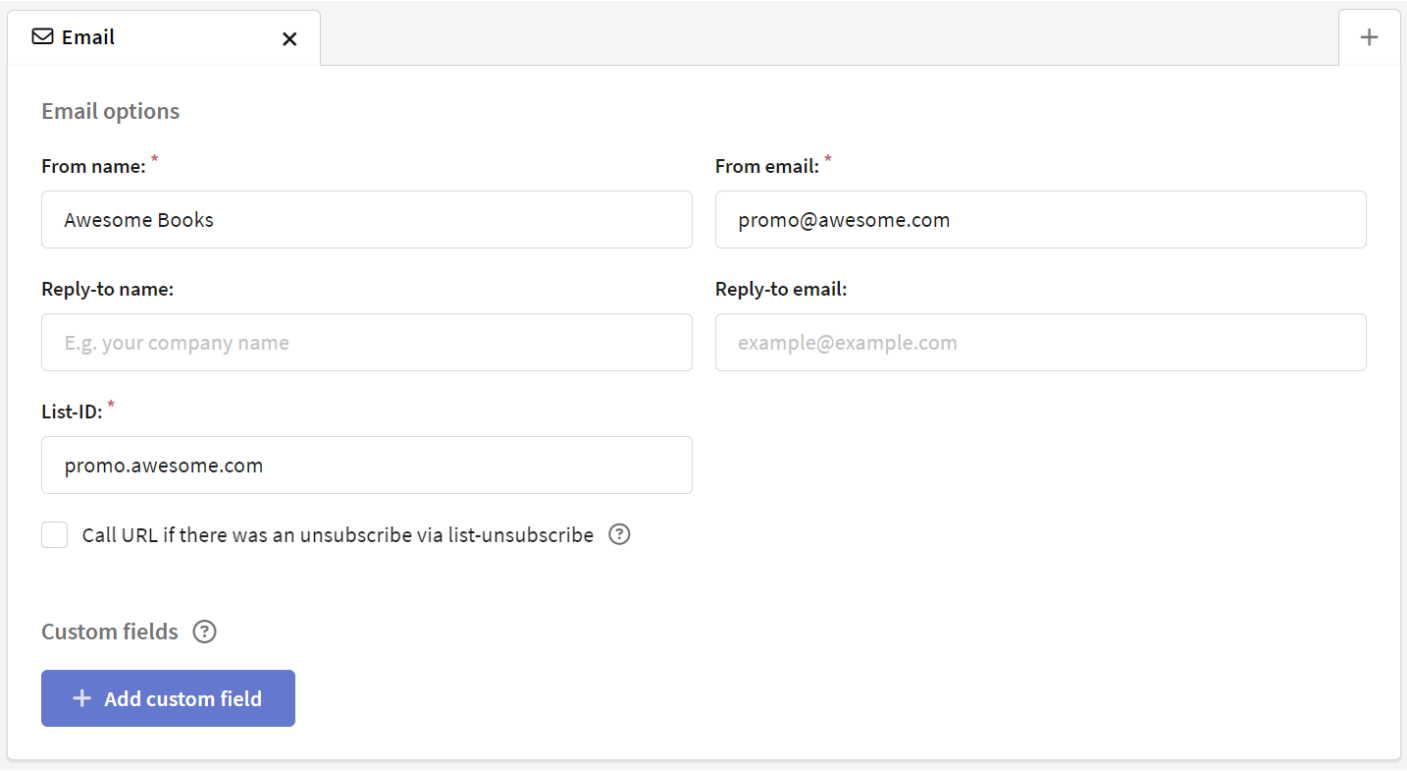
You can add **no more than 8 communication channels** for a single resource!

Each resource can form subscriptions via one or several communication channels. 7 channels are available out of the box: Email, SMS, Push, Telegram, WhatsApp*, MAX, Viber and Notify. Custom channels are preconfigured by platform administrators. To add a channel configuration tab, press **plus** on the right side:



Email

To communicate with customers by email, enter **From name**, **From email**, **Reply-to name** and **Reply-to email** - in case you will gather replies to your mailings:



You also need to fill in the **List-ID** field. List-ID is an additional header included in a message that transmits a unique mailing list identifier. Email clients use information from the header for filtering and sorting incoming messages. Details of the List-ID standard can be found here.

Enter the List-Id in the format `[list-label].[domain]`. Examples:

- `list-header.nisto.com`
- `commonspace-users.list-id.within.com`
- `0Jks9449.list-id.cmu.edu`

List-ID is specified without `<` and `>` symbols. The length of the identifier should not exceed 255 characters. The domain should match the sender's domain.

INFO

It is recommended to use different List-IDs for each resource.

Unsubscribe settings in the Email channel

The platform provides two mechanisms for unsubscribing from the Email channel in the resource:

1. **Unsubscribe link in the List-Unsubscribe** is transmitted in the header of the letter for use by mail services.
2. **Unsubscribe link in the body of the letter** is added to the template using the variable `{unsubscribe}`.

The following settings are available to manage unsubscribe via **List-unsubscribe**:

- **Don't use list-unsubscribe in email** — unsubscribe list-unsubscribe link is not added to the header parameter.
- **Call URL if there was an unsubscribe via list-unsubscribe** — the required field will be displayed **Called URL**. It should specify the URL that the user will go to after unsubscribing via the unsubscribe link of the mail service. If the URL is omitted, the user will be redirected to the system unsubscribe page.

CAUTION

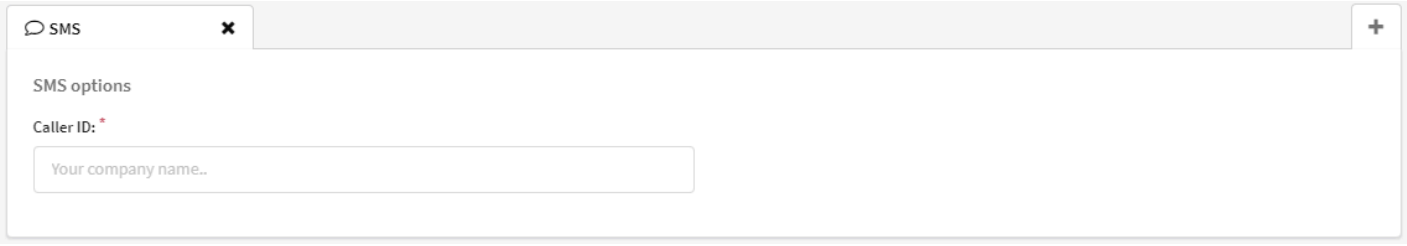
Unsubscribe option **Don't use list-unsubscribe in email** is available only for users of the On-premises solution. To activate this option, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the configuration file `main.json`.

TIP

To communicate with subscribers via the email channel, you need to use email addresses formatted as john@example.com. Email addresses must comply with RFC standards. You can find the main requirements at this link.

SMS

To communicate with short telecom messages, activate the **SMS channel** in the resource and enter **operator name (Caller ID)**. Generally,it needs to be agreed on with your SMS provider, unless you use your own telecom gateway.



TIP

Communication with subscribers through the SMS channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`. Learn more.

Push

CAUTION

If a custom unsubscribe link is specified in the email template, then it will be a priority. Read more about user links in the article.

Push channel combines settings for all supported platforms:

Web browsers: Google Chrome, Mozilla Firefox, Apple Safari and Opera browser

Mobile applications: Android and iOS.

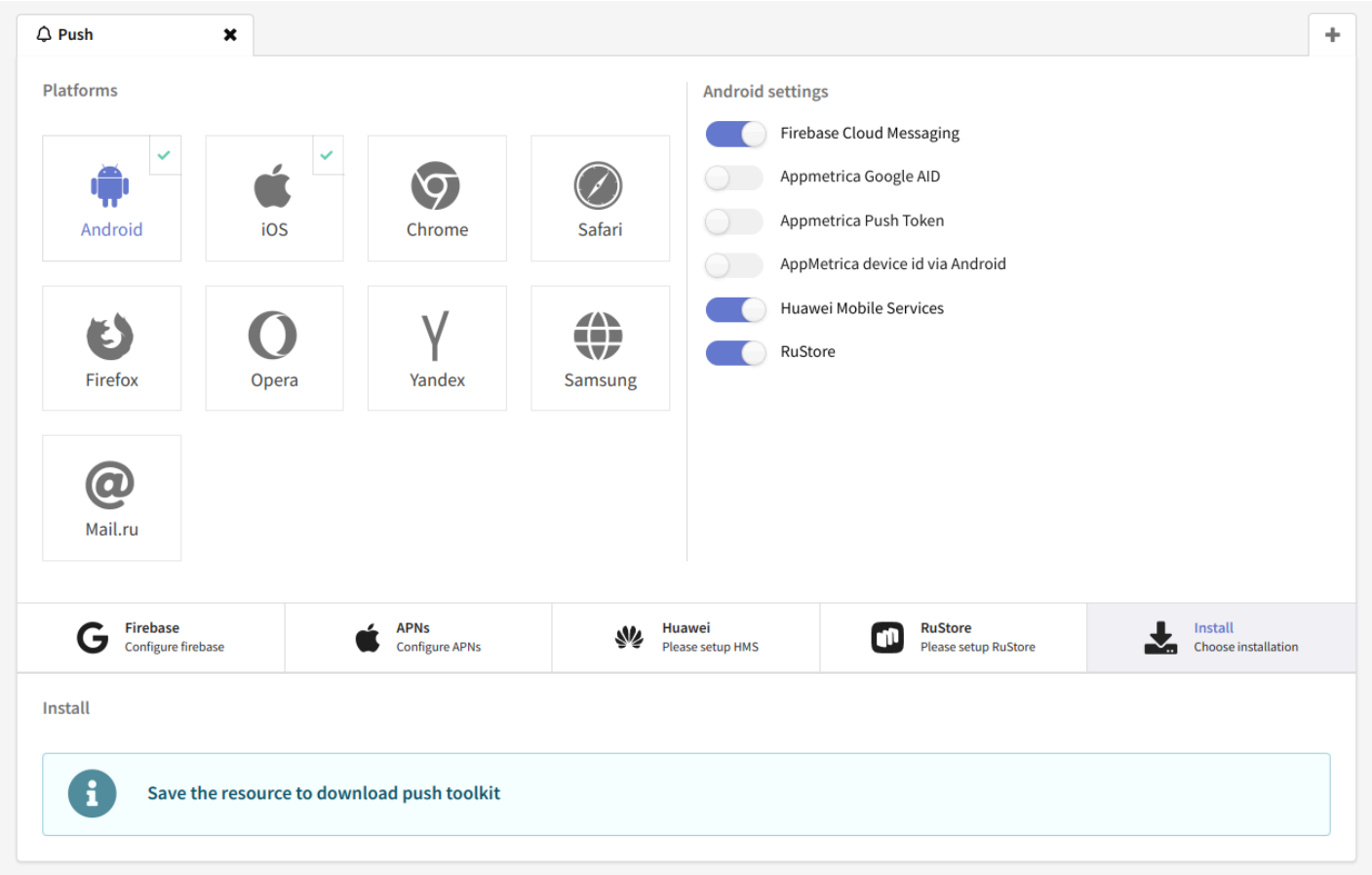
The following providers are available out of the box:

- **Google Firebase Cloud Messaging** — all platforms, except Safari
- **Apple Push Notification Service** — Safari & iOS
- **Mozilla Services** — Mozilla Firefox
- **Yandex AppMetrica** (AppMetrica integration) — Android & iOS with Google FCM SDK.
- **Huawei Mobile Services** — Android & iOS
- **RuStore** — only Android

To work with push notifications, activate the **Push channel** in resource and configure the necessary platforms and providers.

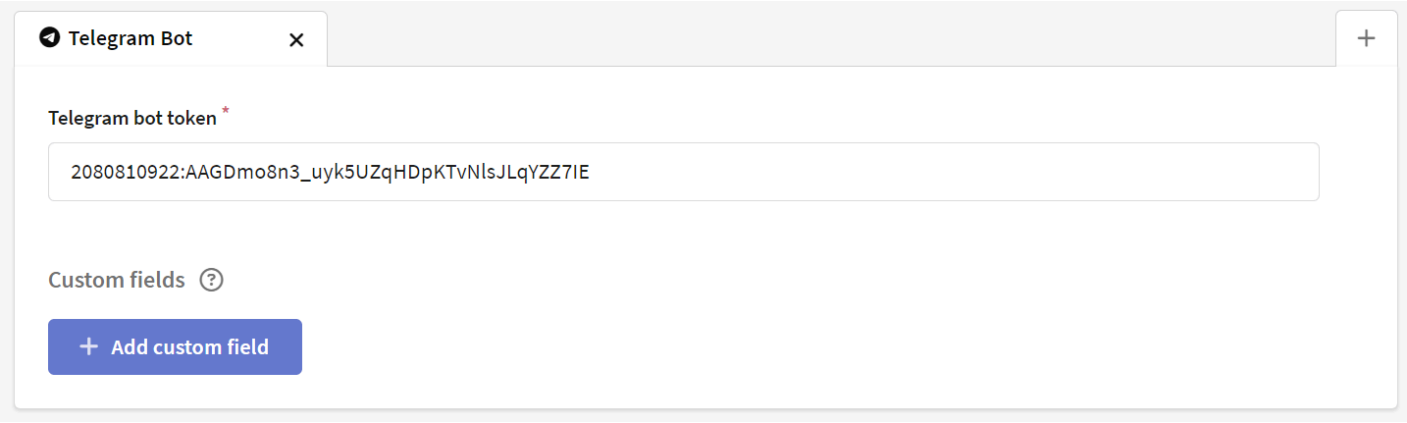
TIP

Configuration options for the supported platforms are described in detail in this article.



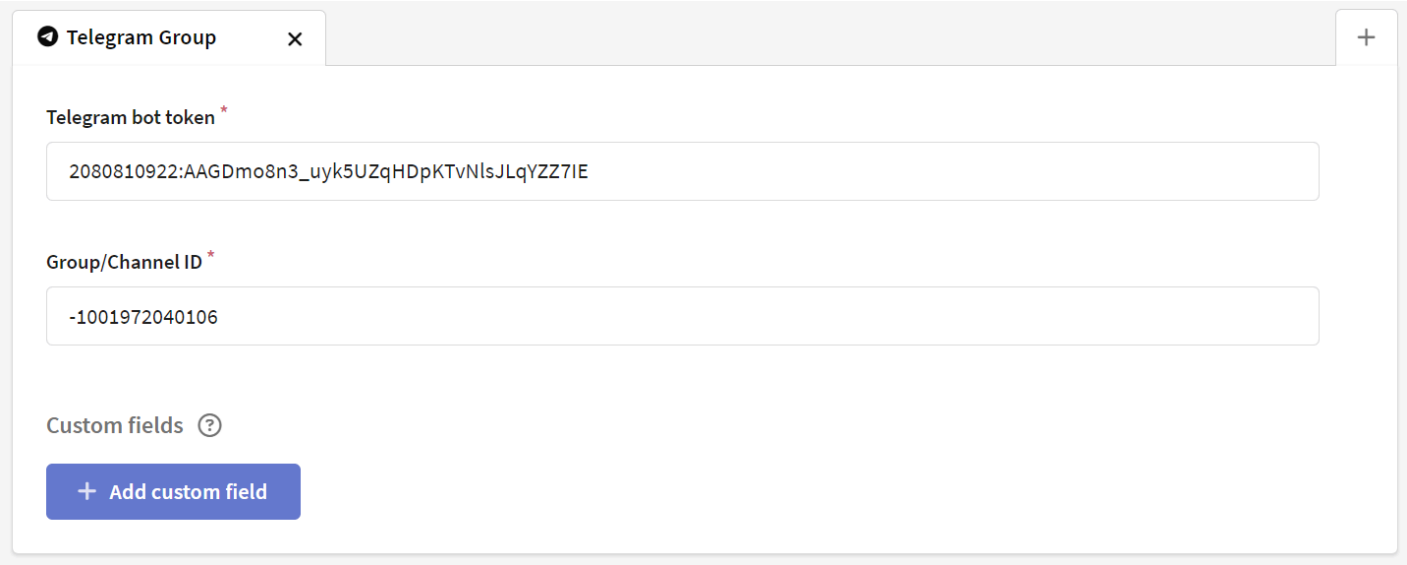
Telegram Bot

Use this channel to send messages to subscribers via the Telegram bot. You need to enter the bot token. Learn how to get a token in this guide.



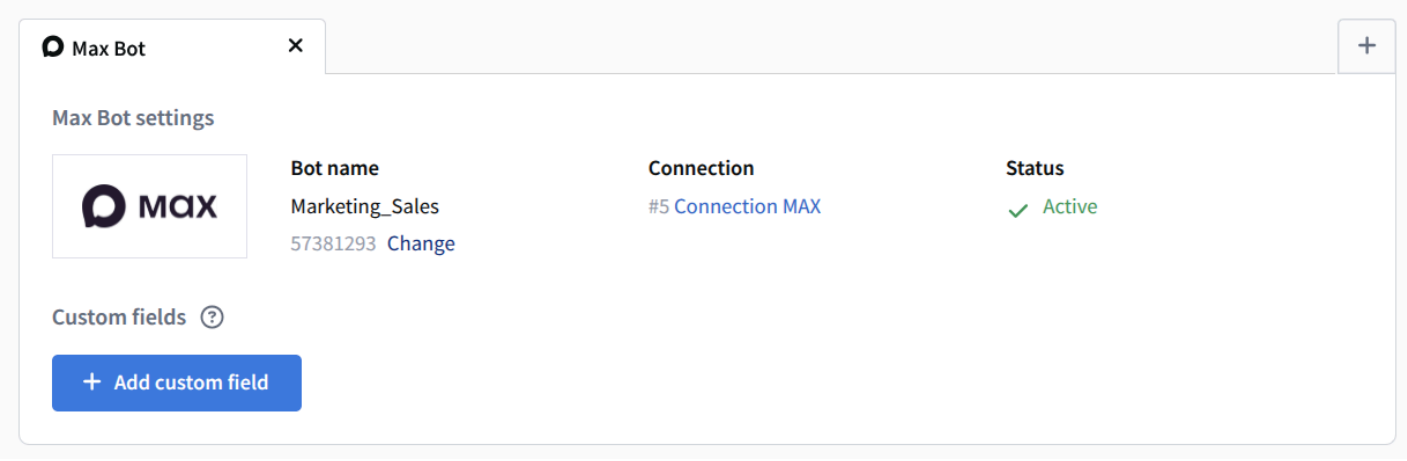
Telegram Group

You can send messages to this channel using placement mailings. Specify the bot token and group or channel ID. More information about channel setup can be found right there.



MAX Bot channel

Use this channel to send personalized messages to subscribers via a bot in the MAX messenger. To create Campaigns in this channel, you need to create a bot connection and configure the MAX integration. In the resource settings, add the MAX Bot channel and select the bot linked to the integration:

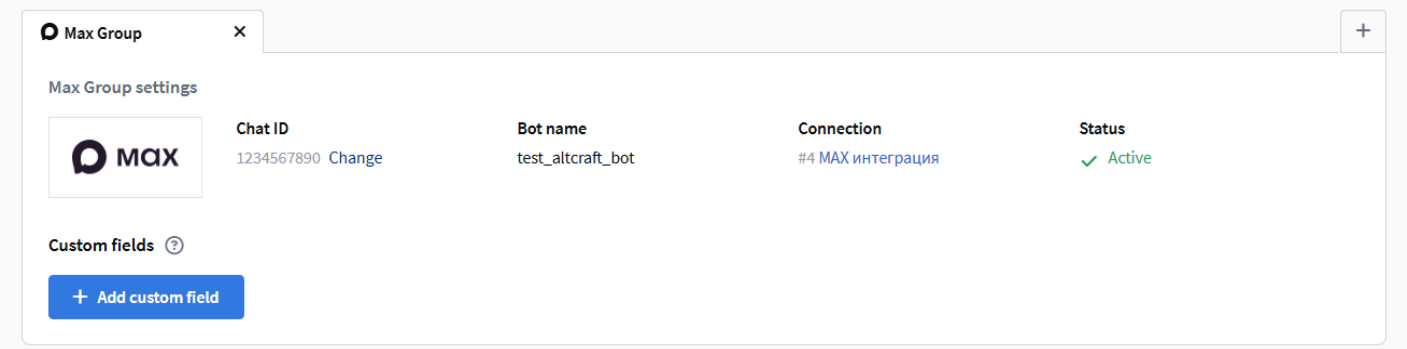


A full step-by-step guide is available on this page.

MAX Group channel

Use this channel to send messages to MAX chats using placements. This is an **indirect communication channel**, meaning Campaigns in this channel cannot be personalized using profile data variables.

To create Campaigns in this channel, you need to create a bot connection and configure the MAX integration. In the resource settings, add the MAX Group channel and select the chat linked to the integration:

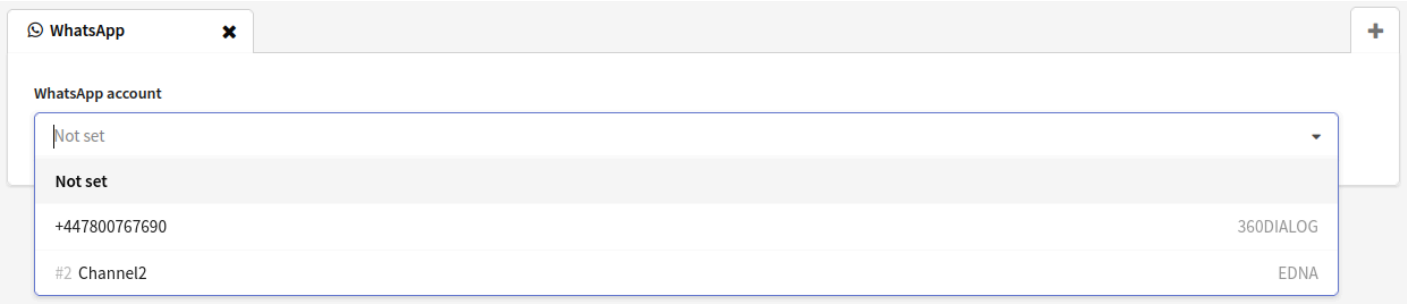


A full step-by-step guide is available on this page.

WhatsApp

To send messages to subscribers in the messaging app, you need to set up integration with the WhatsApp* Business API. Integration can be done through such providers as edna or 360dialog.

After setting up the integration, select the connected account (360dialog) or channel (Edna) in the drop-down list:



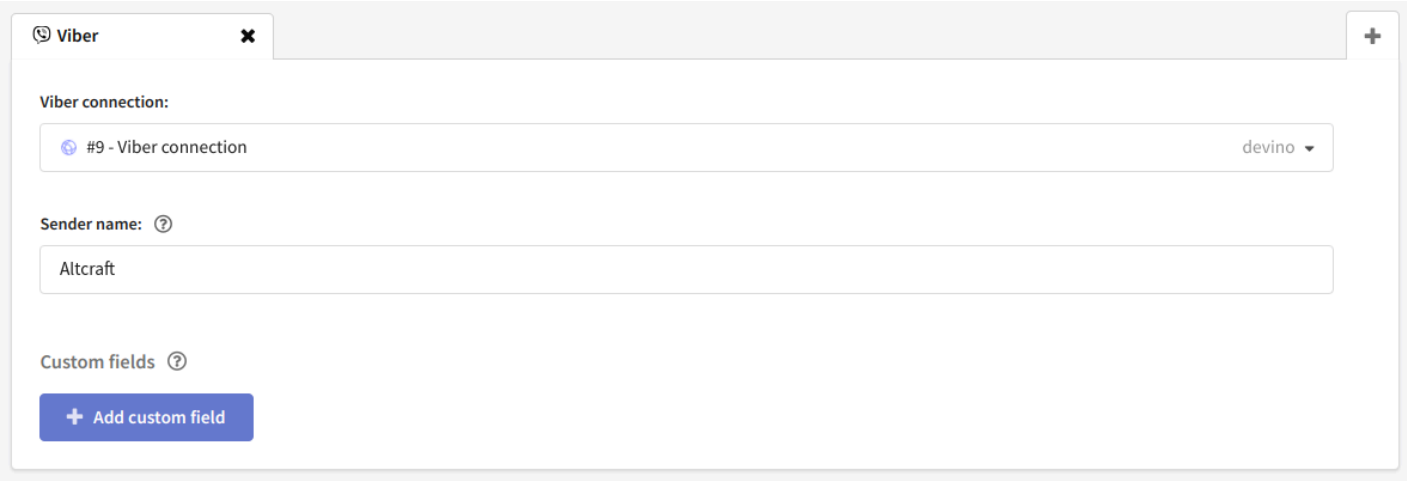
TIP

Communication with subscribers through the WhatsApp* channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`.

Viber

To send messages on Viber, you need to set up an integration through the provider, such as Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto.

In the channel settings, select the Viber connection with Devino Telecom, SMSTraffic, MTS OmniChannel or Rapporto you created earlier and specify the **sender's name**. This name will be visible to subscribers when they receive a message. You can find the sender's name in your provider dashboard, and it should be verified by your provider.



INFO

To set up callbacks with message statuses, please contact your provider's account manager or technical support (Devino Telecom tech support, SMStraffic tech support, MTS OmniChannel, Rapporto).

TIP

Communication with subscribers through the Viber channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`.

Notify

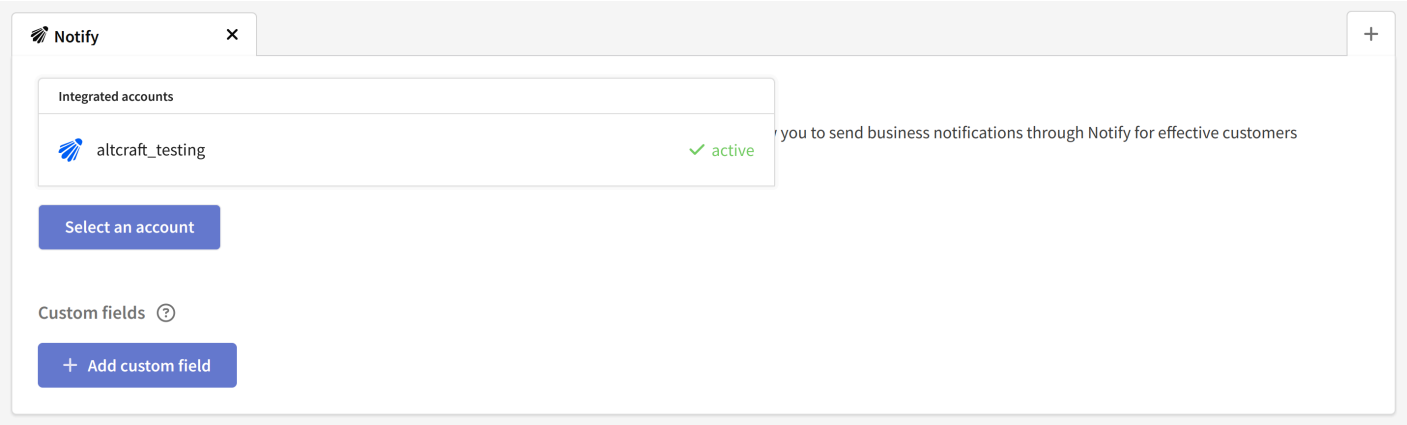
INFO

According to the service policy, new access to the Notify service personal account is currently **not provided**.

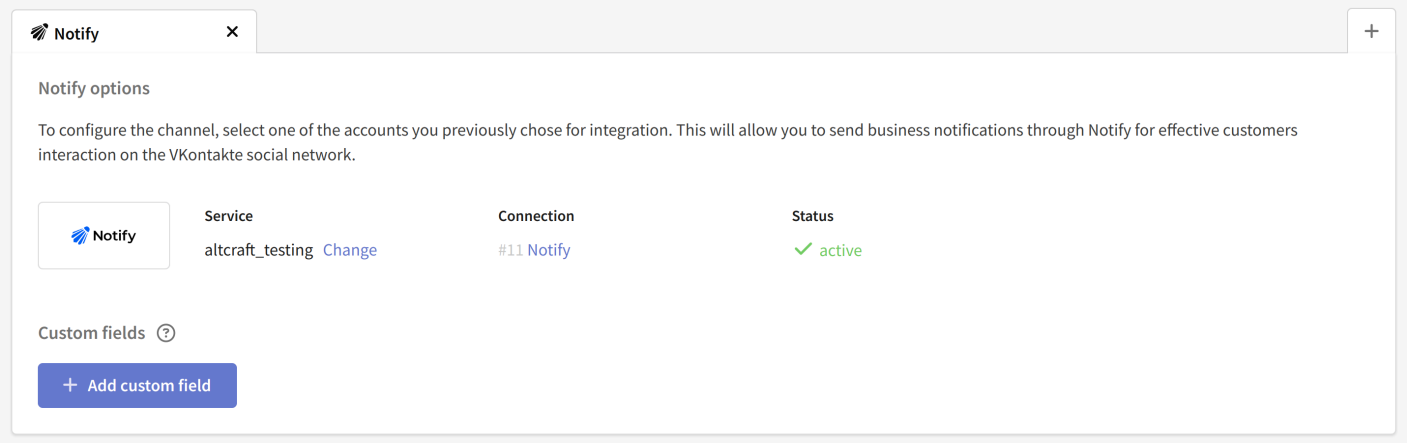
Support for current users is provided by MAX Business tech support.

Notify service allows sending business notifications via VK and OK for effective communication with customers. To use Notify in the platform, you need to pre-configure connection and integration of the service.

In the channel settings, select the previously integrated Notify account:



Check if the account is active and save the resource:

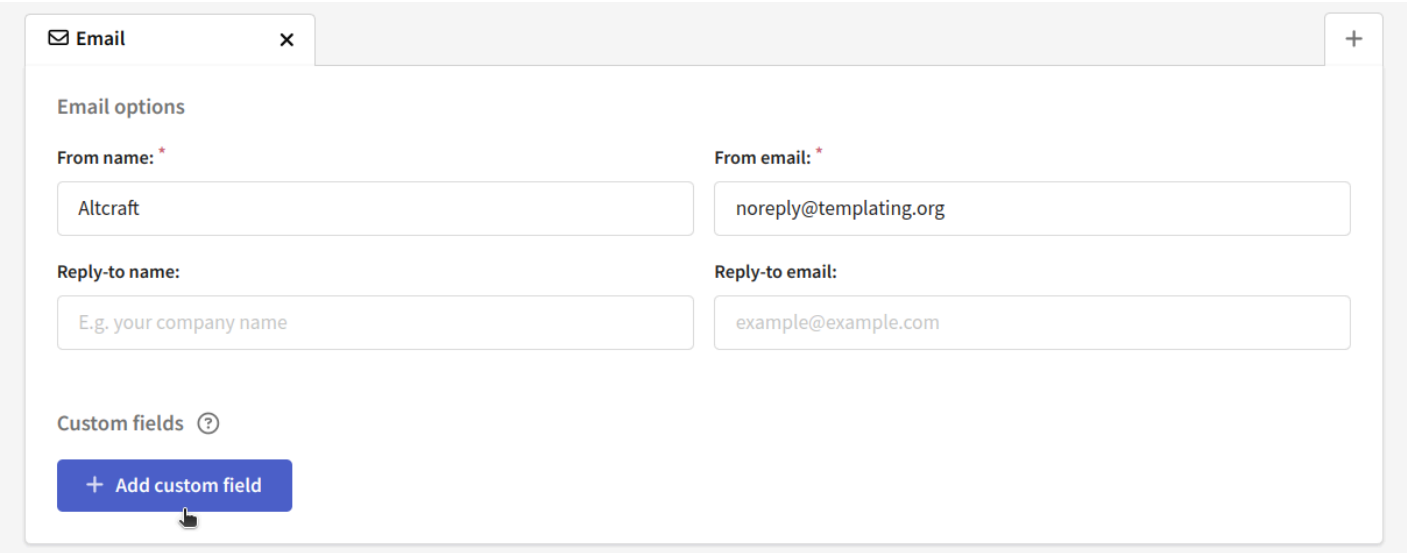


Custom fields in subscription channels

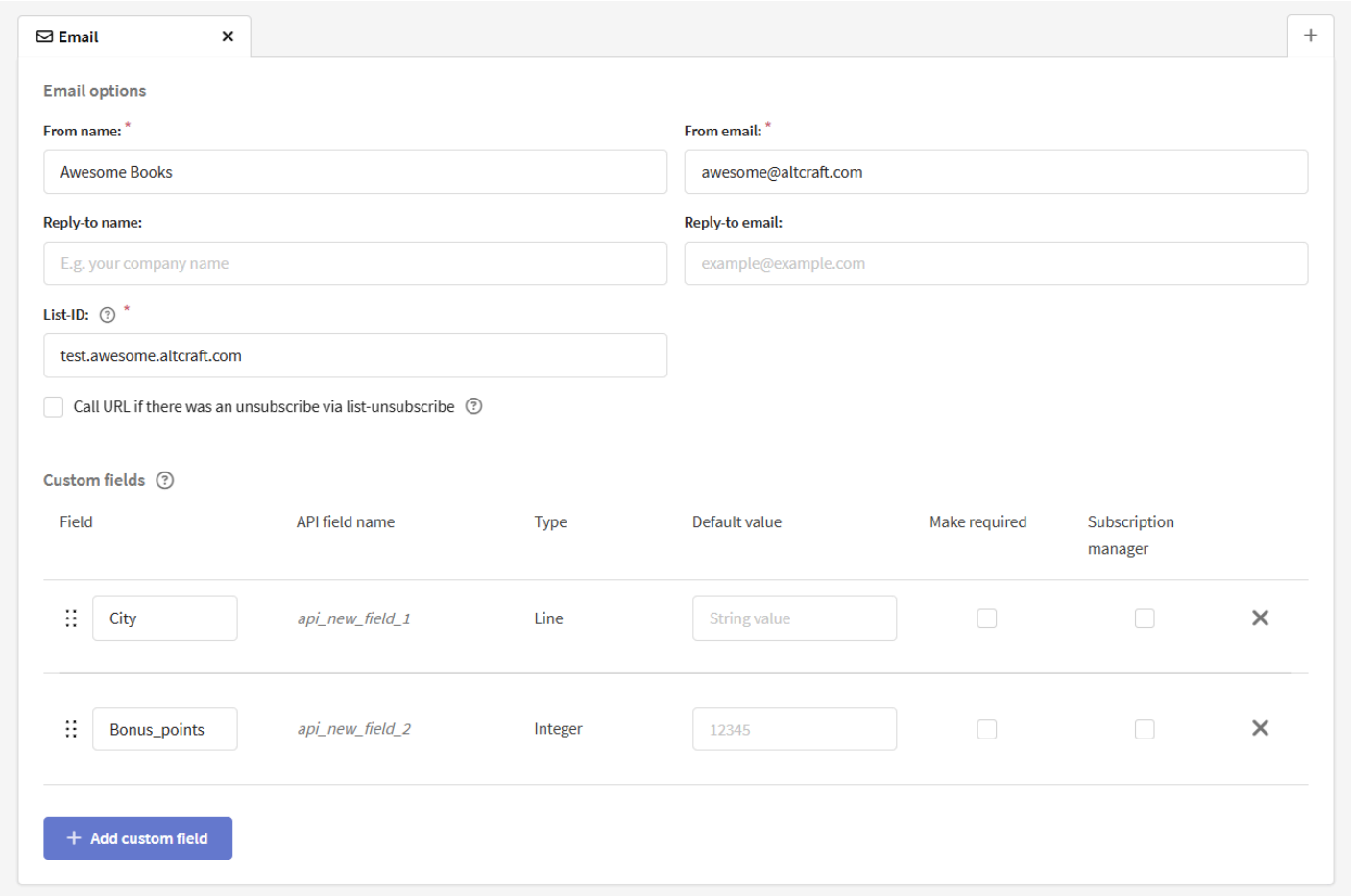
You can add custom fields to any channel resource. They are similar to additional fields in the profile database. Use them to store extra information related to the subscription to the channel.

To create a field, click the **Add custom field** button and select its type:

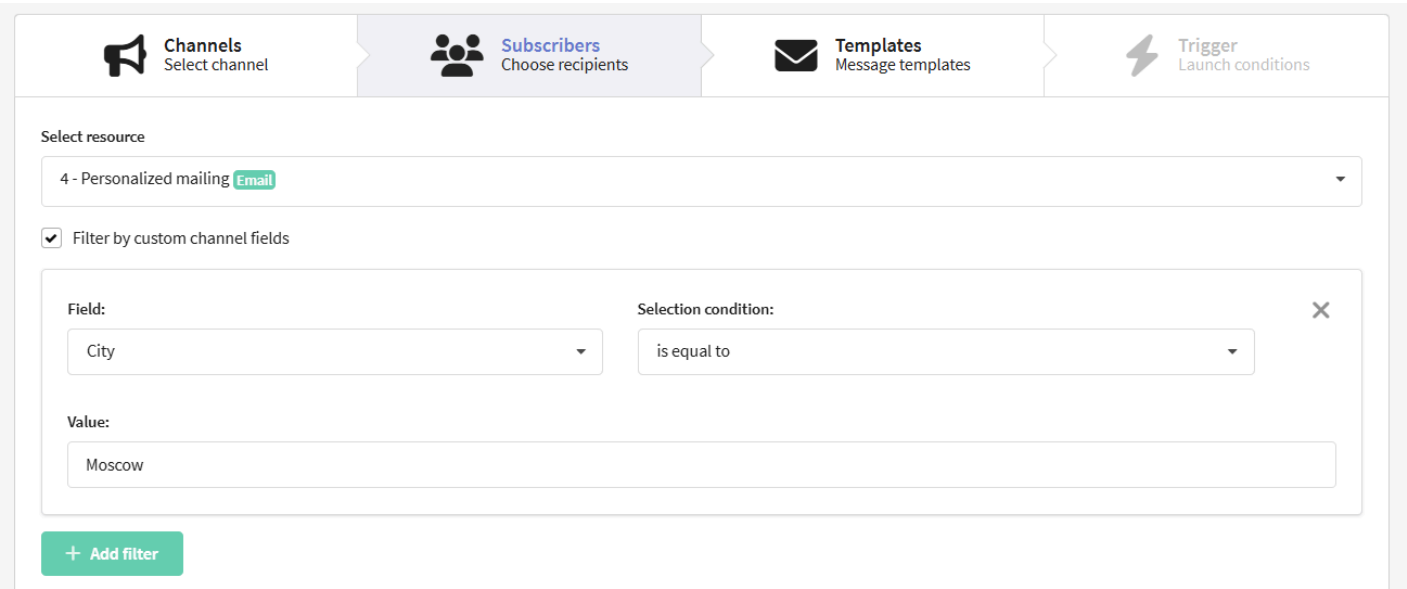
- **Integer** — positive and negative whole numbers.
- **Float** — positive and negative floating-point numbers.
- **Line** — text in any language, including numbers, special characters, and spaces. It uses the Unicode standard.
- **Boolean** — a field that provides a choice between two options: True or False (Yes or No).



Next, specify the **field name** and the **name** to be used in **API requests**. The field name can include any characters, while the API name should only contain Latin characters, numbers, and underscores. It also must begin with a letter.



After creating custom fields you can apply them to your trigger mailing. This will make it more personalized.



Additional settings:

- **Default value** — used when no data for this field is provided during profile import.
- **Make required** — data for this field will be required during import.
- **Subscription manager** — the field will be added on the Subscription Manager page.

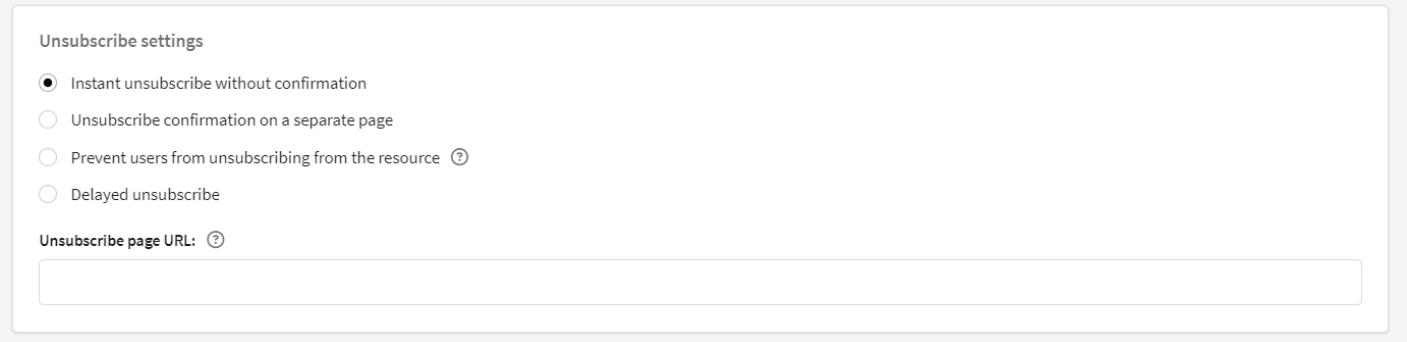
To delete a field, click the cross. The number of additional fields is not limited. After saving the resource, you can edit the field name and its additional settings, but changing the API name is not possible.

Unsubscribe settings

You can set up a separate unsubscribe mechanism for each resource. The unsubscribe page URL specified in the resource will override the system unsubscribe page. However, if a custom unsubscribe link is specified in the email template, then it will take priority. Learn more about user links.

Instant unsubscribe without confirmation

When clicking on the unsubscribe link, the profile goes to the unsubscribe system page and its status immediately changes to **Unsubscribed**. This type of unsubscription is used by default.



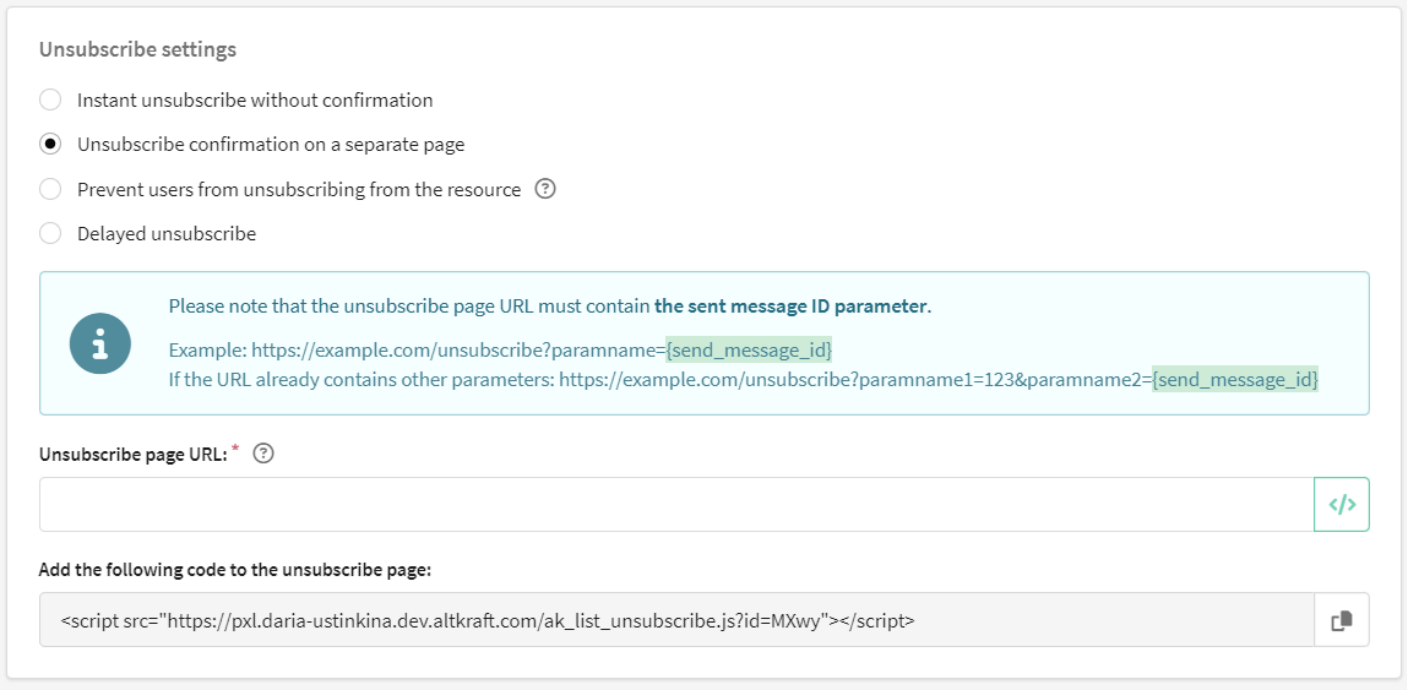
You can redirect the profile to your own unsubscribe page by specifying a different URL in the **Unsubscribe page URL** field.

Unsubscribe confirmation on a separate page

CAUTION

This unsubscribe setting is only available for users of the On-premises solution. To activate these settings, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the 'main' configuration file.json`.

When you click on the unsubscribe link in the message, the profile will go to the user's unsubscribe page. Its status will change to **Unsubscribed** only after he confirms the unsubscription. Such a page can be used to clarify the reason for unsubscribing.



Setting up fields:

Specify **Unsubscribe page URL** — this URL will override the URL of the system unsubscribe page.

In the URL of the unsubscribe page, add the parameter **ID of the sent message** `{send_message_id}`. This is a required parameter that will allow you to identify the resource and profile for which you want to unsubscribe.

Select the unsubscribe mechanism:

Unsubscribe mechanism using the Altcraft SDK

CAUTION

This mechanism is implemented **on the client side** (in the browser). It is safe to use on the unsubscribe page accessible to the user. All operations are performed via the SDK, without exposing tokens or other sensitive data.

Paste the code provided in the resource to your unsubscribe page to connect the Altcraft SDK. Next, initialize the `AkUnsub` class.

Example:

```
<!-- Connecting the Altcraft SDK -->
 <script src="https://pxl.altkraft.com/ak_list_unsubscribe.js?id=MXw3Mw=="></script>

<!-- Initializing the AkUnsub class -->
<script>
   const akUnsub = new AkUnsub();
</script>
```

INFO

If necessary, you can redefine the resource token and pixel domain in the script:

```
new AkUnsub({
    rtoken: 'example_token',
    pxl_domain: 'your_pxl_domain',
 })
```

The AkUnsub class has 2 public methods:

- `confirmUnsub` — method for confirming unsubscription
- `cancelUnsub` — method for canceling unsubscription. Used only **for delayed unsubscribe**, more on this in the next paragraph of the article.

Each of the methods accepts the following **arguments:**

- **SMID** is the identifier of the sent message (required parameter).
- **An array of categories** of the resource from which unsubscribing occurs (optional parameter).

Example:

```
AkUnsub.confirmUnsub('SMID', ['cat_to_unsub'])
AkUnsub.cancelUnsub ('SMID', ['cat_to_unsub'])
```

To call methods, you can add buttons to the unsubscribe page. Example:

```
<!-- Unsubscribe confirmation button -->
<button onclick="akUnsub.confirmUnsub('SMID',['category_1', 'category_2'])">Confirm unsubscribe</button>

<!-- Unsubscribe button -->
<button onclick="akUnsub.cancelUnsub('SMID',['category_1', 'category_2'])">Cancel unsubscribe</button>
```

Replace `SMID` and `['category1', 'category2']` with the corresponding values.

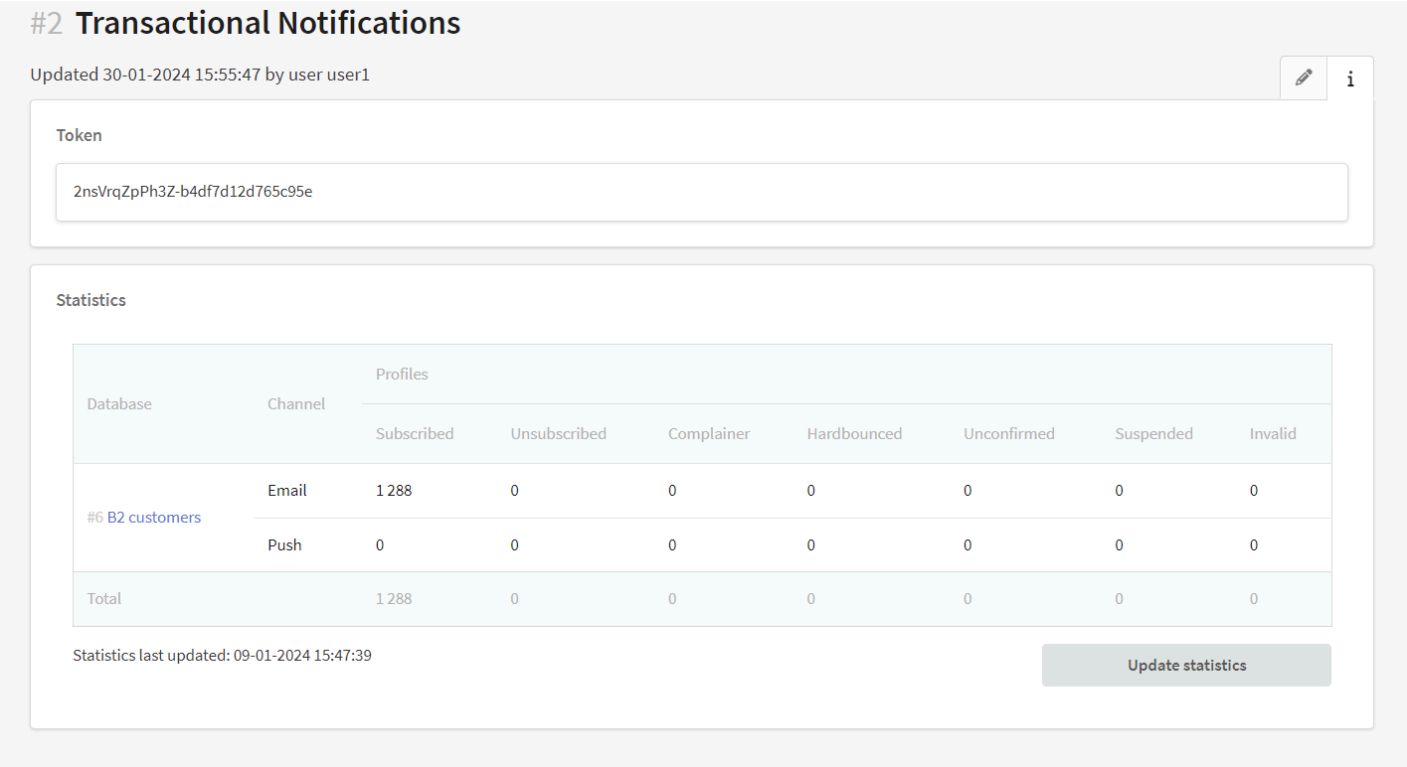
Unsubscribe mechanism without using the Altcraft SDK

CAUTION

This mechanism is implemented **on the server side**. It uses an API token and a resource token, which are confidential. They must not be included in page code or sent directly from the browser.

You can implement the unsubscribe mechanism on your side. To do this, add 2 parameters to the **Unsubscribe page URL**:

- **Send message ID** `{send_message_id}`. This is a mandatory parameter that allows you to identify the resource from which you want to unsubscribe.
- **Resource token** `{resource_token}`. This parameter is required for additional protection and to implement your own unsubscribe mechanism without using the Altcraft SDK. The resource token can be passed not in the URL of the unsubscribe page, but in the body of the API request. The token can be obtained in the resource editing window on the “i” tab — details:



If the profile has confirmed the unsubscription on your page by its actions, then it is necessary to transfer this data to the platform using the API request `/api/v1.1/profiles/message_unsub`. The status of the profile's subscription to the resource will change to "Unsubscribed". If you pass the `categories` parameter, the profile will be unsubscribed only from the specified resource categories.

Request Example:

```
{
"token": "abcdefghijklmnqrstuvwxyz", // API token
  "smid": "fGmXiWU9Y4NSIpCwl4QG6QXAAAQE", // ID of the sent message
  "rtoken":"2fHLS9YR9daS-a90b371eecc2952f", // Resource token
  "categories": ["cat_to_unsub"] // Resource categories to unsubscribe from
}
```

TIP

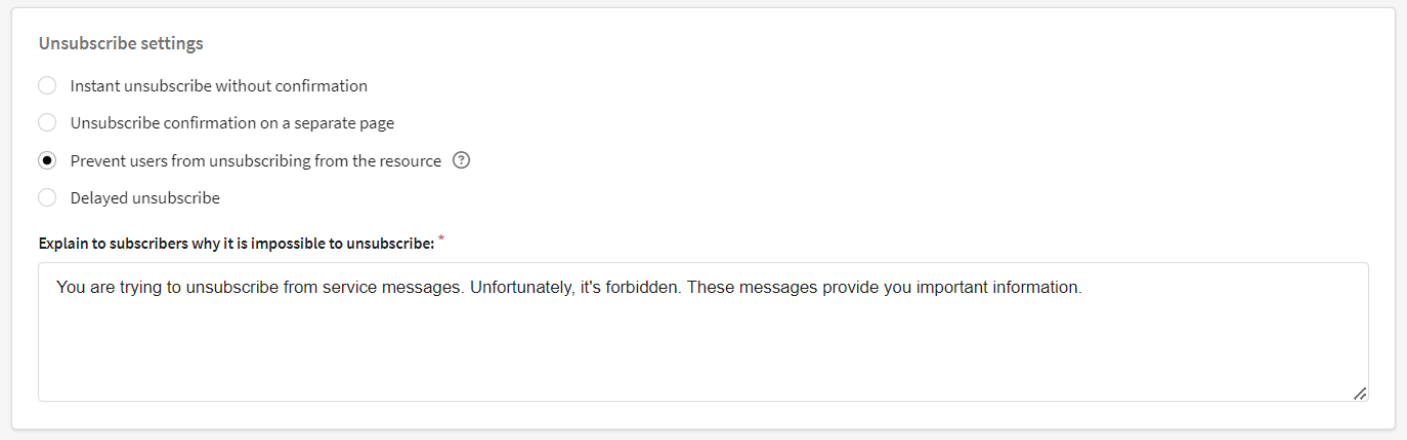
API request **/api/v1.1/profiles/message_unsub** is described in **documentation**.

Prevent users from unsubscribing from the resource

CAUTION

This unsubscribe setting is only available for users of the On-premises solution. To activate these settings, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the 'main' configuration file.json`.

For certain types of mailings, you can set a limit on unsubscribing. When this option is selected, all unsubscribe links specified in the template will lead to a page where the reason for not unsubscribing is displayed.



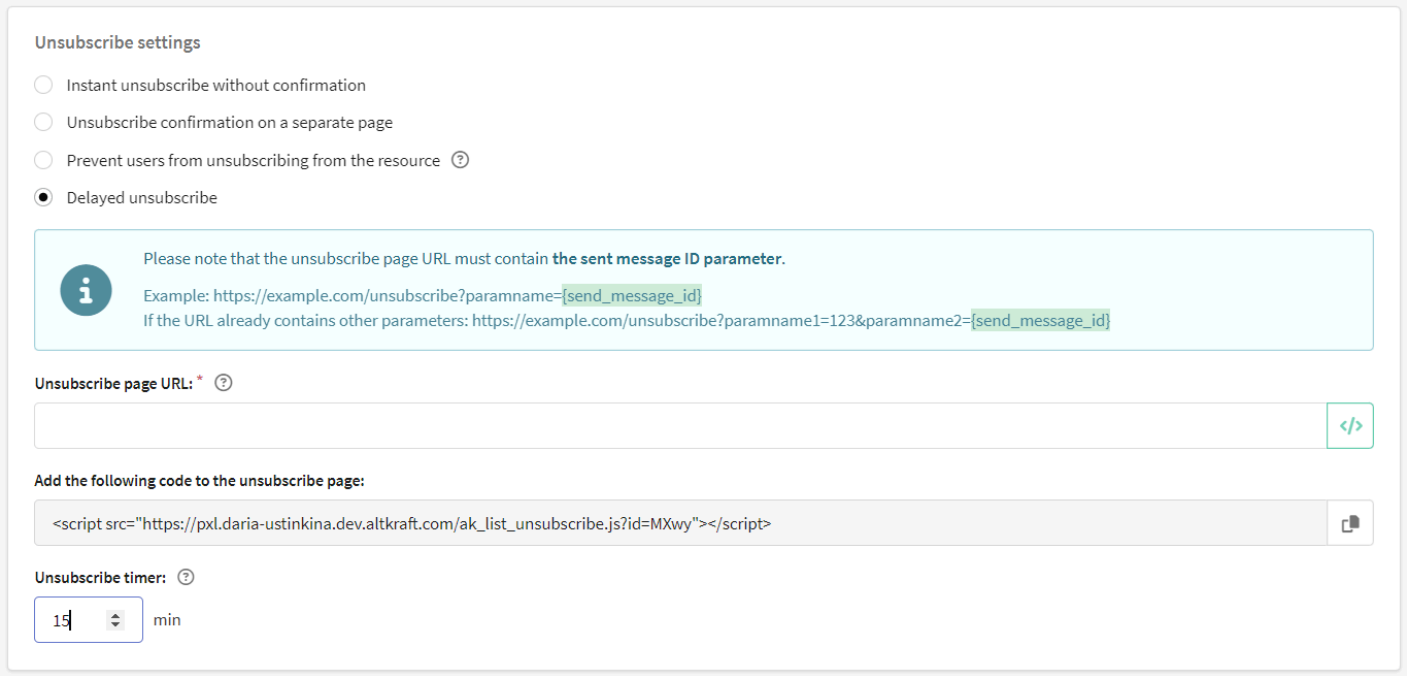
INFO

Unsubscribing via the list-unsubscribe link in an Email will also be prohibited.

Delayed unsubscribe

When you click on the unsubscribe link in the message, the profile will go to the user's unsubscribe page. Its status will change to "Unsubscribed" if:

- the profile will confirm the unsubscription by its actions;
- The profile will not cancel the unsubscribe during the time specified in the Timer field (the maximum timer value is 30 minutes).



Setting up fields:

Specify **Unsubscribe page URL** — this URL will override the URL of the system unsubscribe page.

In the URL of the unsubscribe page, add the parameter **ID of the sent message** `{send_message_id}`. This is a required parameter that will allow you to identify the resource and profile for which you want to unsubscribe.

Select the unsubscribe mechanism:

Unsubscribe mechanism using the Altcraft SDK

CAUTION

This mechanism is implemented **on the client side** (in the browser). It is safe to use on the unsubscribe page accessible to the user. All operations are performed via the SDK, without exposing tokens or other sensitive data.

Paste the code provided in the resource to your unsubscribe page to connect the Altcraft SDK. Next, initialize the `AkUnsub` class.

Example:

```
<!-- Connecting the Altcraft SDK -->
 <script src="https://pxl.altkraft.com/ak_list_unsubscribe.js?id=MXw3Mw=="></script>

<!-- Initializing the AkUnsub class -->
<script>
   const akUnsub = new AkUnsub();
</script>
```

INFO

If necessary, you can redefine the resource token and pixel domain in the script:

```
new AkUnsub({
    rtoken: 'example_token',
    pxl_domain: 'your_pxl_domain',
 })
```

The AkUnsub class has 2 public methods:

- `confirmUnsub` — method for confirming unsubscription
- `cancelUnsub` — method for canceling unsubscription

Each of the methods accepts the following **arguments:**

- **SMID** is the identifier of the sent message (required parameter).
- **An array of categories** of the resource from which unsubscribing occurs (optional parameter).

Example:

```
AkUnsub.confirmUnsub('SMID', ['cat_to_unsub'])
AkUnsub.cancelUnsub ('SMID', ['cat_to_unsub'])
```

To call methods, you can add buttons to the unsubscribe page. Example:

```
<!-- Unsubscribe confirmation button -->
<button onclick="akUnsub.confirmUnsub('SMID',['category_1', 'category_2'])">Confirm unsubscribe</button>

<!-- Unsubscribe button -->
<button onclick="akUnsub.cancelUnsub('SMID',['category_1', 'category_2'])">Cancel unsubscribe</button>
```

Replace `SMID` and `['category1', 'category2']` with the corresponding values.

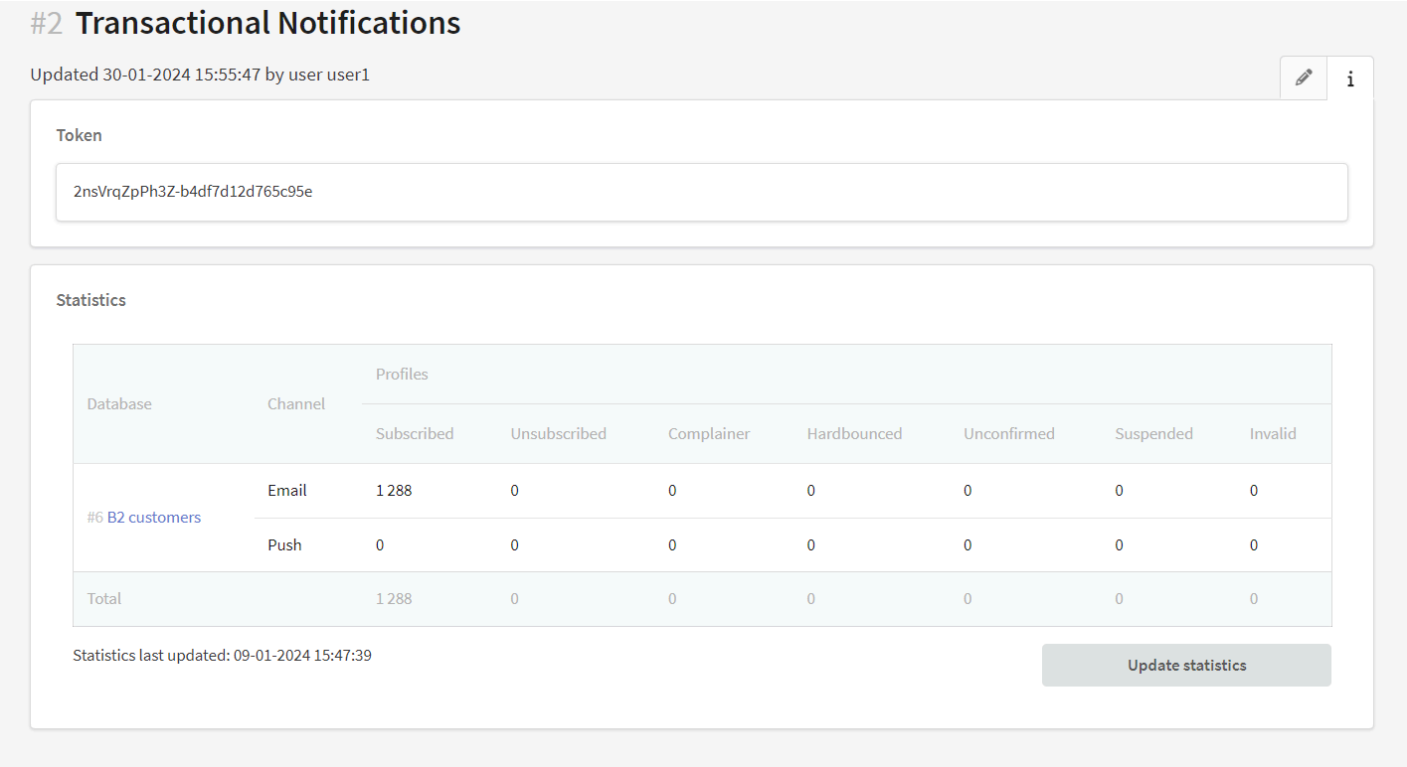
Unsubscribe mechanism without using the Altcraft SDK

CAUTION

This mechanism is implemented **on the server side**. It uses an API token and a resource token, which are confidential. They must not be included in page code or sent directly from the browser.

You can implement the unsubscribe mechanism on your side. To do this, add 2 parameters to the **Unsubscribe page URL**:

- **Send message ID** `{send_message_id}`. This is a mandatory parameter that allows you to identify the resource from which you want to unsubscribe.
- **Resource token** `{resource_token}`. This parameter is required for additional protection and to implement your own unsubscribe mechanism without using the Altcraft SDK. The resource token can be passed not in the URL of the unsubscribe page, but in the body of the API request. The token can be obtained in the resource editing window on the “i” tab — details:



If the profile has confirmed the unsubscription on your page by its actions, then it is necessary to transfer this data to the platform using the API request `/api/v1.1/profiles/message_unsub`. The status of the profile's subscription to the resource will change to "Unsubscribed". If you pass the `categories` parameter, the profile will be unsubscribed only from the specified resource categories.

Request Example:

```
{
"token": "abcdefghijklmnqrstuvwxyz", // API token
  "smid": "fGmXiWU9Y4NSIpCwl4QG6QXAAAQE", // ID of the sent message
  "rtoken":"2fHLS9YR9daS-a90b371eecc2952f", // Resource token
  "categories": ["cat_to_unsub"] // Resource categories to unsubscribe from
}
```

TIP

API request **/api/v1.1/profiles/message_unsub** is described in **documentation**.

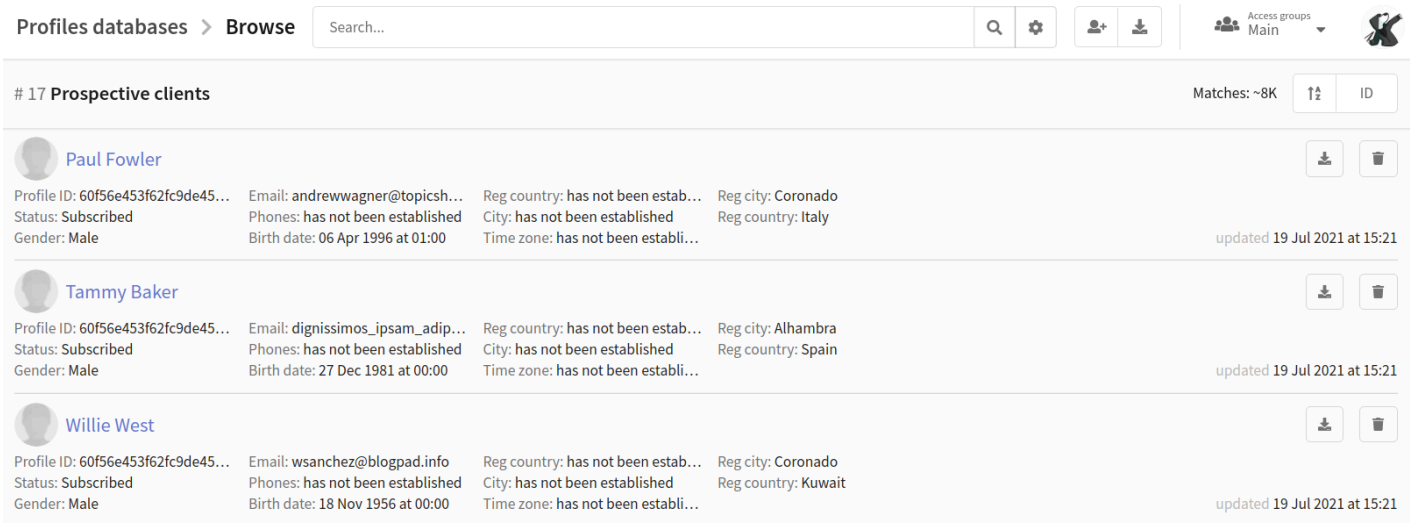
**The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.*

Managing databases

Description

Modern people-based marketing techniques use real time behavior tracking and customers personal information.

Altcraft database management provides you with powerful and flexible tools to collect and make use of customer profiles data.



Flexibility and usefulness of database depends on how it is created, filled, and maintained. If data is correctly collected and organized, then it becomes possible to segment subscribers depending on certain attribute to separate groups and send targeted content to each group.

This process is called Segmentation, and for this we have certain element - segment, which has many intuitive settings.

Think about your marketing for a minute. What goals do you have, who is your audience and what information is interesting for them.
For example:

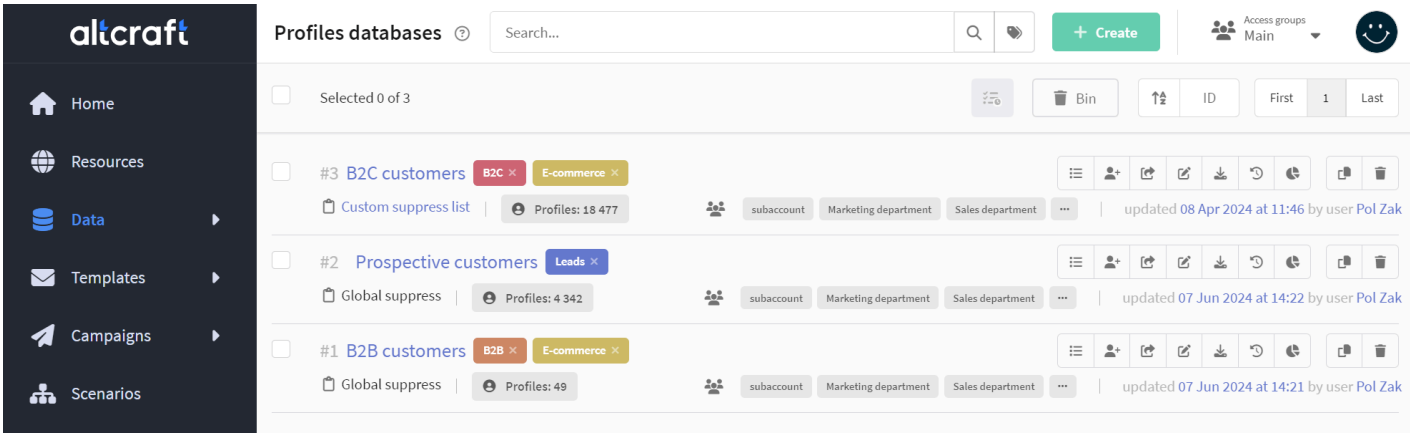
- Do you have your seasonal customers? Do they prefer certain product types?
- Is it necessary to send different content templates depending on customers occupation or hobbies;
- What other parameters affect your customers behaviour?

It can be convenient to figure out what customer information is necessary for you, and write it. You can use it to create custom fields for storing information about your customers in the database.

Thoughtful database setup will bring skyrocketing impact, especially in the long run.

Managing a database

To manage databases, select **Data —> Profiles databases** in the main menu. To create a new database, click **+ Create** in the top panel. To edit an existing database, click on its name.



The following functions are available in the context menu:

— view profiles

— import customer profiles

— copy profilesto another database

— update multiple profiles in the database. Using this instrument, you canupdate profiles, subscribe them to resources, attach or detach promo codes.

— exportcustomer profiles into a file. Beforeexport, you can specify data you need to download.

— export profile history. Download data on profile activity.

— create a new empty database with the same structure. Custom fields will be copied into a new database.

— delete database altogether with profiles.

TIP

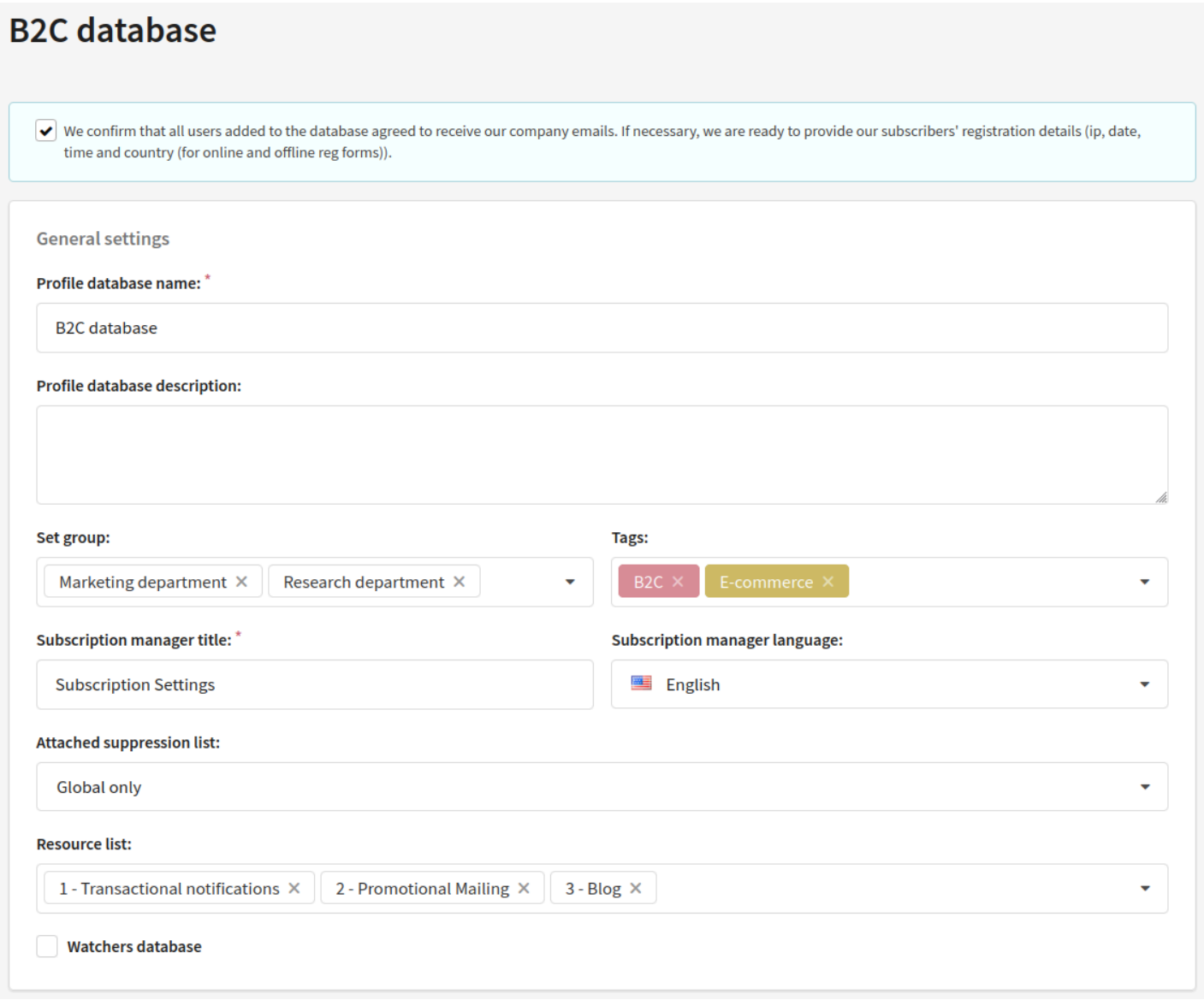
You can delete a database only if it does not contain any profiles that have had interactions within the account. If there have been interactions, such as sending mailings to these profiles, you must first delete those profiles from the database.

Creating a profiles database

General settings

The following settings can be configured for a database:

- **Database name:** name displayed in platform database lists.
- **Description:** database purpose description.
- **Set group:** database access group.
- **Tags:** custom tags that can be used for quick search and analytics.
- **Subscription manager title:** title displayed only in subscription manager.
- **Subscription manager language:** subscription manager page language for database profiles.
- **Attached suppression list:** a custom suppression list for this database.
- **Resources:** enable communication with database profiles through various channels within a specific topic. Instructions on how to configure the resource are provided in this article.
- **Watchers database:** select this option for a database of observers. They will receive messages when a mailing starts together with your customers. Databases with this option can be selected in Smart mailing settings.



Default fields

Standard profile fields are designed to keep the most frequently used customer information.

Default — this value is used when field value is unknown after profile import.

Make required — this field will be required during import. If you enter a Default value, profile import without this field value will be possible.

Subscription manager — this field value will be shown in subscription manager page.

Default fields

Field	Type	Default	Make required	Subscription manager
Email	Line		☑	☑
Phones	Phones list		☐	☑
First name	Line		☐	☐
Last name	Line		☐	☐
Birth date	Date	%Y-%m-%d	☐	☐
Gender	Enum	Not set	☐	☐
Reg date	Date	%Y-%m-%d	☐	☐
Reg IP	IP	127.0.0.1	☐	☐
Reg city	Line		☐	☐
Reg country	Line	Not set	☐	☐
Reg URL	Line		☐	☐
IP	IP	127.0.0.1	☐	☐
City	Line		☐	☐
Country	Line	Not set	☐	☐
Region	Line		☐	☐
ZIP code	Line		☐	☐

INFO

Fields with customer geodata are divided into two types:

- The first (“reg.” In the field name) is the information received during registration: based on the IP address or static import data.
- The second is the latest data received from the client’s actions history: also based on IP address or transmitted data.

Custom fields

Custom fields are created by users of the Altcraft system to store specific for business customer information. You can create unlimited custom fields. They can contain 10 different data types:

- **Number** — positive and negative integer numbers.
- **Float** — floating-point number: it has a decimal place.
- **Line** — text in any language, including numbers, special characters, spaces. Unicode standard is used.
- **Date** — date in RFC 3339 format, like `2019-08-22T00:00:00.000Z`.
- **Boolean** — this field provides a choice of two options: `True` or `False` (Yes or No). It is used to define if customer has certain attribute, like VIP status.
- **Enum** — this field is similar to Boolean, but allows you to choose one option from unlimited number of options created by user. For example, customer tariff plan. Options are written in a comma separated line.
- **IP** — IP in IPv4 format. For example, `127.0.0.1`.

- **Tags** — in this field, several values can be written simultaneously. They may be products or services purchased by customer, or other values, composition of which may vary. Tags are written in line, separated by `"," "|" or ";"`. Therefore, these characters can not be used in tag itself. For example: `Product-1`, `Product 2`, `Product_3`.
- **Object** is a field for storing undefined data: a set of flags or complex data structures.
- **Array of objects** is a field for storing a subtable, which, for instance, may contain a list of managers, contracts, etc.

INFO: Validation in objects and arrays

1. **Naming rules.** In object field names, only Latin letters (in any case), digits from 0 to 9, the underscore `_`, and the hyphen `-` are allowed.

2. **Data type.** You can store data of `int`, `float`, `string`, `boolean` types in the “Object array” or “Object” fields.

Example

```
[
    {
        "key1": 12.3,
        "key2": 12,
        "key3": true,
        "key4": "text"
    }
]
```

3. **Object structure.** If you use such types as “Array of objects” or “Object”, you can only store peer objects or peer objects that are nested within other objects. If you use objects with a more complex structure, connect the SQL database and create queries to select profiles.

Details

Valid values:

```
[
        {"x":1},
        {"y":2}
]

//OR

{
  "key_1":
    {
      "sub_key_1":
        {"x":1,"y":2}
    },
  "key_2":
    {
      "sub_key_1":
        {"x":5,"y":3}
    }
}

//OR

[
    {"key": "value_1", "new_key": "new_value1"},
    {"key": "value_2", "new_key": "new_value2"}
]
```

Invalid values:

```
[
        ["x","y"],
        ["z","w"]
]

//OR

{
  "key":
    [
      ["x","y"],
      ["z","w"]
    ]
}

//OR

[
    {"key1": "value_1", "new_key10": "new_value1"},
    {"key7": "value_2", "new_key300": "new_value2"}
]
```

Custom fields

Field	Type	Default value	Make required	Subscription manager	
Custom_Integer	Integer	123	☑	☐	✖
Custom_Float	Float	123.45	☐	☑	✖
Custom_String	Line	test string	☐	☐	✖
Custom_ISO_date	Date	2022-01-10	☐	☐	✖
Custom_Boolean	Boolean	True	☐	☐	✖
Custom_List	Enum	Option 1, Option 2, Option 3	☐	☐	✖
Custom_IP	IP	127.0.0.1	☐	☐	✖
Custom_Tags	Tags	Tag 1, Tag 2, Tag 3	☐	☐	✖
Custom_Object	Object	{"X":1,"Y":2,"Z":3}	☐	☐	✖
Custom_Array	Objects array	[{"key":"value_1"},{"key":"value_2"}]	☐	☐	✖

+ Add custom field

TIP

You can find detailed information about the types of fields in the database and the allowed values in **this article**.

Email settings

These fields are required for creating a new database:

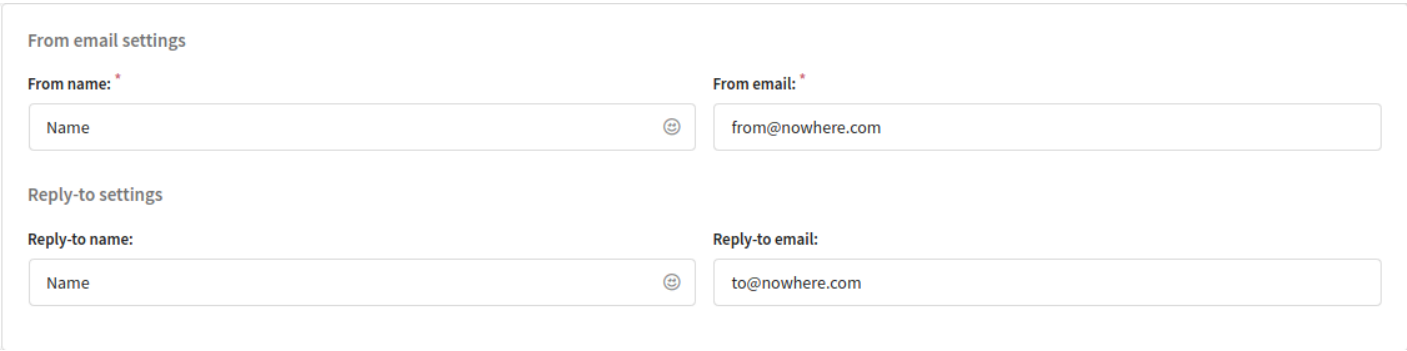
- From name

```
  <label>From email</label>
</strong>
```

And these you will need to gather replies:

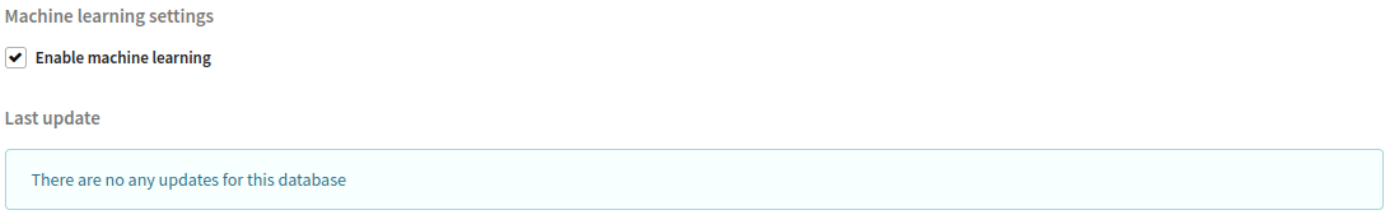
```
  <label>Reply-to name</label>
</strong>
```

```
  <label>Reply-to email</label>
</strong>
```



Machine learning settings

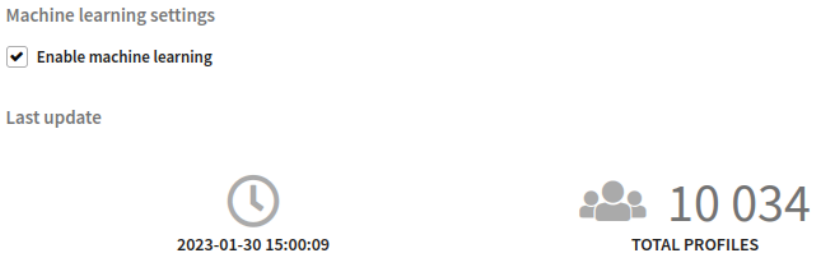
Enabling machine learning in the database allows you to determine the best time to send a message to a profile. This data can then be used to segment profiles and set up scenarios.



After saving the settings, you will see a window with information about when the recalculation of profile data in the machine learning database is scheduled. The first best time data will be displayed after the first calculation.

INFO

After enabling or disabling machine learning, you need to save the database.



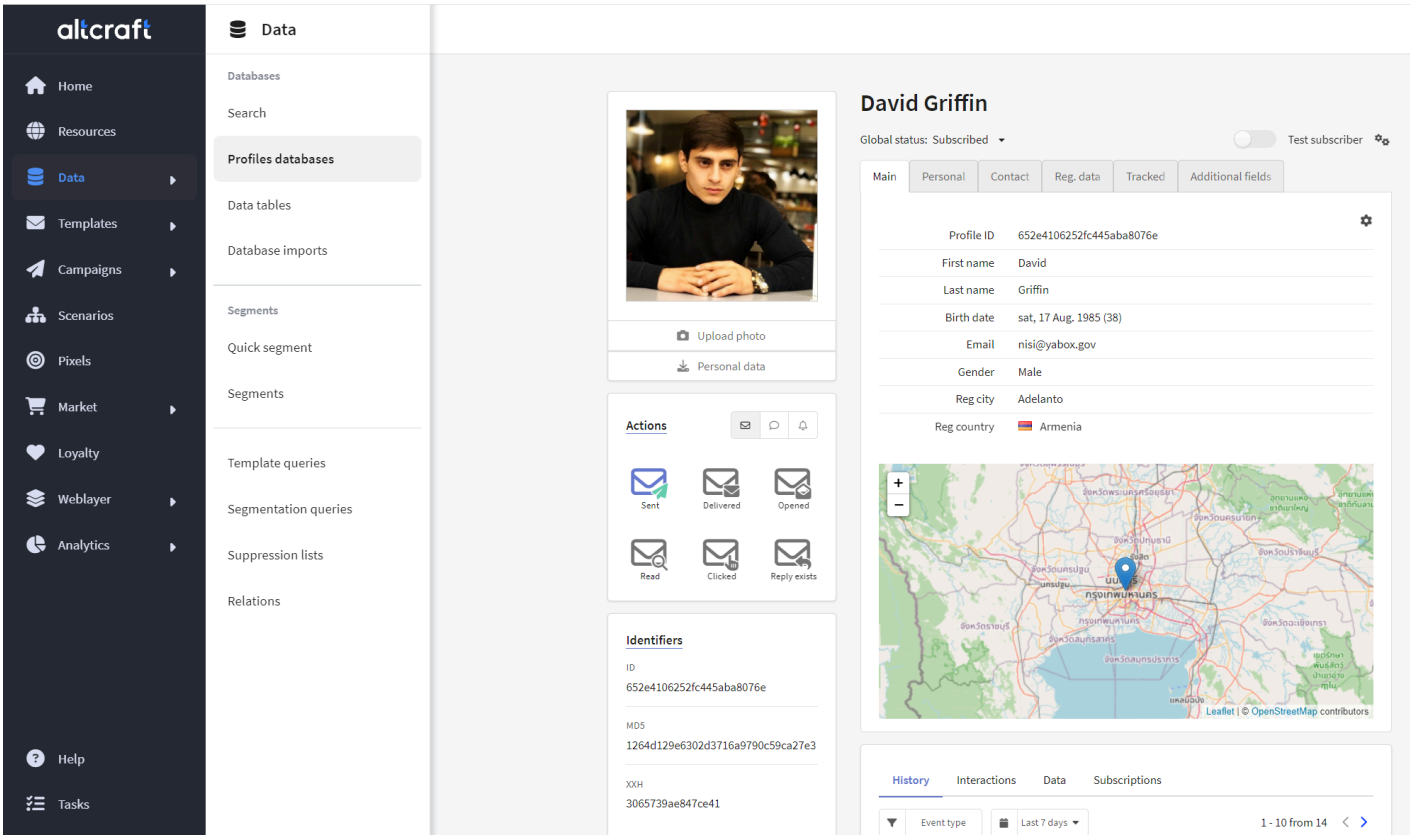
INFO

The schedule for updating machine learning data is set in the configuration file in the `ML_REFRESH_TIME_HOURS` (UTC +0) field. The default is at 00:00.

Subscriber profile

Description

Information about mailing recipients — customers, partners, mailing watchers — is stored in profiles organized in databases. Profile database determines the structure of general and custom data fields of each profile. Resources make communication possible.

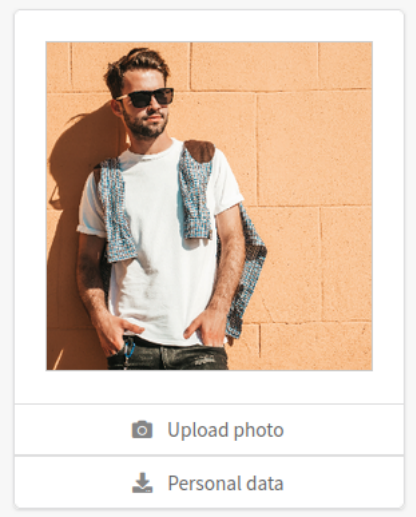


Subscriber profile combines several data types. Static data is displayed on the main page and is organized into several tabs. Other information is displayed below and contains:

- **History — events, connected to this customer: mailing and browsing action history.**
- **Interactions — profile participation in static segments and automation scenarios; you can manually send a profile to a scenario, add to a segment or run a trigger mailing.**
- **Data — information about promo codes, orders and relations of the profile, as well as a list of forms he or she filled out.**
- **Subscriptions — profile's subscriptions to various Resources.**

Avatar and data export

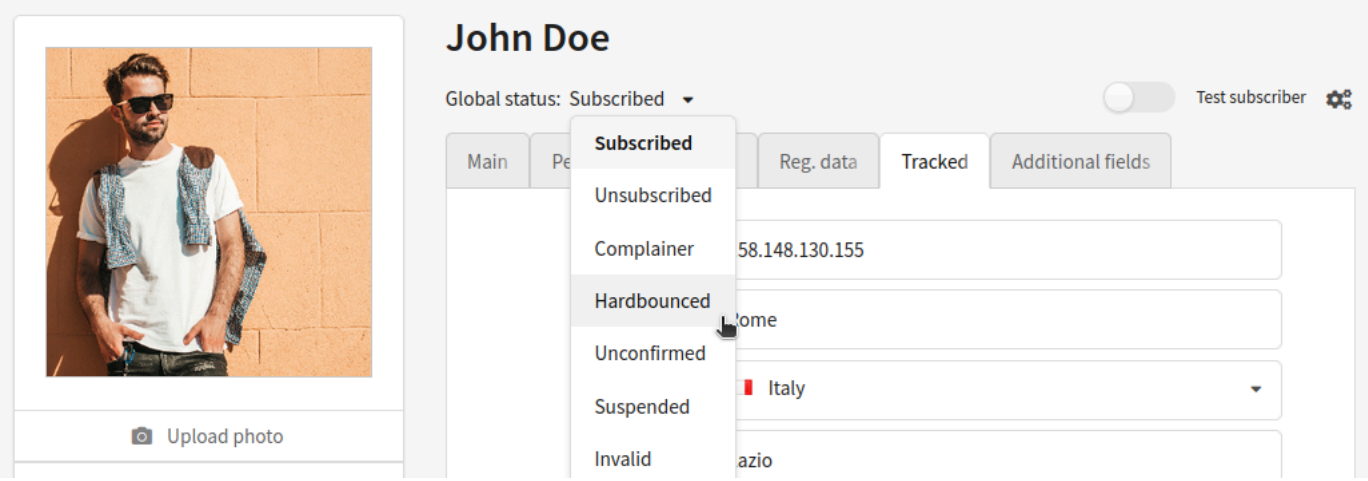
The top left side of the profile page contents your client's avatar and buttons that help you change it manually or export profile's personal data.



Global profile status

Global profile status is defined by the ability to communicate with customers through any communication channel. Individual subscription statuses depend on global status. If the profile has several active subscriptions, but its global status does not allow communication, a message will not be sent. 7 subscription statuses are available:

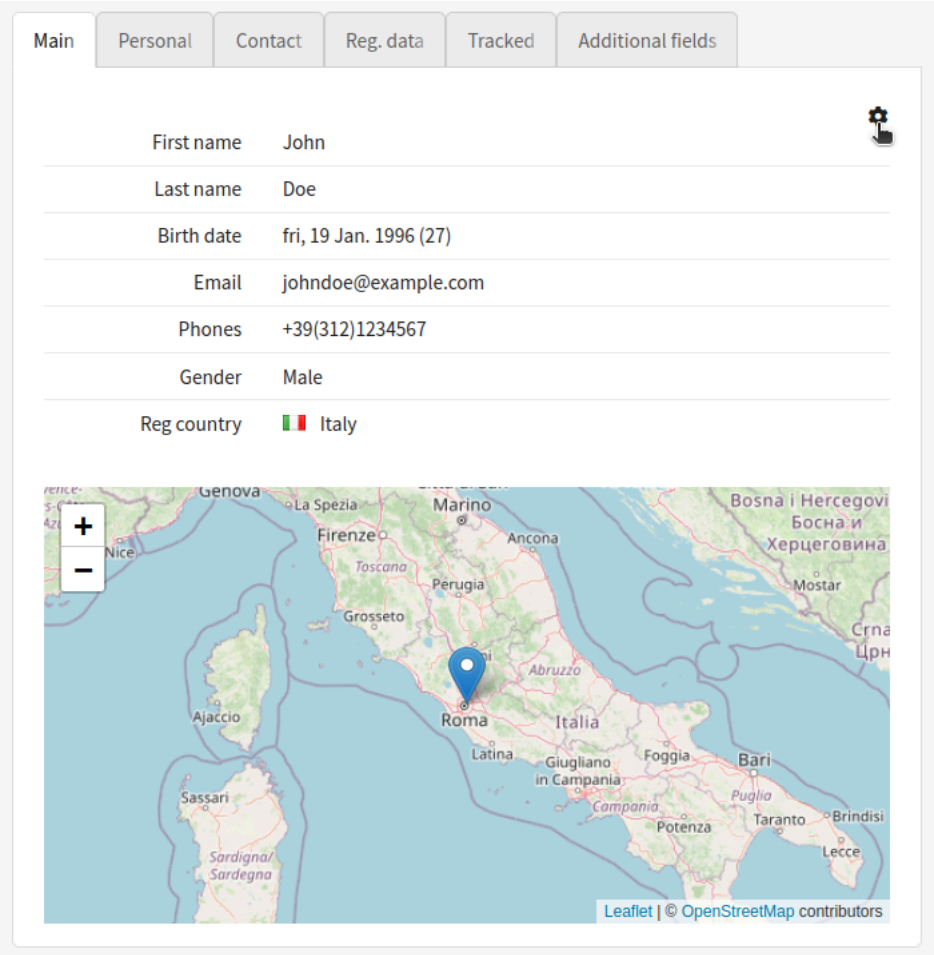
- **Subsribed — available for communication by all channels**
- **Unsubscribed — unsubscribed from receiving messages by all channels**
- **Complainer — reported to Spam. Communication is stopped.**
- **Hard bounced — the provider completely refused to accept messages to this address. Most likely mailbox does not exist and attempts to send something will decrease your reputation. Invalid addresses are recorded in the global suppression list.**
- **Unconfirmed — subscriber registered but did not confirm his consent to receive messages. It is used with Double opt-in subscription. After confirmation, the status is changed to Subscribed.**
- **Suspended — subscription is suspended for some time.**
- **Invalid — excluded for marketing mailings due to other reasons.**



Profile data

- **Main — general information and location pin. Pressing a cog button will allow you to add or remove fields to customize tab appearance.**
- **Personal — customer's demographic data.**
- **Contact — customer's basic contacts.**
- **Reg. data — opt-in information.**
- **Tracked — IP and location of the latest transaction.**
- **Additional fields — custom database fields containing business information.**

Email and Phones fields in the Contact tab are used to identify profiles in the database. Marketing mailings use resources.



TIP

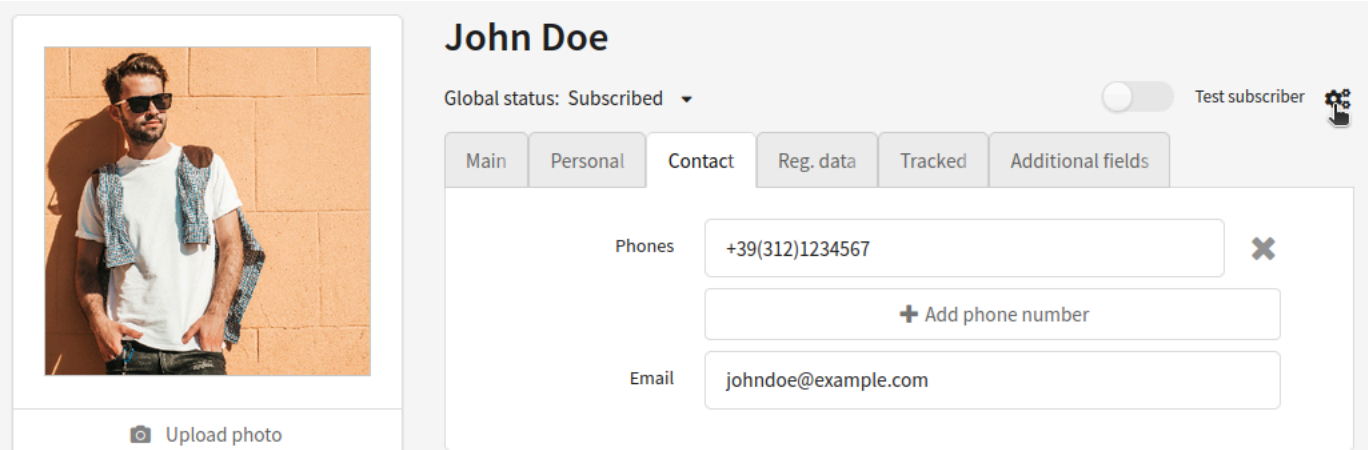
Subscriber location data can be set manually. Two types of default fields are used for this:

- **Latest action data — Country, Region, ZIP code, Timezone**
- **Registration data — Reg Country, Reg City**

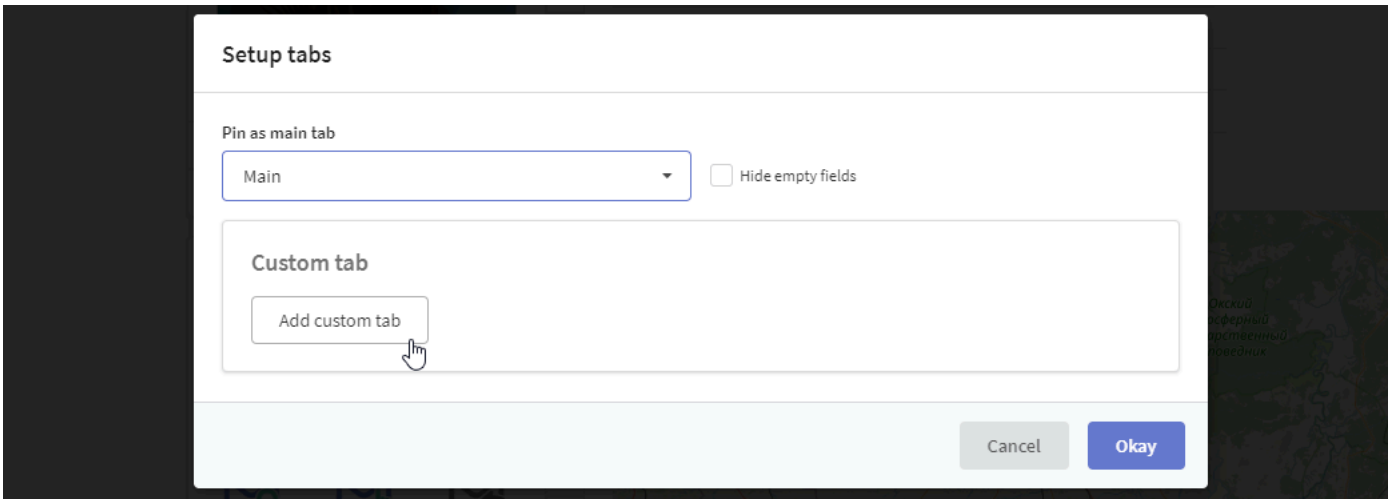
Location in Altcraft Platform is identified automatically using the subscriber's IP address. This happens when customers open letters, click links or perform actions on a website. Geographical coordinates are displayed on the built-in map widget.

Automatic location is also available during profile import. For registration data, Reg IP field is used. Auto-detection option must be activated beforeimport from file. For API import, this option is enabled by default.

Whenever you open a customer profile - the main tab will be displayed by default. To customize it press the cogs button on the right:



Select a tab to be displayed by default - or create your custom tab and customize data to be shown:



INFO

Custom data fields are created to store business-specific customer information. Thesefields may contain 8 types of data:

- **Number — positive and negative integer numbers.**
- **Float — floating-point number: it has a decimal place.**
- **Line — text in any language, including numbers, special characters, spaces.**
- **Date — date in RFC 3339 format, like `2019-08-22T00: 00: 00.000Z`.**
- **Boolean — thisfield provides a choice of two options: True or False (Yes or No). It is used to define if the customer has a certain attribute, like VIP status.**
- **Enum — this field is similar to Boolean, but allows you to choose one option from the unlimited number of options created by the user. For example, customer tariff plan. Options are written in a comma-separated line.**
- **IP — IP in IPv4 format. For example, `127.0.0.1`.**
- **Tags — in thisfield, several values can be written simultaneously. They may be products or services purchased by a customer, or other values, composition of which may vary. Tags are written inline, separated by `","` `"|"` or `";"`. Therefore, these characters can not be used in the tag itself. For example `Product-1`, `Product 2`, `Product_3`.**

Custom fields are configured in the profiles database structure.

You can find detailed information about the types of fields in the database and the allowed values in this article.

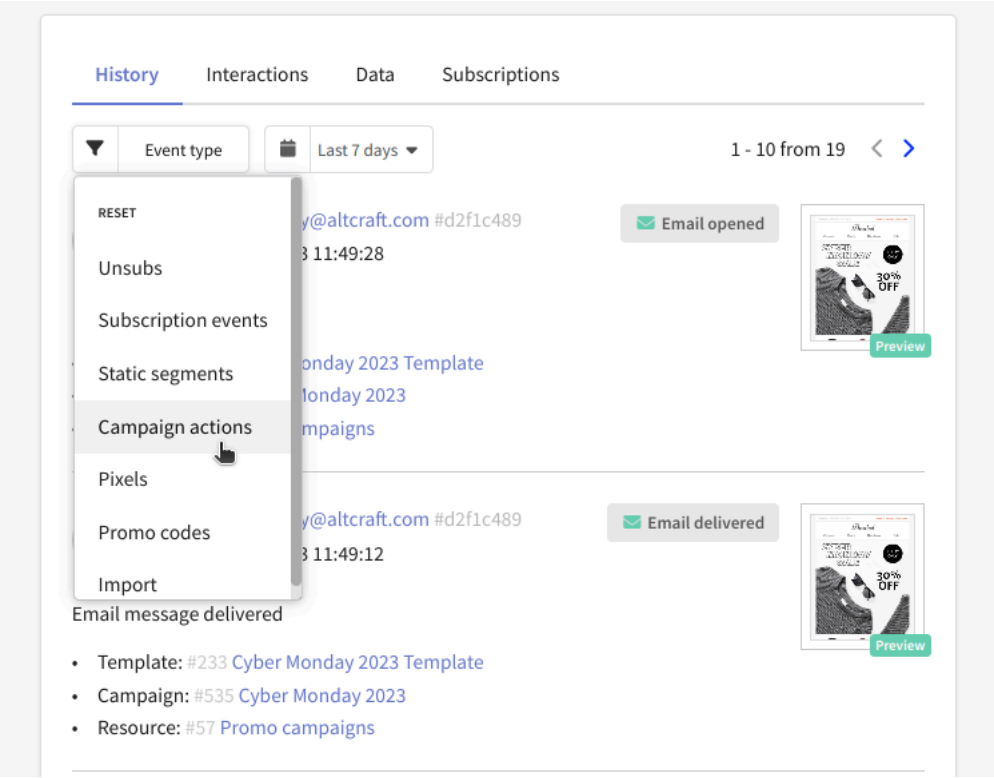
History

Profile's action history collects all the events connected with the profile. It can be used in behavioral segmentation:

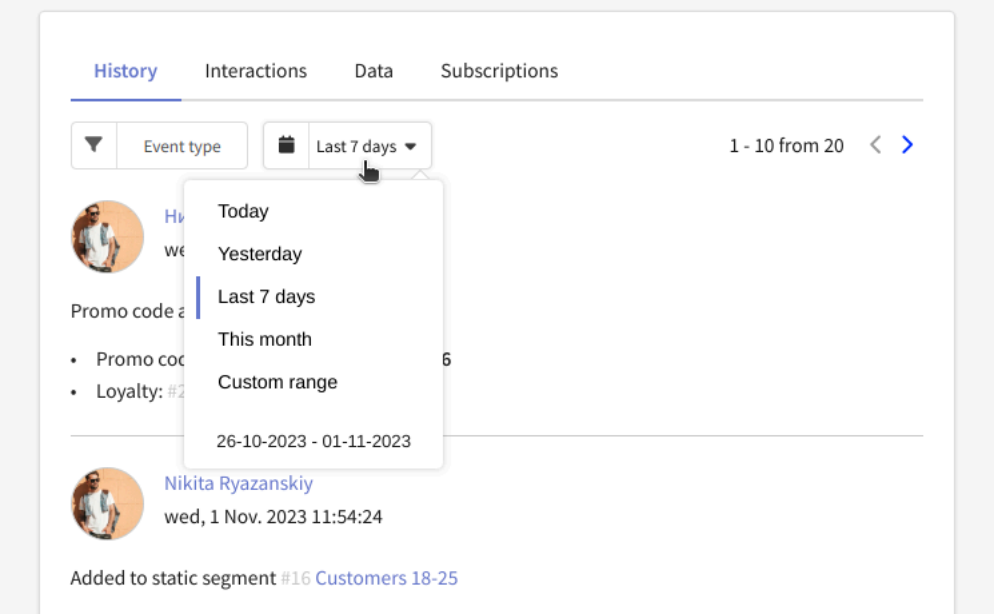
- **Import. How the profile was added to the database (manually, via API import,**
- **Subscription events. The profile was subscribed to a specific channel of the resource (Email / SMS / Push / Telegram / WhatsApp* / Notify / MAX / custom channel), or the profile was unsubscribed from the channel in the resource by changing the subscription status to "Unsubscribed".**
- **Mailing actions. Includes key events tracked in mailings (Sends, Deliveries, Opens, Reads, Clicks, Confirmation Link Clicks), as well as change of subscription status to "Hardbounce" or "Complainer".**
- **Unsubs. Unsubscribe events that are fixed when a profile clicks on an unsubscribe link in a message.**
- **Pixels. The profile performed an action on the site or in the mobile app that triggered the tracking pixel. The achievement of the goal and its cost are fixed.**
- **Promo codes. Includes events such as promo code attached, detached or activated.**
- **Static segments. The profile entered or exited a static segment.**

**To check a specific event type, use a filter. When you click on **

the conditions in the filter will be reset.



By default, profile activity history is displayed for the last 7 days, but you can select the period you need:



TIP

When the subscription status changes (in the interface, via API, or during import), events are registered only for the following statuses:

- **Subscribed**
- **Unsubscribed**
- **Complainer**
- **Hardbounced**

When the status changes to Subscribed, the event is registered only if the previous subscription status was Unsubscribed, Complainer, or Hardbounced.

For example, if the subscription status is set to Suspended and then changed to Subscribed, the event will not be registered.

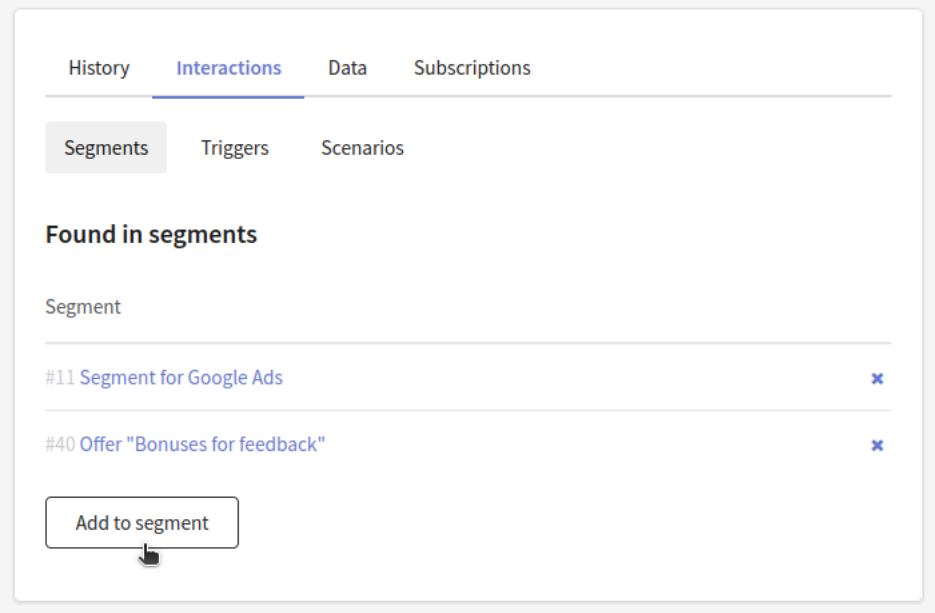
Events are not registered for the following statuses:

- **Unconfirmed**
- **Suspended**
- **Invalid**

Interactions

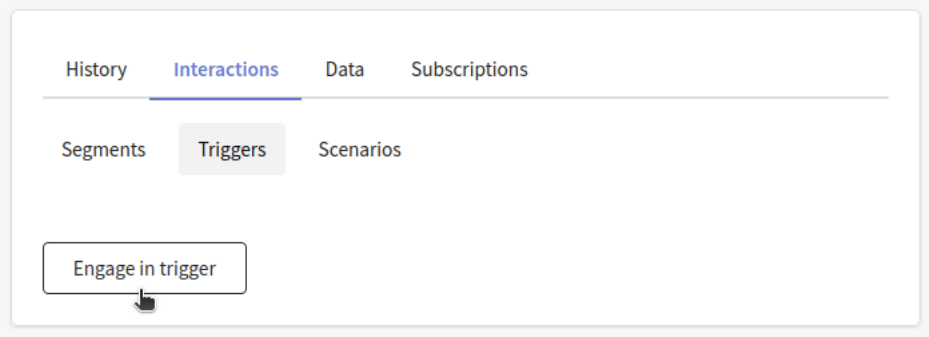
Segments

Static segments are used for advanced analytics of customer fixed groups. There is a ready-made integration to export static segments data via SQL. Here you can add the profile to a segment or remove.



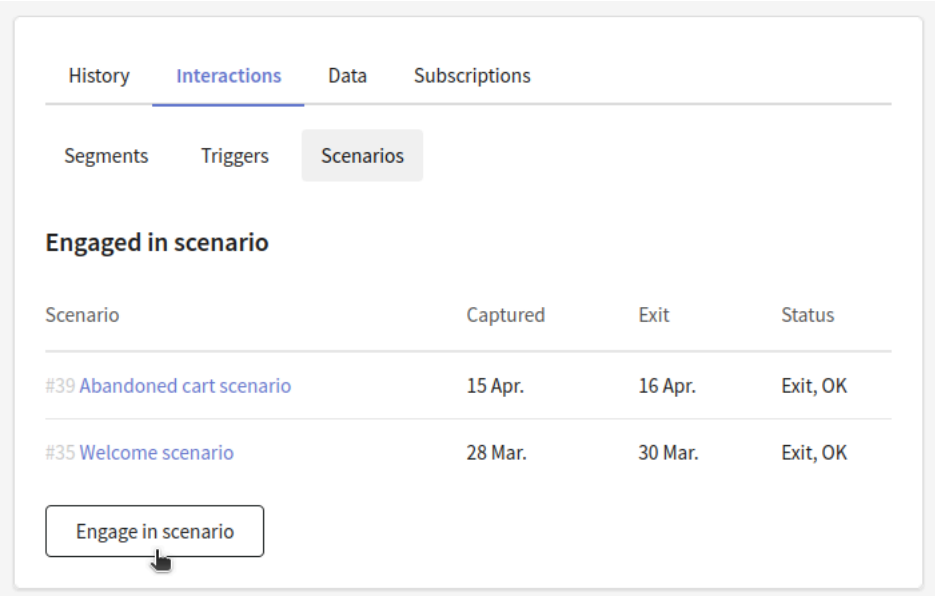
Triggers

Triggers are mailings that are launched when a certain event occurs (the profile clicks on a link, fills out a form, etc.). On this tab, you can manually send a trigger message to a profile.



Scenarios

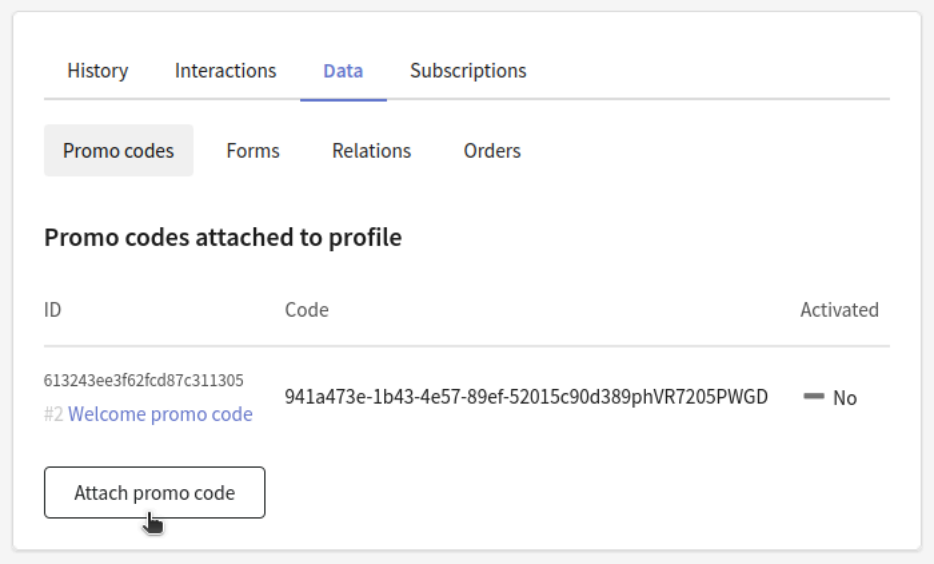
Scenarios help you automate your marketing workflows. This tab contains the history of a customer's scenario participation and results. And you can engage a customer in a scenario - manually.



Data

Promo codes

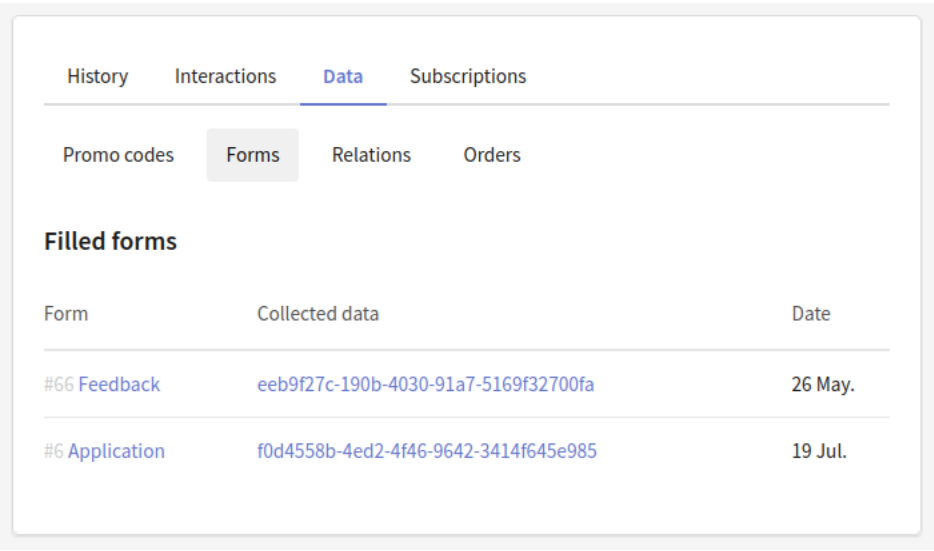
Promotion codes can encourage deals and they can be managed by Altcraft Platform loyalty programs. On this tab all the relevant information is displayed. You can manually assign a promo code as well.



Forms

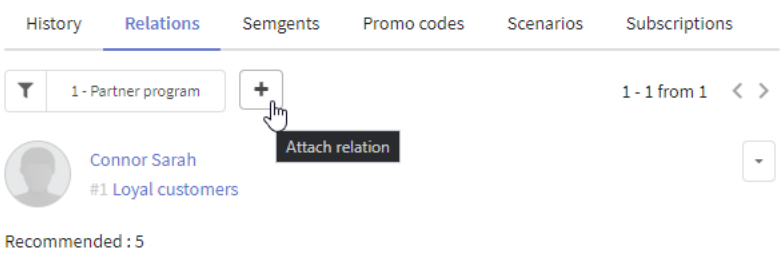
Forms are a tool for collecting data from your customers. With its help, you can implement various mechanisms using external data, for example, add new profiles to the database, manage subscriptions to various resources, and receive feedback from customers.

On this tab, you can see which forms the profile has already filled out. When you click on the collected data, you will be taken to a new window that displays all the data left by the profile in the form.



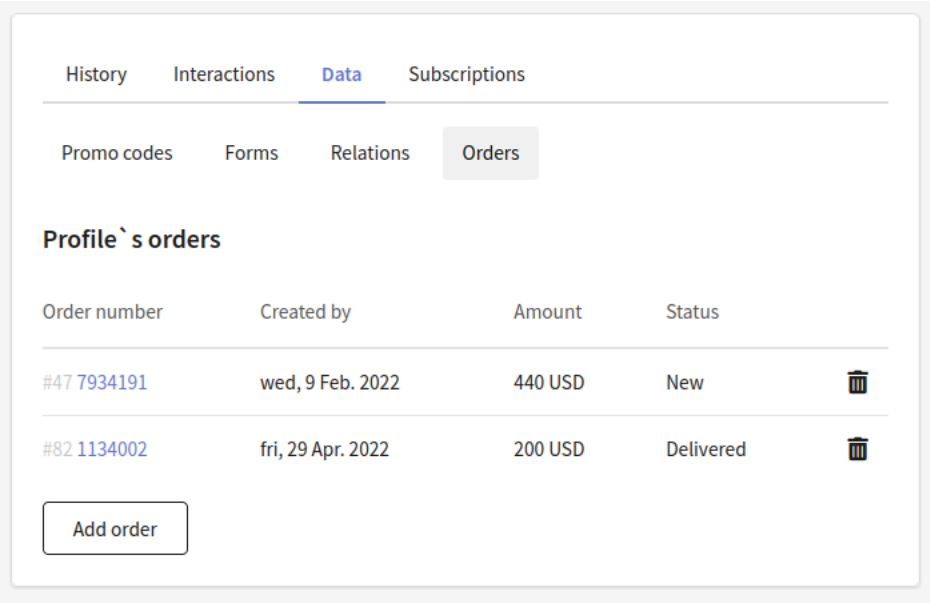
Relations

Relations between profiles can be used to register interactions between your clients — likes in a social network or service recommendations can serve as examples here. On this tab you can add a relation manually.



Orders

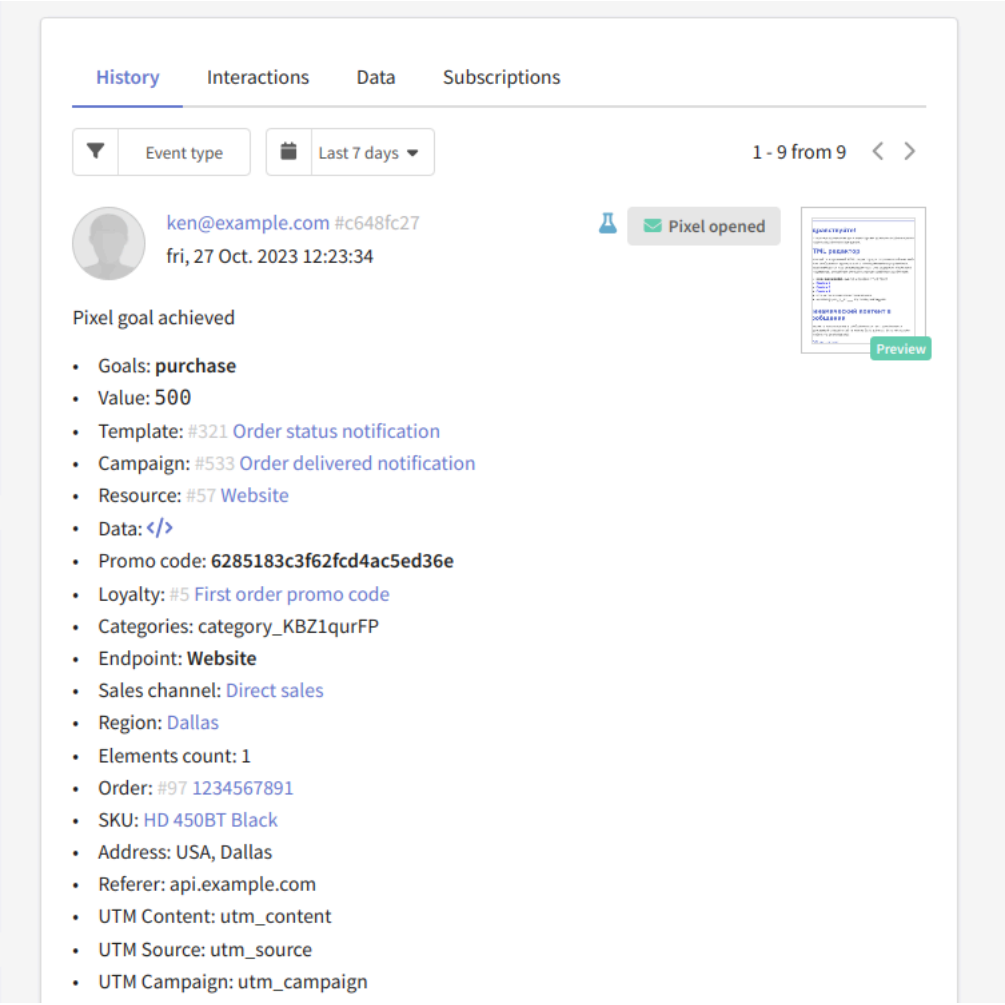
Orders are an element of the Market module. This module allows you to import product and order data to the platform and use this data to communicate with profiles. To activate this module, you must configure the appropriate access roles.



Pixels

A pixel is an HTML or JavaScript code placed on a website or in an application designed to track user actions. When a subscriber performs a specific action on the website or in the application, a request is sent to the server to record this action.

The pixel can also transmit information related to this action, such as the achieved goal, the value of the action, the URL of the website from which the subscriber came, profile geodata, information about sent mailings, and more. These data are recorded in the profile history and used for segmentation.



TIP

To view the data passed in the `pixel_data` object, press `</>` to the right of Data.

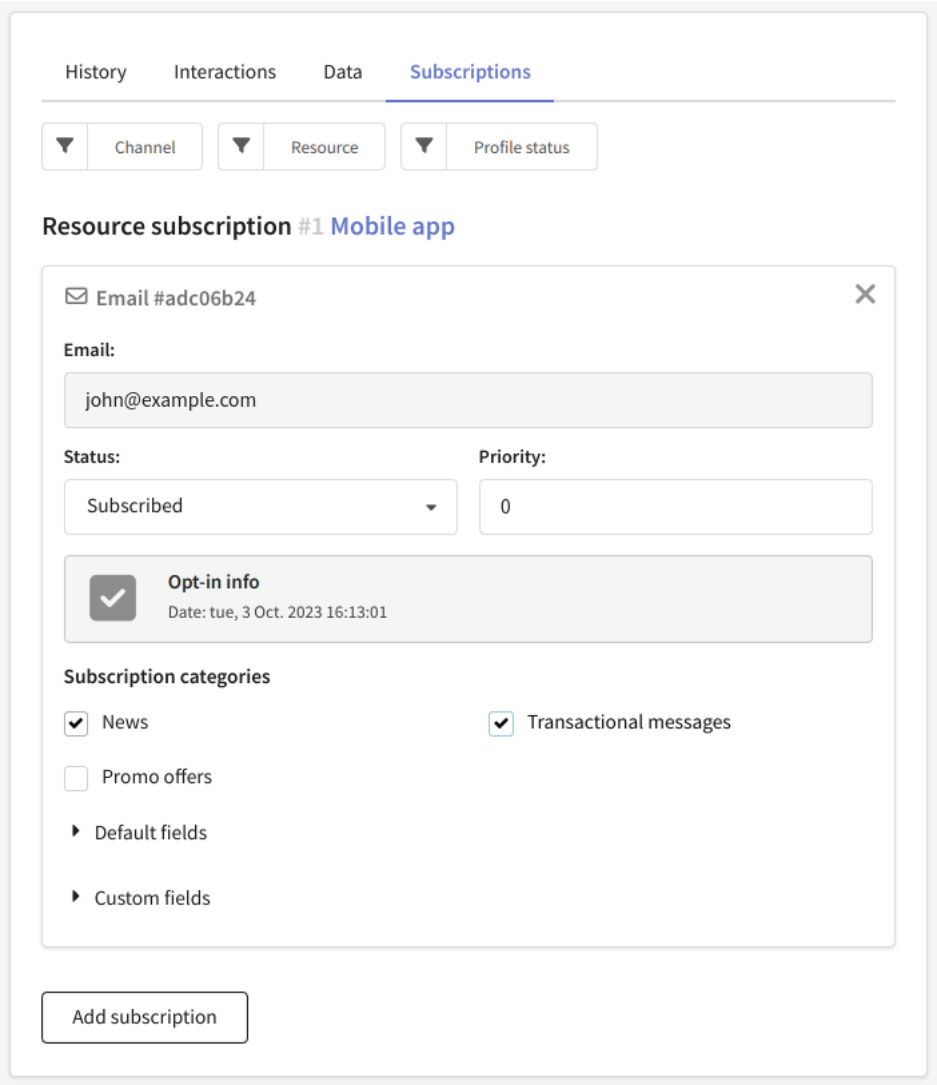
INFO

Goals can be registered for clients whose contact and personal information you do not yet have. Such events are called anonymous. Altcraft distinguishes them from each other based on connection data (cookies). If a client later registers using the same browser, their data is added to the already existing profile.

Profile subscriptions

Profile subscription is a record that combines contact data for a specific communication channel and information about registration. Every profile can have a number of subscriptions grouped by resource - the source of subscription. Every profile has a limit of 200 subscriptions or 20 subscriptions per resource in a channel. Also, different resources can use different contact information. For example, if a customer unsubscribes from one resource, they will continue to receive content from another resource.

Here you can manually change subscription status and priority:



Status

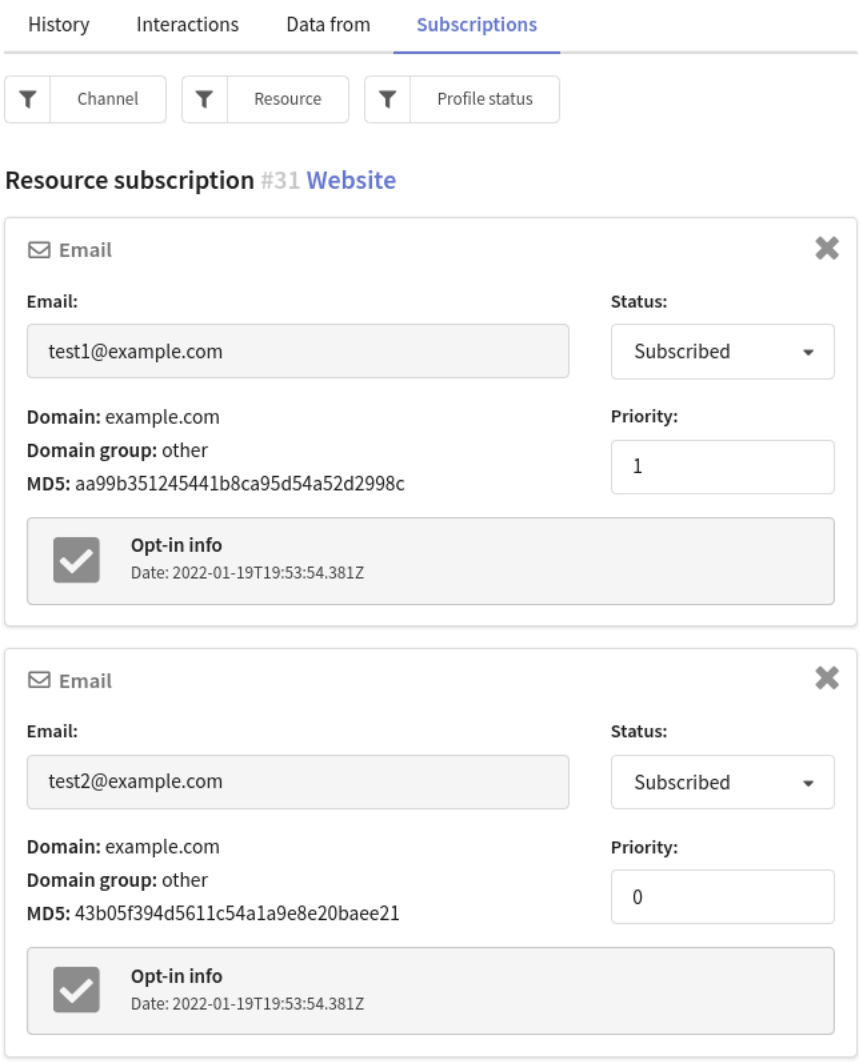
Subscription status defines communication possibility with a customer by specific contact data within a specific resource. Here, like for global profile status, there are 7 types:

- **Subscribed — available for communication by all channels**
- **Unsubscribed — unsubscribed from receiving messages by all channels**
- **Complainer — reported to Spam. Communication is stopped.**
- **Hard bounced — the provider completely refused to accept messages to this address, most likely the mailbox does not exist and attempts to send something will spoil your rating with the provider. Invalid addresses are recorded in the global stop list of the platform.**
- **Unconfirmed — the subscriber registered but did not confirm his consent to receive the newsletters. It is used with Double opt-in subscription. After confirmation, it changes to Subscribed.**
- **Suspended — subscription is suspended for some time.**
- **Invalid — excluded for marketing mailings due to any other reasons.**

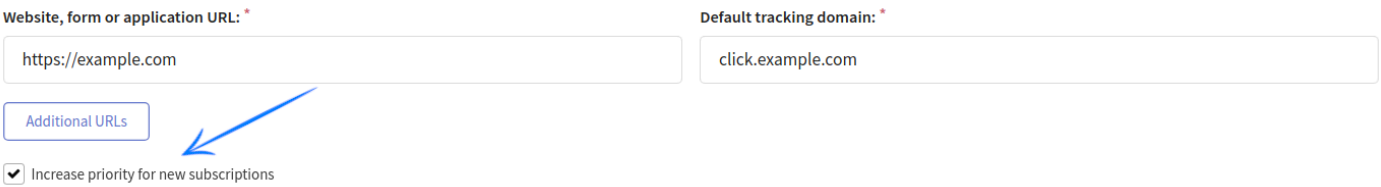
Priority

A user can subscribe to one resource several times by entering different contact details (email or phone number). The higher the value in the Priority field, the lower the priority of the subscription.

In the example below, mailings within the “Website” resource will be sent to `test2@example.com`, because a subscription with these contact details has the highest priority — 0.



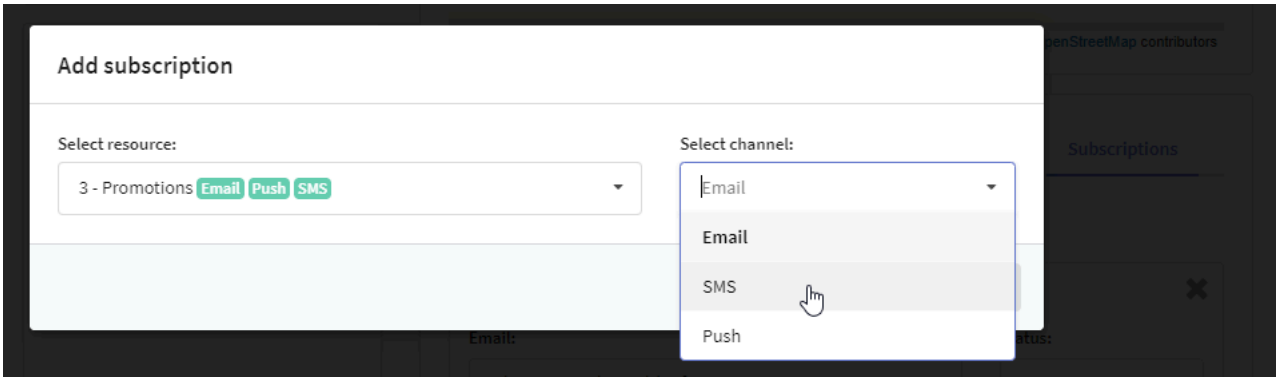
You can increase the priority of a subscription automatically. To do this, activate the option in the resource settings.



CAUTION

Please note that the "Increase priority for new subscriptions" option applies only if the priority is not explicitly specified when adding a new subscription. If the priority of the new subscription is explicitly specified (for example, in an API request), then the priorities of existing subscriptions remain unchanged.

If you want to subscribe a customer to a resourse manually, press the Add subscription button, specify a resource and a channel, enter contact data and save your changes:



TIP

The subscription process (opt-in) can be organized in two ways:

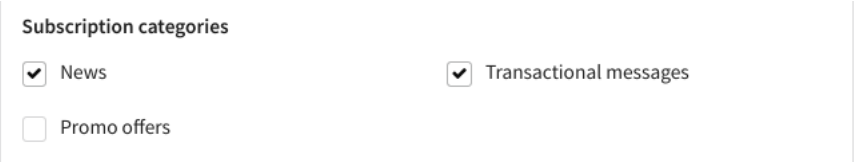
- **Single opt-in (SOI) - Without an agreement to receive content. It is enough to receive customer information (or create a subscription to a resource in Altcraft).**
- **Double opt-in (DOI) - Creating a subscription is not enough. Customers must send an email confirming their agreement to receive messages. This process allows you to exclude potential complaints from customers on marketing mailings, increase their effectiveness and improve sender reputation.**

Click here to find instructions on how to organize Double opt-in subscriptions in Altcraft.

Subscription categories

A resource subscription can include multiple categories. Categories enable personalization of offers. The client will receive messages only within the categories to which they are subscribed.

Unlike a subscription to a resource, a subscription to a category has only 2 statuses: “Subscribed” and “Unsubscribed”. To subscribe a profile to a category, check the appropriate box. You can also manage category subscription status via API requests.



Channel subscription fields

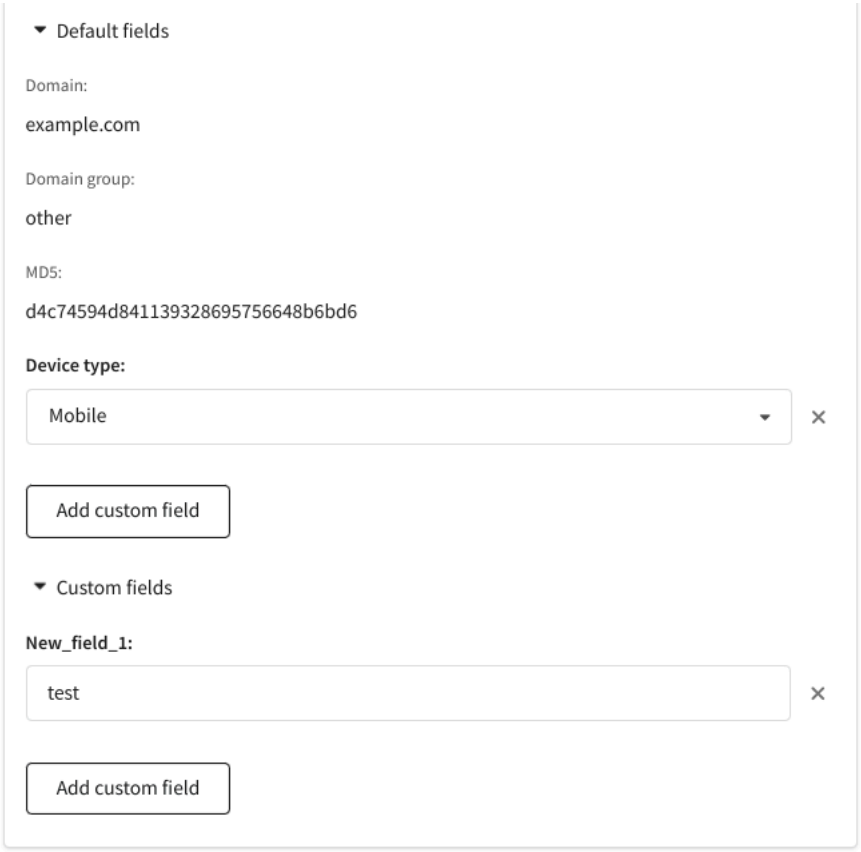
Data about a profile’s subscription to a specific channel is stored in standard and custom fields.

You can also include additional standard fields that store information about the device and profile application:

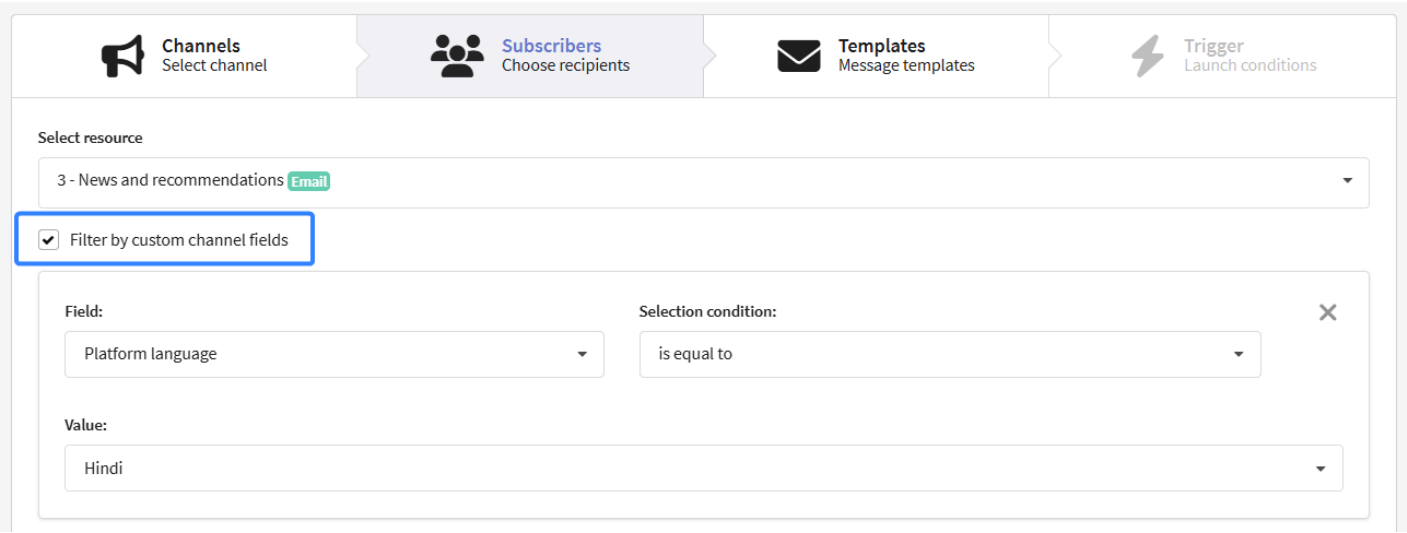
- **Device type (Desktop, Mobile, Other)**
- **Device model**
- **Device name**
- **Platform (Windows, Linux, Mac, Opera, iOS, BlackBerry, Android)**
- **Platform version**
- **Platform time zone**
- **Platform language**
- **Ad tracking allowed (True or False)**
- **Platform language**
- **Ad identifier**
- **Application ID**
- **Application installation ID**
- **Browser: Bots, Internet Explorer, Firefox, Safari, Facebook*, Chrome, Opera, Outlook, Thunderbird**
- **Application version**
- **Browser version**

To add a new field, click the Add custom field button and select a field from the list, then enter its value.

To store additional information about your channel subscription, use custom fields. They are created in resource settings.



You can use them to filter the recipients of a trigger mailing. This will make the mailing more personalized.



Subscriptions Limit

There is a subscriptions limit for a profile. A profile can have no more than 200 subscriptions, or no more than 20 subscriptions per resource in a channel. If the maximum number has been reached, a new subscription will replace one of the old ones.

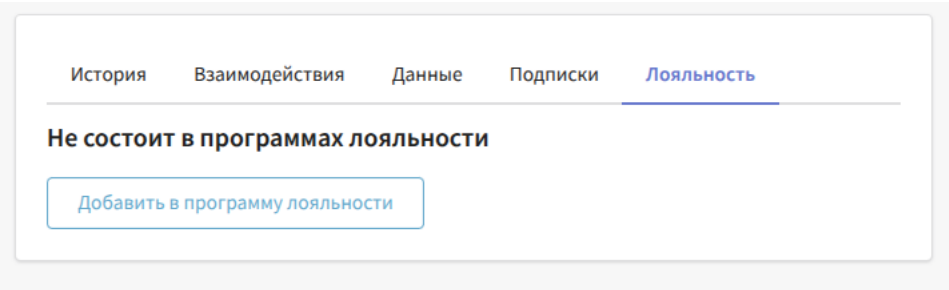
The following categories of subscriptions are being removed at first place:

- **All deleted subscriptions**
- **All subscriptions with statuses:**
  - **Invalid**
  - **Hard bounced**
  - **Complainer**
  - **Suspended**
  - **Unconfirmed**
- **Subscriptions with the lowest priority**
- **Oldest subscriptions by creation date**

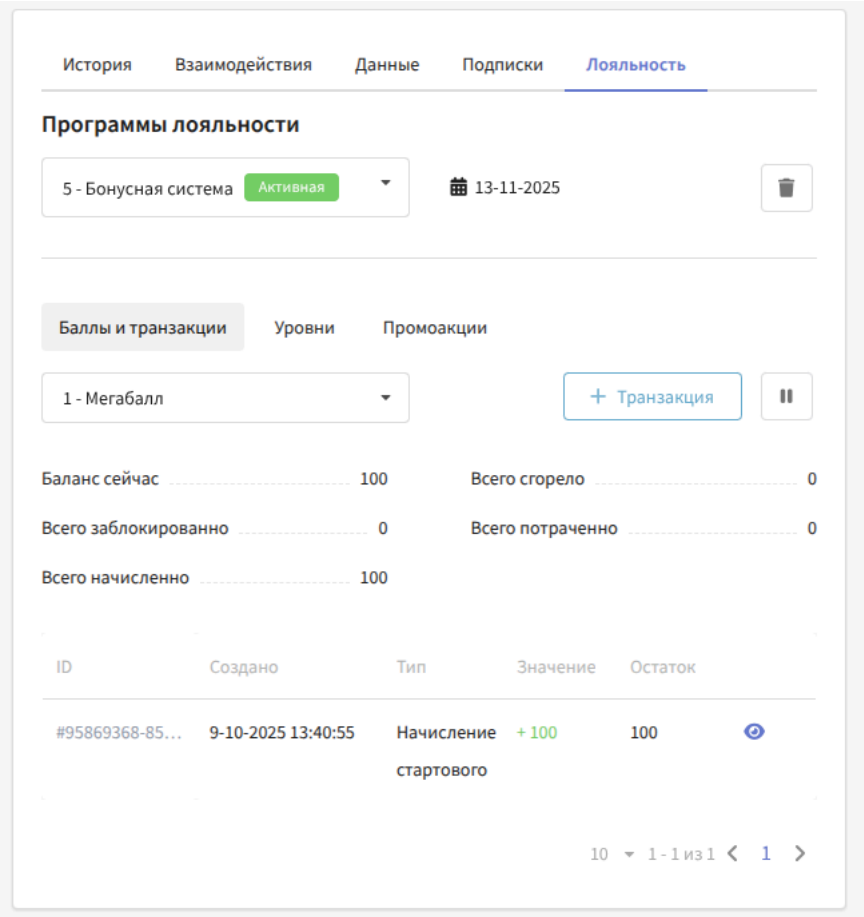
If you try to add replaced subscription, the platform will consider it a new one and re-add it to the list of subscriptions.

Loyalty programs

This section displays information about a profile's participation in loyalty programs. If the profile is not enrolled in a program, you can register it:



If the profile is already registered in one of the loyalty programs, this tab shows the profile's points and transactions, its tier in the loyalty program, and participation in the program's promotions. From here, you can also perform points transactions for the profile:



Identifiers

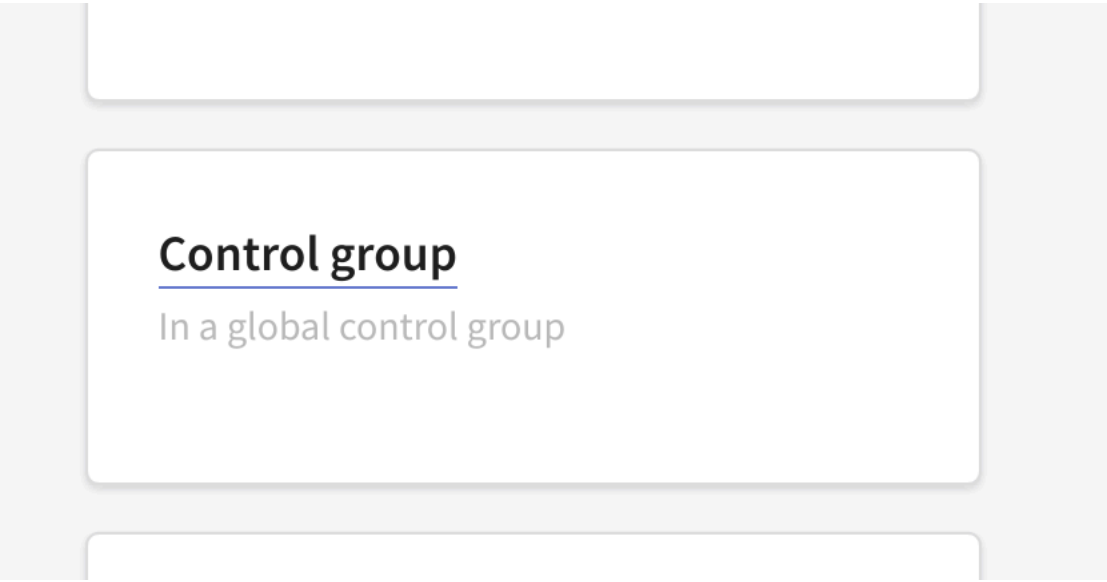
![img](/img/attachments/3114129/22708328.png)	**Every subscriber profile in Altcraft Platform has a unique profile identifier profile_id — for all databases. It also gets unique MD5 and XXH identifiers. Within a database, the standard Email field is unique for profiles. However it is not required:if email address is missing, a new profile will get a **profile_id@localhost** record.**

UTM

![img](/img/attachments/3114129/37847347.png)	**UTM tags are updated in the subscriber's profile if they clicked on a link with UTM tags in the message template or submitted a [form](/en/user-guide/weblayers/forms/form-creating/37848225) with configured UTM tags (the "Update UTM data to profile" option is enabled in the form).**

Participation in the control group

If the client profile is currently in the global control group, the corresponding mark will appear in their profile:



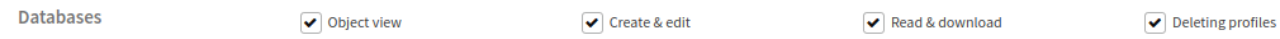
Also, the action history will show every time it gets in the GCG:

![img](/img/attachments/gcg/en/gcg-profile-2.png)	

Deleting profile

![img](/img/attachments/3114129/28936560.png)	**If you need to delete a profile from the database, you can do it using the "Delete profile" button. This action will also remove profile actions history and relations.**

Special permission is necessary for deleting profiles. It can be configured in the user roles. You can read more about roles here.



You can also use an API request to delete a profile (see more).

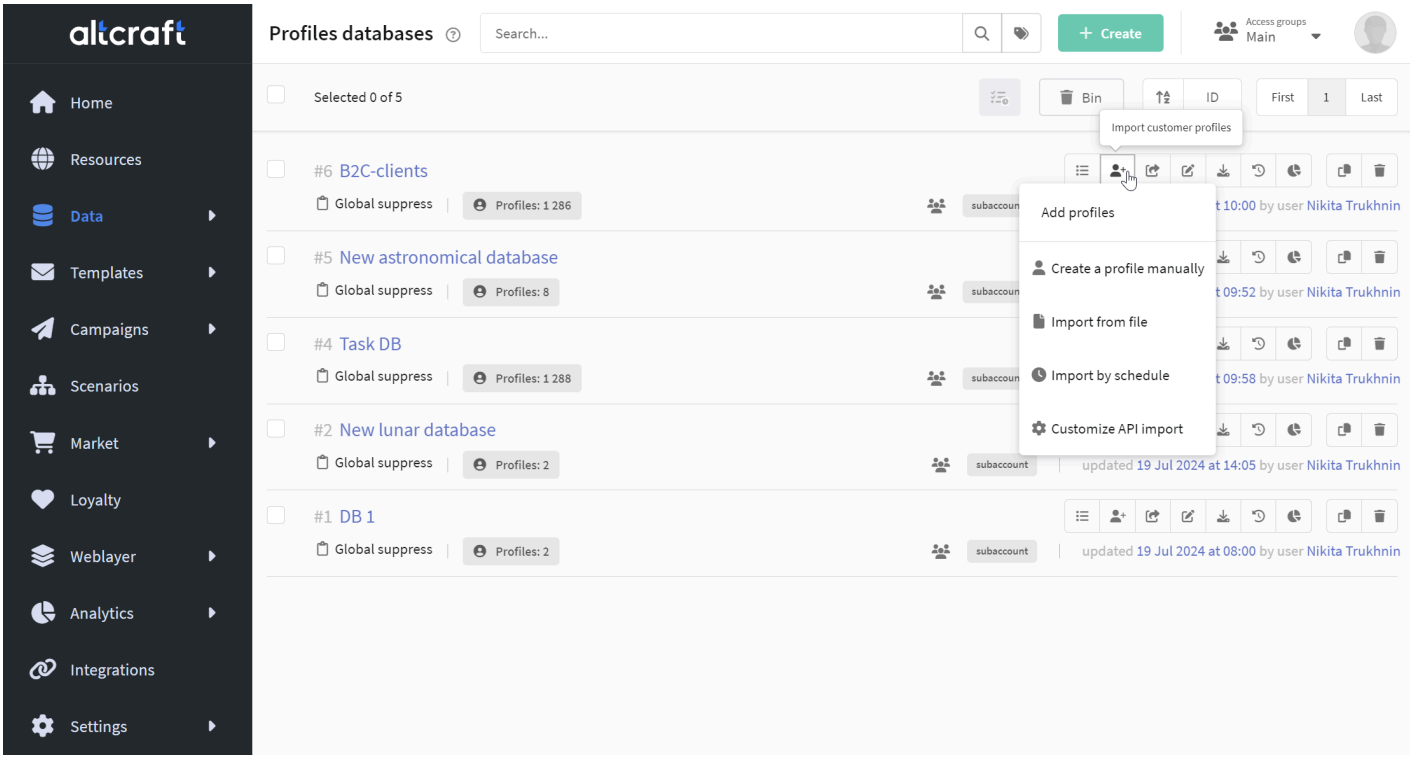
The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Profiles import and data update

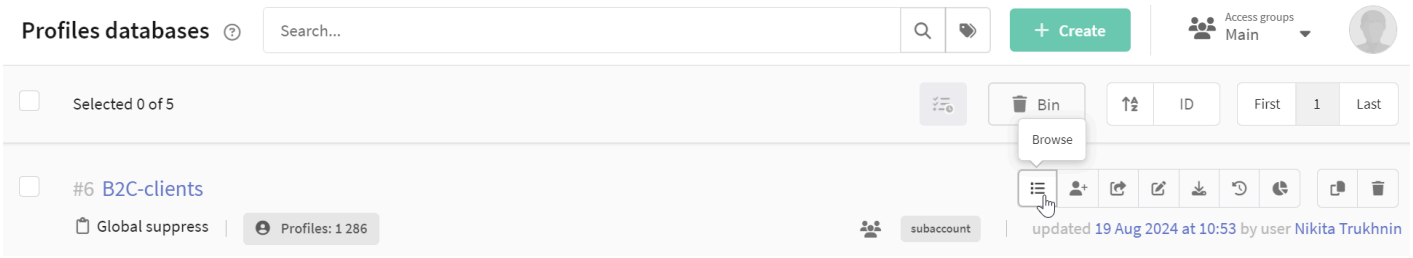
How to import data

There are three ways to proceed to import data:

1. **Under Data, select Profile Databases. Click on “Import Customer Profiles” next to the profile database you want and select the appropriate way to load the data:**

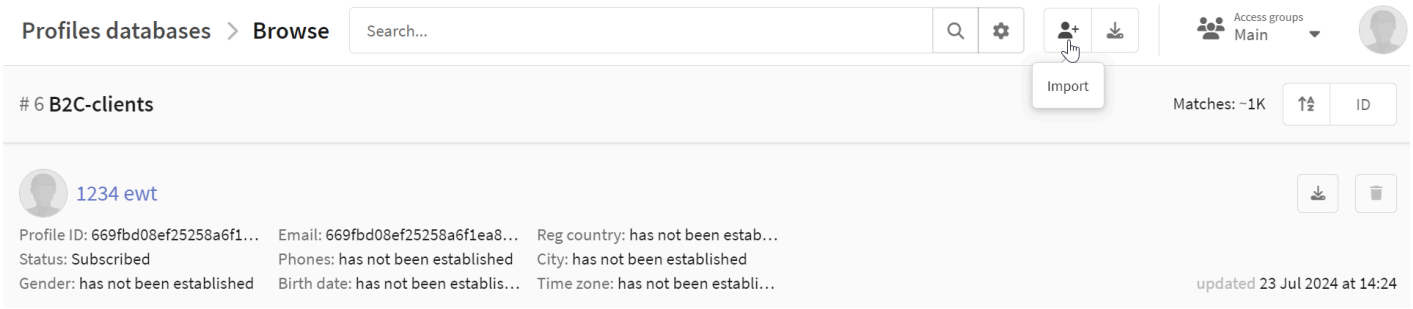


2. **Select your desired Profile Database and click on the Browse tab**



Click on the icon

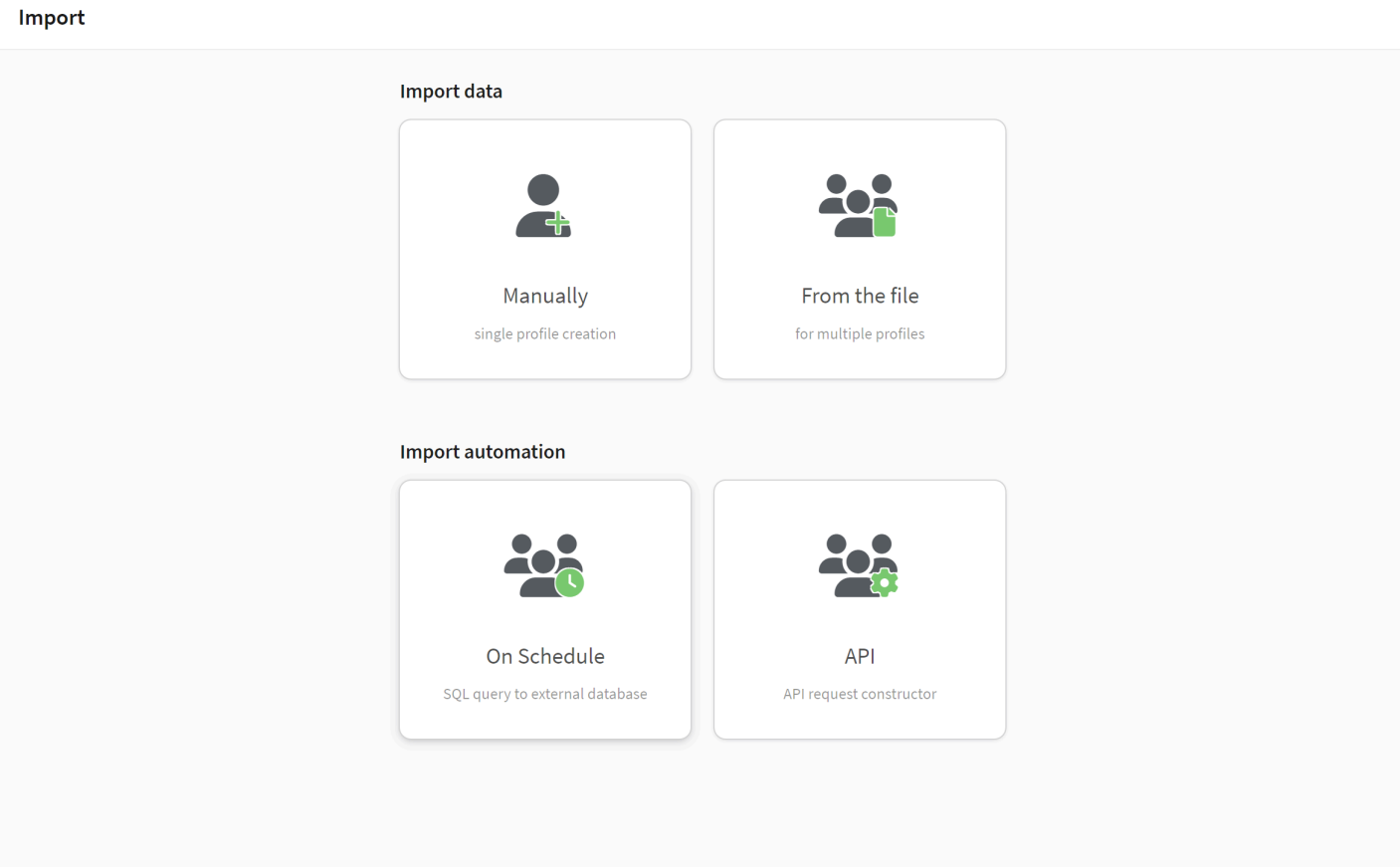
at the top of the screen and select the import method that suits you:



3. **In the Data section select Import.**

Customer profiles import and update can be organized depending on database size, new subscribers processing scenario (directly or from CRM), and company technical capabilities. There are 4 ways to import profiles into a database:

- **Manual import — add one profile at a time using the edit form**
- **From a file — import profiles from a file with customer data**
- **Via API — import profiles using API integration**
- **Schedule — import profiles from the external SQL database on schedule**



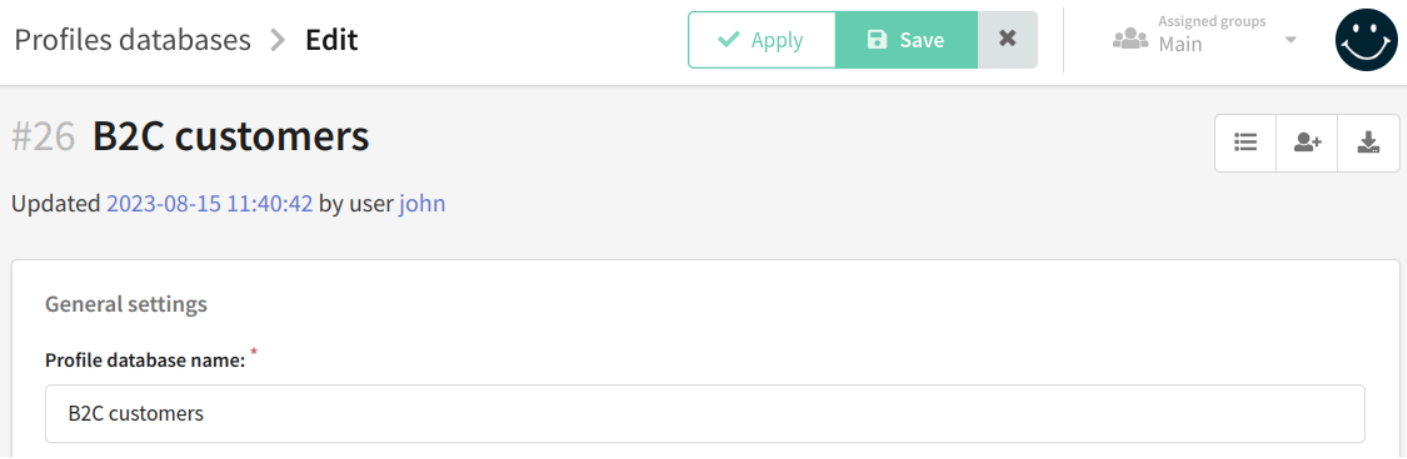
TIP

Altcraft Platform also has integrations with external services:

- **Tilda — integration with Tilda data collection forms.**
- **LPGenerator — integration with LPGenerator data collection forms.**
- **Yandex AppMetrica — integration with Yandex application metrics service.**

Manual profile creation

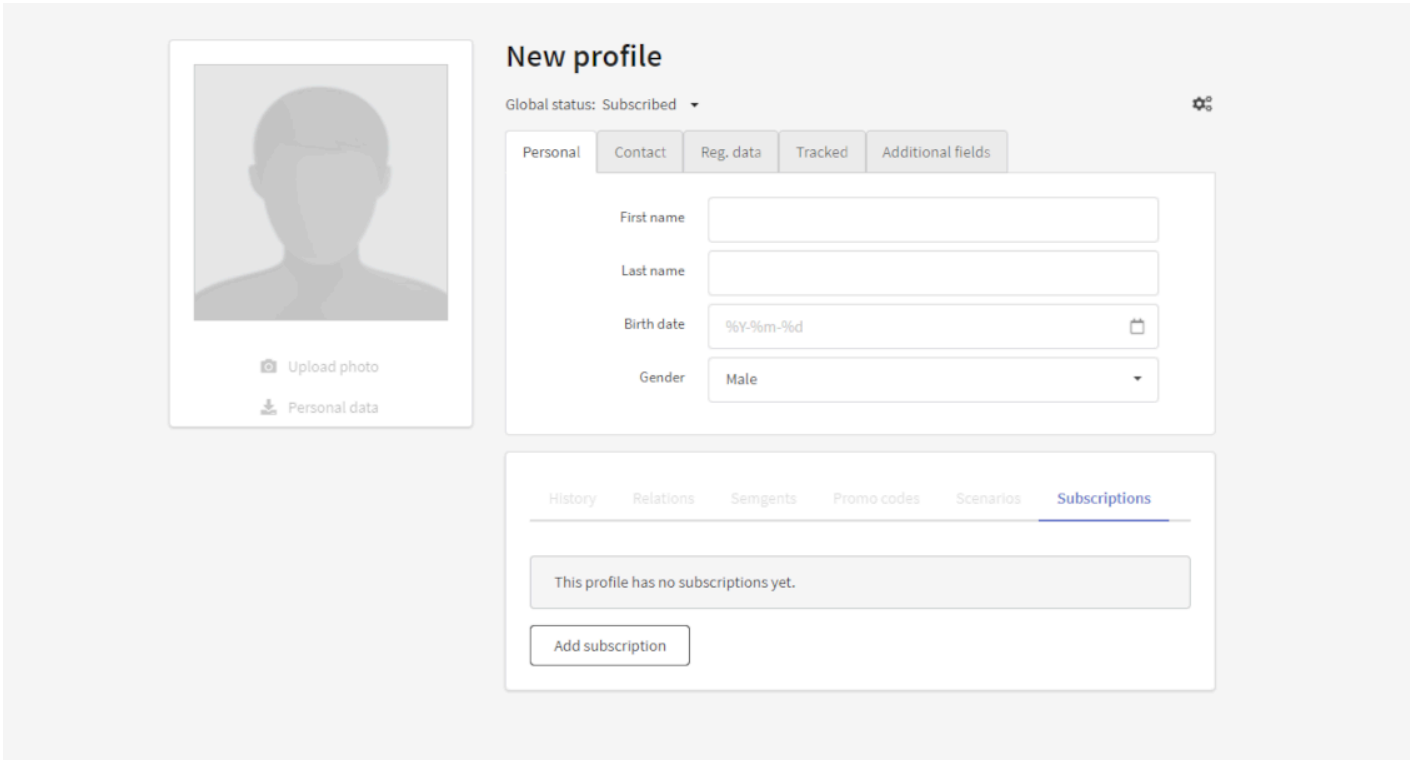
Select the import method - Manually.



The profile creation/editing form will open. During profile creation, you can add personal, contact, registration, and tracked data. This information is recorded in standard database fields.

Values for additional (custom) fields are specified in the "Additional fields" tab. Mandatory fields are marked with a red asterisk. You can create additional fields and set them as mandatory on the database editing page.

Below, on the Subscriptions tab, you can immediately assign a client to a specific resource when creating the profile.

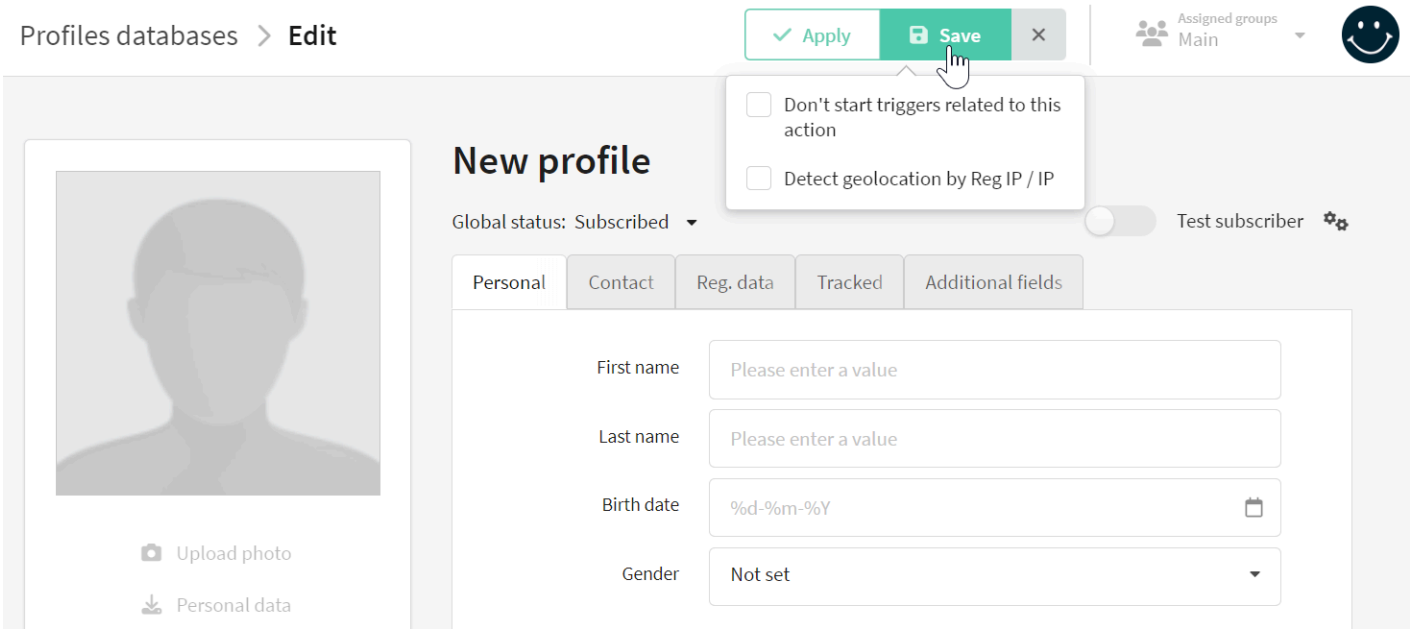


CAUTION

Email and Phones fields on the Profile tab are not used for communications. You can store contacts in these fields and use them in subscriptions. Youcan also leave these fields empty and store customer information in subscriptions.

By default, the Email field is used for profile identification along with profile_id. You cannot create a profile manually without this field.

When saving a new profile, you can disable triggers and enable location by Reg IP (registration IP address)/IP (last known IP address):

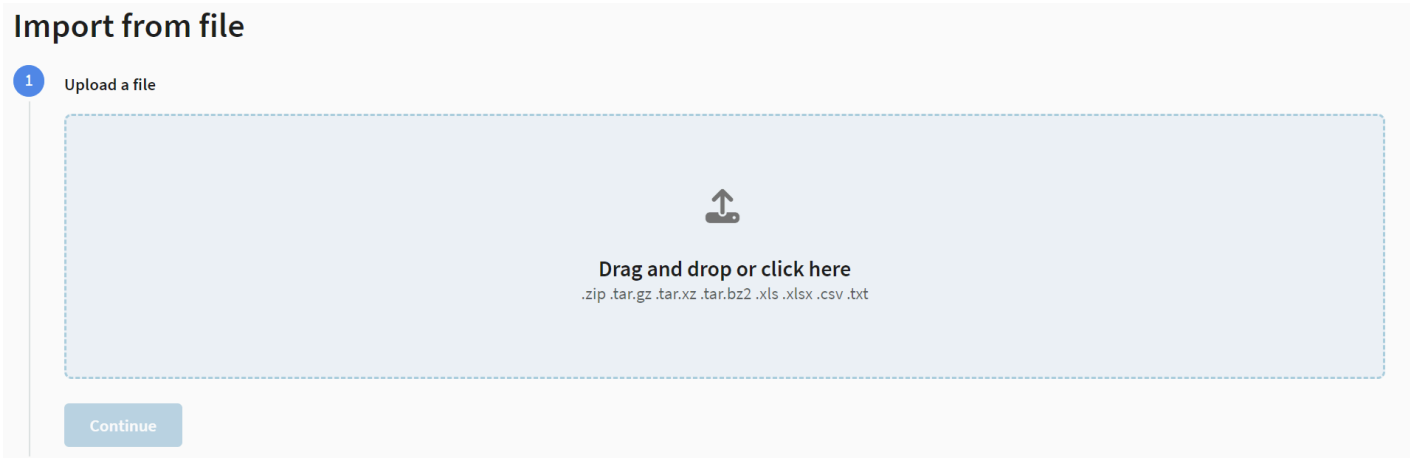


Import from file

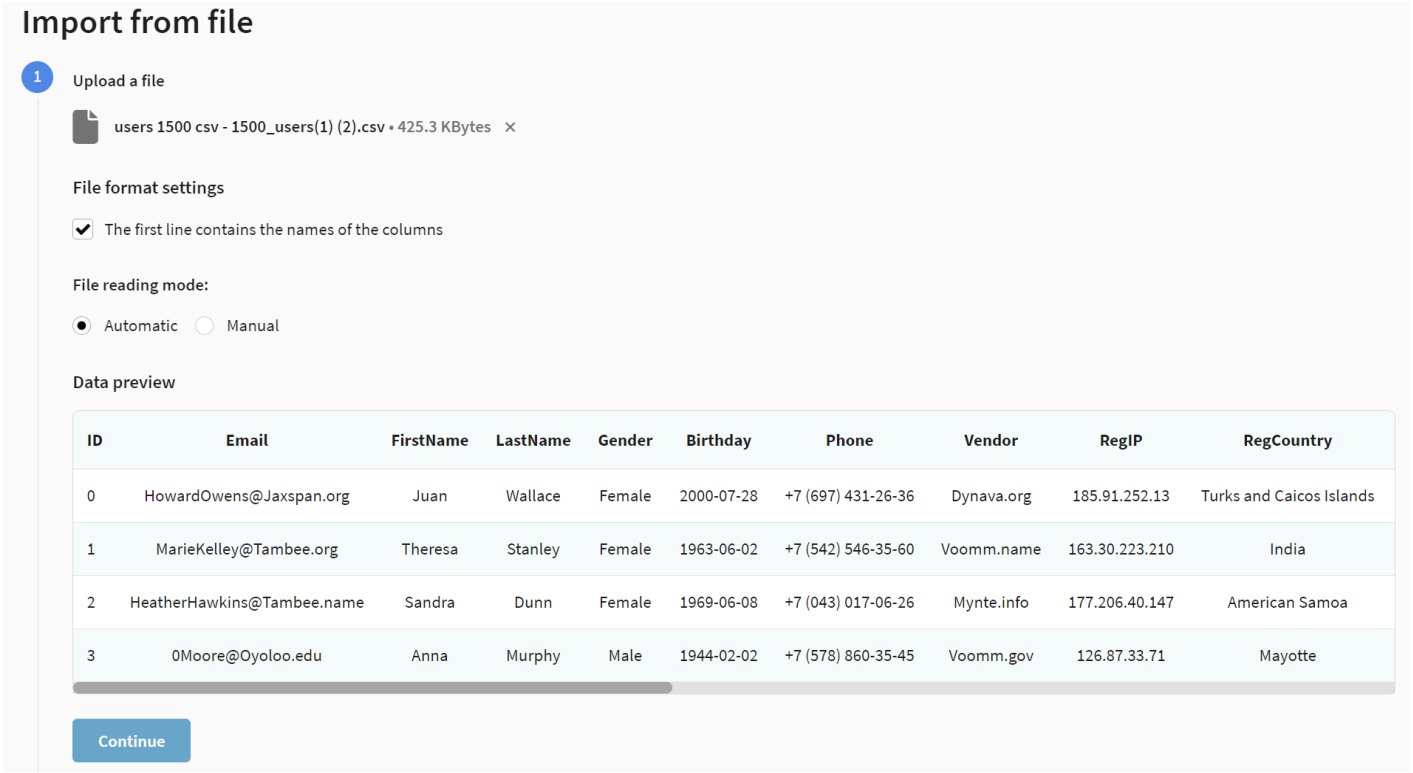
Uploading a file

You can use import from files to upload many profiles or update existing data. csv, txt, xls, xlsx, zip, tar.gz formats can be used.

Multiple table files can be uploaded into import from file form simultaneously. Press X next to file name to remove it. To add a new file, drag it to upload area or click on it and select file:



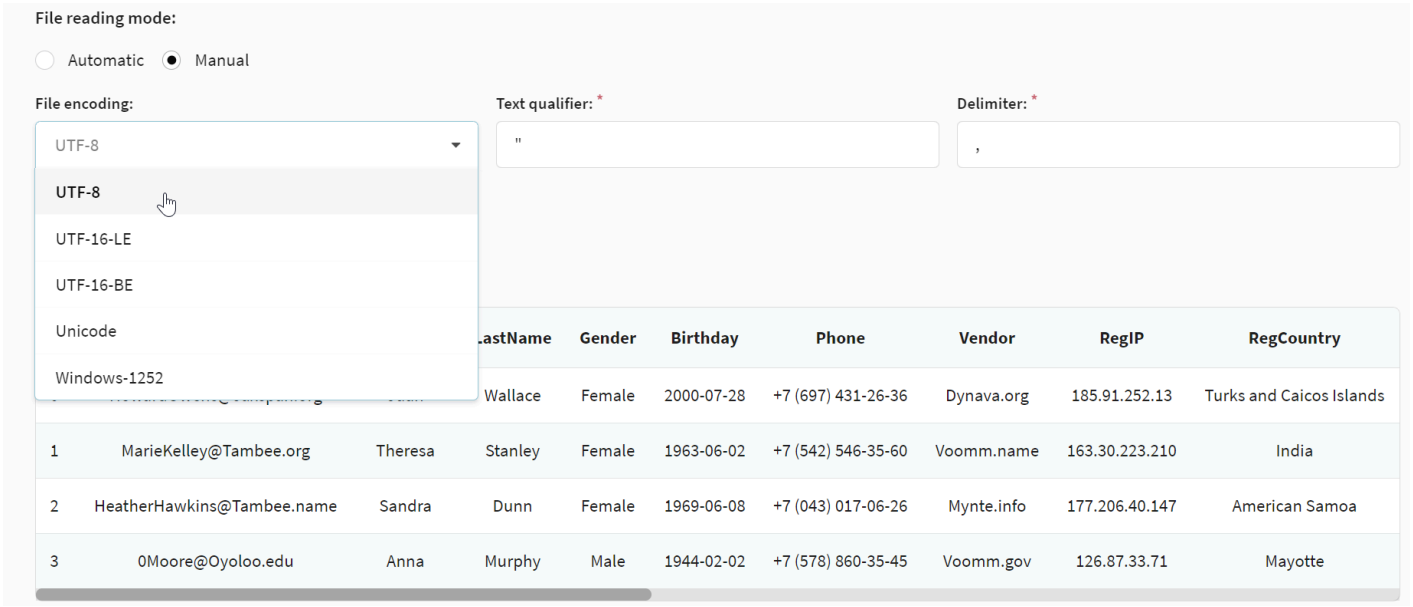
Check the “The first line contains the names of the column” box if the first line in your profile file is used to designate columns.



ID	Email	FirstName	LastName	Gender	Birthday	Phone	Vendor	RegIP	RegCountry
0	HowardOwens@Jaxspan.org	Juan	Wallace	Female	2000-07-28	+7 (697) 431-26-36	Dynava.org	185.91.252.13	Turks and Caicos Islands
1	MarieKelley@Tambee.org	Theresa	Stanley	Female	1963-06-02	+7 (542) 546-35-60	Voomm.name	163.30.223.210	India
2	HeatherHawkins@Tambee.name	Sandra	Dunn	Female	1969-06-08	+7 (043) 017-06-26	Mynte.info	177.206.40.147	American Samoa
3	0Moore@Oyoloo.edu	Anna	Murphy	Male	1944-02-02	+7 (578) 860-35-45	Voomm.gov	126.87.33.71	Mayotte

Select File Reading Mode. Two modes are available: Automatic and Manual.

The automatic mode uses UTF-8 encoding, if your file has a different encoding, you can go to manual reading mode and choose the one that suits you:



			LastName	Gender	Birthday	Phone	Vendor	RegIP	RegCountry
			Wallace	Female	2000-07-28	+7 (697) 431-26-36	Dynava.org	185.91.252.13	Turks and Caicos Islands
1	MarieKelley@Tambee.org	Theresa	Stanley	Female	1963-06-02	+7 (542) 546-35-60	Voomm.name	163.30.223.210	India
2	HeatherHawkins@Tambee.name	Sandra	Dunn	Female	1969-06-08	+7 (043) 017-06-26	Mynte.info	177.206.40.147	American Samoa
3	0Moore@Oyoloo.edu	Anna	Murphy	Male	1944-02-02	+7 (578) 860-35-45	Voomm.gov	126.87.33.71	Mayotte

- **Text qualifier — a character used to indicate the beginning and end of text data within a table cell. Qualifiers are required when the data within a cell contains characters that can be seen as delimiters or other special characters.**
- **Delimiter — a character that is used to separate data into separate fields (cells) in a table row. Delimiters are used to indicate the boundaries between columns in a table.**

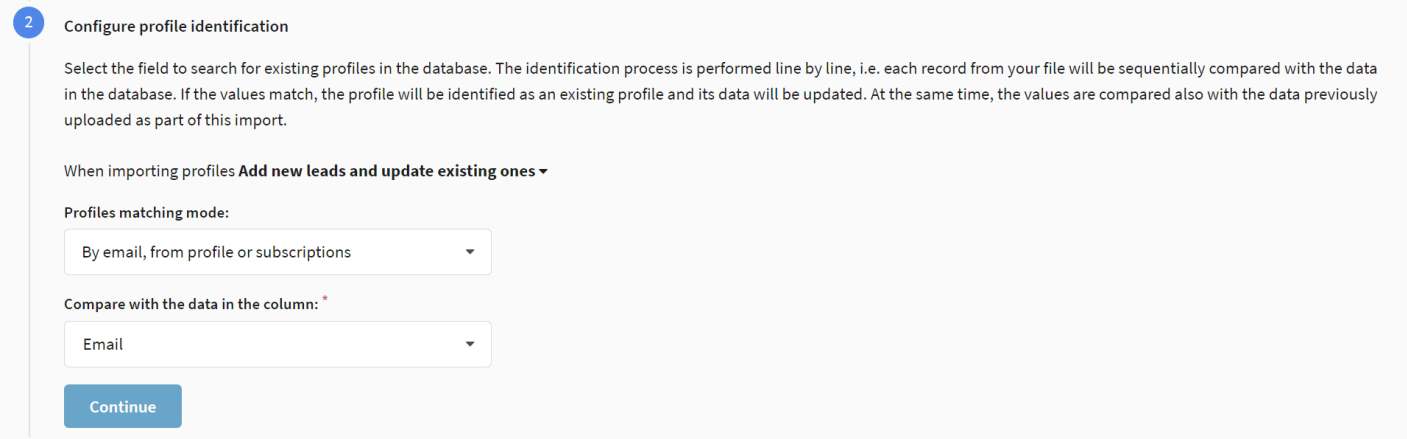
Profile identification

Select Import mode — it will depend on whether you add new records or update existing in the Altcraft profiles database:

- **Add new leads and update existing ones**
- **Add new leads, do not update existing ones**
- **Update only**

Profile matching mode defines what information from your file will be used in Altcraft for profiles search. This is necessary to update data without duplicating records. The following modes are available:

- **By email, from profile or subscriptions — search by email from database or subscriptions.**
- **By email from profile — search by database profile email.**
- **By phone, from profile or subscriptions — search by phone in database field or SMS subscriptions.**
- **By custom field — search by custom database field, like customer ID in company CRM.**
- **By subscription — serach subscription data.**
- **By profile ID - by unique profile ID (only when updating data).**



TIP

It is necessary to take into account the following peculiarities of matching:

- **The value of the search field should be unique for each profile in the database, otherwise the platform will not be able to unambiguously identify the profile being searched for**
- **Only one search mode should be used for one profile database. This is to avoid creating duplicate profiles in the database.**
- **If you are going to use custom field matching, you should contact the platform administrator to create an index for this field. Thanks to the index, searching for a profile in the database will take much less time.**

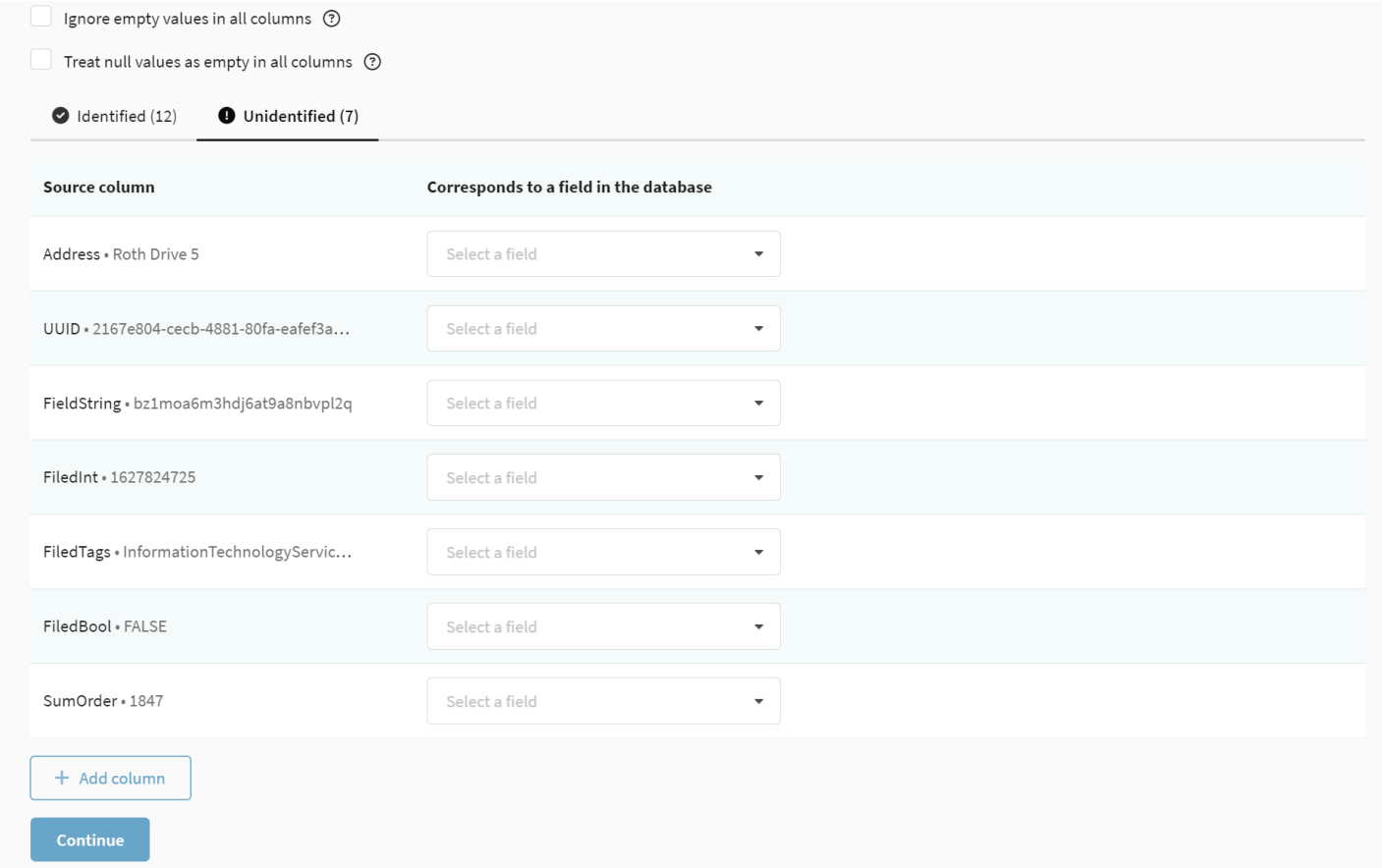
Fields customization

TIP

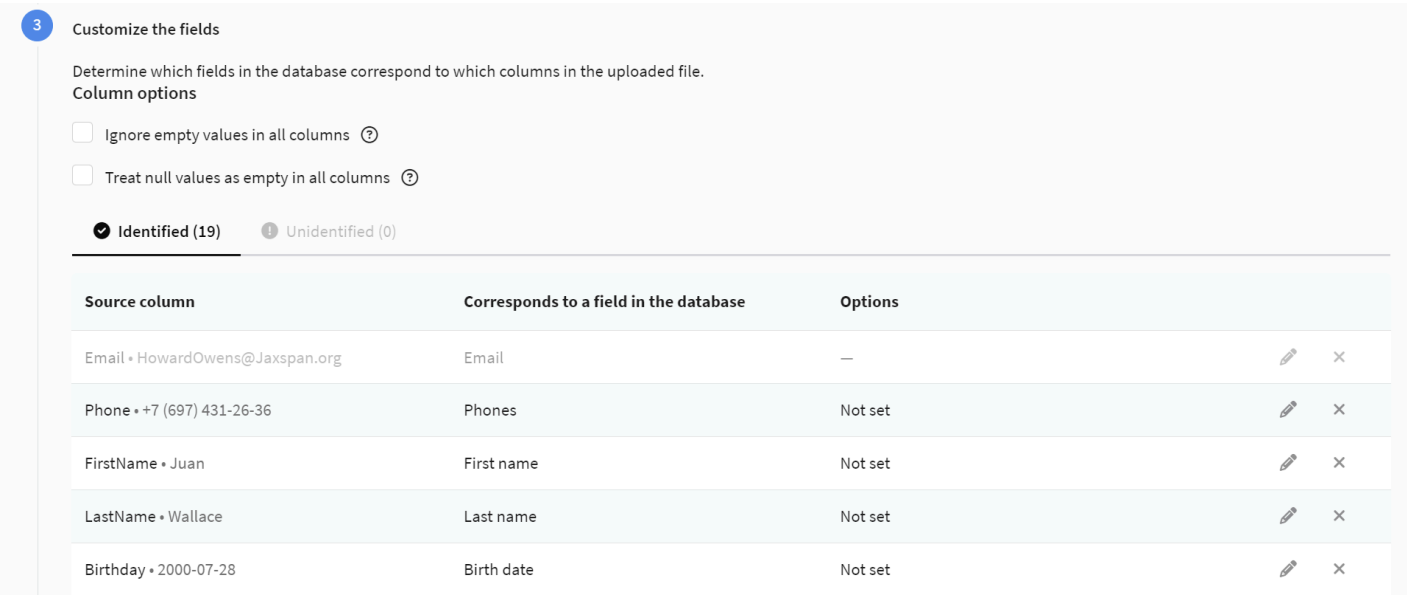
Pass profile contact information to Altcraft resources subscriptions (see the previous paragraph). You can store this data in database fields in order to search profiles. If customer identification is configured using CRM or other identifiers, email and phones may not be stored in appropriate fields.

You need to match columns in Altcraft database and columns in your file. Altcraft will detect the columns automatically if their names match the system ones, however, if your columns do not match the system ones, you can customize them manually by going to the “Unindentified” column:

- **Source column — select data source from your file. The firstrecord will be displayed for each column for data type proper selection.**
- **Corresponds to a field in the database — select customer profile field to store data. You can select standard and custom fields.**



After you select matches for the undefined columns, they will automatically be moved to the Identified column. If you have set an incorrect match, click on the cross to the right of the field you want, then it will be moved back to the Unindentified column.



The “Ignore empty in all columns” option is used when updating profile data that is already in the database. For example, the profile field “vendor” already contains the value “test.com”, but now you are importing a file where the “vendor” field for this client is empty. The platform will overwrite the field with an empty value. To avoid this, enable the “Ignore empty in all columns” option for the corresponding column. In this case, the value in the “vendor” field will remain the same.

If your file has unknown client data written with values such as `null`, `nil`, `none` or `undefined`, the platform can also proccess these values as empty. To do this, you need to enable both the "Treat null value as empty and Ignore empty values fields. In this case, the platform will not update the field.

Options

When configuring the import of data from a file, certain options are available for each field type that determine how the data will be processed and saved in the system:

Option	Description	Type of field
Ignore empty values	**Prevents overwriting profile data with empty values during import, preserving existing information if the corresponding fields in the imported file are empty.**	**All**

Option	Description	Type of field
Treat null values as empty	**The platform will treat values of type null, nil, none or undefined as empty if this option is enabled.**	**All**
Automatic register correction	**Converts letters to the selected register. For example: All letters are lowercase, Each word starts with an uppercase letter, or All letters are uppercase.**	**First name, Last name, Subscriptions(type of field - Line)**
Select the date format from the file	**Allows you to select the date format that is represented in the imported file.**	**Date**
Default value	**Preset value that is automatically filled in the field if there is no data for this field in the uploaded file.**	**Date**
Match the values of columns and fields	**If the values in your column do not correspond to the system values of the platform, you can specify their correspondence.**	**Gender(type of field - List)**
Increase the field value by the value from the column	**Adds a value from the imported file to an existing value in the specified database field, instead of replacing it.**	**Integer, Float**
Decrease the field value by the value from the column	**Subtracts the value from the imported file from the existing value in the specified database field, instead of replacing it.**	**Integer, Float**
Round values to a specified number of decimal places	**Automatically reduces a numeric value to a specified number of decimal places during data loading to ensure accuracy or format compliance. Rounding is applied to the data from the file, not to the total value that will be written to the profile field.**	**Float**
Tag delimiter	**A character or set of characters used to separate multiple tags (labels) in a single file field so that the platform can properly recognize and import them as separate values.**	**Tags**
Add or update data by object field	**An option that either creates a new value for the specified object field if it does not exist, or updates an existing value in the database based on data from the imported file. Please note that this update will work correctly only if the objects in the array have the same structure ([{key":“value_1”},{“key”:“value_2”}]).**	
Tags, Array of objects		
Delete data by object field	**An option that removes objects with the same value in the specified field.**	**Array of objects**
Import to property	**Writes a value from a file column to the specified property of the object.**	**Object**

Field import specifics

When importing data into the system, the platform expects specific value formats for each field type. All valid formats are described in this article. Make sure the data in your source matches these requirements, otherwise the platform will not be able to process it correctly.

For Object and Array of Objects fields, the platform expects the value to be a string containing valid JSON. To set this value, you need to first convert the JSON object into a string and then escape it according to the CSV rules described in RFC 4180 (double quotes " are replaced with “”).

Example: `"[{""field1"": 123, ""field2"": {""subfield"": false}, ""field3"": ""abc""}]"`

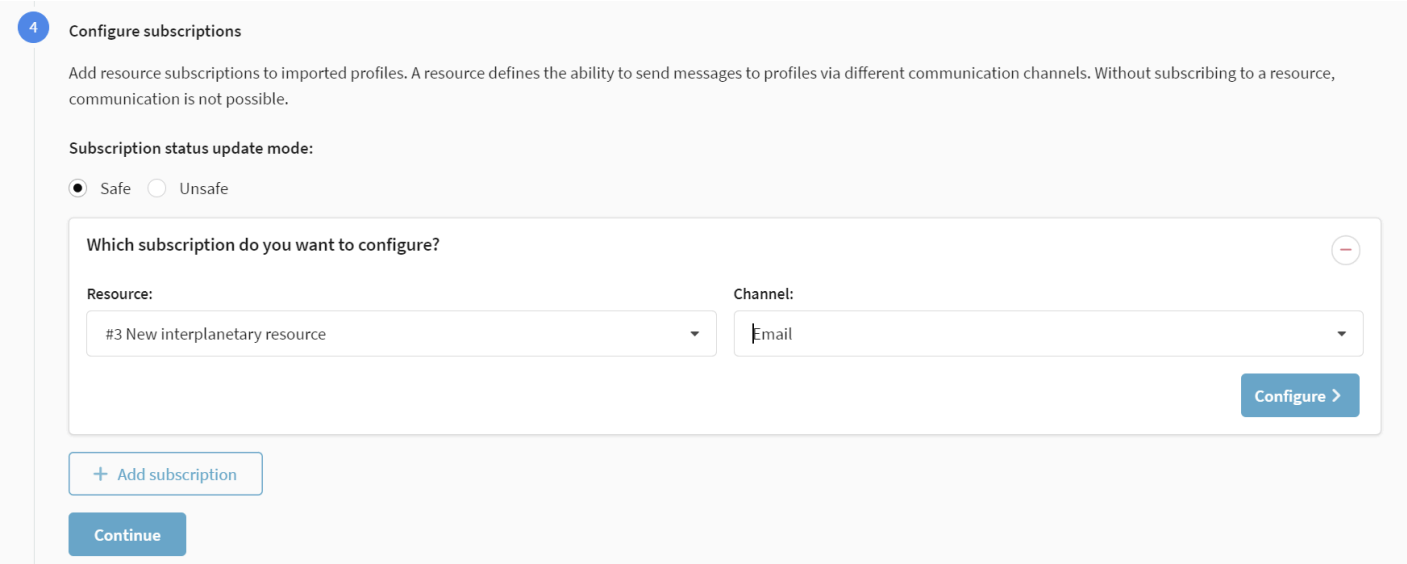
For Enum fields, you can import either the values themselves or their indexes. For example, if you have created a enum list in the database with options “tariff_1”, “tariff_2”, “tariff_3”, you can specify the same values in the source column or use the indexes: *1 — “tariff_1”, 2 — “tariff_2”, 3 — “tariff_3”, 0 — “Not selected”* (default value).

Please note that values consisting only of digits cannot be imported directly — the platform will treat them as indexes. For example, if the list contains the values 100 and 200, you need to import the corresponding indexes: 1 — 100, 2 — 200.

Subscriptions configuration

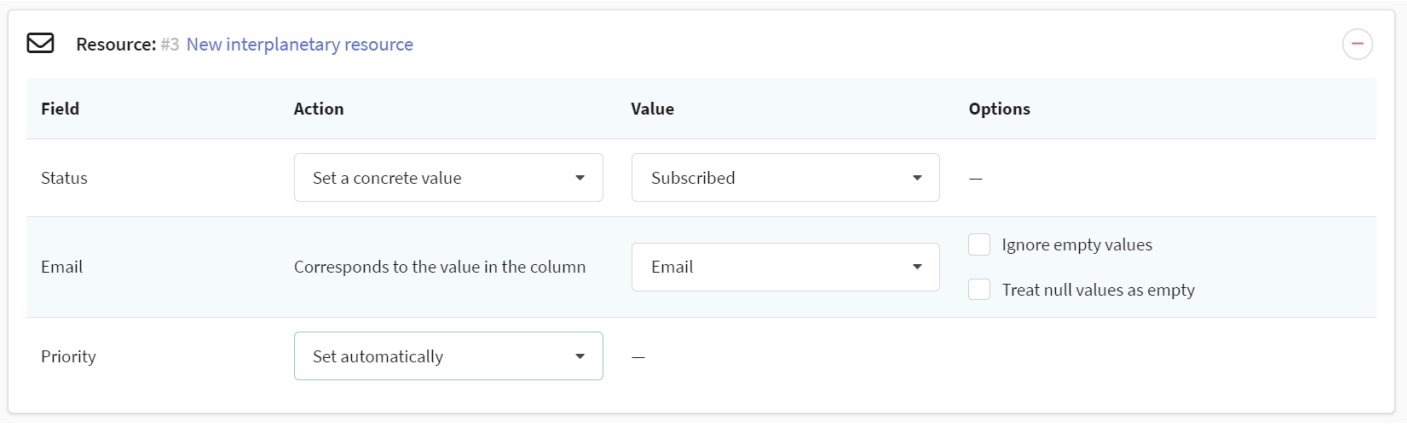
When importing from a file, subscribe your clients to the necessary resources. A resource defines the possibility to send messages to profiles via different communication channels. Without subscribing to the resource, communication is impossible. In the Customize subscriptions block, click the + Add subscription button.

You have two modes of updating the status of your subscriptions: safe and unsafe. If you choose unsafe mode, subscriptions of profiles with the following statuses will be updated: Unsubscribed, Complainer, Hardbounced, Not valid. If a profile already has a subscription to the selected channel in the resource with one of these statuses, after import its status will be changed according to the specified settings.

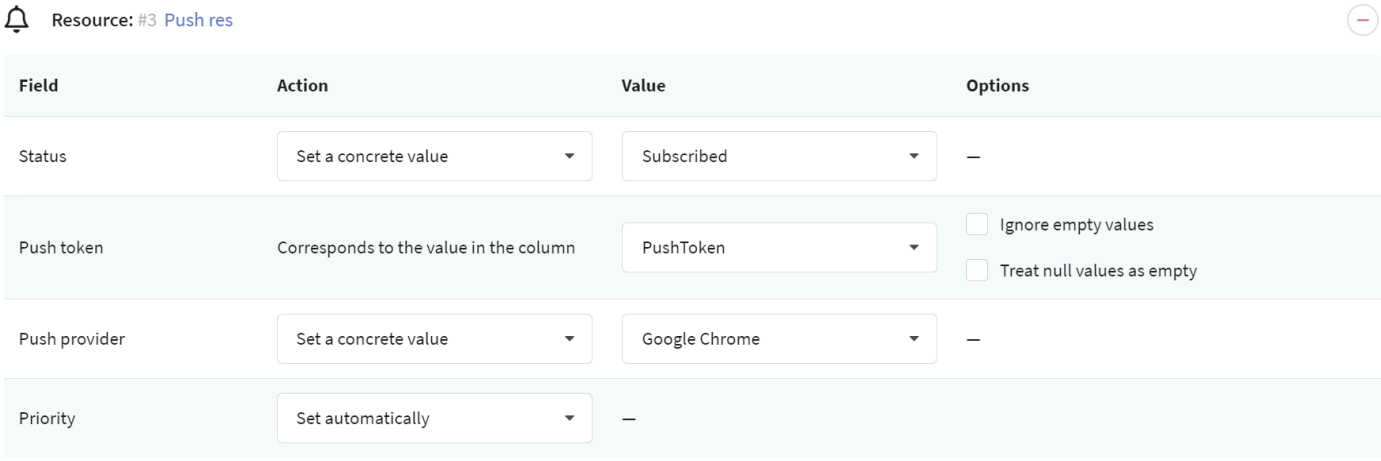


For each subscription, select Resource, Communication channel, Subscription status and set the field matches.

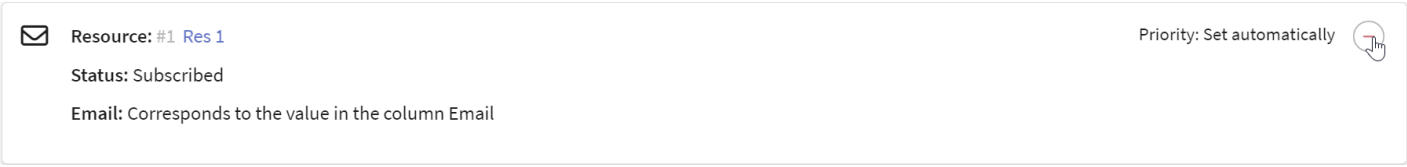
Each profile is limited to no more than 20 subscriptions per resource.



For subscriptions to push notifications, additionally specify push service provider and push token:

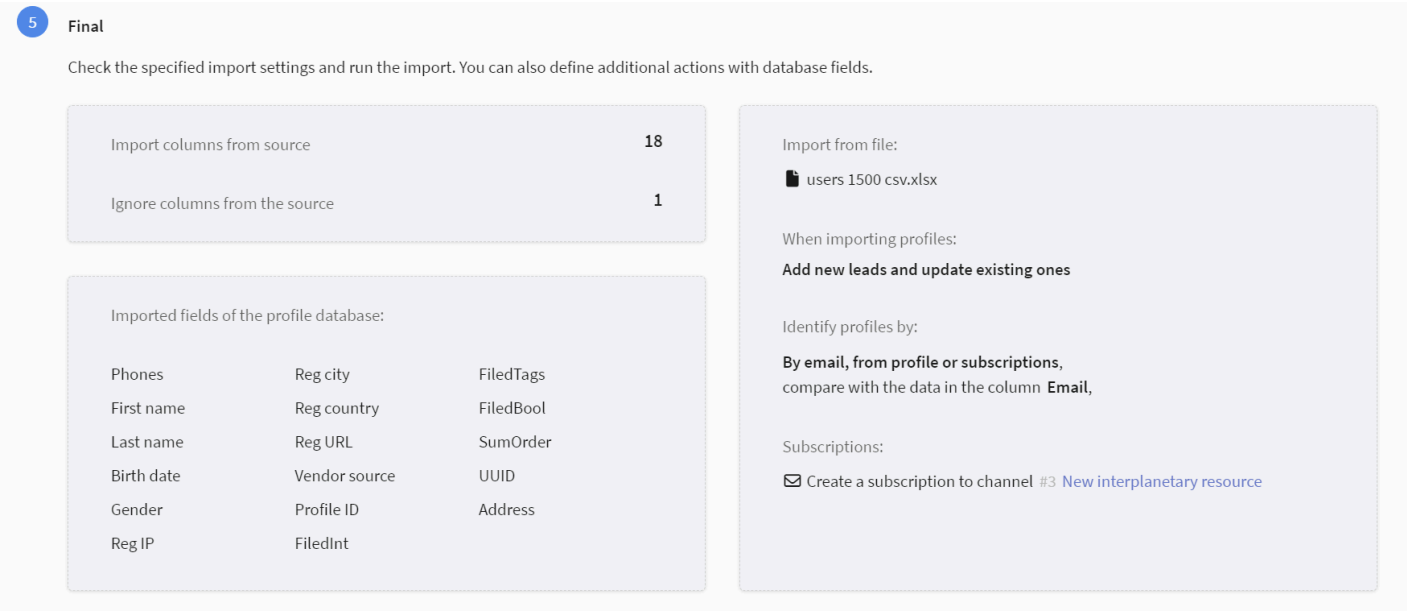


Once a subscription is set up, you can edit it by clicking on it, or delete it by clicking on the red minus icon.



Final

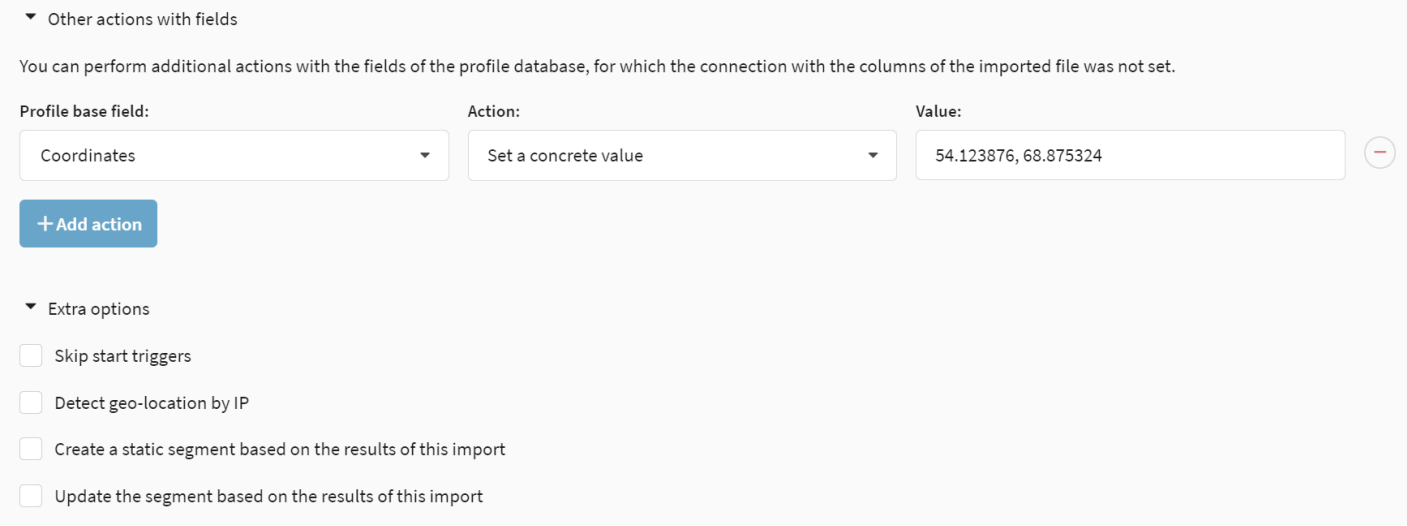
In the final step of the import you can check your settings and run the import, as well as define additional actions with database fields.



In the Other actions with fields block, you can do additional actions on the profile database fields that have not been matched.

Furthermore, additional functions are included in the Extra options block:

- **Skip start triggers — disable on-import triggers and scenarios for this operation.**
- **Detect geo-location by IP — When you submit data to the Reg IP or IP fields in IPv4 format, Altcraft will automatically detect geolocation and you will be able to select profile fields such as Coordinates, Country, City - or Country Reg. and City Reg. to fill in, and profile location on map widget will also be noted.**
- **Create a static segment from the results of this import — when importing profiles static segment is automatically created.**
- **Update a segment from the results of this import — when importing profiles, the segment is automatically updated.**



Transferring status on import

When customizing fields and subscriptions, you can pass the profile global status and subscription status, respectively. The status in the file column can be specified in either text or numeric format.

Possible field values:

- **0, “subscribed” - available for communication via all communication channels**
- **1, “unsubscribed” - unsubscribed from receiving messages on all channels**
- **2, “complained” - complained about spam, communication stopped**
- **3, “hardbounced” - ISP has completely refused to accept messages to this address**
- **4, “unconfirmed” - subscriber registered but did not confirm consent to receive mailings**
- **5, “suspended” - subscription is suspended for some time**
- **6, “invalid” - excluded from marketing mailings for any other reasons**

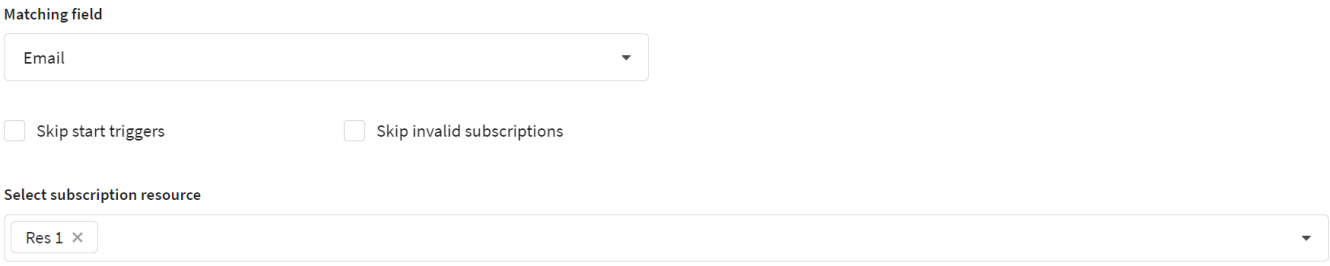
Import via API

API is used for integration with external systems — sites, applications, customer data management (CRM) services, data collection forms. In the API request constructor you can set certain parameters for importing data and pass them to your developers.

In the “Matching field”, select the data that will be used to search for existing profiles:

- **Email - by profile field Email.**
- **Phone - by profile field Telephones.**
- **Profile ID - by unique profile ID (only when updating data).**
- **Custom field - by custom database field. For example, customer ID in a CRM company.**

Also select resource to subscribe.



Check the Skip start triggers box if you do not want to run scenarios and trigger mailings configured to import into the selected database.

Check the Skip invalid subscriptions box if you do not want to add a profile subscription if invalid data is present in the query.

Below you will see a template of the API request:

Template v1.1 | cURL v1.1

POST URL

https://nikita-trukhnin.dev.altkraft.com/api/v1.1/profiles/import

HTTP Headers

Token name	Value
Content-Type	application/json

Query body

```
{
    "token": "<Your previously configured token>",
    "data": {
        "email": String ,
        "phones": phones ,
        "_fname": String ,
        "_lname": String ,
```

TIP

- **Use API query to import or update a single profile.**
- **Other import API queries are described in the Profiles section.**
- **Managing profile subscriptions is described in the Subscriptions section.**

Scheduled Import from External SQL Databases

TIP

Altcraft can automatically collect data to import new customers profiles and update existing profiles from ClickHouse, MySQL, MS SQL, PostgreSQL and Oracle RDBMS.

Instructions for configuring scheduled import are in a separate article.

Common Errors When Importing Profiles

When importing profiles from a file, the platform validates data before writing it to the database. If a value does not match the field type, the profile may be rejected, and an error will appear in the import report.

Most often, errors occur when the data in the file does not match the field type in the database: for example, a date value is provided in a different format, text appears in a numeric field, or a list field contains a value that is not among the available options.

Error `incorrect ip`

This error occurs if a value that is not a valid IPv4 address is provided in an IP type field.

Examples of invalid values:

```
999.999.999.999
127.0.0.1.1
abc.def.ghi.jkl
```

The error may also appear if an empty value is provided in an IP field that expects a valid address.

How to fix:

Use an IPv4 address: four numbers from 0 to 255, separated by dots.

Valid examples:

```
127.0.0.1
95.165.10.20
```

If the IP address is unknown, leave the column unmapped from the IP field or configure empty value handling during import.

Error `Incorrect phone`

This error occurs if the phone number fails format validation.

Examples of invalid values:

```
12345
phone
+000000000000
+799912345678901234567890
```

How to fix:

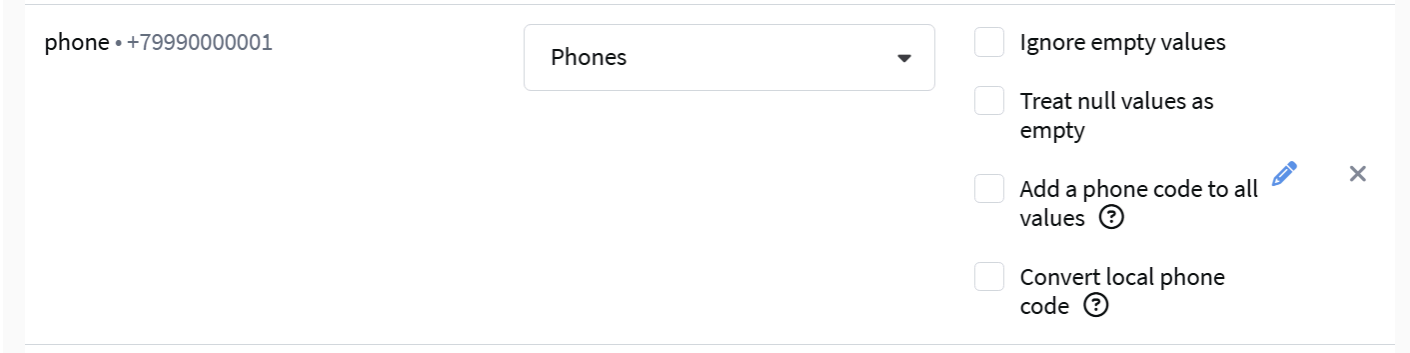
Provide phone numbers in international format or configure phone number transformation during import.

Valid example of international format:

```
+79991234567
```

If the numbers in the file are recorded without a country code or in local format, use the field settings during import:

- **Add phone code to all values — adds the country code to the imported numbers.**
- **Convert local phone code — replaces the local code with the international one.**



For example, a Russian number in local format can be converted to the `+7` format using the local code conversion setting.

If the number contains text, too few or too many digits, fix the value in the file before import.

Error `incorrect date`

This error occurs if the date does not match the format selected during import configuration, or if the date is impossible.

Examples of invalid values:

```
2024/99/99
0001-01-01 00:00:00 +0000 UTC
31.13.2024
```

How to fix:

Check the date format in the field settings during import and bring the values in the file to this format.

Examples:

Format	Valid Value
`%d-%m-%Y`	`31-12-2024`
`%Y-%m-%d`	`2024-12-31`
RFC 3339	`2024-12-31T00:00:00.000Z`

Do not pass service zero dates if they should not be written to the profile. It is better to replace such values with empty ones and enable empty value handling.

Error `incorrect integer value`

This error occurs if a value that cannot be written as an integer is provided in an Integer type field.

Examples of invalid values:

```
one
NaN
Infinity
1,25
99999999999999999999999999999999999999999999
```

How to fix:

Use only integer numeric values without letters, spaces, fractional parts, or extra characters.

Valid examples:

```
0
1
25
-5
```

If the value in the source may be empty or unknown, configure empty value handling or do not map such a column to a numeric field.

Error `incorrect float value`

This error occurs if a value that cannot be recognized as a floating-point number is provided in a Float type field.

Examples of invalid values:

```
money
∞
12..34
--0.5
1,234.56
```

How to fix:

Use numeric values without currencies, text labels, or extra characters. Use a dot for the fractional part.

Valid examples:

```
1234.56
0.5
-10.25
```

If the source uses a different decimal separator, convert the values to dot format before import.

Error `incorrect enum field: not found value`

This error occurs if a value that is not among the enum options is provided in an Enum type field.

Examples of invalid values:

```
unknown
male/female
?
unexisting_plan
999
-1
```

This error can appear in both system fields with predefined values and additional Enum type fields.

How to fix:

Check the list options in the field settings and provide one of the existing values. The value must match the list option exactly, including case, spaces, and characters.

For example, if the field has the following options:

```
basic
standard
premium
```

then the CSV should contain one of these values:

```
standard
```

You can also pass the option index for a list. Indexing starts from `1`: first option is `1`, second is `2`, third is `3`.

If the list value does not need to be updated, leave the cell empty and enable Ignore empty values and Treat null values as empty for this column. In this case, an empty value will not overwrite the current field value for an existing profile.

For a new profile with an empty value, the default value will be used if it is set in the field settings. For example, for an Enum type field, this value might be “Not selected”.

> INFO
>
> **If the list options consist only of digits, the platform may interpret them as indices rather than text values. To avoid confusion, it is better to use text values, such as `plan_100` or `tariff_200`.**

JSON Errors in “Object” Type Fields

For custom fields of type Object, the value must be valid JSON.

In the import report, such errors may look different:

```
invalid character 'b' looking for beginning of object key string
invalid character 'T' looking for beginning of value
invalid character 'u' looking for beginning of value
unexpected end of JSON input
invalid character 'q' after object key:value pair
```

Examples of invalid values:

Value	Why It’s Invalid
`{bad json}`	**An object key must be a string in double quotes.**
`{"vip": True}`	**In JSON, boolean values are written in lowercase: `true` or `false`.**
`{"field": undefined}`	**JSON has no `undefined` value. If the value is unknown, use `null`.**
`{"city": "Moscow",}`	**There should be no comma after the last property of an object.**
`{"nested": {"x": 1}`	**The outer curly brace is not closed.**
`{"quote":"bad " quote"}`	**Quotes inside a string must be escaped.**

How to fix:

Provide valid JSON:

- **keys and string values must be in double quotes;**
- **boolean values must be passed as `true` or `false`;**
- **empty values must be passed as `null`;**
- **there should be no trailing comma at the end of an object;**
- **all curly and square brackets must be closed.**

Valid example:

```
{"vip": true, "city": "Moscow"}
```

Error `wrong data type for custom field ...`

This error occurs if an inappropriate data type is passed to a custom field. For example, a string, single object, number, or invalid JSON is passed to an Array of objects type field.

Examples of invalid values:

```
{bad json}
[1,2,]
"unterminated
{"field": undefined}
```

How to fix:

For an Array of objects type field, provide an array of objects with the same structure.

Valid example:

```
[
  {"id": 1, "sum": 100},
  {"id": 2, "sum": 250}
]
```

In CSV, such a value should be written as a string with escaped double quotes:

```
"[{""id"": 1, ""sum"": 100}, {""id"": 2, ""sum"": 250}]"
```

If the first object contains fields `id` and `sum`, the remaining objects in this array must also contain the same fields.

Errors Due to Empty Values

An empty value is not always considered valid. An error may appear if an empty cell falls into a field where an IP, date, number, list, or other strictly typed value is expected.

Examples of errors in the report:

```
incorrect ip
incorrect date
incorrect float value
incorrect integer value
incorrect enum field: not found value
```

How to fix:

If an empty value should not update the profile, enable Ignore empty values for the column.

If unknown values in the source are recorded as `null`, `nil`, `none`, or `undefined`, enable null value handling as empty and ignore empty values.

How to Quickly Check a File Before Import

Before uploading the file, check:

1. **The first row contains column names, if this setting is enabled during import.**
2. **The file encoding matches the selected import encoding.**
3. **The delimiter and text qualifier are defined correctly.**
4. **Dates are written in the selected format.**
5. **Phone numbers are written in international format.**
6. **IP addresses are written in IPv4 format.**
7. **Numeric fields do not contain text, currencies, or special characters.**

8. **List values exist in the settings of the corresponding field.**
9. **JSON in Object and Array of objects fields is valid and properly escaped for CSV.**
10. **Columns are mapped to fields of the appropriate type.**

What to Do If the Error Persists

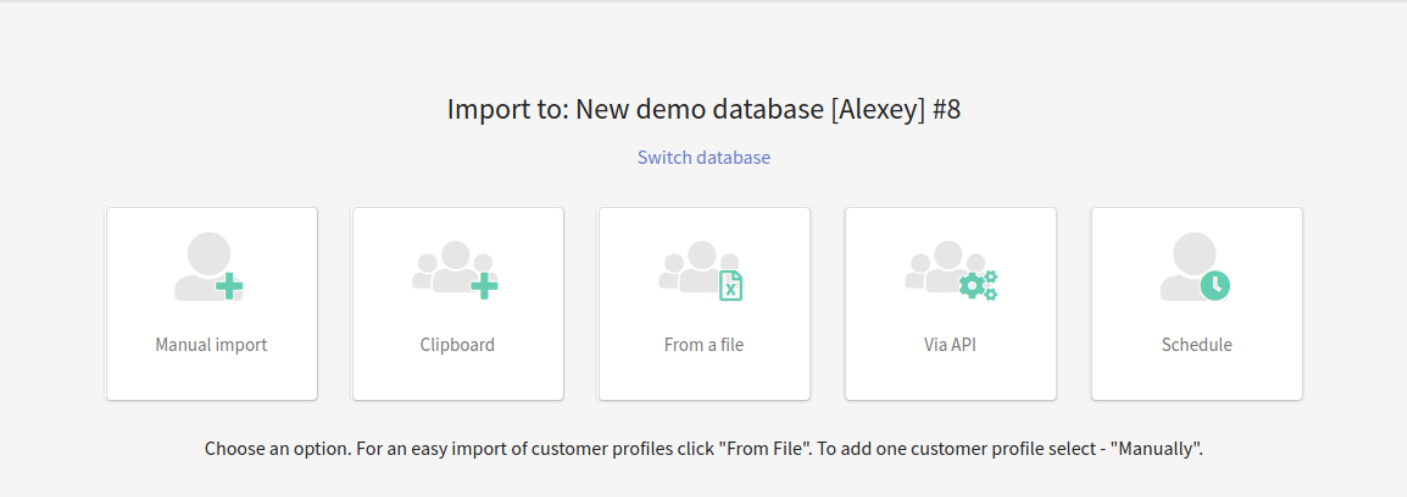
If the error repeats after fixing the file:

1. **Check which field the column is mapped to.**
2. **Make sure the field type in the database is suitable for the data from the file.**
3. **Check the date format, tag delimiter, and empty value handling settings.**
4. **Check the options for Enum type fields.**
5. **Validate the JSON in a validator and then escape it for CSV.**
6. **Download the import error report and check the `error_field` column: it indicates the field that caused the profile to not be imported.**

Scheduled customer data import

Customer data import can be automated using scheduled Database import. The import module takes data from external data tables. To configure automatic data collection, you need to configure SQL connector in admin panel and bind it to your account. In your account, to proceed to import creation, go to Data —> Database Imports.

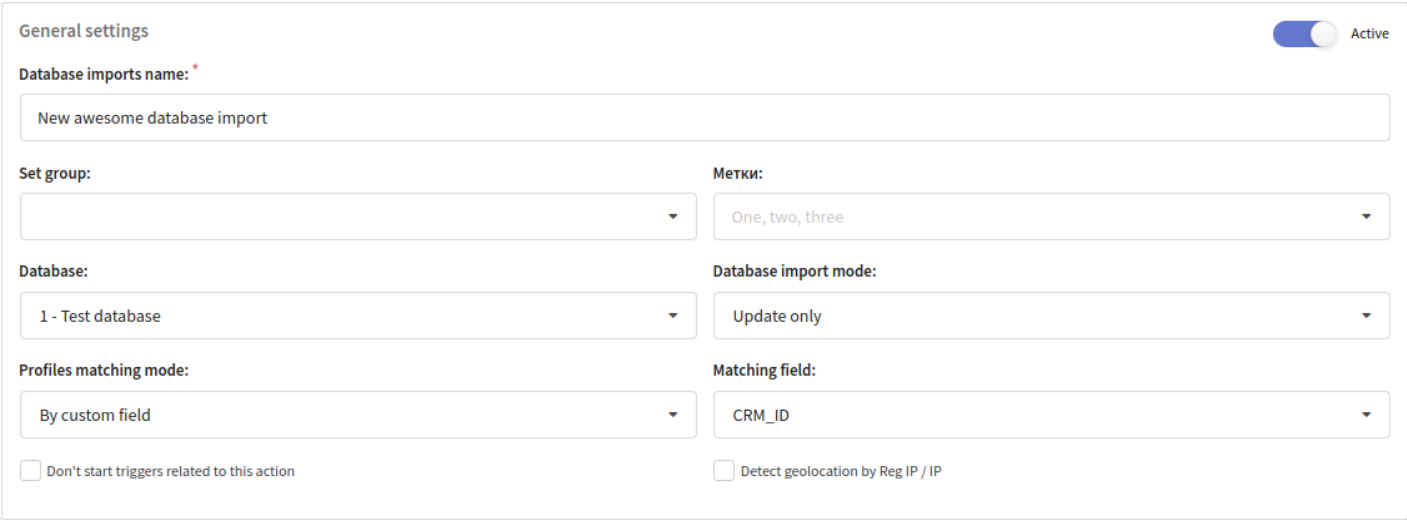
You can also go to this section from the list of databases by choosing “Import customer profiles” in the context menu on the right:



General settings

In general settings of import, you can set import name, assign access groups and labels and set how the import will process data in the platform account:

1. **Database — import will add or update fields in the selected database of customer profiles. The set of available fields for storing information depends on the database.**
2. **Database import mode — import can work in three modes:**
   - **Add new leads and update existing ones**
   - **Add new leads, do not update existing ones**
   - **Update only**
3. **Profiles matching mode — You can search by contact data, subscription data (including Push), by profile ID to update existing profiles — and by custom database field.**
4. **Matching field — if you search for profiles by custom field of the database, you need to select it in this menu.**



For scheduled import IP location is also available. If you have configured trigger mailings to import or update profiles, you can restrict their triggering for scheduled import.

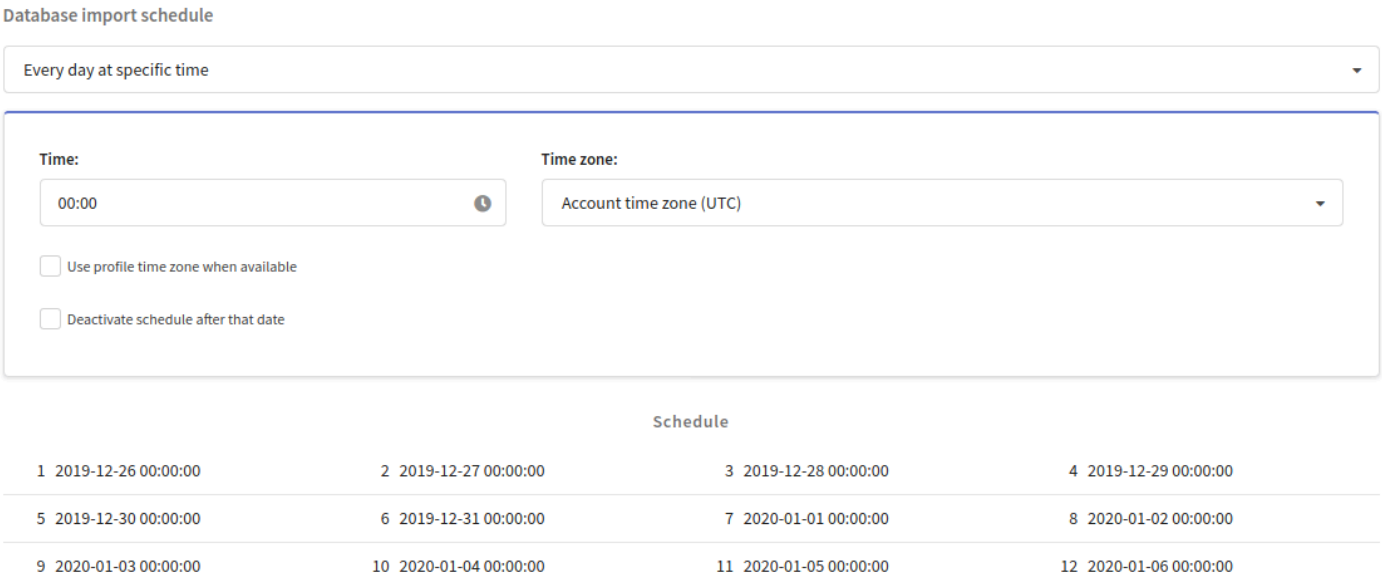
In the main settings there is an option to instantly start import in addition to the schedule. If this item is checked, after clicking the Apply or Save button, the import will be launched.

Database import schedule

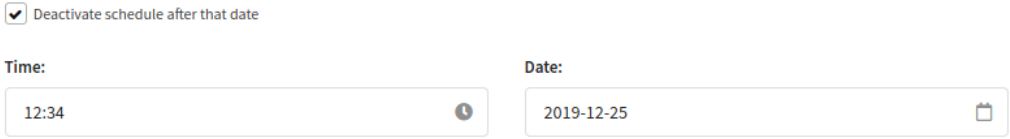
You can flexibly configure schedule for regular import. Choose schedule plan from the menu and configure it depending on your import purpose.

TIP

Mailing schedule is described in detail on [this page](#).

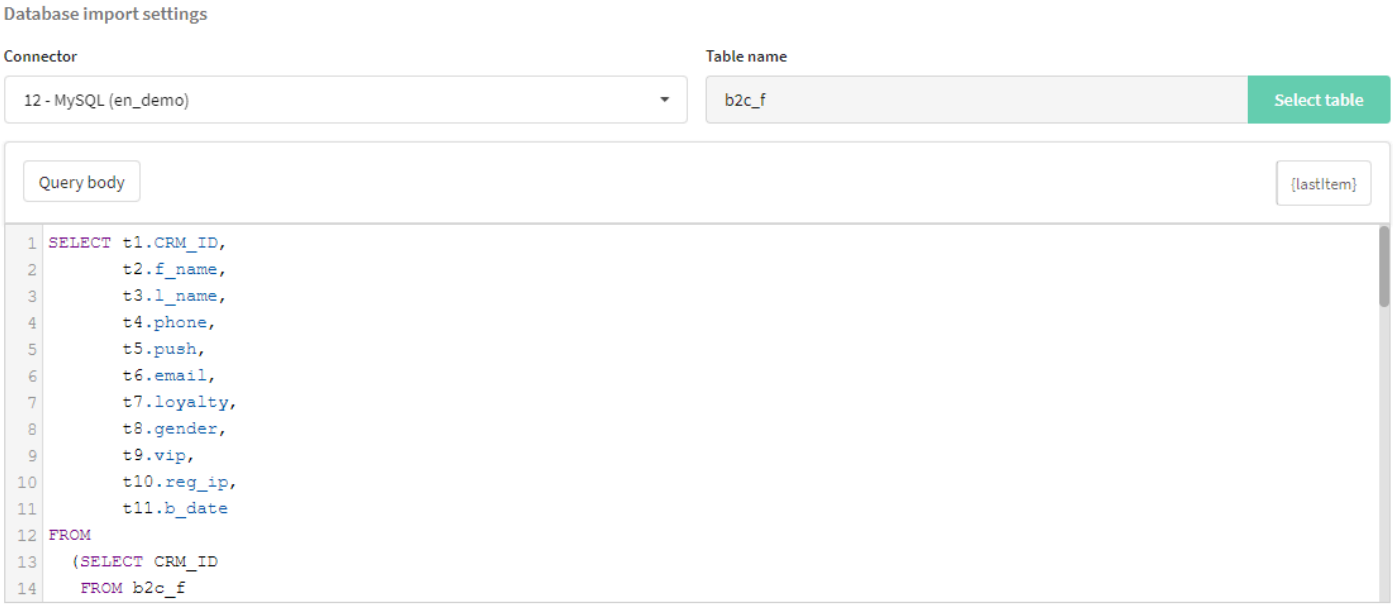


For each schedule plan, you can set the time zone for the calculation. You can also set import deactivation date:



Database import settings

Choose the necessary database connector (configured in administrative panel) in database import settings and the data table to take data from:



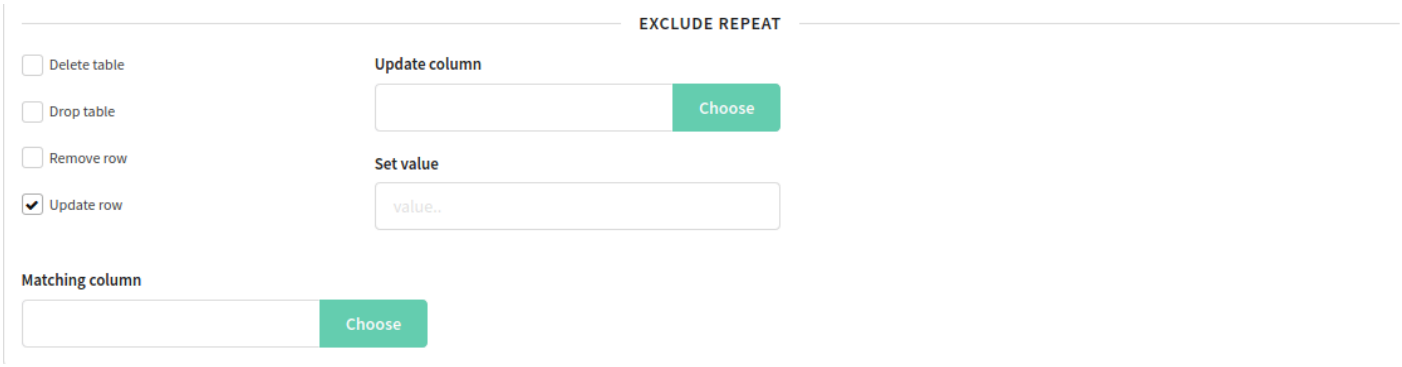
Create Query body depending on DB management system you use: MySQL, PostgreSQL, MSSQL, Oracle, Hive or ClickHouse. SELECT operator is necessary on your queries. There are no any other restrictions.

Prevent duplication

These functions are available to prevent repetative data update:

- **Delete table — deletes all records in data table**
- **Drop table — deletes table from database**
- **Remove row — if the value of the specified column in the record from the table and in the profile in the database coincides, such record will be deleted.**

- **Update row — updates values in selected column for imported records. It identifies records by Matching column (unique for every customer profile and differs from profile search column) and updates data in Update column.**



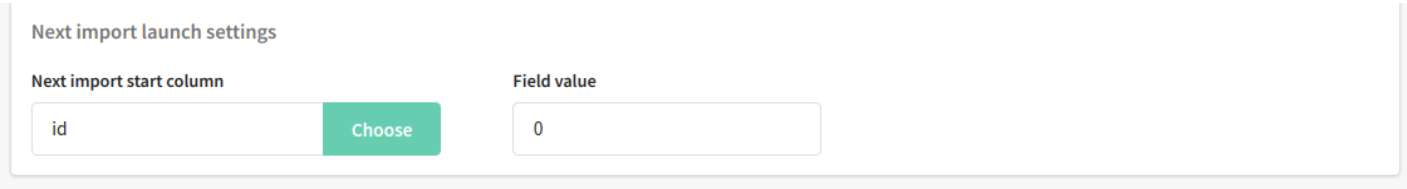
Splitting import into parts

When you import big amount of records, you can divide this process, to distribute loading. To do this, you need to add `{lastItem}` variable into Query body for the column with unique ID. It is recommended to sort IDs when making a query.

```
Query example: SELECT * from TestTable where id>{lastItem} LIMIT 10
```

Next, you need to specify from which row in the table you want to start the first import. To do this, select the column with the ID in Next import start column and specify Field value.

An example of settings for starting import from the first row (if the data in the `id` column is sorted in ascending order):

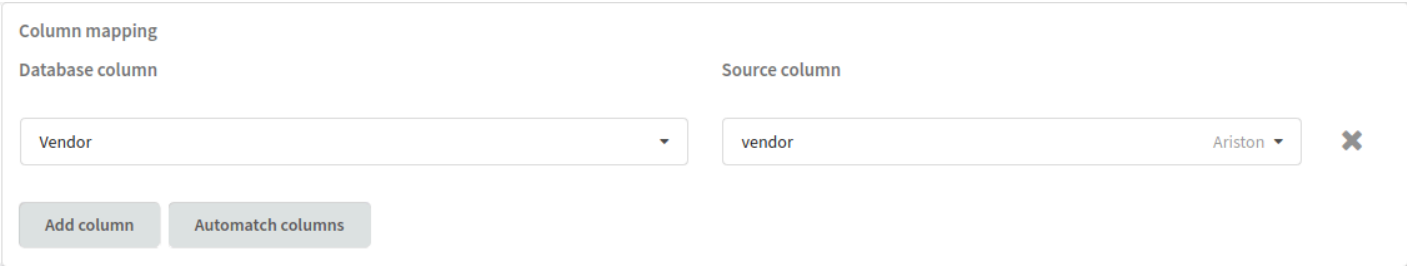


After starting a regular import, Field value will be automatically updated — it will be filled with the last ID after the previous run.

Column mapping

In this block, you need to determine the correspondence of profile fields in the Altcraft database and columns of the external SQL database. After clicking on the Add column button on the left of the drop-down menu, select Altcraft profile field for updating or recording data.Dur

When you choose column of external SQL database, its first value will be shown to the right as gray text, like in example below:



Field import specifics

When importing data into the system, the platform expects specific value formats for each field type. All valid formats are described in [this article](). Make sure the data in your source matches these requirements, otherwise the platform will not be able to process it correctly.

For Object and Array of Objects fields, the platform expects the value to be a string containing valid JSON. To set this value, you need to first convert the JSON object into a string and then escape it according to the CSV rules described in [RFC 4180]() (double quotes " are replaced with “”).

Array of objects example:

```
[
    {
        "field1": 123,
        "field2": {"subfield": false},
        "field3": "abc"
    }
]
```

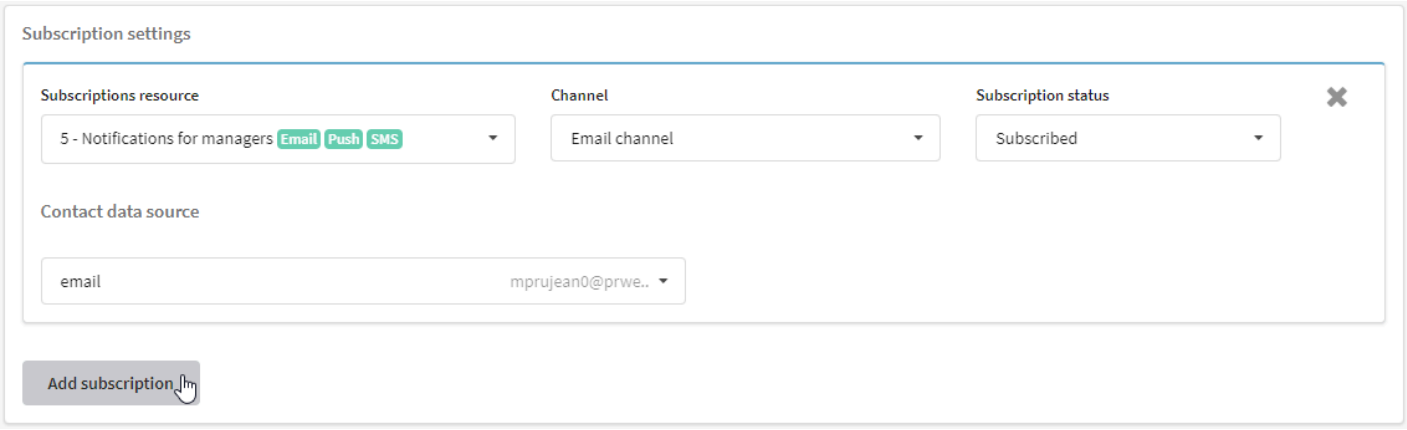
Expected value: `"[{""field1"": 123, ""field2"": {""subfield"": false}, ""field3"": ""abc""}]"`

For Enum fields, you can import either the values themselves or their indexes. For example, if you have created a enum list in the database with options “tariff_1”, “tariff_2”, “tariff_3”, you can specify the same values in the source column or use the indexes: *1 — “tariff_1”*, *2 — “tariff_2”*, *3 — “tariff_3”*, *0 — “Not selected”* (default value).

Please note that values consisting only of digits cannot be imported directly — the platform will treat them as indexes. For example, if the list contains the values 100 and 200, you need to import the corresponding indexes: 1 — 100, 2 — 200.

Subscription settings

During import, you can automatically subscribe customers to certain resources - through one or more channels. For every subscription you need to select a resource, communication channel and select column in the data table from which contact data for subscriptions will be imported.

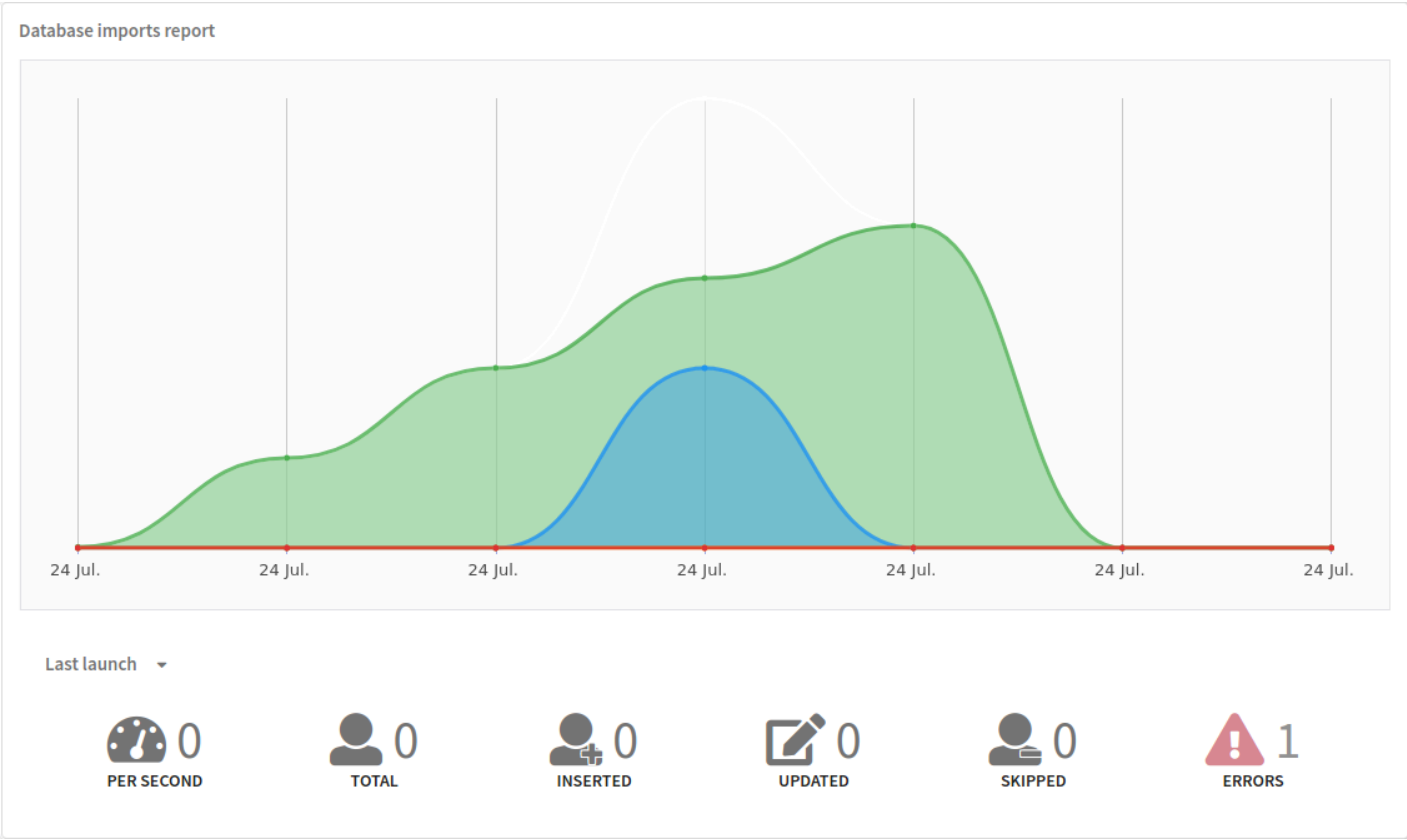


You can set status for every subscription:

- **During import, you can give "Subscribed" status for Single opt-in, or "Unconfirmed" status for Double opt-in logics.**
- **During data update, you can change subscription status depending on user actions, like suspending or unsubscribing**

Database imports report

In this block, you can see information about last 10 imports. If any errors occurred during last import, you can load report about them.

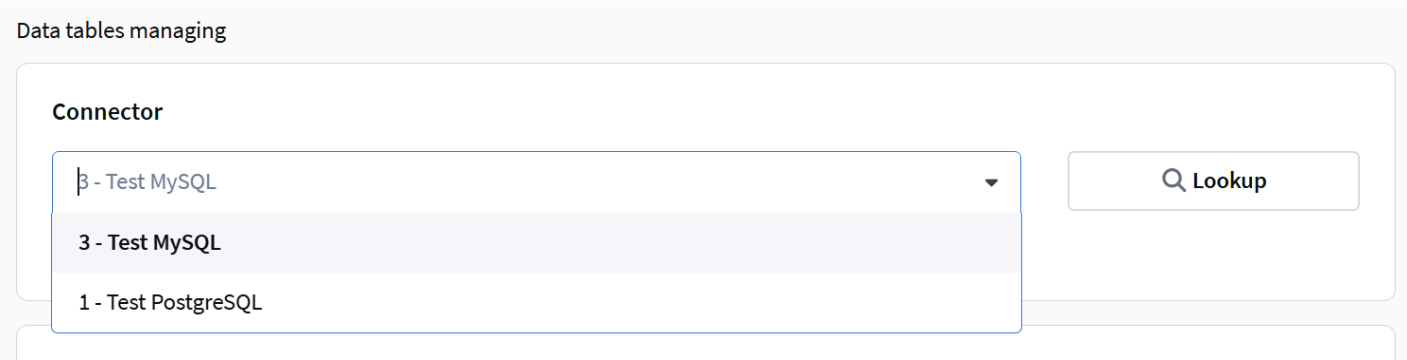


Managing Data Tables

Data Tables — an interface for working with external SQL databases. You connect to tables, write a query, and can immediately view the result or save it for further use.

Supported database management systems:

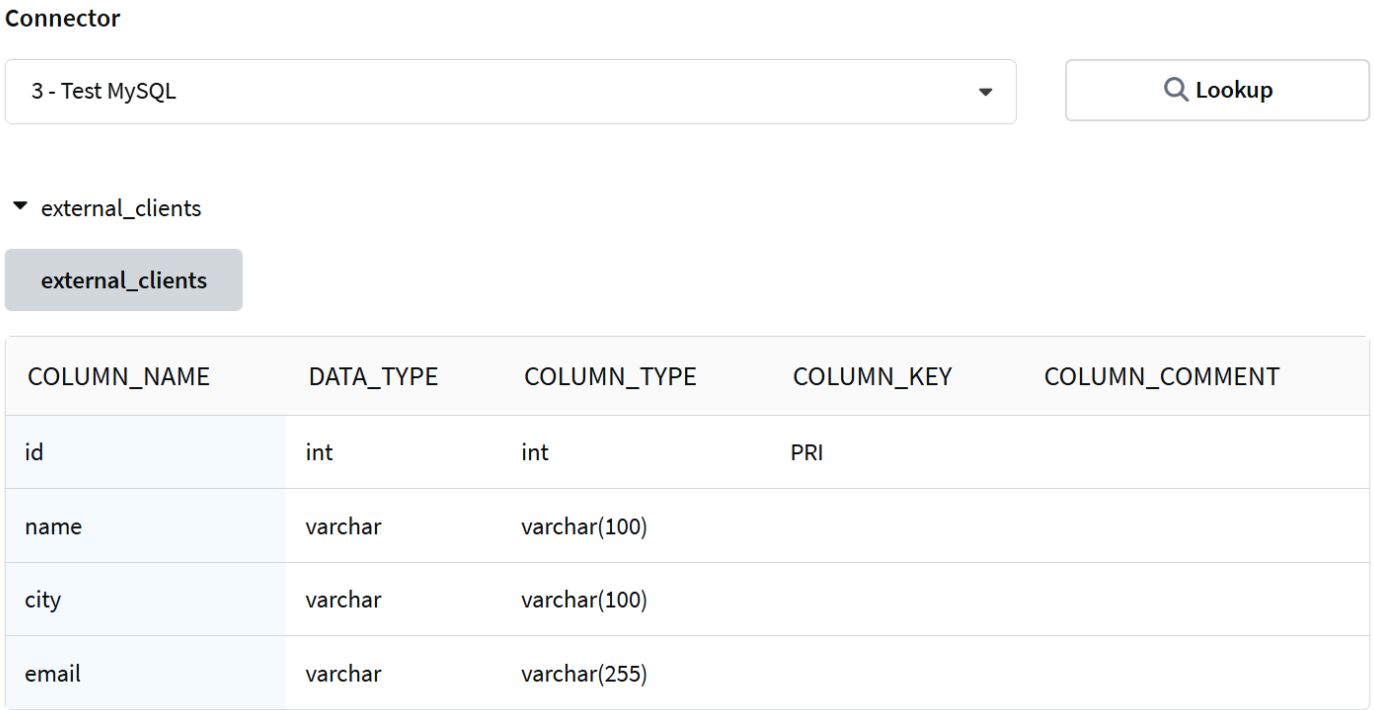
- **MySQL;**
- **PostgreSQL;**
- **MSSQL;**
- **Oracle;**
- **ClickHouse.**



Connector

A connector is a connection to an external database. Connectors are created and configured by the platform administrator.

To work with data tables, you need to select a connector from the dropdown list. After selecting a connector, you can click the Lookup button — the platform will display a list of available tables in this database.



You can learn more about connecting external databases in this article.

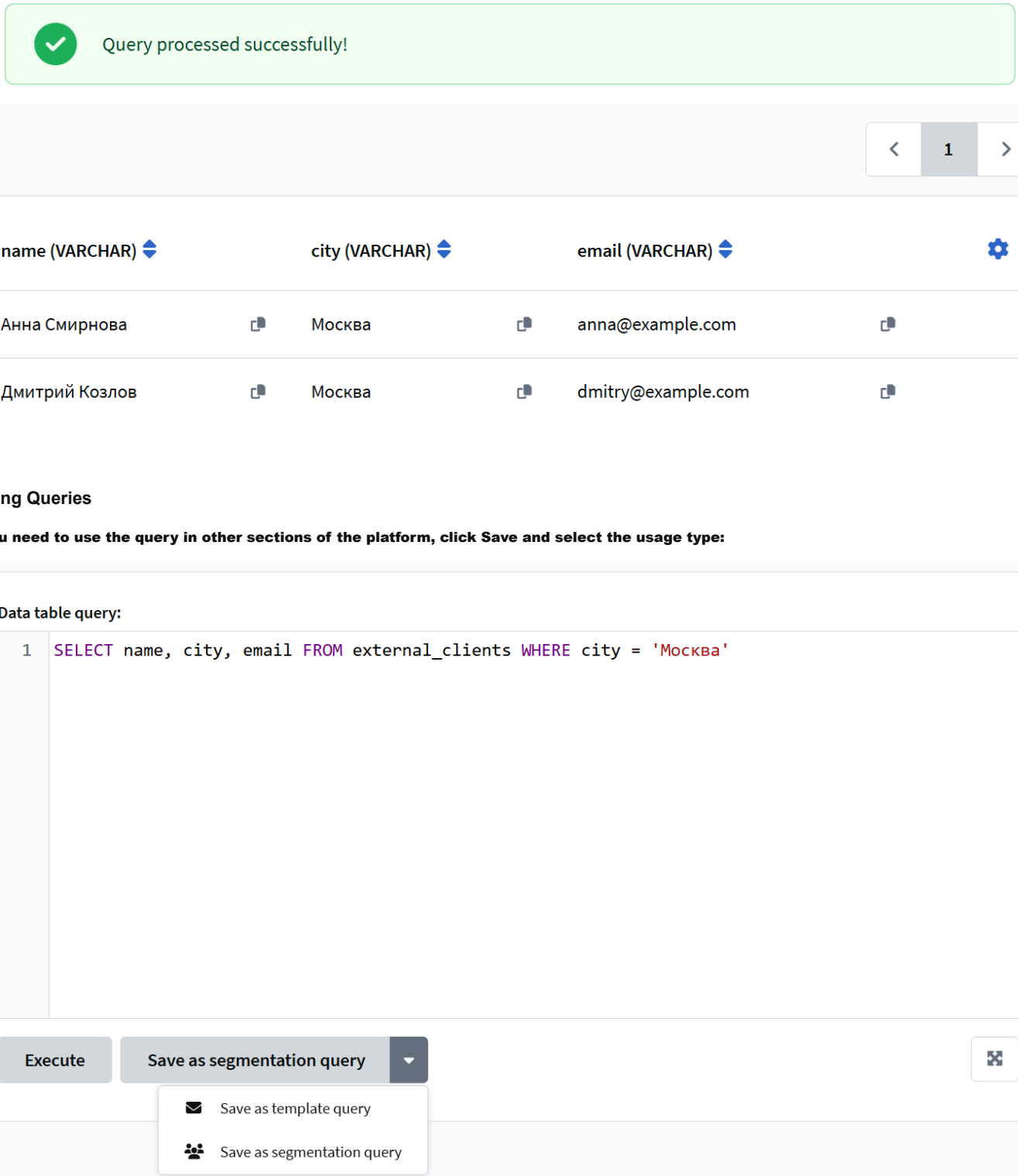
Table Queries

To send a query to an external table, you need to write the query in the “Data Table Query” field.

Query example:

```
SELECT id, email, name FROM external_clients WHERE city = 'Moscow'
```

To simply view the result and ensure the query works, click Execute. The result will appear as a table below.



Saving Queries

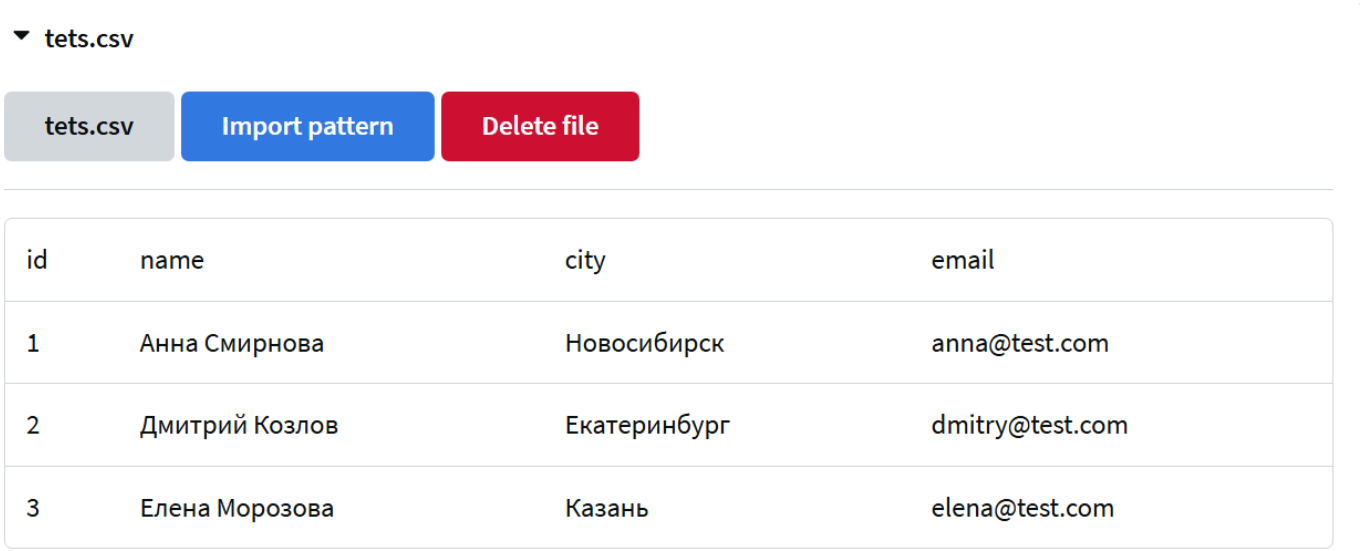
If you need to use the query in other sections of the platform, click Save and select the usage type:

- **For Segments — the query will be available in segmentation for filtering subscribers based on data from the external database.**
- **For Templates — the query will be available in dynamic content for inserting values into message templates.**

Working with CSV and TXT Files

The file upload area is available for all connector types, but works correctly only for MySQL connectors.

To upload a file, select a MySQL connector, then drag and drop a CSV or TXT file into the upload area or click on it to select the file. After uploading, three buttons will appear next to the file:



The button with the filename inserts the variable `{filename.csv}` into the query. The platform replaces it with the actual path to the uploaded file. The Import Template button inserts the base query `LOAD DATA LOCAL INFILE`. The button with the cross removes the file from the interface.

Query example using a file variable:

```
SELECT * FROM {clients.csv}
```

Query example for importing data into a table (MySQL):

```
LOAD DATA LOCAL INFILE '{clients.csv}'
INTO TABLE target_table
FIELDS TERMINATED BY ','
IGNORE 1 ROWS
```

The target table must exist in the database. Import parameters (delimiter, encoding, wrapping characters) are configured manually depending on the file structure.

INFO: Note

When a file is uploaded, saving the query is unavailable.

Automatic data collection

Altcraft Platform platform provides a set of tools that keep profile information up to date and enrich it with user behavior and activity details.

This data can be further used for precise targeting segmentation or analytic reports.

Message tracking

The basic type of data collection automation is tracking message action history. It works by default for all available communication channels.

Message tracking records basic actions for messages like sent, delivered, opened or clicked.

Email tracking

Email action history tracking is processed by preset pixels — special tracking elements — that can be transparent images, code fragments or parts of URL links.

They can capture the following events:

- **message opened — a subscriber has opened the message**
- **message read — a subscriber has opened the message and kept it open for at least 10 seconds**
- **message clicked — a subscriber has clicked any link within the message**
- **link clicked — a subscriber has clicked a specific link within a message**
- **confirmed — a subscriber has clicked the subscription confirmation link**
- **unsubscribed - the unsubscribe link has been clicked by a subscriber**

Email sender provides you with email carrier information for profiles:

- **Delivered — the message has been delivered.**
- **Undelivered — the message hasn't been deliverd.**
- **Softbounced — sending client errors. The platform changes the profile status to "Hard bounced".**
- **Hardbounced — the provider completely refused to accept messages to this address.**
- **Complained — a subscriber reported to Spam. Communication is stopped.**

A single message link click history can be analyzed using Altcraft clicks map with content change history.

Email tracking data can also be used for cohort and lifetime value analytics.

SMS tracking

SMS links are also tracked by Altcraft platform. They are automatically shortened for SMS message versions:

`example.com/s/GrlX73`

SMS sender also tracks events like Delivered or Undelivered provided by SMS carrier.

Web Push tracking

Web push actions are processed by push providers: Mozilla, Google Firebase or Apple Safari.

As a result you receive events like "Push delivered", "Push opened"and "Push link clicked".

App Push tracking

App push actions are tracked by the push providing application.

For events like "Push delivered" and "Push opened" there are two links in each message `"ack"` and `"open"`, activated by the application GET request.

Optionally you can use event "Push link clicked" with `"click_action"` parameter.

AK Pixel

Pixel tracking scripts are implemented into your web site and track site visitors' custom actions. For more information see the specified documentation page.

API integrations

Using the API, you can create and edit profiles.

- **Profiles**
- **Profile subscriptions**

Scenarios

A scenario is a customizable logical process of automatic data processing.

The Refresh Data node directly modifies the data in the selected profile fields.

The Feedback node sends a message to the specified server, which can use the Altcraft API or work directly with the database.

Action hooks

Action hooks are used for sending events information to custom interfaces.

It can process more events than triggers — and they can both make http requests and send messages directly to RabbitMQ.

Profile update link attributes

You can use your messages links to update specific profile fields. The attribute notation looks like:

`update-field_name="value"`

It changes values in existing profile fields.

```
<a href="http://www.example.org" name="link" update-clicked_link="true">Update clicked_link!</a>
```

To subscribe your customer to a different resource on click — use the following link attribute:

```
<a href="http://www.example.org" subscribe-to="1">Subscribe me to resource #1</a>
```

Altcraft Platform platform provides a set of tools that keep profile information up to date and enrich it with user behavior and activity details.

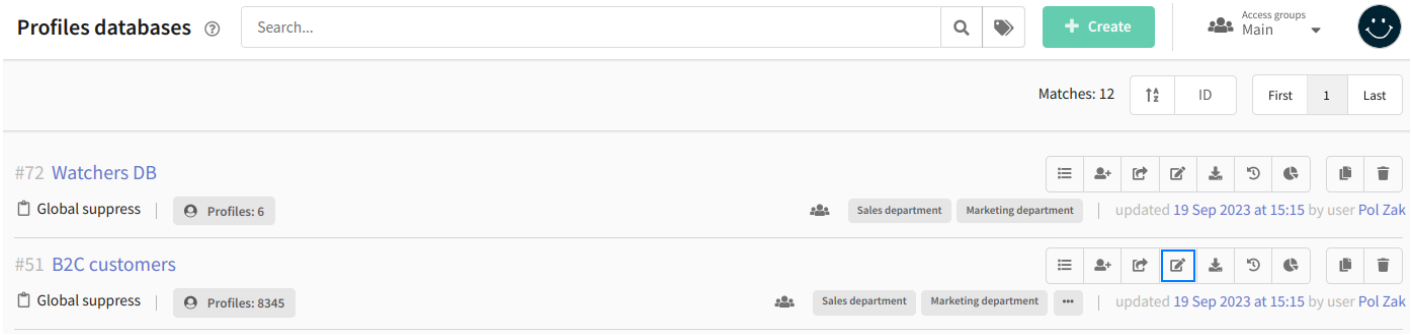
This data can be further used for precise targeting segmentation or analytic reports.

Bulk customers profiles update

Bulk profiles update allows you to update manually Database or the Segment of customers. In this case, conditions can be configured, to make update query more accurate. The following functions are available:

- **Adding and changing customer profile fields value**
- **Adding and updating [profiles subscriptions](/en/user-guide/profiles-and-databases/9405665)**
- **Managing [promo codes](/en/user-guide/loyalty-programs/)**
- **Deleting profiles from a database**
- **[Global profile unsubscribe](/en/user-guide/profiles-and-databases/9405665#Subscriberprofile-Globalprofilestatus)**

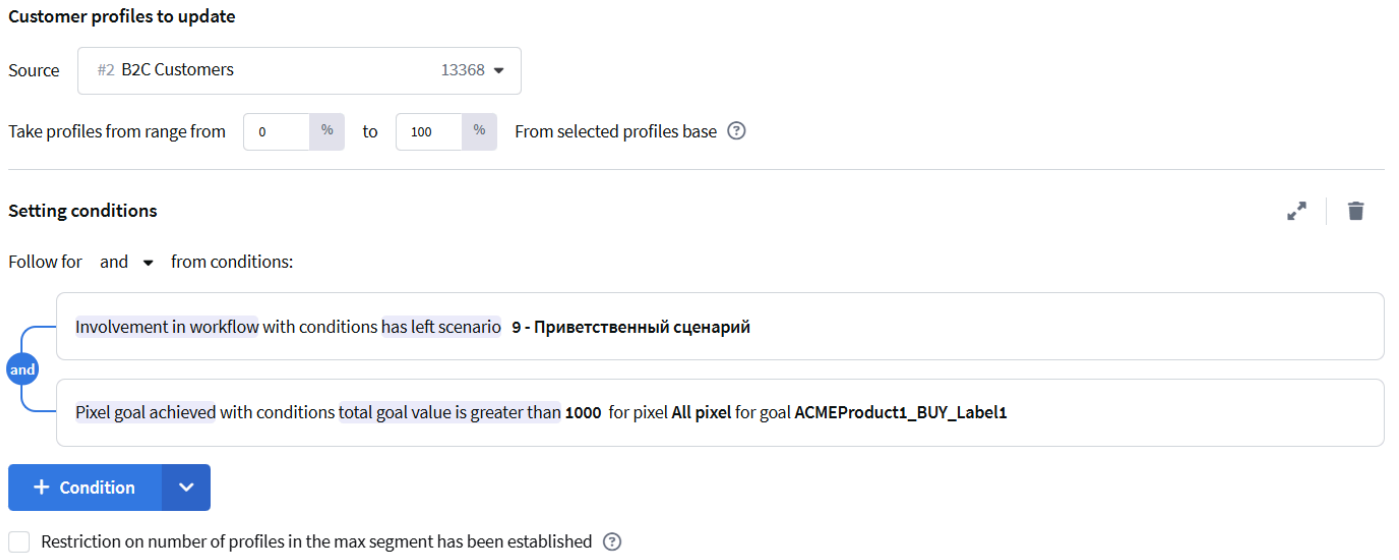
To open the bulk update tool, select Data —> Profile databases or Data —> Segments in the main menu. In the context menu of the profile group, select Update profiles:



Selecting customer profiles

Select the database where profiles that need to be updated are stored. If you want to update only a part of the profiles in the database, add specifying segmentation rules.

You can also add multiple rules and combine them using logical AND (all conditions must be met) and OR (at least one condition must be met).



TIP

In this section of the documentation, you can find information on segmenting customer profiles based on personal data, subscription data, and action history.

You can also set a limit on the resulting sample using two options that can be applied simultaneously — the maximum profile count limit and the range.

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can be included. If more people match the segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 profiles will be included in the segment. This restriction applies to the sample result of the segment. If a range is also used, the range restriction is applied first, and then the profile count limit is applied to the selected portion of the database.

The range determines which portion of the entire database will participate in segmentation. By default, it is set to 100%, meaning the conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in the database is assigned a numeric value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls within the selected range. This allows you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% and a range from 80 to 100% will never intersect.

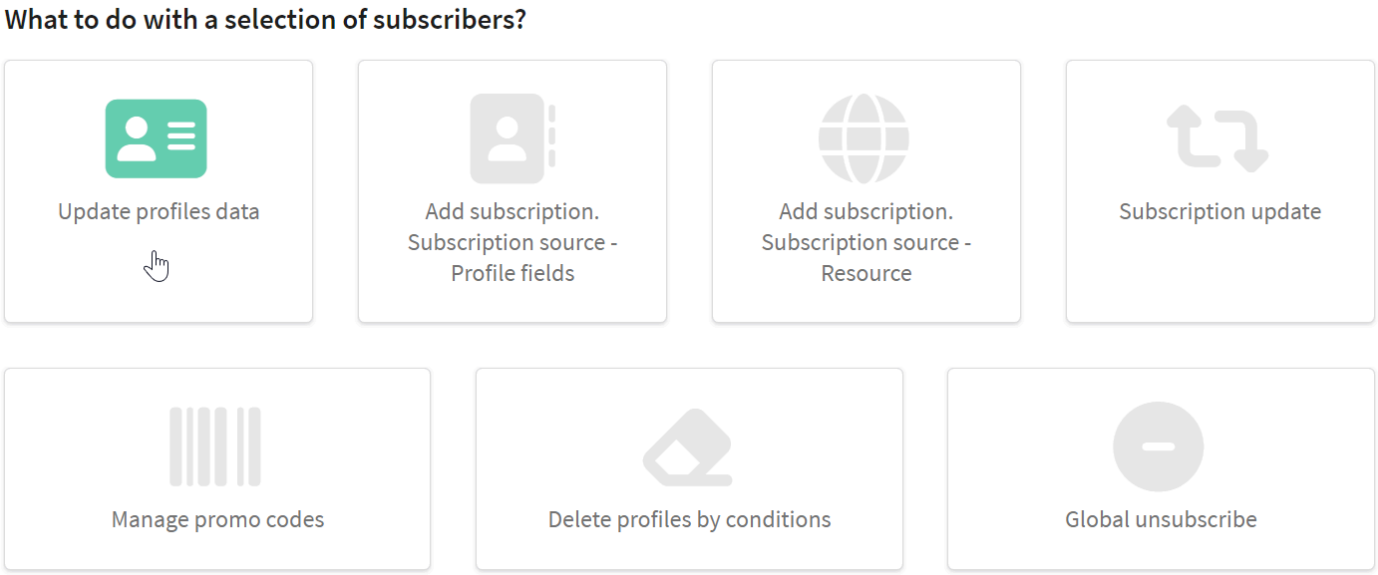
When new users are added to the database, they are distributed according to the same rules: if a 20% range is selected, a new profile has the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the database, and then the limit takes effect within that portion.

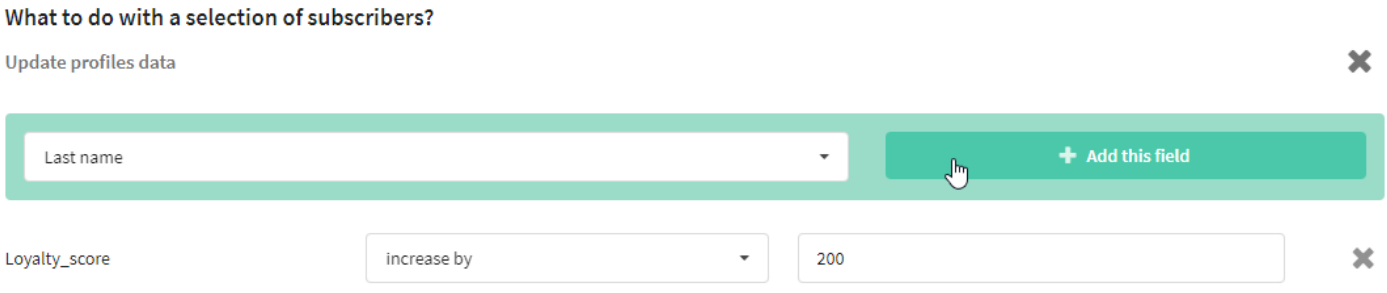
Customer profiles update

Updating profiles data

Select the option Update profiles data:



All database fields are available in the dropdown list. Choose the field you want to update and click + Add this field. Then select the update option: change or delete the value. If you want to update, choose a new value depending on the field type:



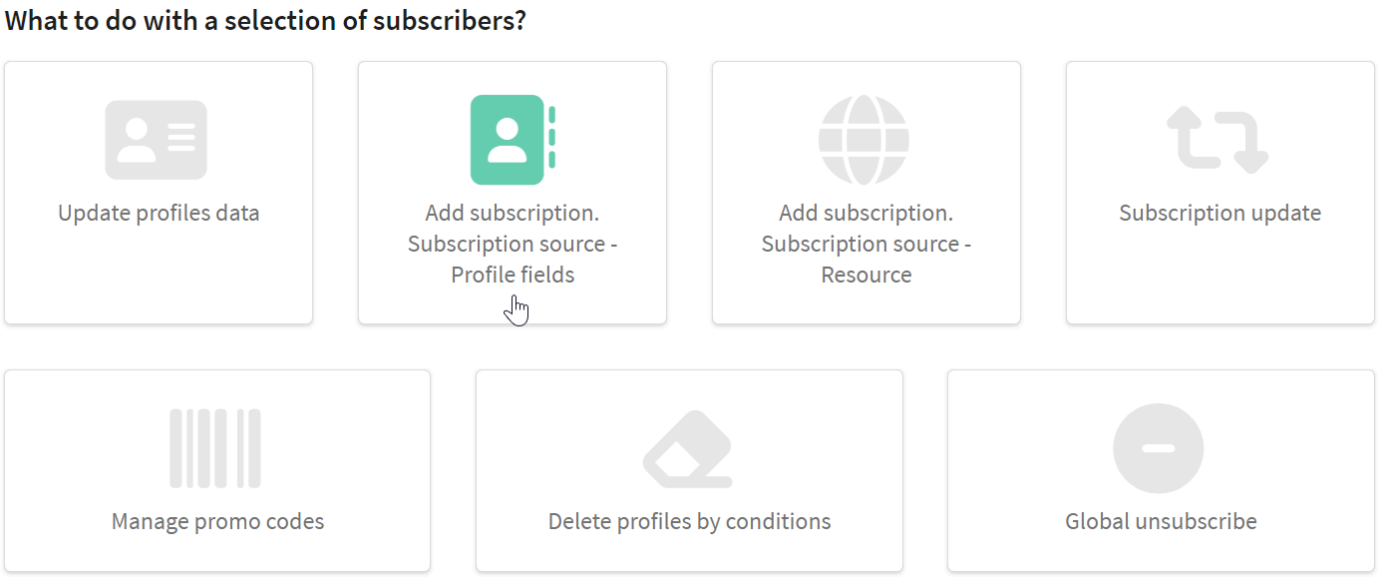
Adding a subscription to a new resource

Depending on the source of your contact information, there are two ways to add a subscription:

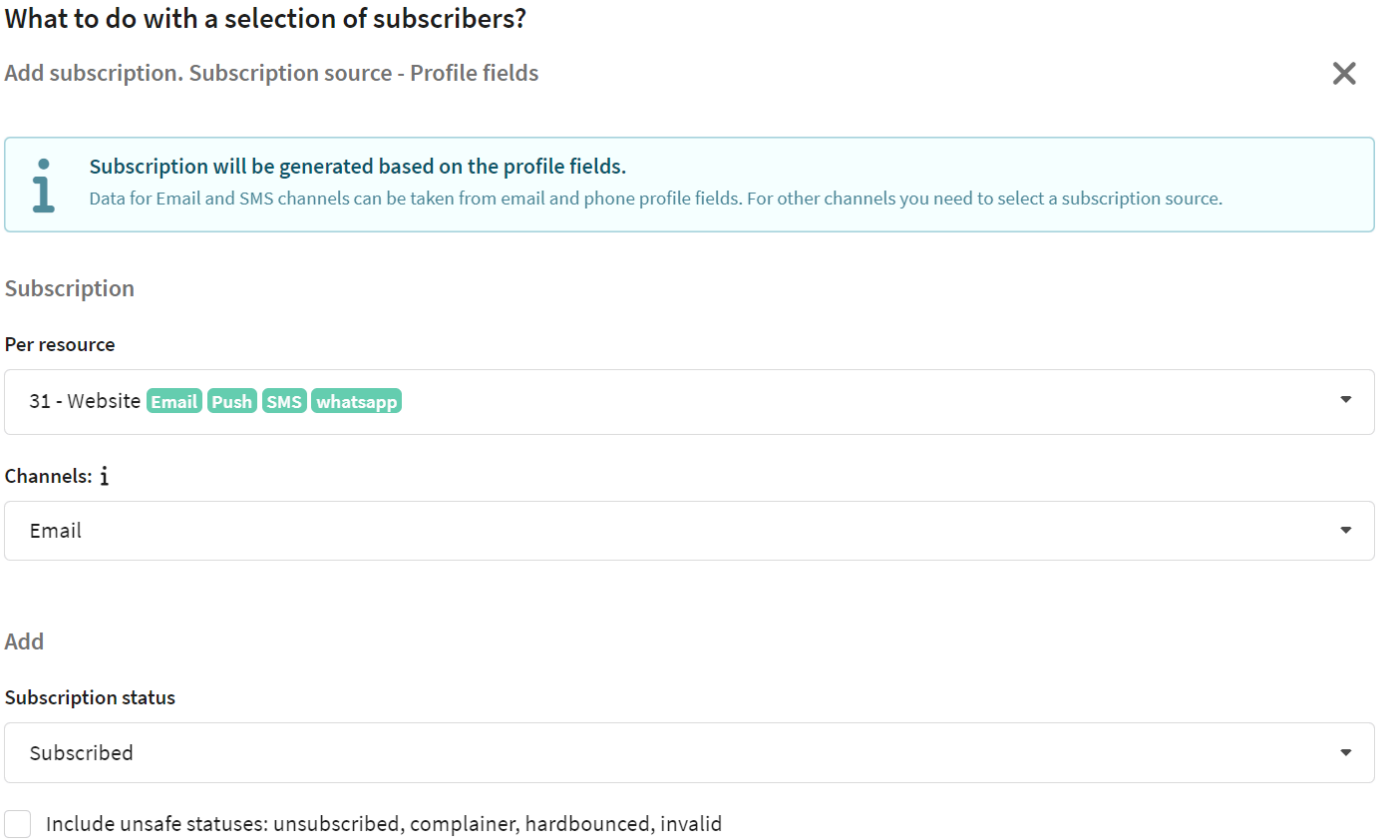
- **subscription source — profile fields**
- **subscription source — resource**

Subscription source — profile fields

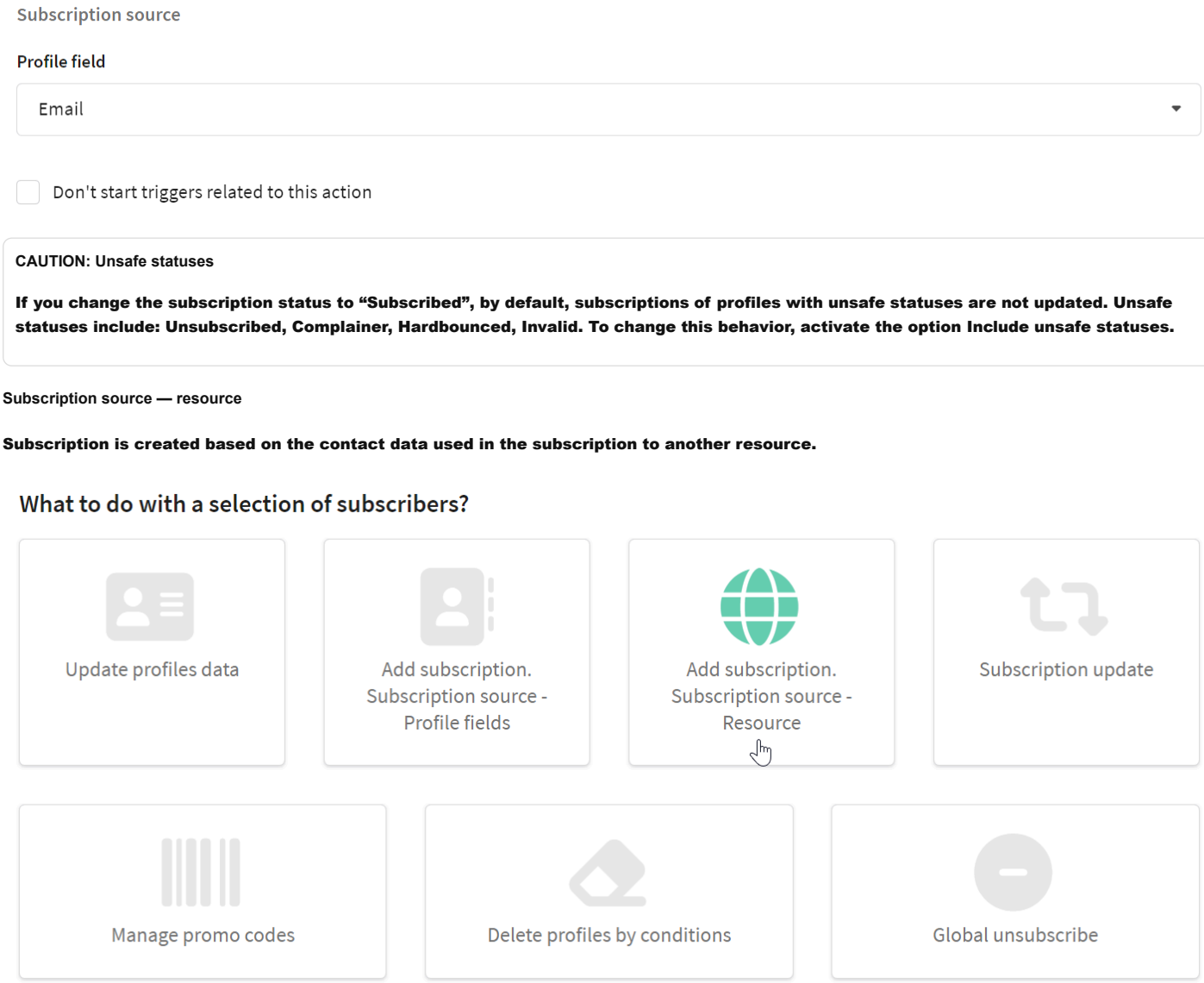
Subscription is created based on contact details (phone number, email address, etc.) provided in the profile fields.



Select the resource and channel to which you want to add the subscription. The resource must be attached to the database. Then specify the subscription status.

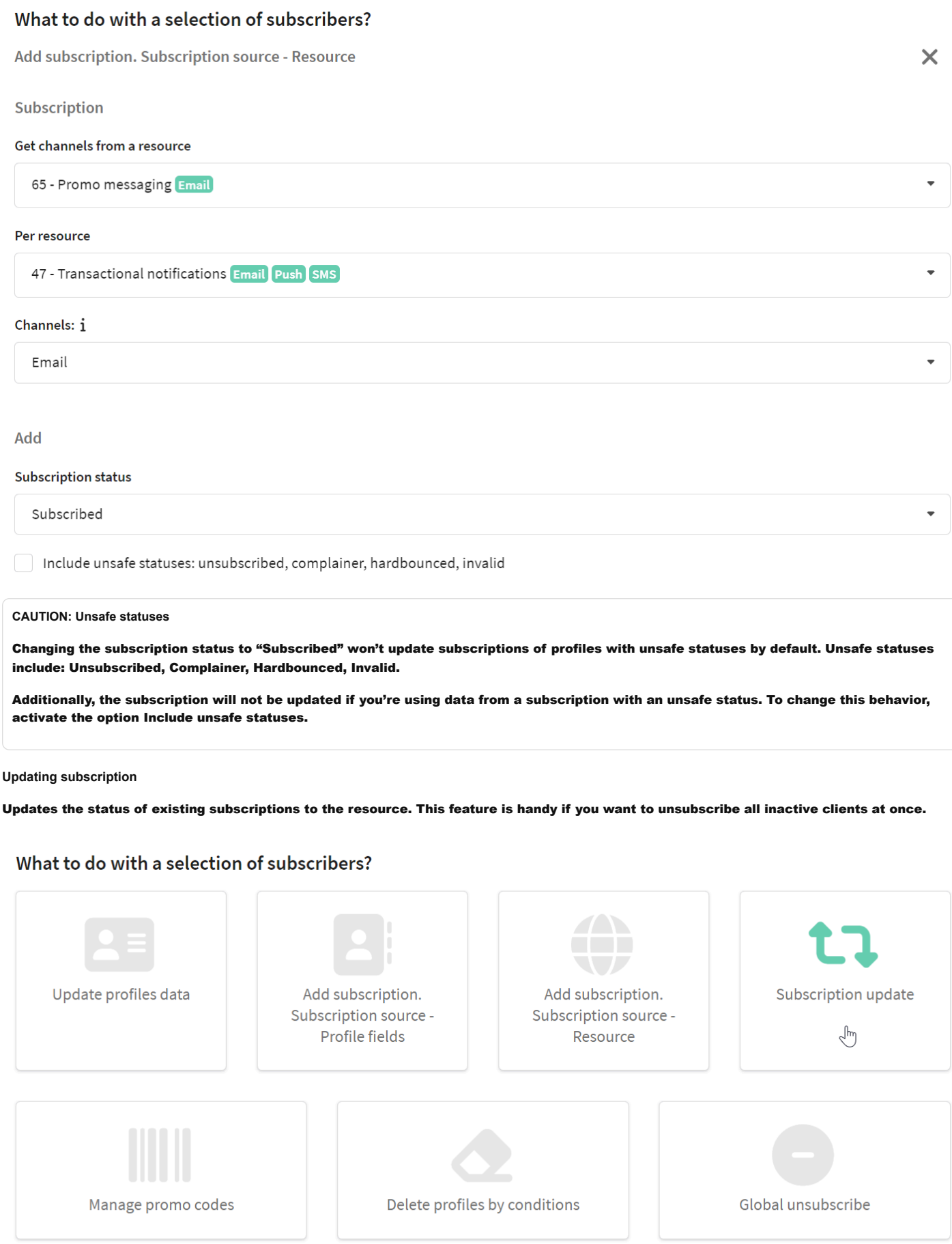


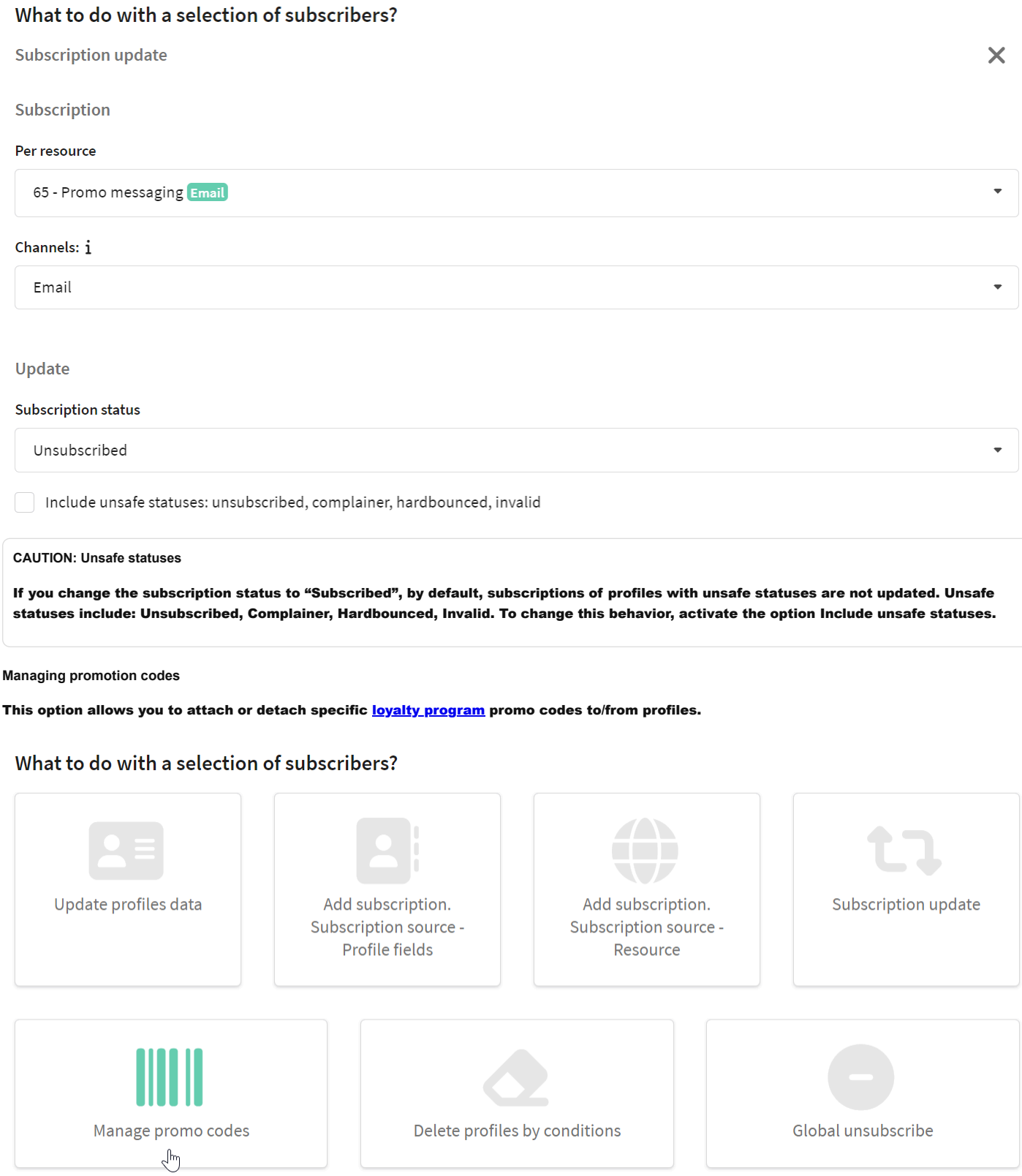
Choose the subscription source from the profile field. For subscribing to the Email channel, use the `Email` field; for SMS, WhatsApp*, Viber channels, use the `Phones` field. You can store data for other communication channels in the additional database fields.



You need to onfigure the following fields:

- **Get channels from a resource. The platform updates contact data using the subscription from the selected resource. If a profile lacks a subscription to this resource, the platform won’t add a new subscription.**
- **Per resource. The platform adds subscriptions to the selected resource for profiles.**
- **Channels. The platform adds subscriptions to the selected channel in the resource. Note that this channel must be configured in both resources.**
- **Subscription status. The platform adds a subscription with the selected status.**

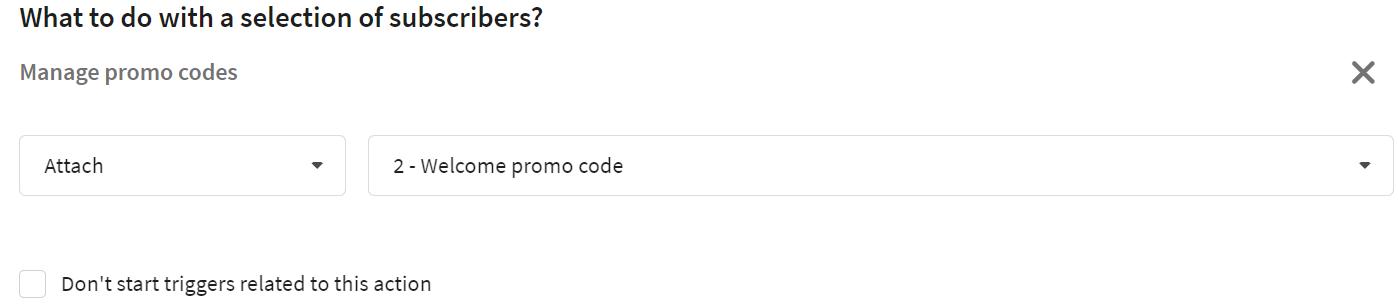

What to do with a selection of subscribers?
Add subscription. Subscription source - Resource
Subscription
Get channels from a resource
65 - Promo messaging Email
Per resource
47 - Transactional notifications Email Push SMS
Channels:
Email
Add
Subscription status
Subscribed
Include unsafe statuses: unsubscribed, complainer, hardbounced, invalid
CAUTION: Unsafe statuses
Changing the subscription status to “Subscribed” won’t update subscriptions of profiles with unsafe statuses by default. Unsafe statuses include: Unsubscribed, Complainer, Hardbounced, Invalid.
Additionally, the subscription will not be updated if you’re using data from a subscription with an unsafe status. To change this behavior, activate the option Include unsafe statuses.
Updating subscription
Updates the status of existing subscriptions to the resource. This feature is handy if you want to unsubscribe all inactive clients at once.
What to do with a selection of subscribers?
Update profiles data
Add subscription. Subscription source - Profile fields
Add subscription. Subscription source - Resource
Subscription update
Manage promo codes
Delete profiles by conditions
Global unsubscribe
Choose the resource and channel, then specify the status:



Managing promotion codes

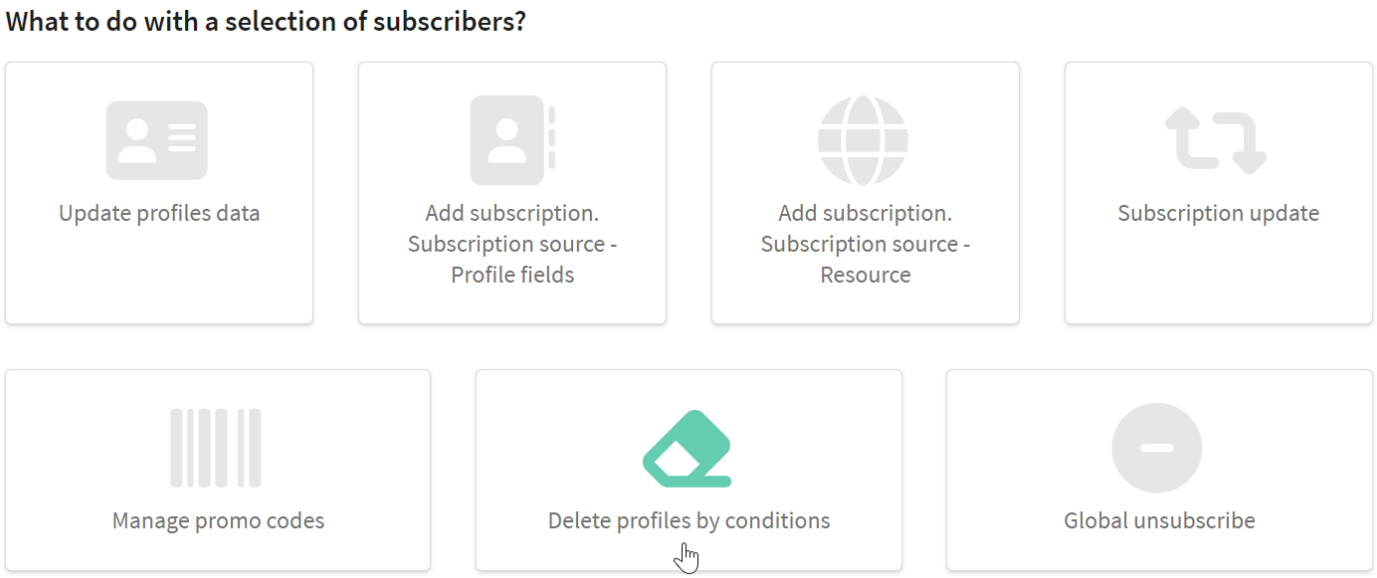
This option allows you to attach or detach specific loyalty program promo codes to/from profiles.

Select the action and the loyalty program:

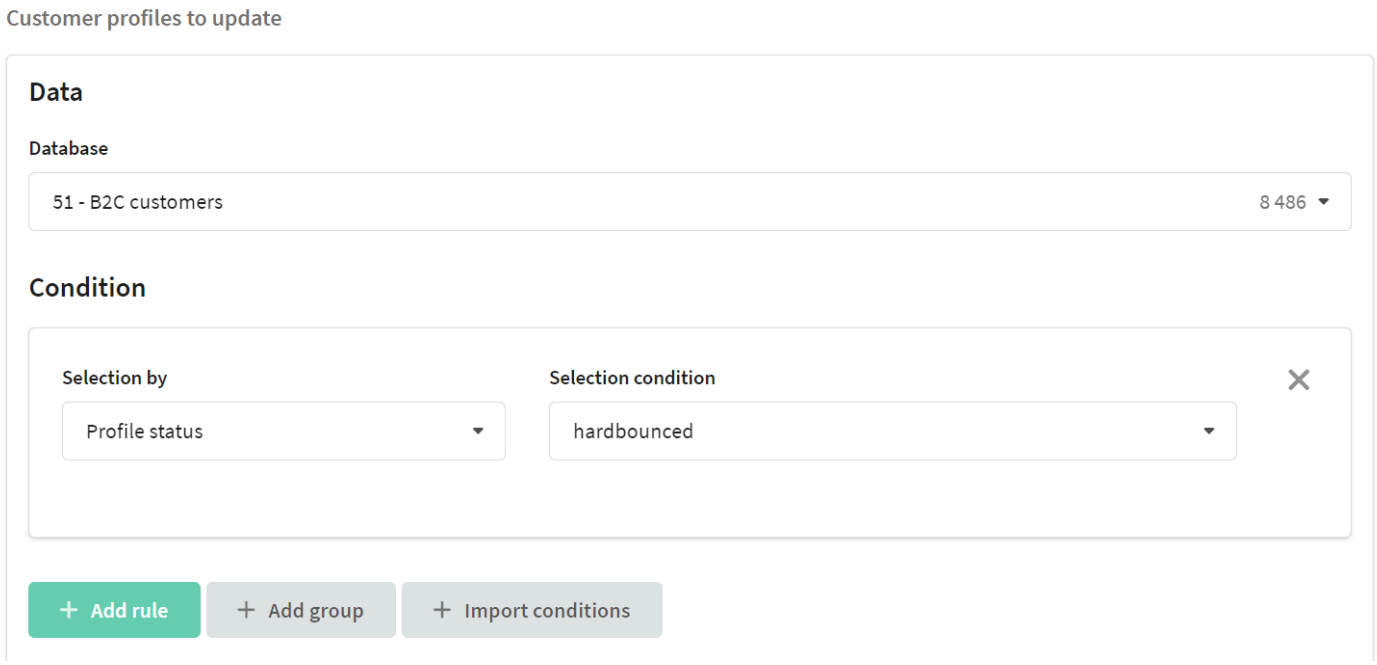


Deleting profiles by conditions

This option allows you to remove profiles from the database that match the specified conditions. The statistics of actions of deleted profiles is saved in reports.



Using segmentation rules, select the profiles you want to delete, then click the "Update" button in the top right corner:



The platform does not delete all selected profiles, but only those that had no interactions during a specific period.

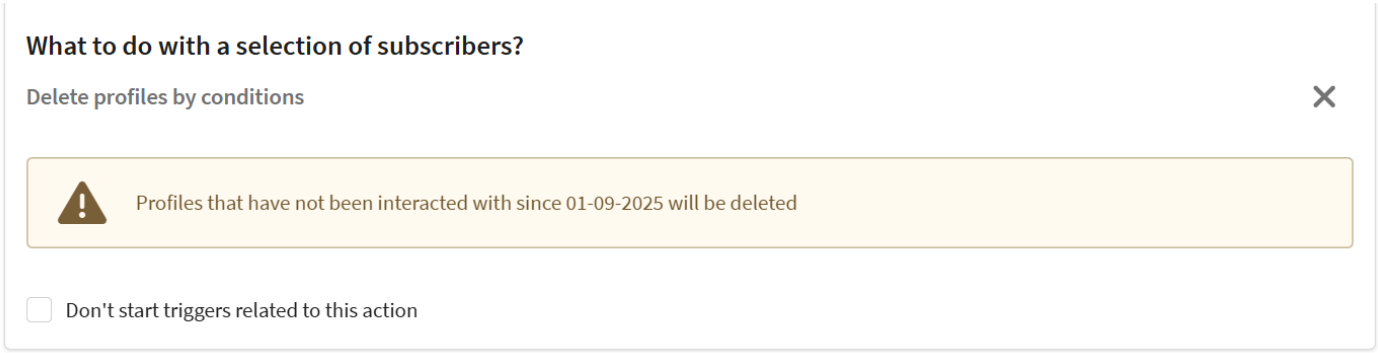
In the Cloud version, this period is fixed and equals one month before the start of the current month. For example:

- **If today is October 1, 2025, the system will delete profiles that had no interactions since September 1, 2025.**
- **If today is October 31, 2025, the same profiles will be deleted — the period does not change within the month.**

TIP

In the On-premises version, this period is configured in the system settings file using the parameter `PROFILES_REMOVAL_MIN_MONTHS`. Learn more.

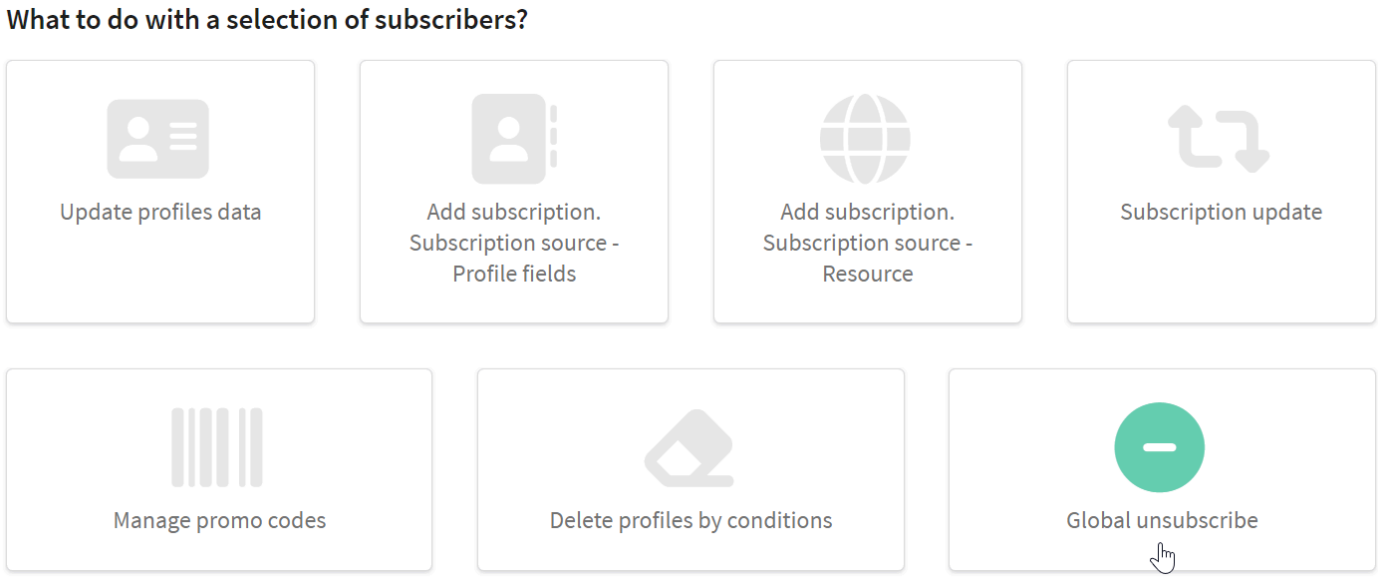
An interaction means any action recorded in the profile's activity history. Information about the current period is displayed in the interface:



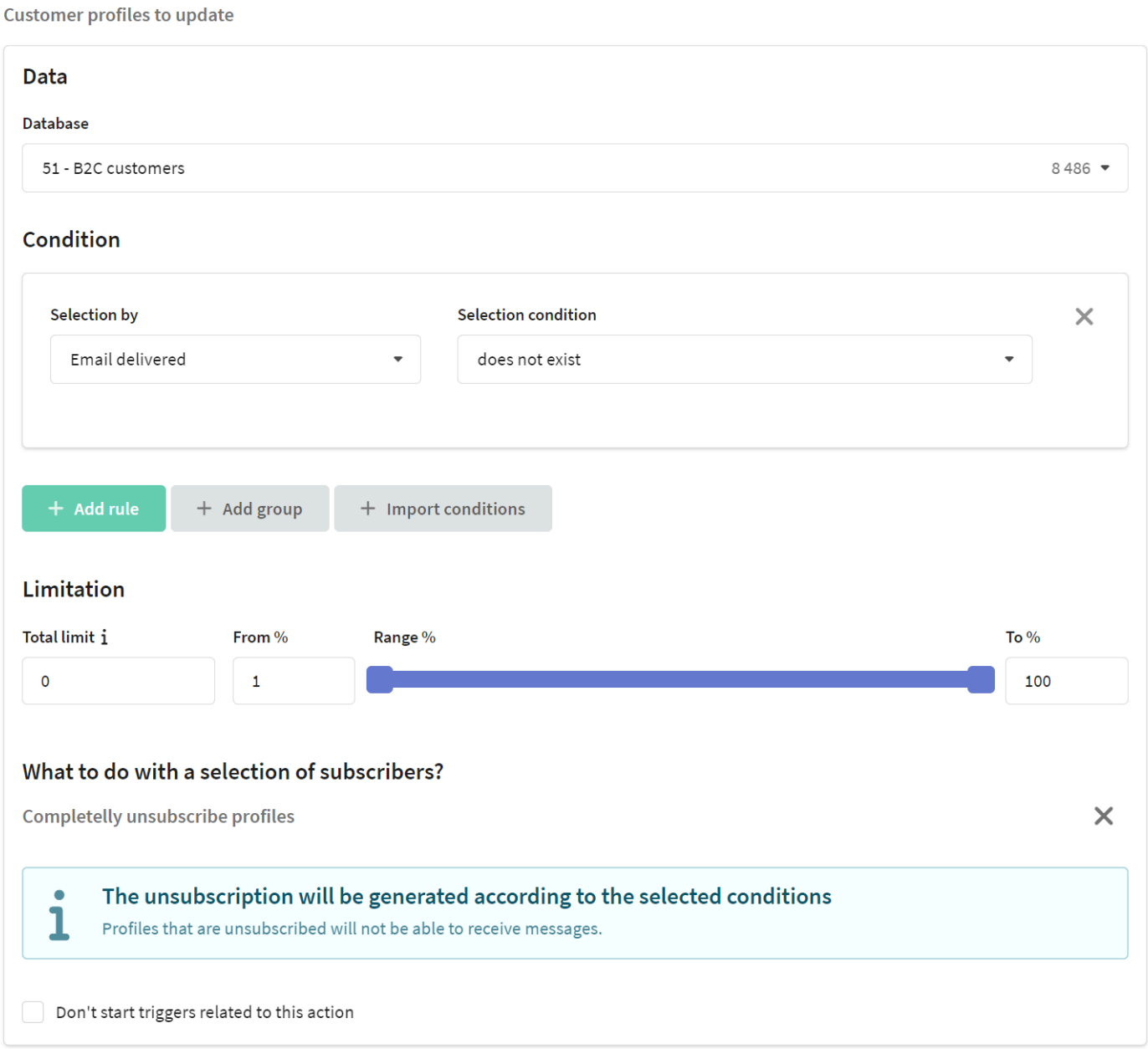
After launching the update, you can check the Tasks section to see how many profiles have been deleted. If a profile matches the selection conditions but was not deleted, it means that some interactions with it were recorded during the specified period.

Global unsubscribe

This option sets the "Unsubscribed" global status for all profiles, completely stoping communication via any channel. Even if a profile has active subscriptions to resources, mailings won't send messages to them.



Using segmentation rules, select the profiles you want to unsubscribe, then click the "Update" button in the top right corner:



In the platform, trigger mailings or scenarios can be configured to activate upon subscription addition or update, profile field update, or promo code attachment. If you don’t want these triggers to activate after the current update, enable the option Don’t start triggers related to this action.

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Double opt-in subscription

Overview

Double opt-in (DOI) is a form of mass communication subscription, organized in two steps:

1. **Subscribers provide their contact data and an agreement to receive content.**
2. **Subscribers confirm their agreement, clicking a special message hyperlink**

Double opt-in is applied over Single opt-in (SOI) when several legal or loyalty factors matter:

- **ISP's sender reputation**
- **Compliance with messaging legal regulations**
- **Higher marketing mailing ROI**
- **Clients' loyalty and lifetime value**

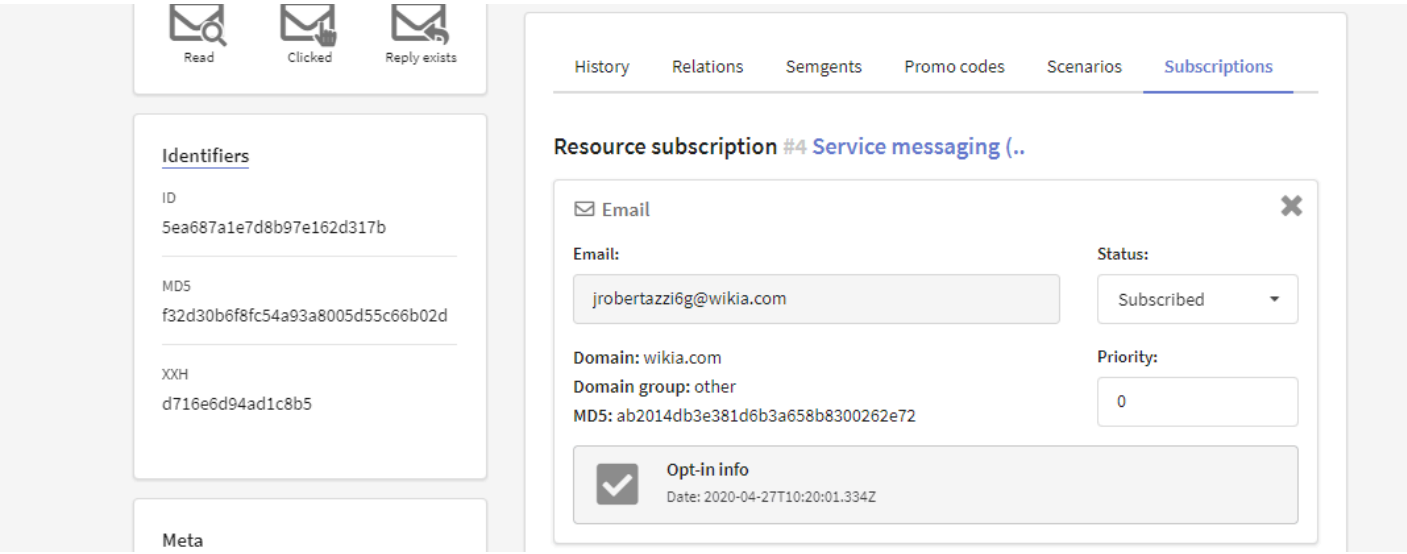
> CAUTION
>
> **Double opt-in is available only for subscriptions to resources.**

Step 1: Subscription

When a client gets subscribed to a Resource, the Subscription status should be set to Unconfirmed.

Adding an Unconfirmed subscription to a single profile

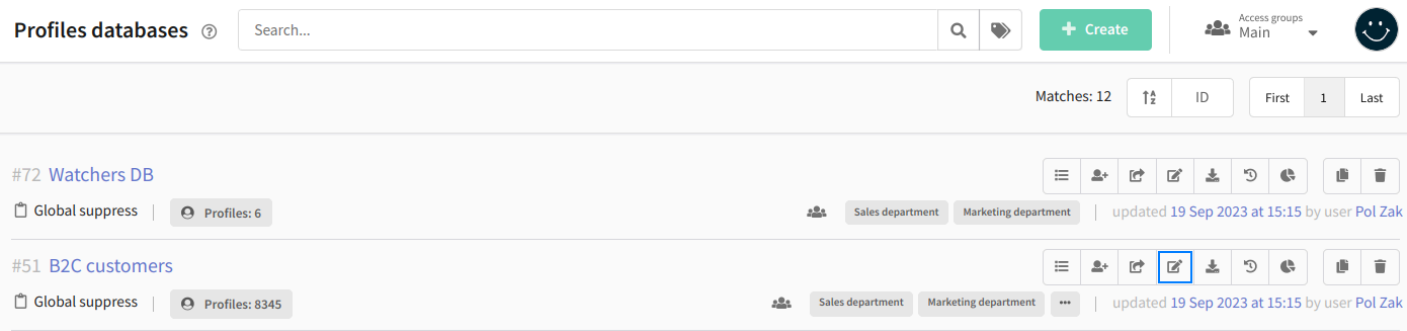
Open a client's profile in a Database and go to Subscriptions tab. Select a Resource and its Email channel. Enter a subscription email address or copy it from profile default fields. Select "Unconfirmed" in Status menu and save your changes.



After you save the subscription it obtains an md5 hash identifier and opt-in information block. It may be necessary for subscribers who doubt their subscription legal status. The opt-in info contains date in UTC and the client's IP address.

Adding Unconfirmed subscriptions to multiple profiles

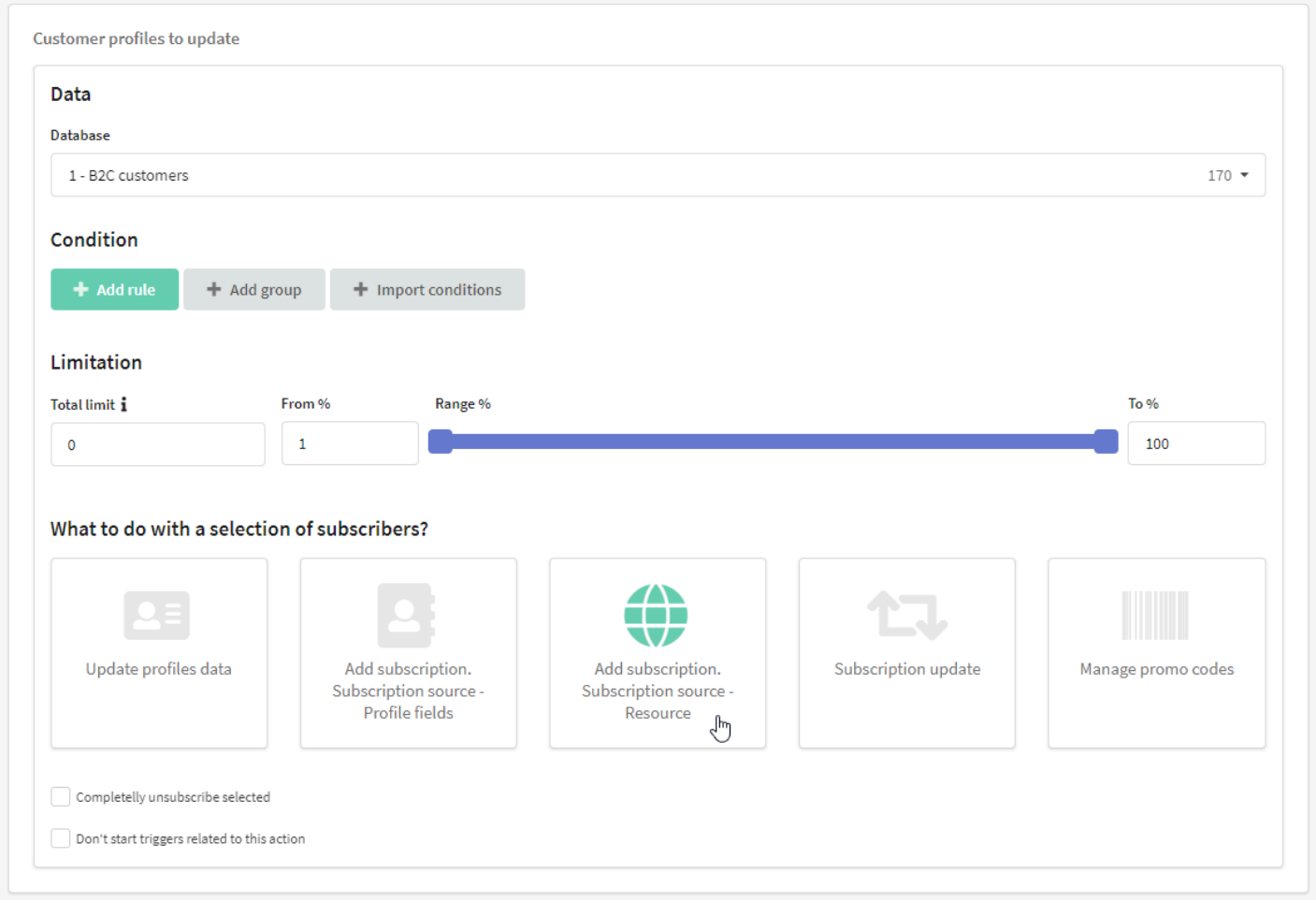
To organize double opt-in process manually for a group of clients, open a list of Databases or Segments and select Update customer profiles from the context dropdown menu on the right side of Database or Segment tile.



On the following screen you can correct a part of audience you want to subscribe. The update screen uses Altcraft Platform segmentation rules.

CAUTION

Profiles bulk update is configured for each Database of a Segment, provided the Segments contains more than one.



In Manage subscription section select Add subscription, specify a resource with Email channel and set Status to "Unconfirmed". You can choose between profile data and other subscriptions data as contacts source:

After all necessary settings are configured, click Update button. A Task will be created where you can watch update process.

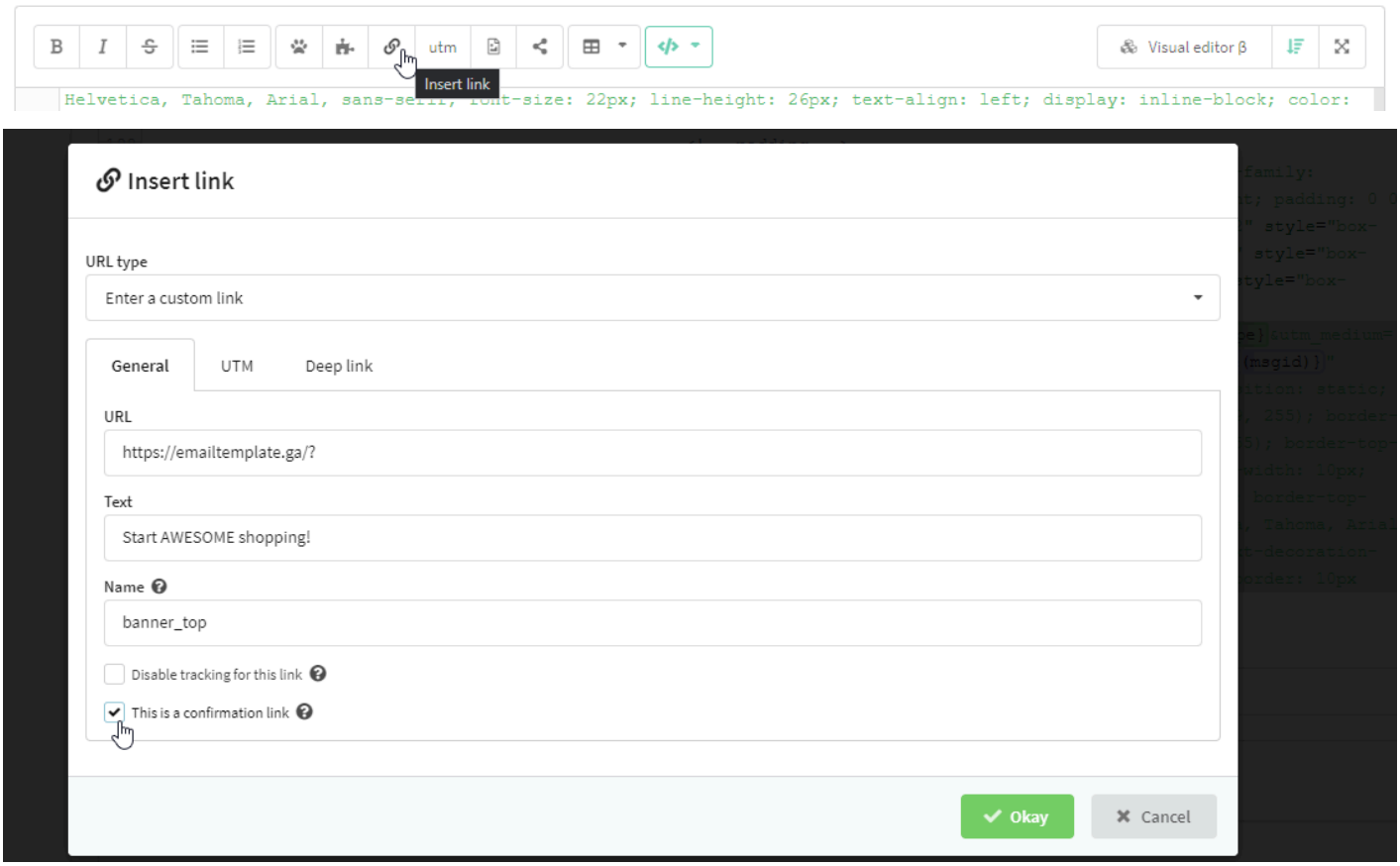
NOTE

For API profile import and subscription requests the following property should be added to subscription objects `"status": "unconfirmed"`

Step 2: Confirmation

Profiles with unconfirmed subscriptions can receive only messages containing a confirmation link. This can be any hyperlink: in the message template add confirm-link attribute to a hyperlink or select Insert link tool and check "This is a confirmation link" option:

HTML VERSION



After the template is ready, add it to a mailing, using the same Resource you have subscribed your DOI audience to. Message recipients by clicking the confirmation link will verify their intention to receive messages and thus their subscriptions' status will be changed from Unconfirmed to Subscribed.

NOTE

A click on the confirmation link can be a capturing event for trigger mailings and automation workflows. In trigger or workflow settings set Type: Message action —> Confirmed.

Suppression lists

![img] (/img/attachments/9407887/9407888.png)	**Suppression list is a list of email addresses, email domains and phone numbers, which you do not want to send messages to. Suppression list filters an email address or a phone number at moment of mailing launch. Suppression lists are formed to follow CAN-SPAM law rules.**

Three types of black lists are used in the platform:

- **Global Platform suppression list — affects mailings in all platform Accounts. If hard bounce is returned for the email address, it goes to global list. Valid addresses, that must be restricted from mailing, are stored separately.**
- **Global Account suppression list — it is created automatically in every account of the platform. Added contacts will be blocked from all Account mailings.**
- **User suppression lists — they are created by users in platform accounts. They can be attached to Profile databases or to Messages Templates.**

Several suppression lists with certain rules are considered, because segment (which can contain several databases with suppresion lists) and several messages can be used in the mailing.

Global suppression list of the platform

Global suppression list of the platform affects all platform Accounts. In consists of two parts:

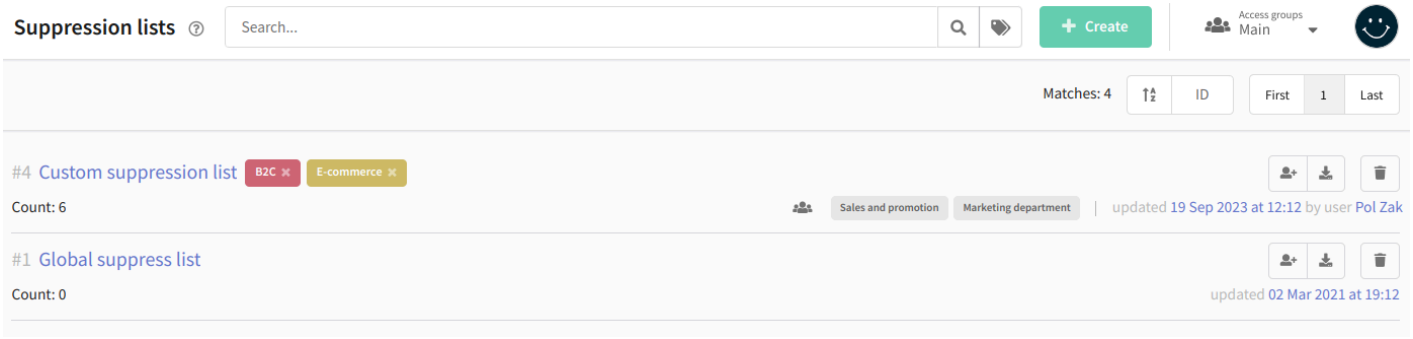
- **Invalid emails list — all emails, for which hard bounce (address does not exist) was received from email provider, go to this list.**

> CAUTION
>
> **Take your notice, that when such answer has been received, Subscription status of the customers changes to hard bounce. These subscriptions will not be used in the mailings, because sending to non-existing addresses makes sender reputation with mail service worse.**
>
> **If you change status of these subscriptions to Subscribed — they will stay in global lost and will not be processed. To resubscribe these customers, you need to remove them from global suppression list of the platform.**

- **Global suppression list — any addresses or any domains for all Accounts of the platform. Platform Administrator can add data in this list, using Administrative panel.**

Account suppression lists

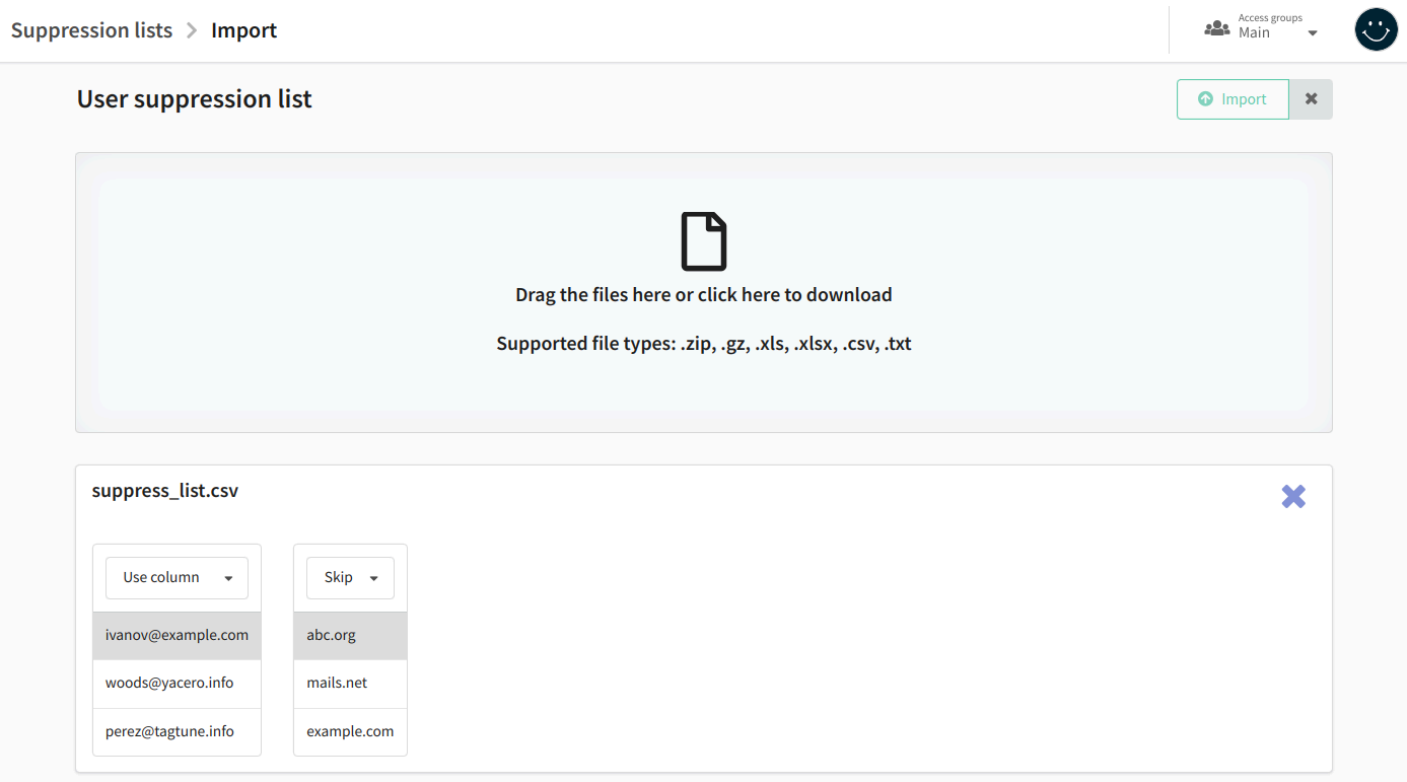
To access Account suppression lists management, choose Data —> Suppression lists in the main menu. This will open a menu displaying the existing suppression lists for your account. From there, you can use the quick actions menu to begin working with the desired suppression list immediately:



File import

Choose appropriate point

in quick actions menu to add contacts from file to suppression list. Drag your file with contacts list in CSV or XLS format into the form opened. You can upload several files, and there can be several columns in these files. Next, you can choose, which column you are going to use for the import to suppression list. After that, click Import:



Manual import

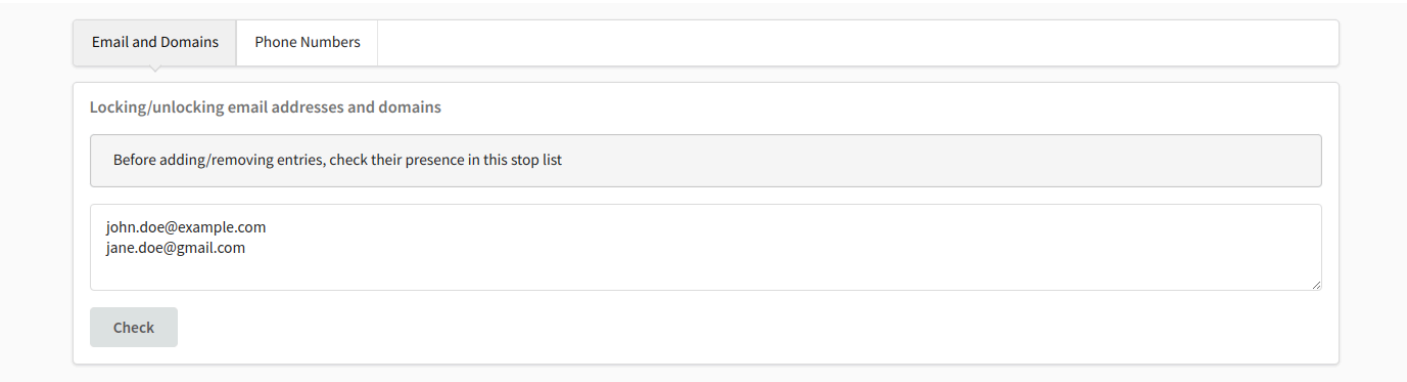
If you want to add customer contacts manually, open the desired suppression list. You can add customer email addresses, email domains, and phone numbers to the blocklist. Depending on the contact type, open one of the tabs and enter the contact details you want to add to the list in the Lock/Unlock section.

TIP

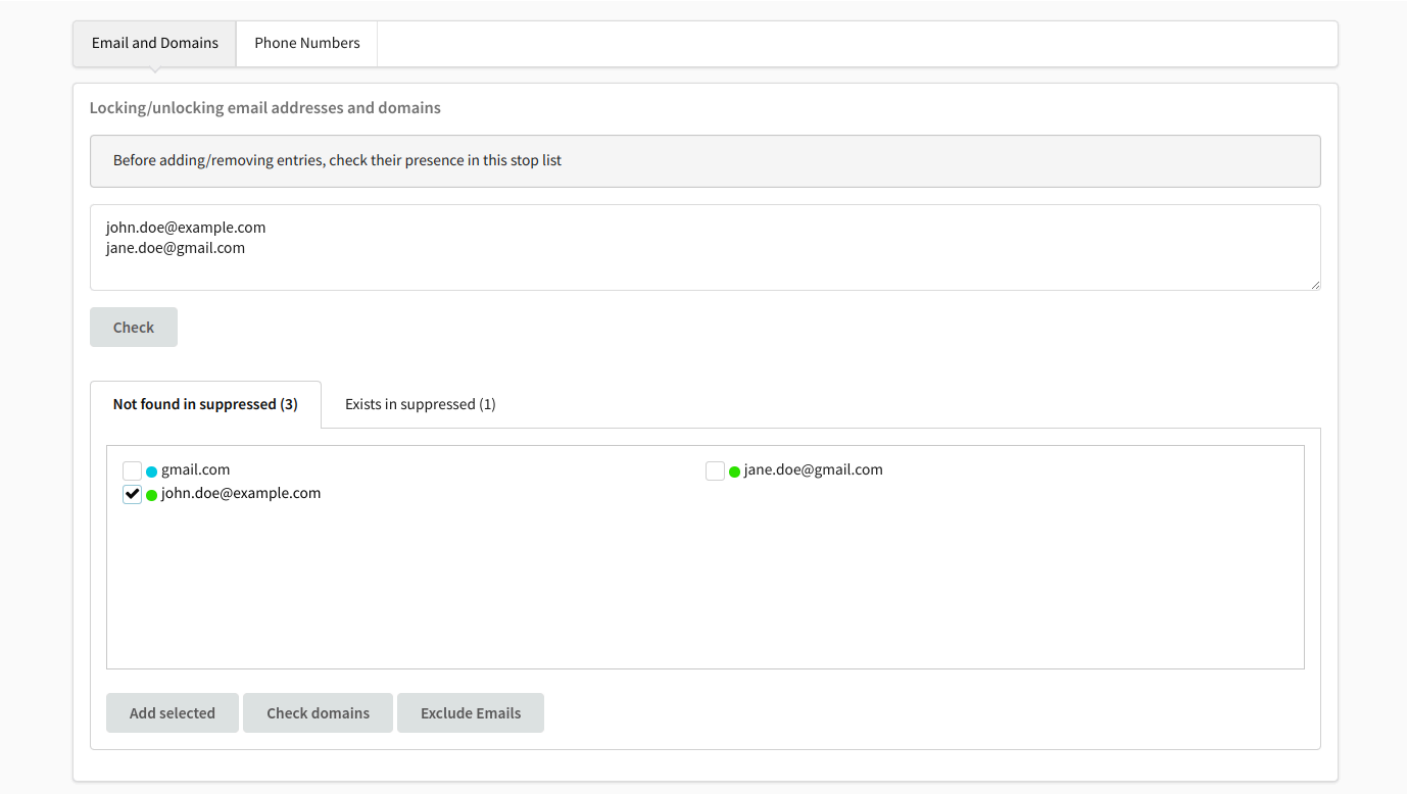
Multiple entries should be separated by commas, spaces, or placed on new lines. You can specify up to 500 email addresses, domains, or phone numbers.

Phone numbers should be in international format (e.g., +79123456789).

Then click Check:



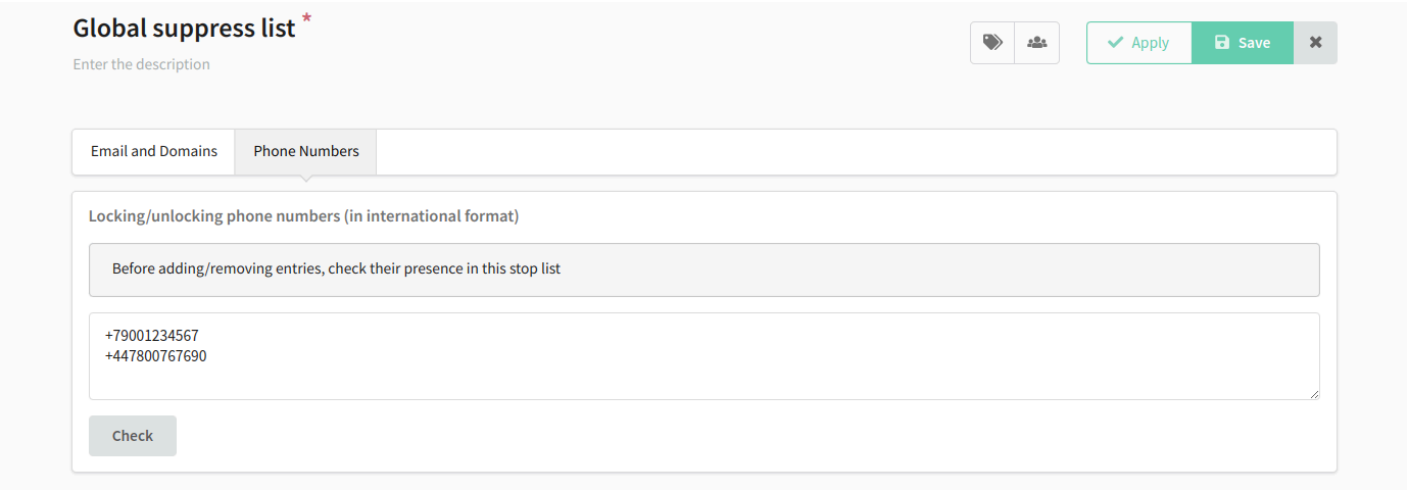
Two tabs will appear: Not found in suppressed and Exists in suppressed. If a domain, email address, or phone number is not in the suppression list (it will appear in Not found in suppressed), you can select it and add it (Add selected). If it is in the suppression list, you can remove it (Exclude). Additionally, you have the option to select all email addresses, all domains, or all phone numbers in tabs (Check):



Global suppression list

In every Altcraft account, one global suppression list is created automatically, which works always - for all mailings within the Account. You do not need to attach it manually to Profiles databases and Messages templates.

You can add subscriptions manually or import from the file to the global suppress list:



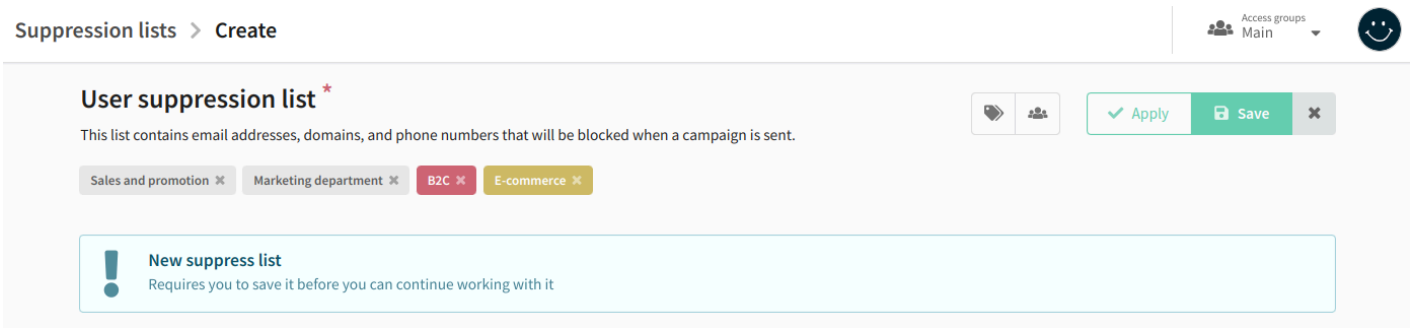
If you want subscriptions to be automatically added into the global suppress list, you need to use a link in the message - URL of global unsubscribe variable `{globalunsubscribe}`. Address of subscribers, who follow this link, will immediately go to global suppress list and will be excluded from all future mailings in the account.

TIP

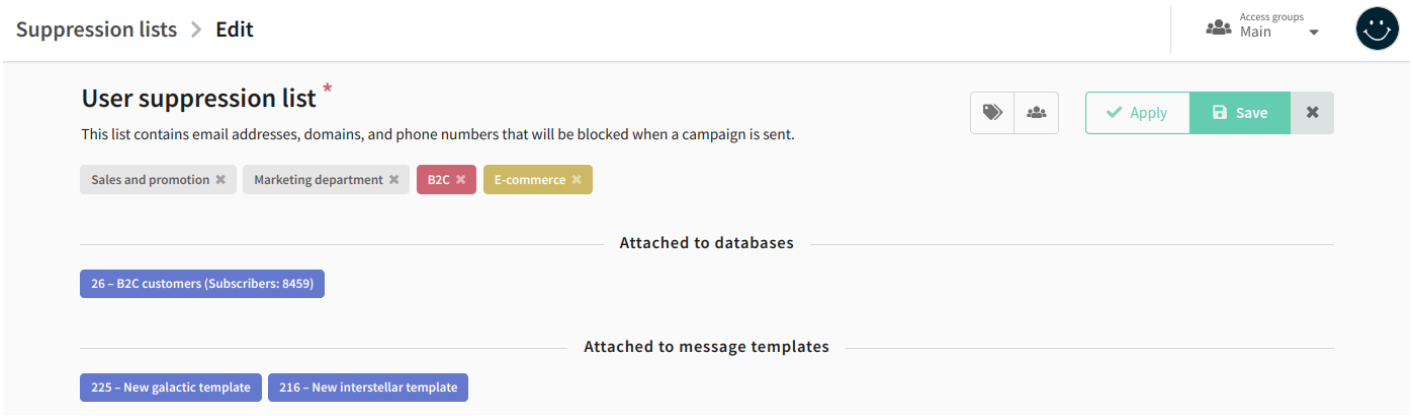
Note that communications will stop only for the email address to which the message with the unsubscribe link was sent. If the profile has a subscription with a different email address, messages will continue to be delivered to it as usual.

User suppression lists

Press + Create in lists menu to add user suppression list into the account. Select Suppression list name, enter Description, set access group and tags for quick search, if it is necessary:

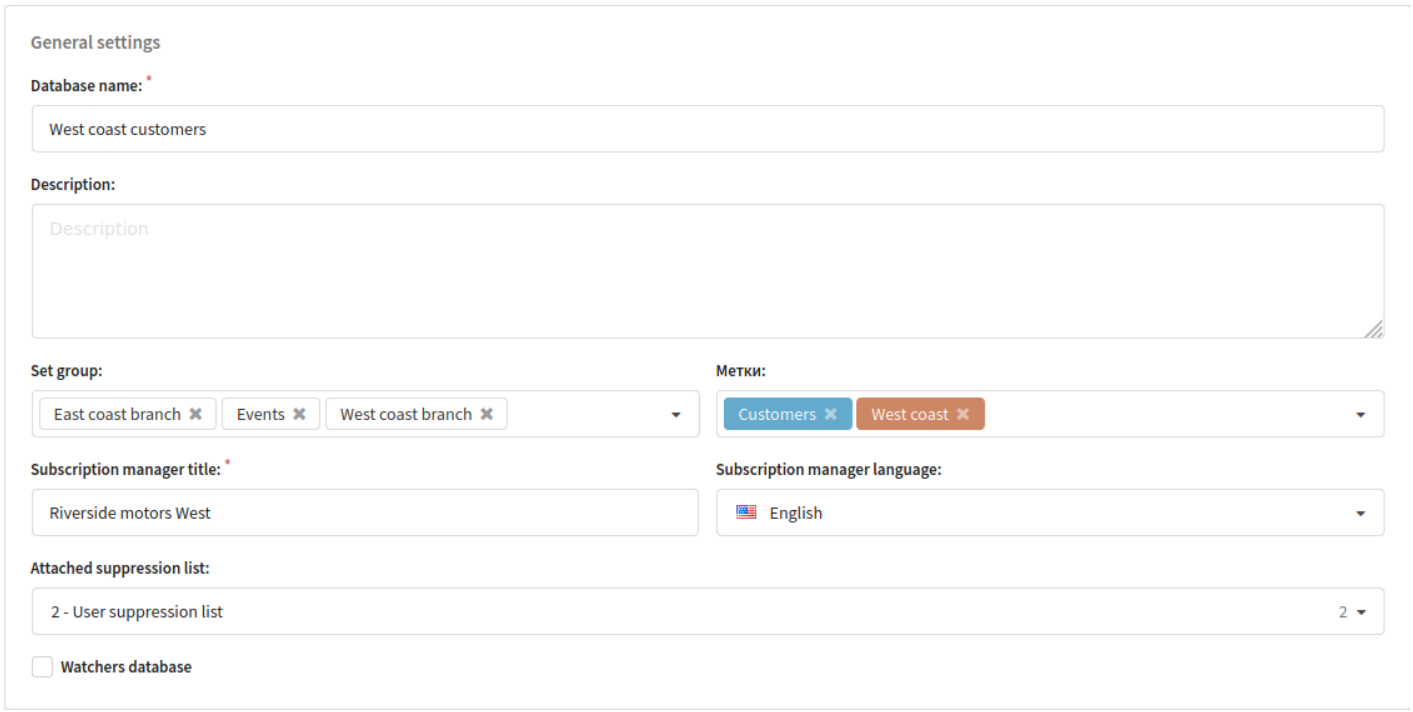


After saving it, you can manually add customer contacts to the created suppression list or import them from a file. Additionally, you can attach the list to profile databases and templates. You also have the option to check at any time which platform objects the list is attached to.



Suppression list for a Profiles database

User suppression lists can be attached to a certain Profiles database of the Account. This list will be active only during orchestrating Mailings for an audience, which includes this profiles database. For this, choose Attached suppression list in General settings of Database editing form:



Suppression list for a Message template

The second option of using custom suppression lists is attachment to certain Messages templates. This will allow you to control, which audience should not receive certain content: all unsubcriptions by template will go this list and therefore, communications will be blocked.

Open the necessary Template, and go to Options tab, and choose necessary suppression list in Extra settings block:

Rules of suppression lists interaction

You can use several suppression lists during mailing. Subscribers are excluded from the mailing in the following way:

1. Global suppression lists are checked, and all subscribers there are excluded from the mailing.

2. Next, suppression lists, attached to Profiles databases, excluded subscribers of their databases from mailing.

INFO

If there are two same subscriptions in two databases, and suppression list, which contains them, is attached only to the first database, then messages from mailing, which contains the segment with this databases, will be completely sent. This happens, because suppression list will act only on the database, that it is attached to, which is the first one.

3. Next, suppression list, which is attached to the Message template, will exclude only subscribes, who would receive this message.

Profile relations

TIP

To attach, detach, edit or modify relations automatically you would need API integration. Check this Section of API documentation.

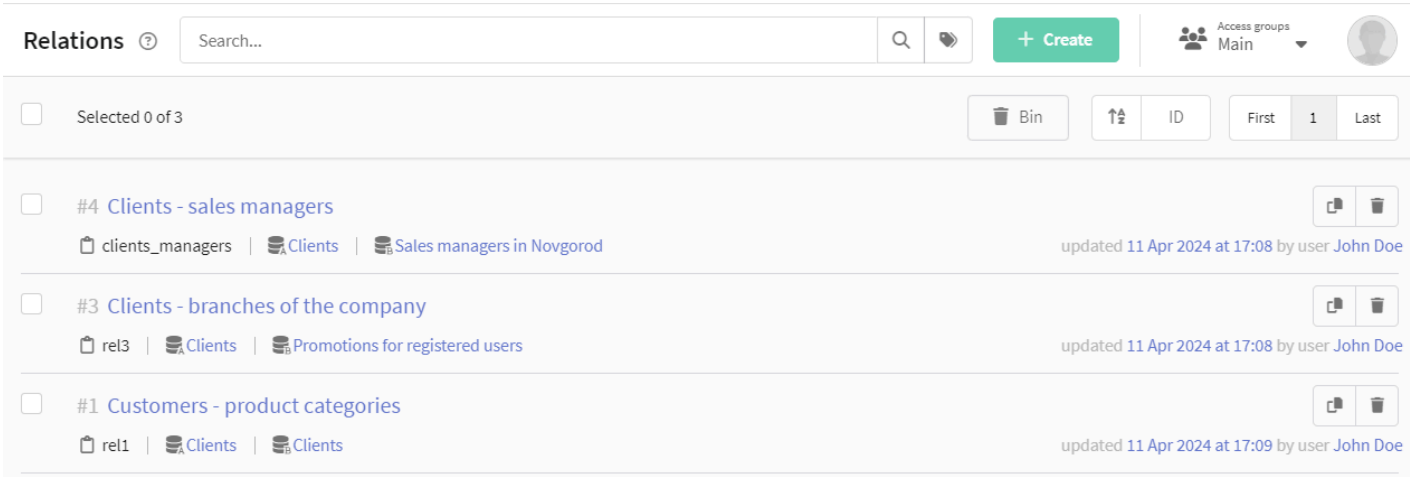
The relations between profiles allow you to implement customer interaction strategies on the platform. You can relation profiles within the same database or relation profiles from different databases. There are two types of communication:

Direct — outgoing. Shows the relationship of the profile to other profiles. For example, the relationship of a referral client with his clients. Feedback is incoming. Shows how other profiles relate to the selected profile. For example, clients of one agent or manager of a company evaluate the quality of his work. Each relationship has properties - relationships between clients. Properties can be in the form of numeric or Boolean (false - true) values.

Relation objects are created in the user interface. You can add, remove, strengthen, or change a relation either manually or using special API requests described in this section of the documentation.

Managing relations

In the main menu search for Relations in the Data section. Here all your existing relations are enlisted, accompanied by created-updated summary and databases involved. Using actions menu on the right side you can quickly clone or delete a relation. To edit an existing relation, click on its name. To add a new record, click + Create at the top of the page:



General settings

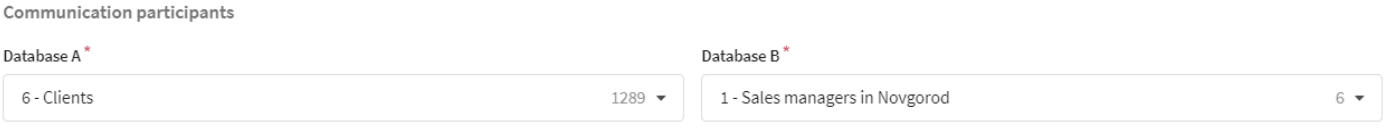
In General settings enter the new relation's name and short name as well. Short name will be used in message template editor and relations API requests.

Below select Databases containing profiles you are going to relate:

- **Database A — contains profiles that are collecting relations. These can be your products, managers, agents — or partners. We can call them objects.**
- **Database B — contains profiles that initiate relations — your or your partners' customers, reviewers or subjects we can call them.**

TIP

If you are using one database relations, select it in both drop-downs!



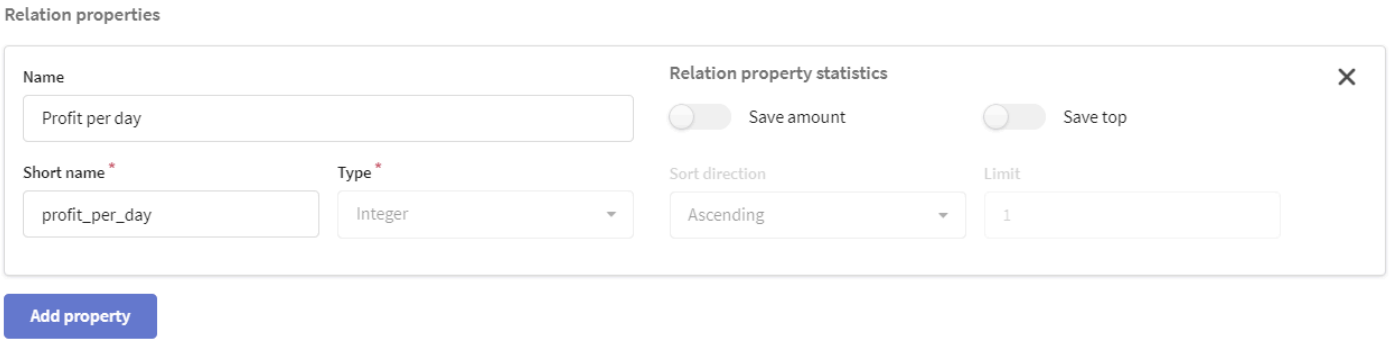
Relation properties

One relation can contain up to 11 properties, describing the value or weight of interaction between profiles. For each new property enter a name and a short name for templates and API.

Property type sets which data will be used to measure relation impact:

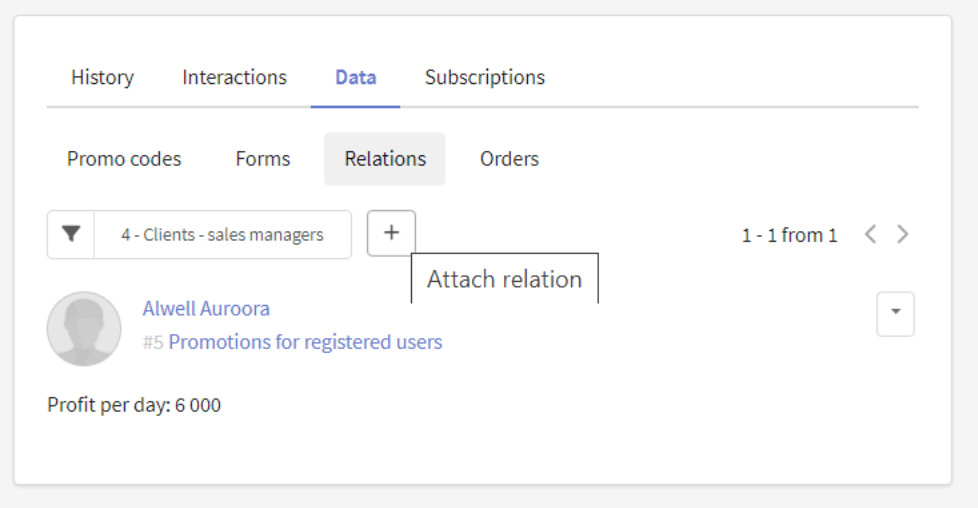
- **Integer — any integer number. Fits well for quality assessment, numbers of purchases, social network profile views and post likes.**
- **Float number — a floating-point number. Monetary values and other more complex numbers can be recorded in this property.**
- **Boolean — a boolean true-false value. Can be used for one-time or on-off actions: subscriptions, recommendations, or availibility.**

For numeric properties you can store statistics: save amount and save top. The sorting direction determines whether to store the highest or lowest values in the top, and the limit (from 1 to 20) sets the number of relations for statistics:



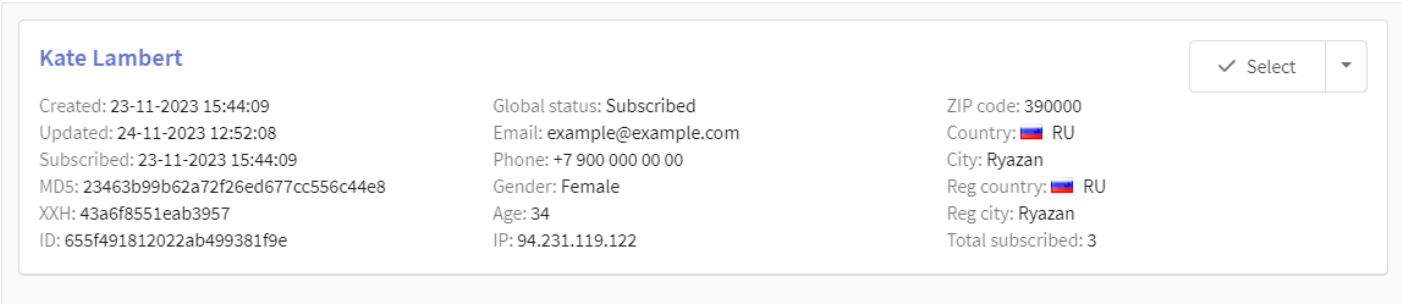
Relating profiles manually

Open the subscriber profile in Database A. On the Data tab, go to the Relation section. The available relations for the profile of the current database are displayed here.

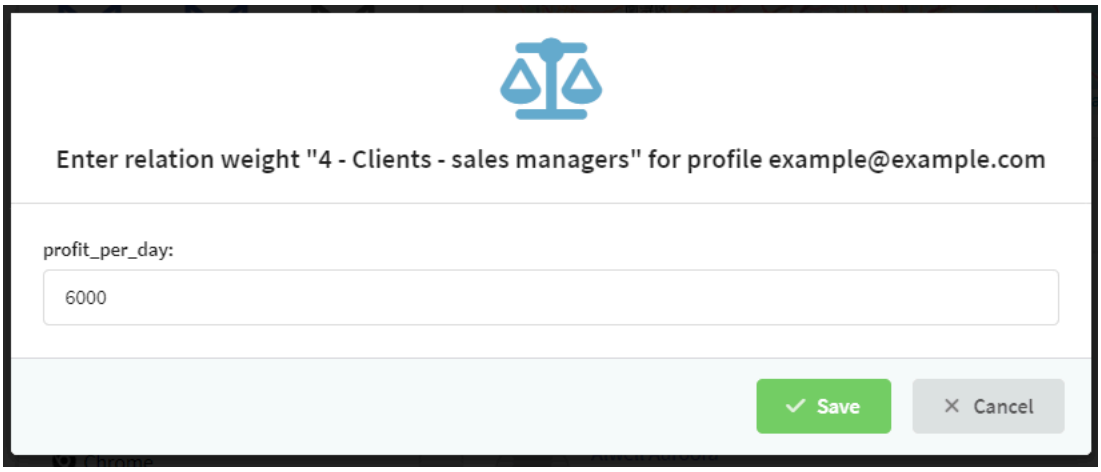


In the window that opens, specify the rule for selecting a profile from the Database B. Standard rules are used, as in the search or segmentation of profiles.

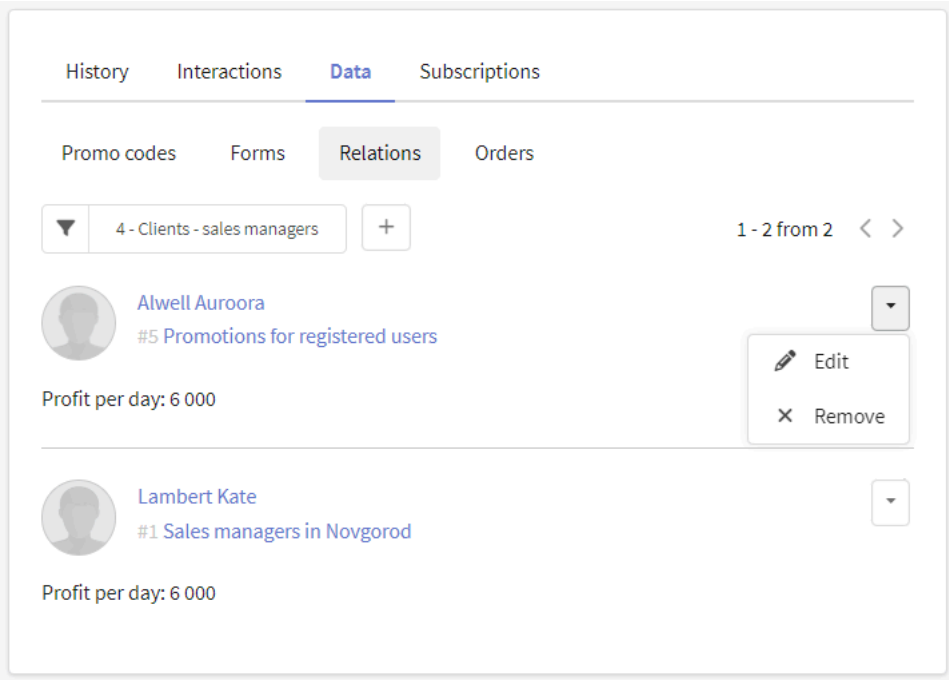
Click **Apply ** and Select:



Then enter relation's property weight:



After adding a relation, a brief summary and a top on the properties of the relation will be available in the profile, as well as a list of related profiles from the Database B. You can change the weight of the relation or delete it using the drop-down menu on the right side of the entry:



What are relations and relation values used on the platform for?

1. Segmentation

They can be used to create a static, dynamic, or updatable segment. Due to this, the target audience is "separated" from all the clients of one database based on the attitude of its participants to each other.

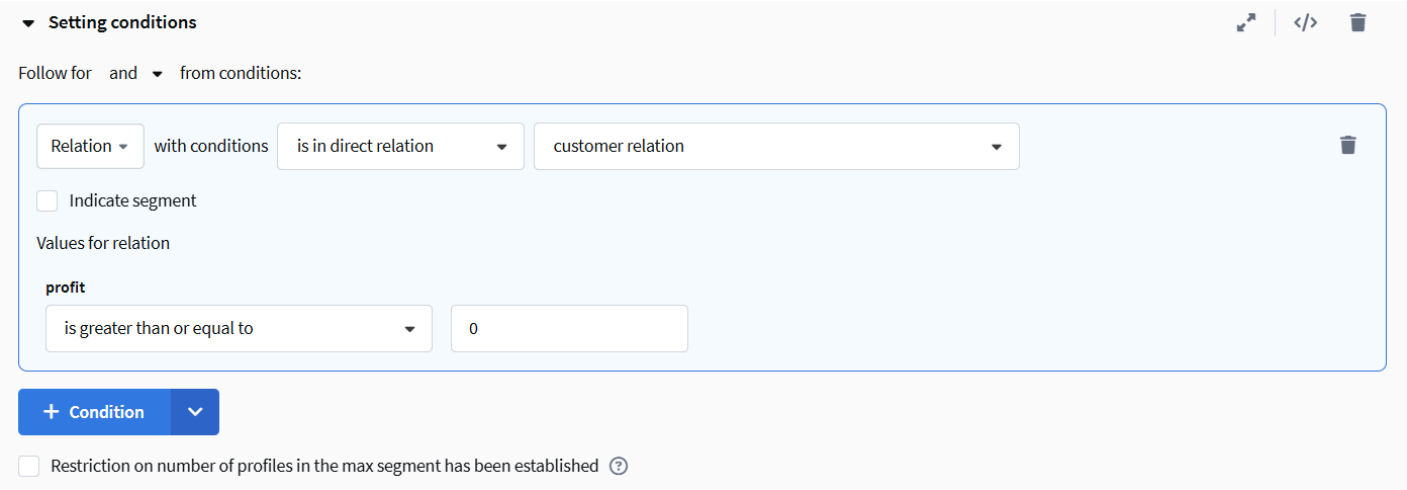
Additionally, you can add checks for certain communication properties:

- **false or true — for Boolean properties**
- **more/less, more/less or equal, equal/not equal — for numeric properties**

In the segmentation rule, select the presence or absence of direct or reverse, and add the necessary parameters.

TIP

For example, you can use the rule below to find customers who approve of your projects and even donate more than 200 currency pieces.



2. Dynamic content in templates

Relation values can be added to [message template] using special variables.

In the variables menu, select direct or feedback and the desired value for substitution.

> **TIP**
>
> **For example, for an online training center, you can notify methodologists about how many hours students spent studying with certain instructors, and for clients - provide an up—to-date schedule of the instructors they prefer.**

Relation variables can also be inserted manually using the following editor script:

```
{relation.[short_connection_name].[direct for direct or reverse for reverse].[property name].[count – quantity, total – sum, top – top relations]}
```

The number of direct relations to the profit property:

```
{relation.managers_loyal.direct.profit.count}
```

Top feedbacks with the `profit` property. A loop is used to output list of values.

> **TIP**
>
> **The loop substitutes the data of those profiles that are specified in the relation, and not from the database through which the mailing is sent. Thus, if you send an email to your agent, it may use personal customer data stored in a separate database.**

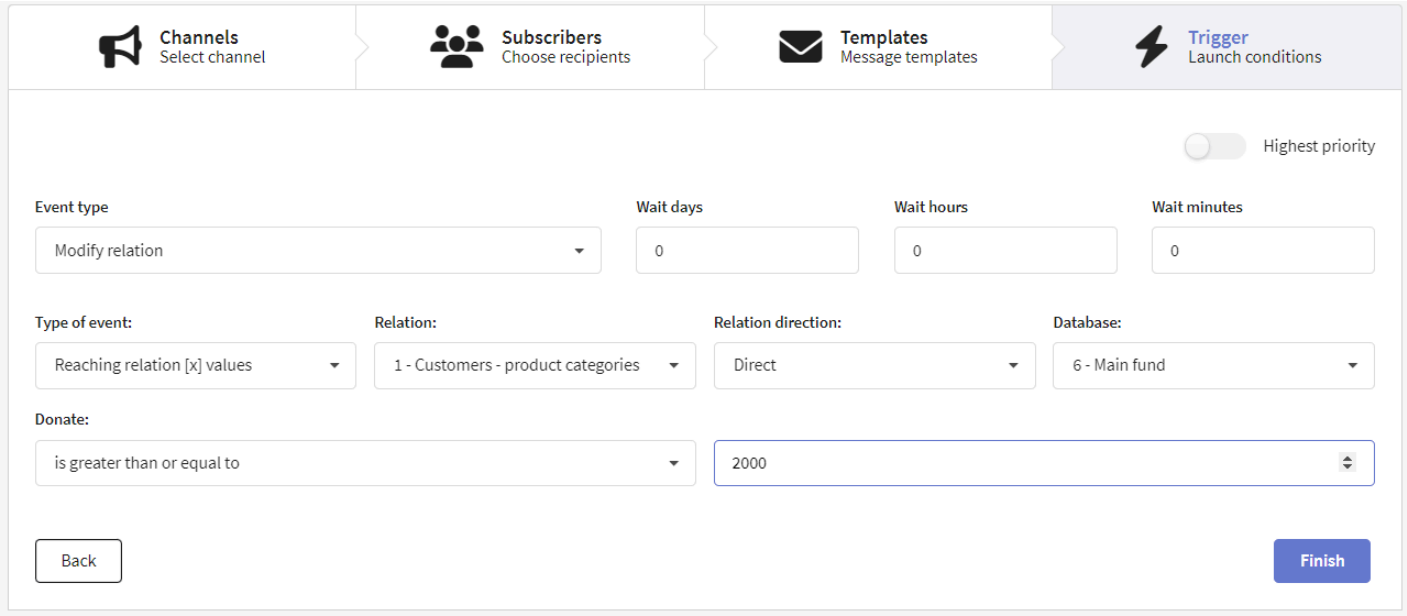
```
{for $index $item = relation.managers_loyal.reverse.profit.top}
{$item.lead._fname} {$item.lead._lname} ({$item.value} profit)
{else} There is no ‘profit’ for you :(
{end}
```

The sum of the value of the `profit` property for direct relation:

```
{relation.managers_loyal.direct.profit.total}
```

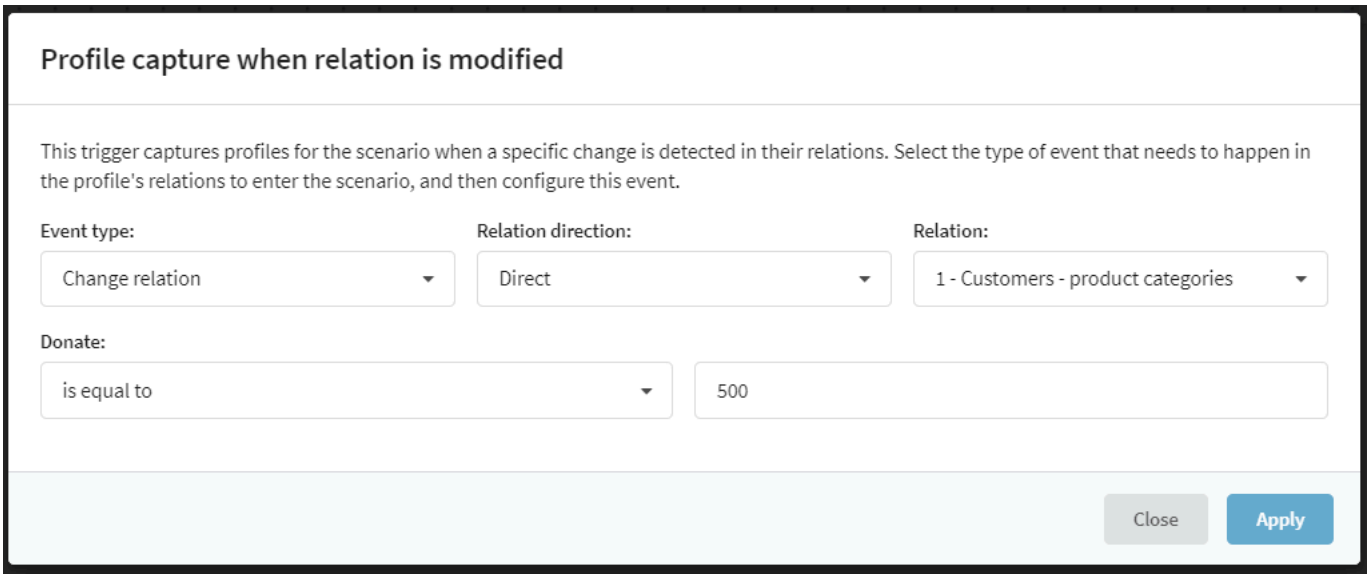
3. Trigger mailing or automation scenario

A [trigger mailing] can be configured to create or delete a relation:



The trigger can be triggered when both the direct and reverse relation between profiles changes. This may take into account the appearance, change or deletion of the relation, as well as the achievement of the relation property set in the trigger value.

**Or connections can be implemented in [automation scenario](): **



TIP

For example, if you have a referral program, set up 2 different automated workflows for both referral and for customers engaged by the referral:

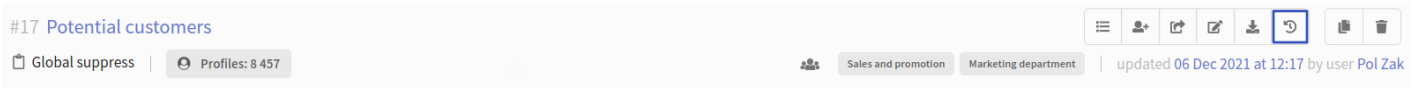
for the referral — a trigger “When a direct relation is attached”. It will launch when a referral engages a new customer for you. Use this scenario or trigger to further motivate your referral. You can additionally specify values, like profit of the initial transaction.

for the engaged customers — a trigger for reverse relation. Use it to promote your products and motivate customers to become new referrals.

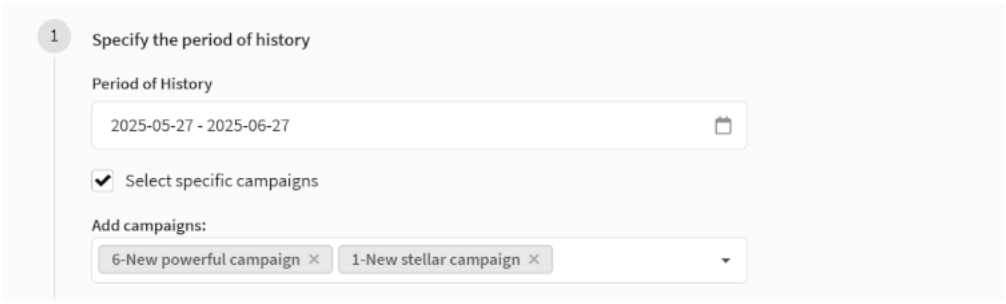
Profile history export

Profile history export allows you to download data on profile activity from the platform. The data is usually in a specific database or a segment. The data will be exported to a CSV file.

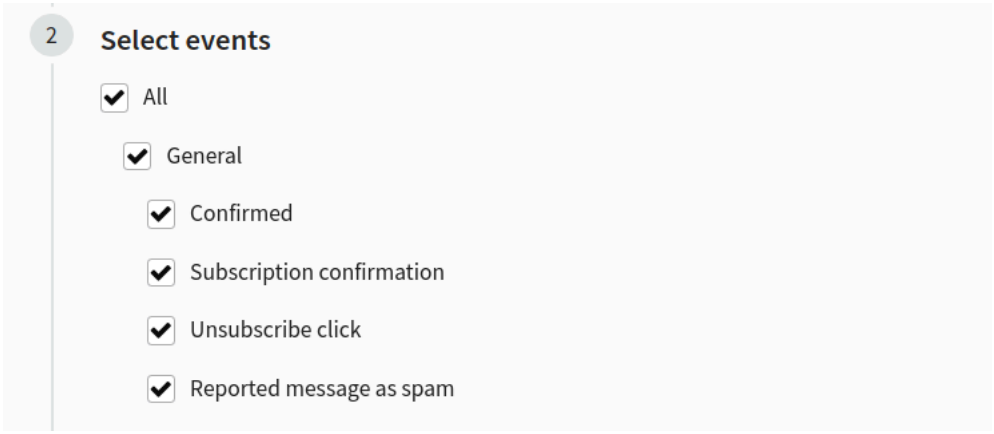
- **Open a list of databases or segments, find a necessary base and click the icon Export profile history.**



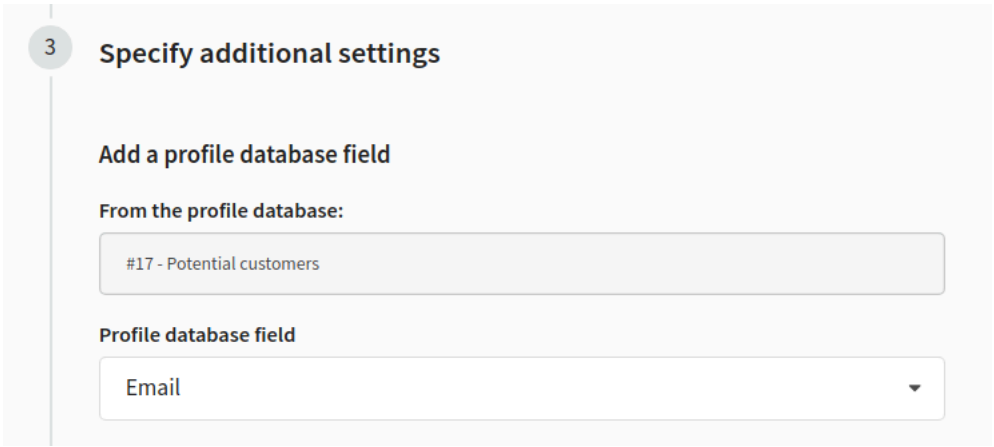
- **Specify the period for which the data needs to be unloaded. You can also select mailings if necessary:**



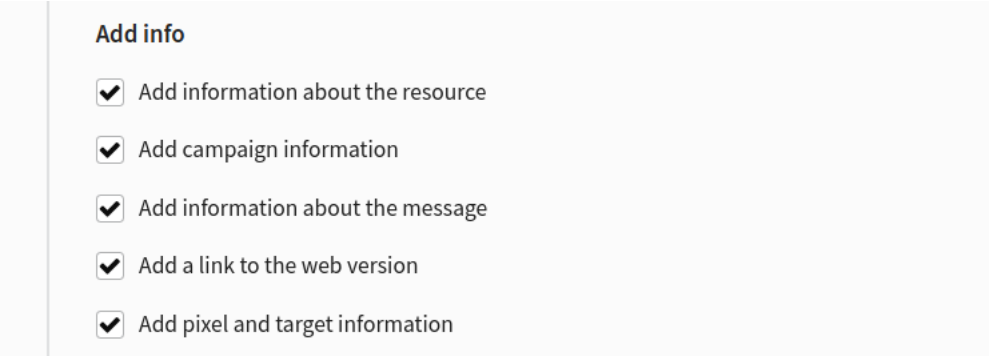
- **Select events to export:**
  - **Basic (subscription confirmation, complaints, unsubscription, etc.)**
  - **Activity in various communication channels (message delivery, openings, clicks, etc.)**
  - **Pixel and goal registration**
  - **Promo codes**
  - **Connections**



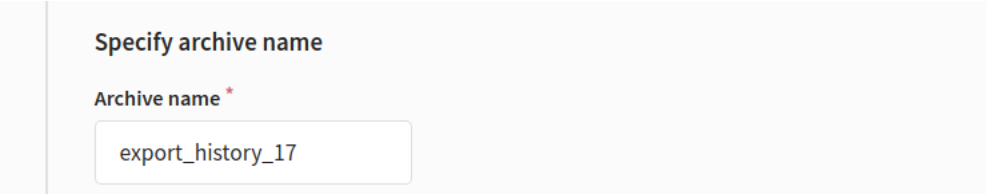
- **Specify the profile database field by which you want to group the unloaded data:**



- **Add additional information for export:**
  - **resource— add ID and the name of the resource to which the profile has subscribed to the target file**
  - **mailing — add ID and the name of the mailing in which the event occurred;**
  - **message— add ID and the name of the template used;**
  - **add a link to the web version of an email;**
  - **pixel and goal — add ID and the name of the pixel as well as the name and value of the goal registered by the pixel**

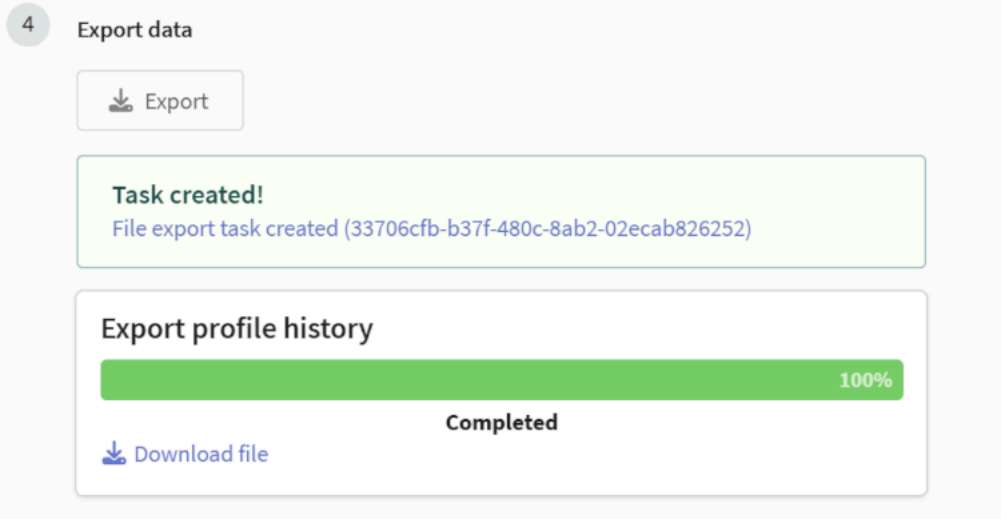


- **Enter an arbitrary name for the archive:**



- **Export the data by clicking Export. An export task will be generated automatically. Its progress will be displayed immediately:**

Once the task is complete, click Download file. Archive in `.zip` format will be downloaded to your device:

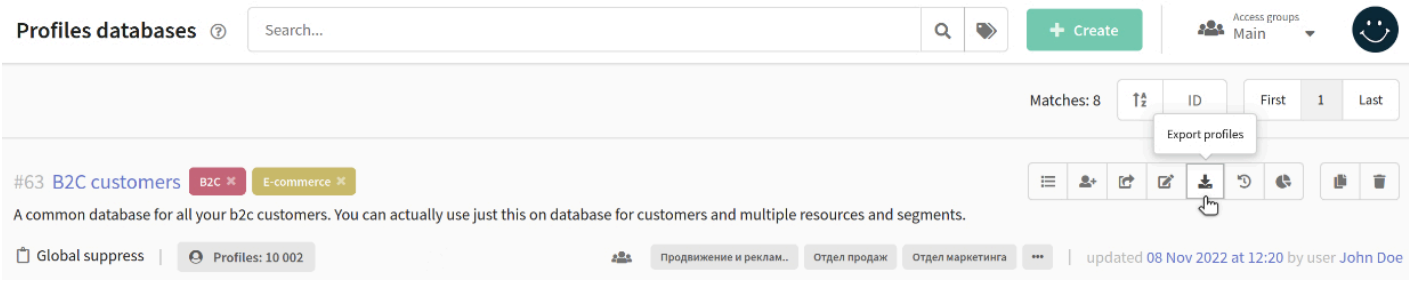


Profile export

Profile history export allows you to download information about all profiles in the database or segment. The data will be downloaded as a csv file. You can choose the data you want to receive.

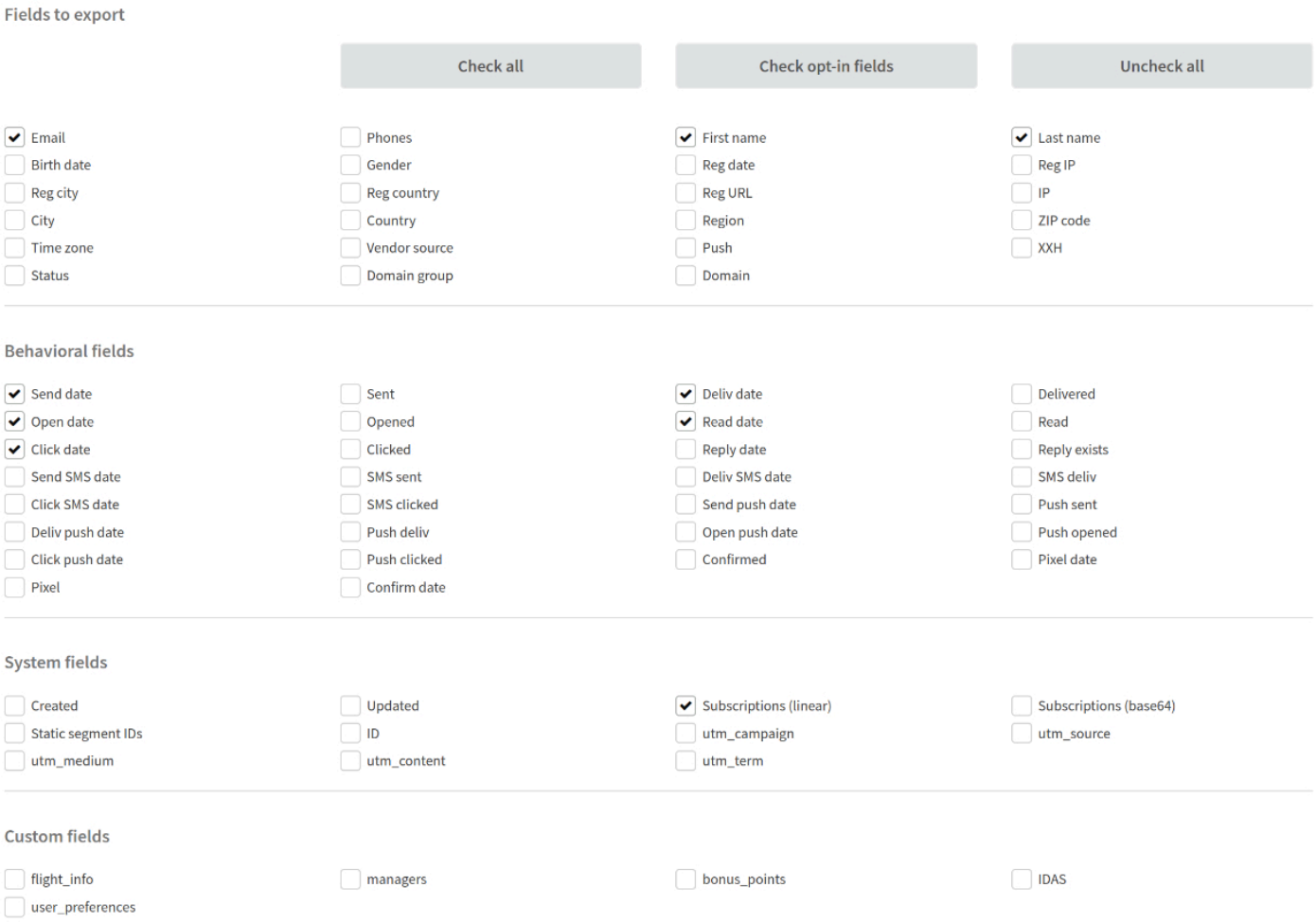
To start exporting, select a database or segment and click

in the toolbar:



Export fields

Check the profile fields to be downloaded. The buttons at the top of the page can help here — check all fields, uncheck all of them, or check only opt-in fields. Opt-in data includes registration information: email, date, links and IP address at the time of registration.



The following profile fields are available for export:

- **default fields;**
- **behavioral fields;**
- **system fields;**
- **custom fields.**

The table below describes behavioral and system fields:

Behavioral fields

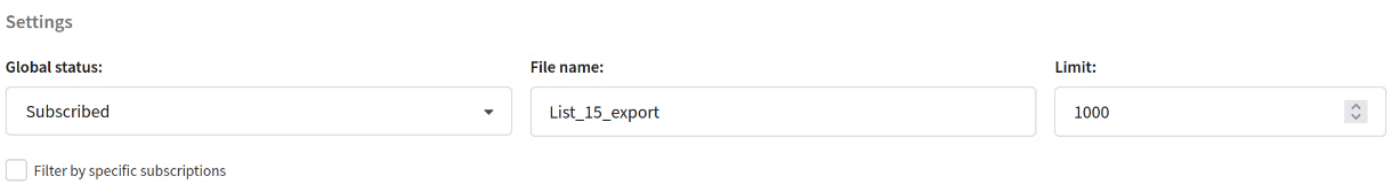
Export field	Column in file	Description
Send date **Send SMS date** **Send push date**	**SendDate** **SendSMSDate** **SendPushDate**	**Date and time of the first message sent (Email, SMS, Push)**
Open date **Open push date**	**OpenDate** **OpenPushDate**	**Date and time when the message was last opened (Email, Push)**
Click date **Click SMS date** **Click push date**	**ClickDate** **ClickSMSDate** **ClickPushDate**	**Date and time of the message was last clicked (Email, SMS, Push)**
Deliv date **Deliv SMS date** **Deliv push date**	**DelivDate** **DelivSMSDate** **DelivPushDate**	**Date and time when the message was last delivered (Email, SMS, Push)**
Read date	**ReadDate**	**Date and time when the message was last read (Email)**
Reply date	**ReplyDate**	**Date and time when the message was last read (Email)**
Confirm date	**ConfirmDate**	**Date and time when the confirmation link was clicked**
Pixel date	**PixelDate**	**Date and time when the pixel was last activated**
Sent **SMS sent** **Push sent**	**SendExists** **SendSMSExists** **SendPushExists**	**Whether a message was sent for the entire period of interaction with the profile or not (Email, SMS, Push)**
Opened **Push opened**	**OpenExists** **OpenPushExists**	**Whether a message was opened for the entire period of interaction with the profile or not (Email, Push)**
Clicked **SMS clicked** **Push clicked**	**ClickExists** **ClickSMSExists** **ClickPushExists**	**Whether a message was clicked for the entire period of interaction with the profile or not (Email, SMS, Push)**
Delivered **SMS delivered** **Push delivered**	**DelivExists** **DelivSMSExists** **DelivPushExists**	**Whether a message was delivered for the entire period of interaction with the profile or not (Email, SMS, Push)**
Read	**ReadExists**	**Whether a message was read for the entire period of interaction with the profile or not (Email)**
Reply exists	**ReplyExists**	**Whether a message was replied for the entire period of interaction with the profile or not (Email)**
Confirmed	**ConfirmExists**	**Whether there was a click on the confirmation link in the message for the entire period of interaction with the profile or not**
Pixel	**PixelOpenExists**	**Whether any pixel was activated for the entire period of interaction with the profile or not**

System fields

Export field	Column in file	Description
Created	**CreateTime**	**Date and time the profile was created**
Updated	**UpdateTime**	**Date and time the profile was updated**
Subscriptions (linear)		**The following subscription data is exported:** • **subscription_resource_id** • **subscription_channel** • **subscription_status** • **subscription_priority** • **subscription_hash_id** • **subscription_is_delete (whether the subscription to the resource was deleted or not)** • **subscription_email** • **subscription_phone** • **subscription_subscription_id (push subscription ID)** • **subscription_provider (push provide)** • **subscription_date** • **subscription_ip** • **subscription_url**
Subscriptions (base64)		**Subscription data is exported as a base64 encoded string**
utm_campaign **utm_source** **utm_medium** **utm_content** **utm_term**	**UtmMailing** **UtmSource** **UtmMedium** **UtmContent** **UtmTerm**	**UTM tags**
Static segment IDs	**StaticSegments**	**ID of the static segment that the profile belongs to**
ID	**Id**	**profile_id**

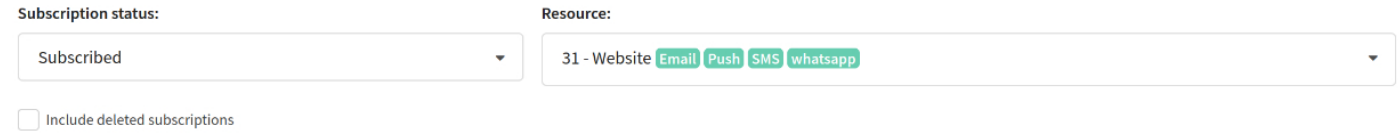
Export settings

In this block, you set additional export settings. You can export the data of profiles that have a certain global status, as well as limit the number of downloaded profiles:



To download information about profiles with a subscription to a specific resource, activate the “Filter by specific subscriptions” option. Select a resource and optionally a subscription status.

In some cases, a resource subscription is added to a profile, but deleted after some time. If you want the file to contain profile data with a deleted subscription to the selected resource, check the box “Include deleted subscriptions”.



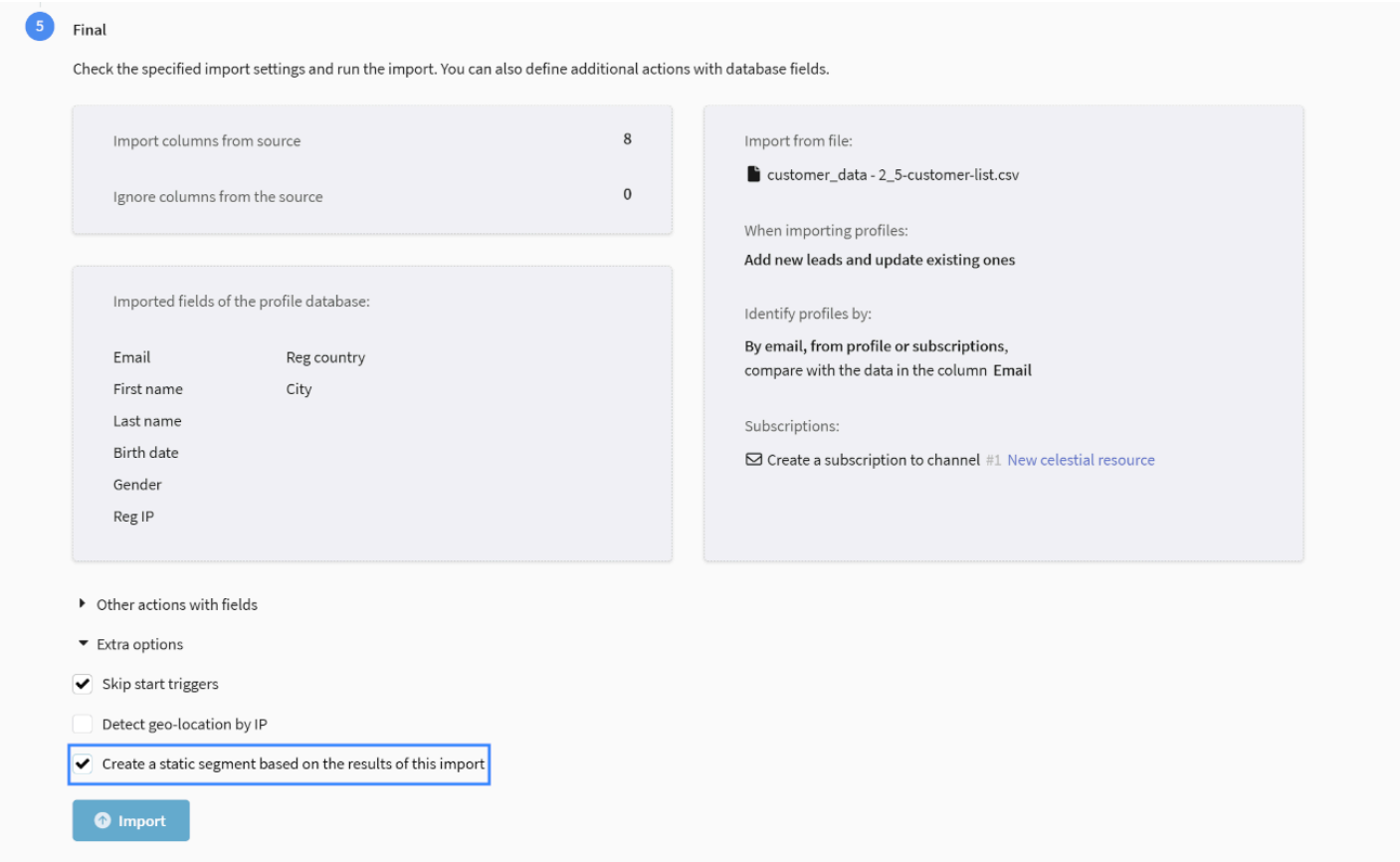
Once you have selected the fields to export and set up the export details, click on the Export button. The platform will start the process. Go to the “Tasks” section to download the generated file containing profile data.

Creating a static segment based on import results

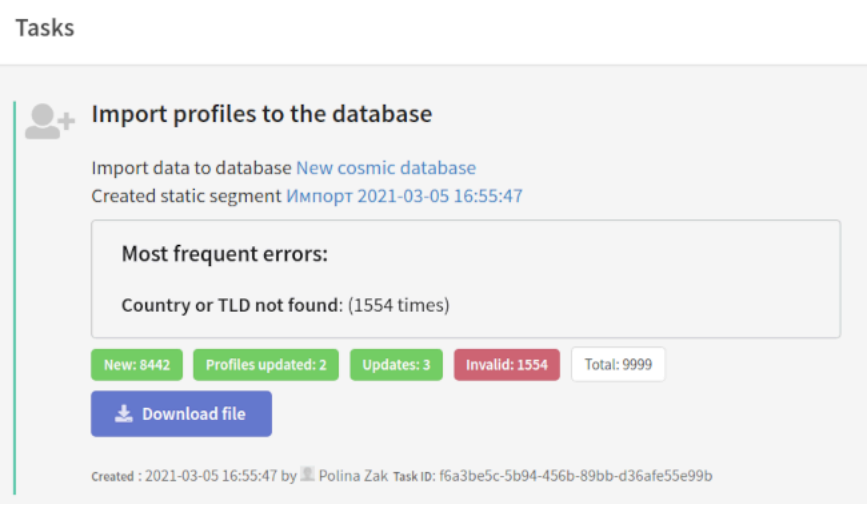
If we import profiles into the database using a file, in the import details, we can enable the option Create a static segment from the results of this import.

INFO

If new profiles are regularly added to the same database, a static segment is an easy way to group them for a one-time mailing



We can check the process of import and the process of creating a segment in Tasks:



This way, as soon as the import is completed, a new static segment will be created. It will include the newly imported profiles. The name of the static segment will be generated automatically (for example, `"Import 2021-03-51 16:55:47"`).

How to open a CSV file

What is CSV

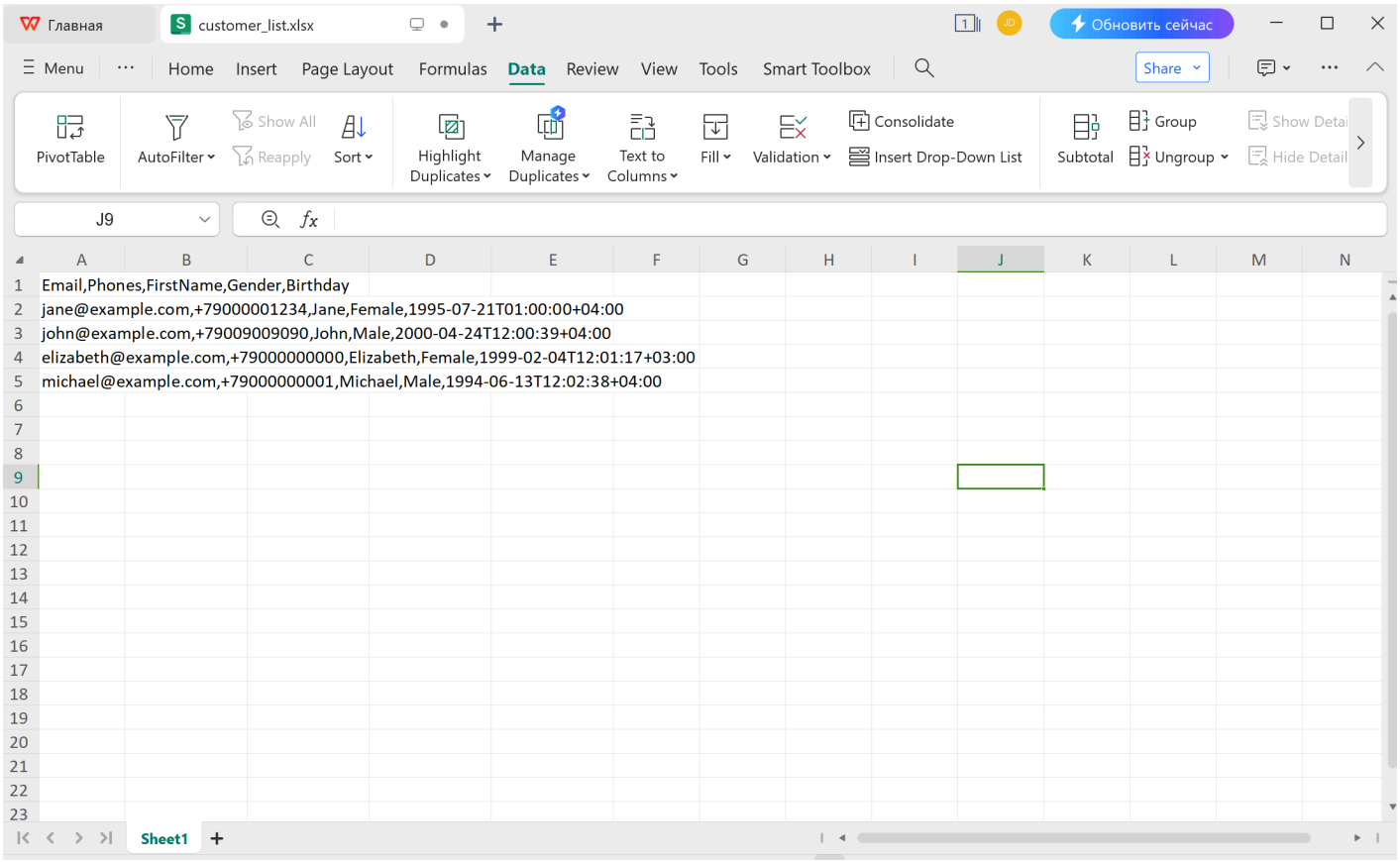
CSV (Comma-Separated Values) is a text format designed to represent tabular data. The data in such a file is separated by commas or other delimiters, such as semicolons or tabs.

```
Email,Phones,FirstName,Gender,Birthday
jane@example.com,+79000001234,Jane,Female,1995-07-21T01:00:00+04:00
john@example.com,+79009009090,John,Male,2000-04-24T12:00:39+04:00
elizabeth@example.com,+79000000000,Elizabeth,Female,1999-02-04T12:01:17+03:00
michael@example.com,+79000000001,Michael,Male,1994-06-13T12:02:38+04:00
```

Using CSV files in the platform

CSV files are used in the platform for the following purposes:

- **Importing profiles into the database from a file**
- **Exporting profiles and their history from the database**
- **Generating reports**
- **Exporting task results, etc.**



Spreadsheet editors like Excel or WPS office are often used for working with CSV files. However, when opening a file, the data may sometimes be displayed incorrectly. The main issue is that Excel incorrectly determines the delimiter used and places all the data in a single column:

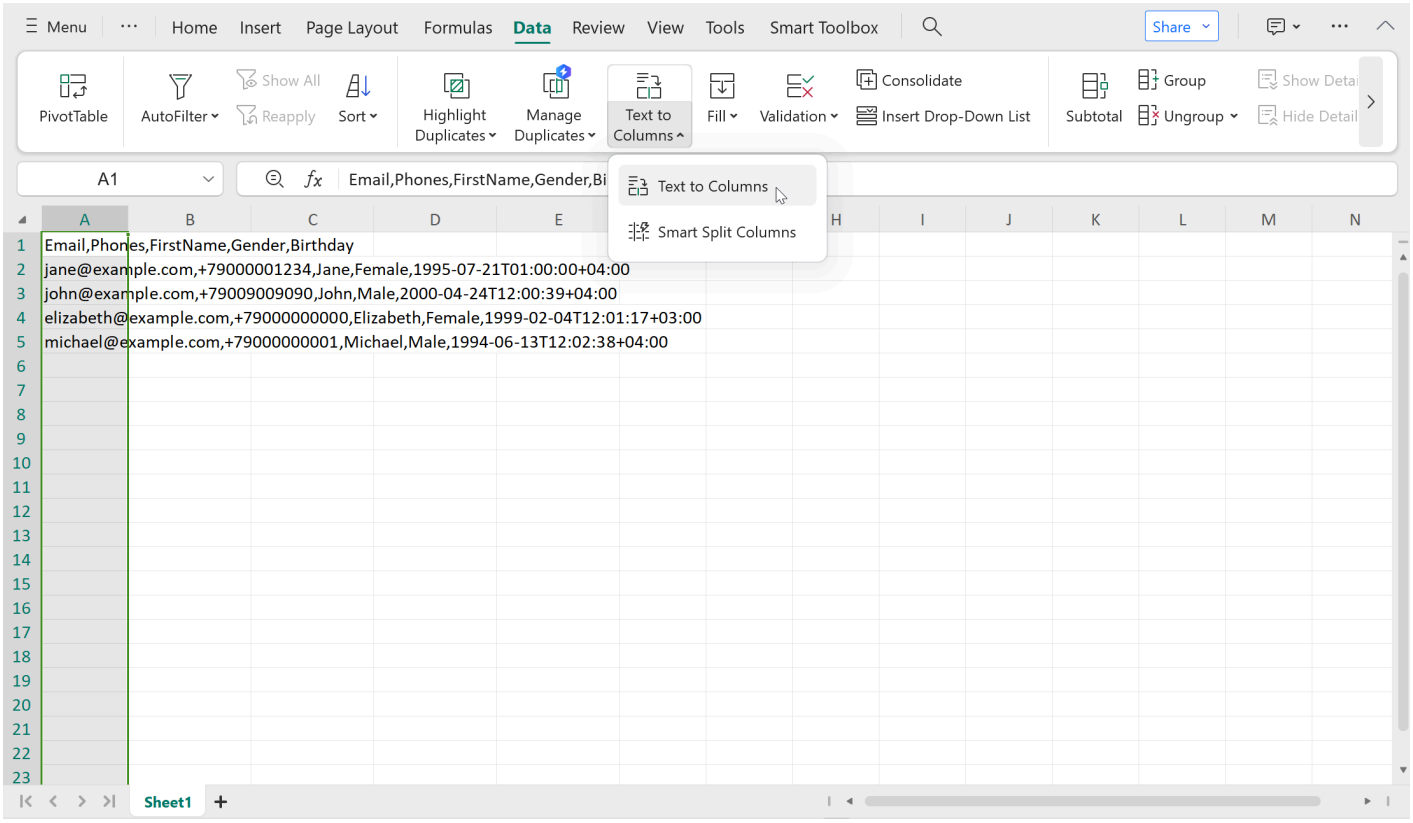
- **Excel incorrectly detects the delimiter used and places all data in a single column**
- **If the file is saved in UTF-8 encoding, Excel may fail to recognize Cyrillic characters and display them as unreadable symbols**

Below you will find instructions on how to properly open a CSV file. The interface may vary slightly depending on the program you use.

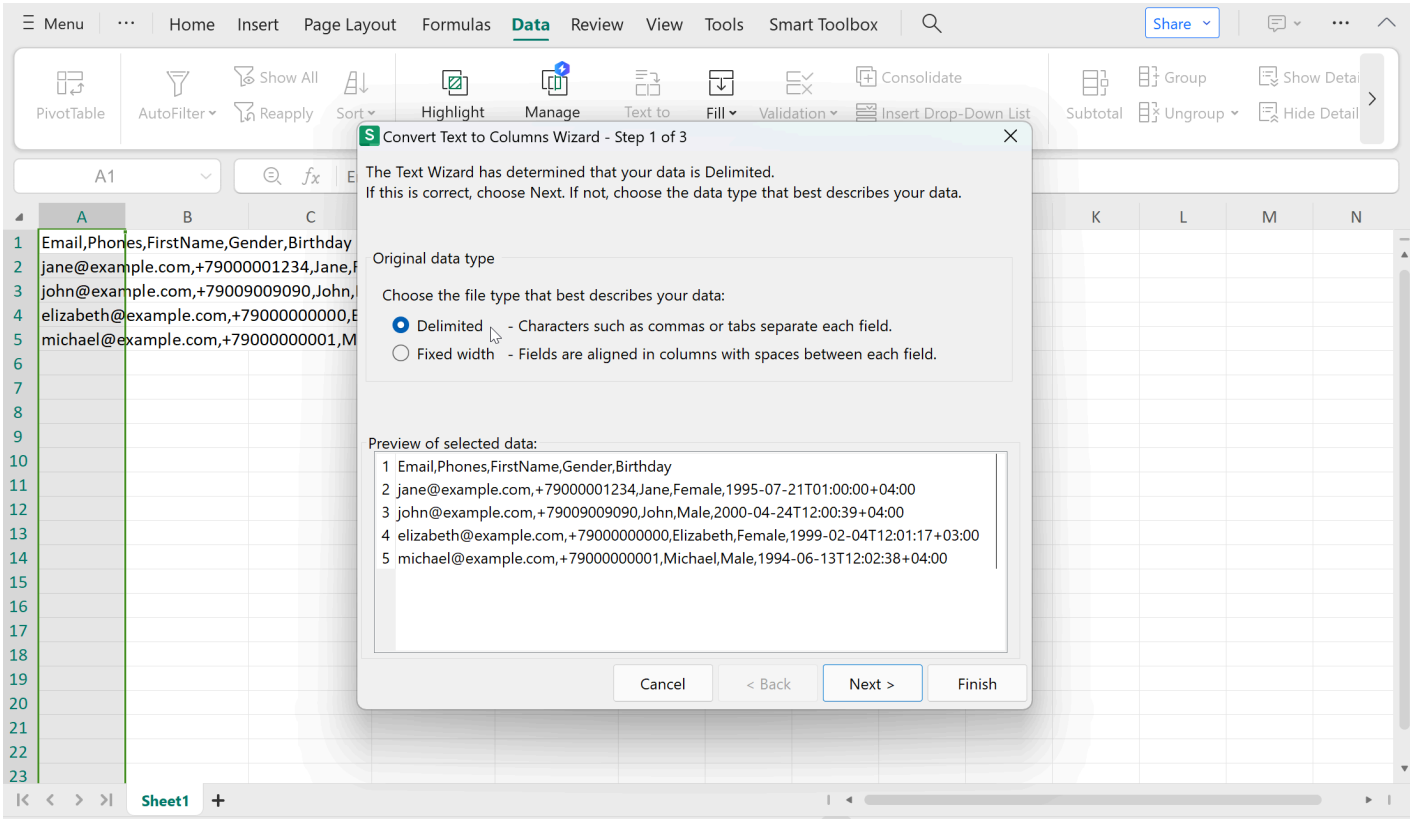
Guide to opening a CSV file

Manual CSV Data Parsing

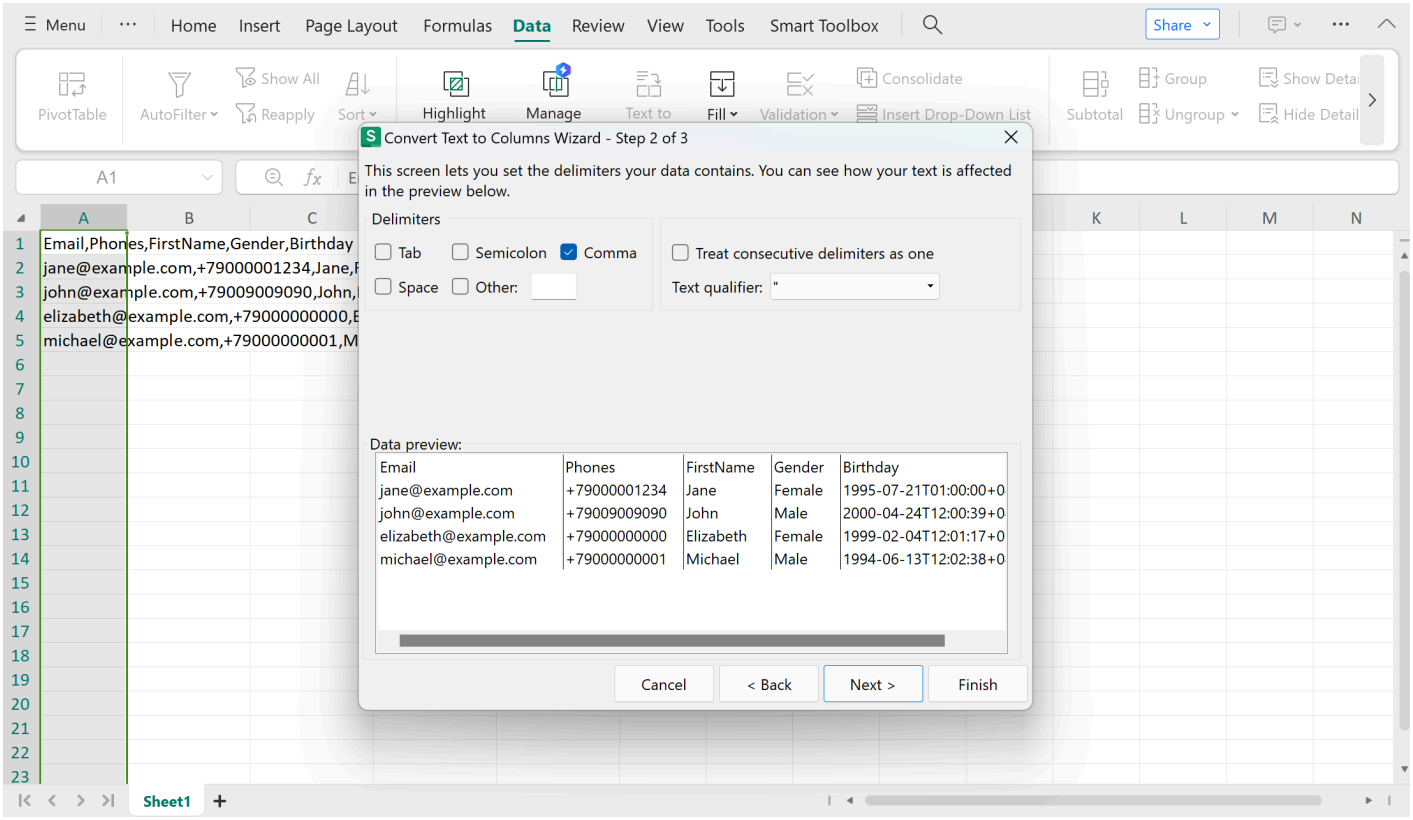
Open the file, select the entire first column, and go to the Data tab on the top panel. Click Text to Columns:



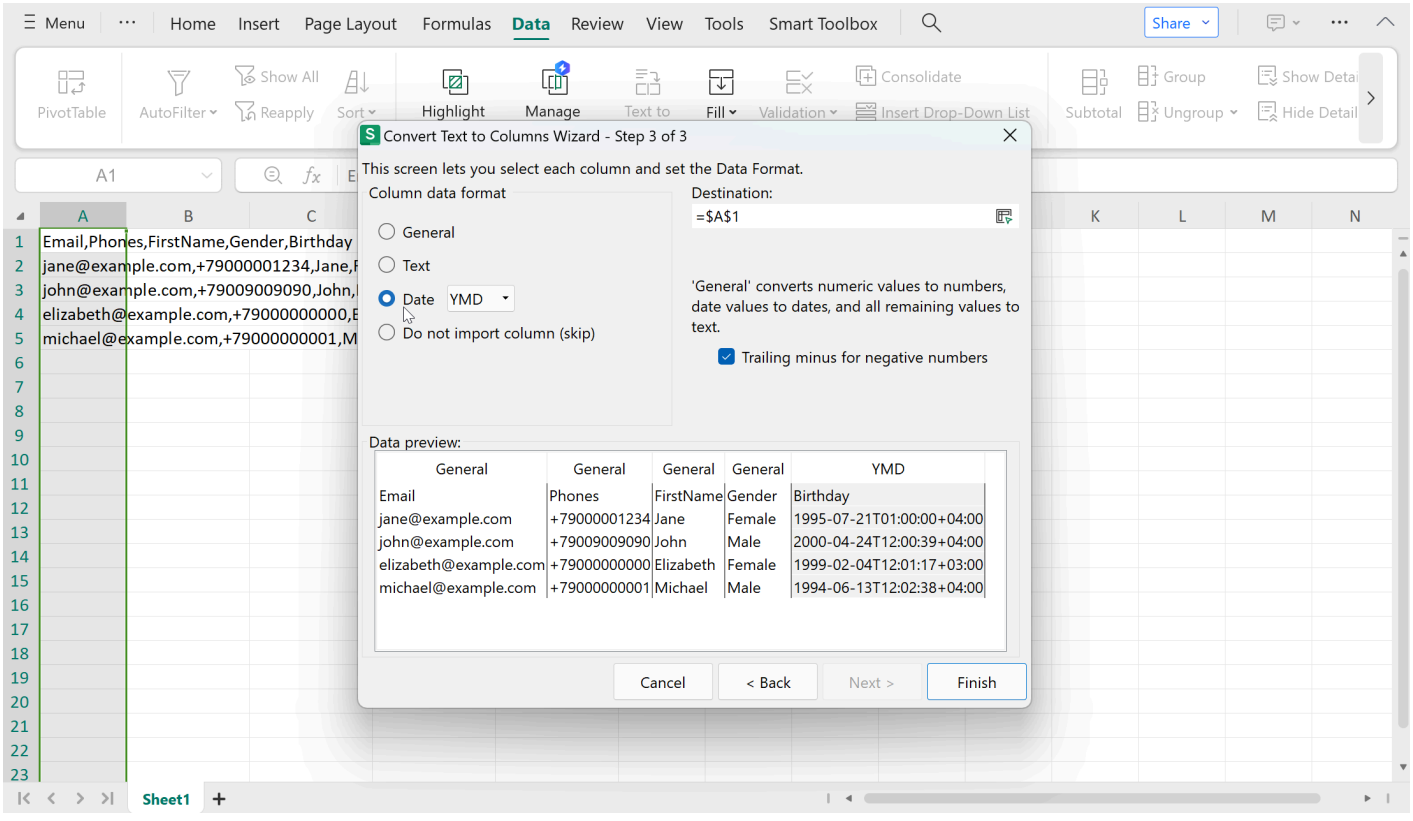
The program will open the text-to-columns wizard to help you correctly interpret the data. In the first step, specify the data format as Delimited. Click “Next”:



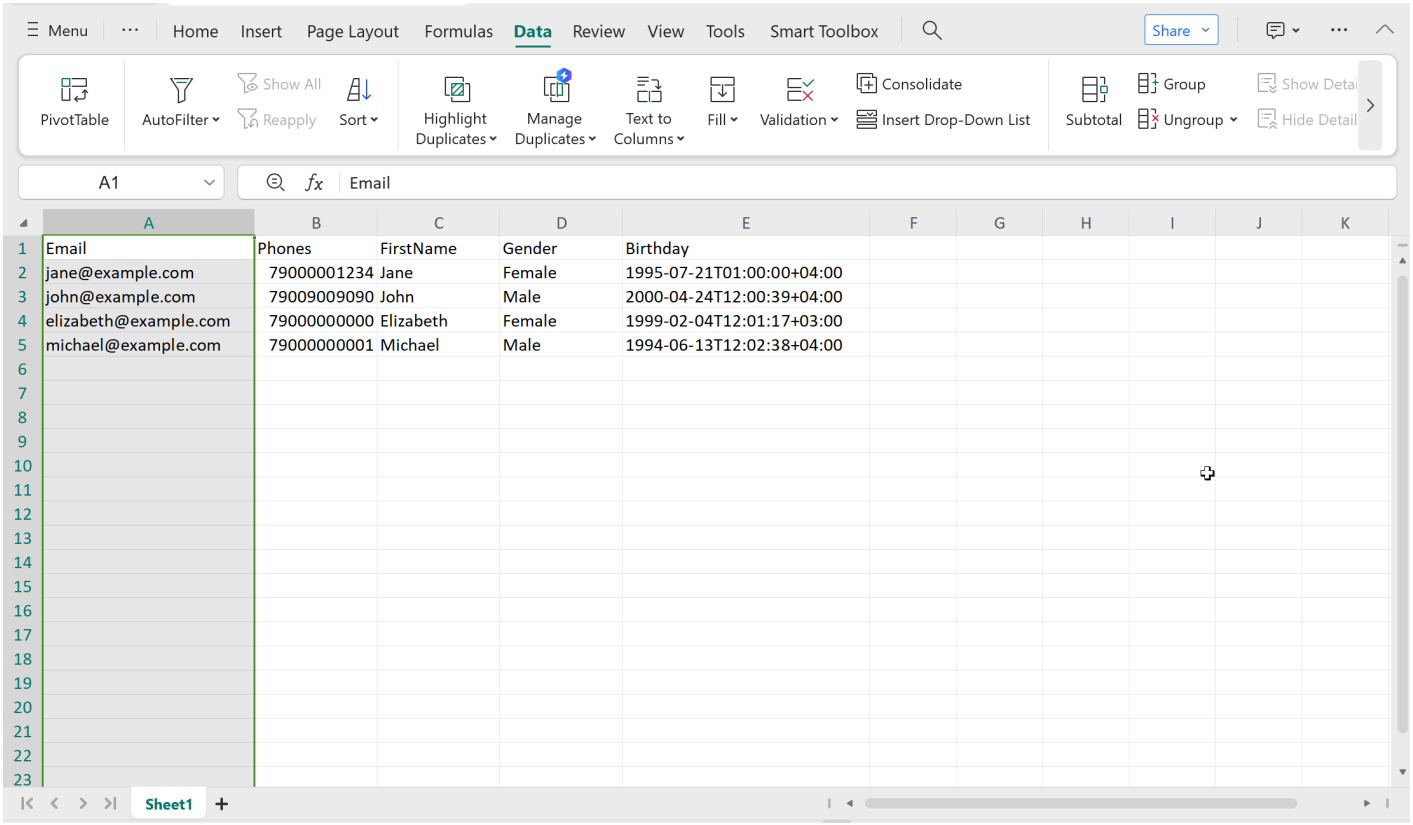
In the second step, choose Comma as the delimiter. Click “Next”:



In the third step, if necessary, you can specify the data format for the columns. In the example below, the Date format (year-month-day) is set for the Birthday column. Then click “Finish”:



If all steps are done correctly, the data in the file will be distributed across the appropriate columns:



	A	B	C	D	E
1	Email	Phones	FirstName	Gender	Birthday
2	jane@example.com	79000001234	Jane	Female	1995-07-21T01:00:00+04:00
3	john@example.com	79009009090	John	Male	2000-04-24T12:00:39+04:00
4	elizabeth@example.com	79000000000	Elizabeth	Female	1999-02-04T12:01:17+03:00
5	michael@example.com	79000000001	Michael	Male	1994-06-13T12:02:38+04:00

UTF-8 Encoding

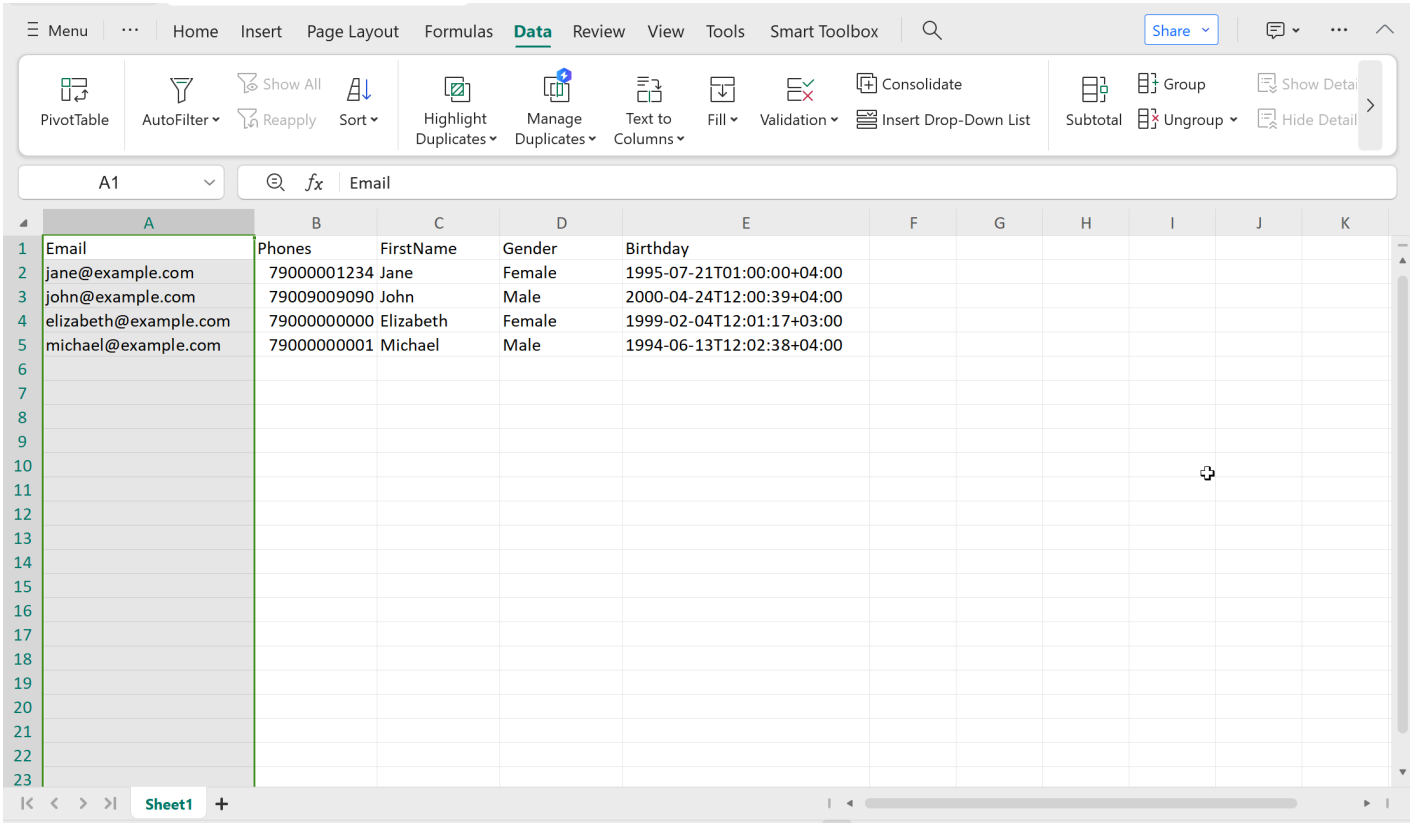
Opening files in UTF-8 (When the file contains Cyrillic characters that display incorrectly)

1. Go to the Data tab —> Import Data (or Edit Data Import if the file is already open):

2. Select the desired CSV file and specify UTF-8 encoding:

4. Select comma as the delimiter:

5. Click Finish, and the data will be properly distributed across columns:

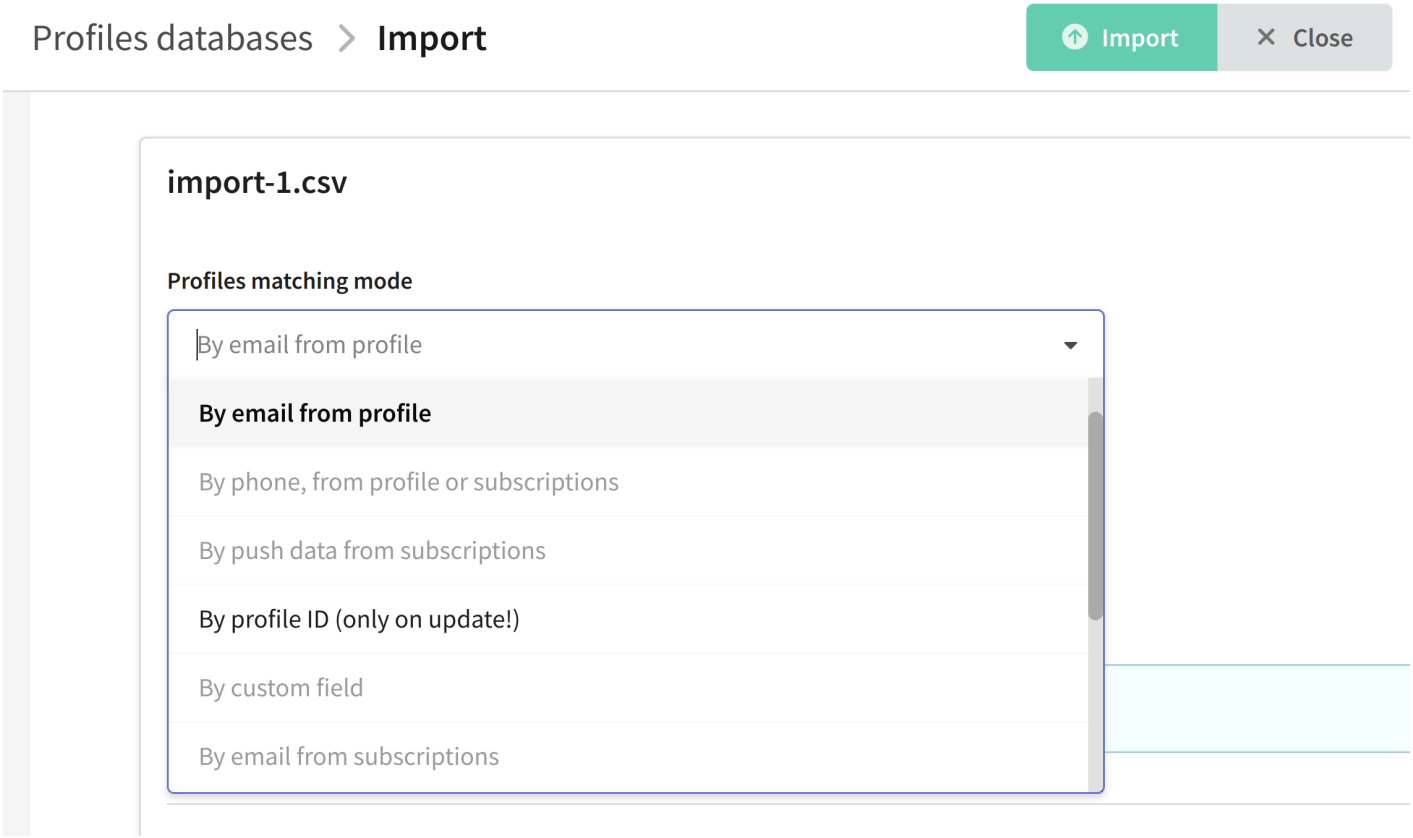


	A	B	C	D	E
1	Email	Phones	FirstName	Gender	Birthday
2	jane@example.com	79000001234	Jane	Female	1995-07-21T01:00:00+04:00
3	john@example.com	79009009090	John	Male	2000-04-24T12:00:39+04:00
4	elizabeth@example.com	79000000000	Elizabeth	Female	1999-02-04T12:01:17+03:00
5	michael@example.com	79000000001	Michael	Male	1994-06-13T12:02:38+04:00

Matching

Matching is the process of searching and identifying existing customer profiles in a database by specific data. It is used to update customer information and prevent the creation of duplicate records in the database.

When importing profiles into the database (from a file, via API query or from an external SQL table), you specify the profile search mode:



.

During the import process, the platform compares this data with what is already in the database. If it finds a profile with the same data, it updates it. If it doesn't find it, it creates a new profile.

Note, the data you are going to use for the search must be in the source. For example, if you are importing data from a file and select email search mode, make sure you have a column with email addresses in your file.

Matching modes

The following modes are available:

- **By email, from profile or subscriptions - search by email from database or subscriptions.**
- **By email from profile - search by database profile email.**
- **By email from subscriptions - search by email from subscriptions only.**
- **By phone, from profile or subscriptions - search by phone in database field or SMS subscriptions.**
- **By phone from SMS subscriptions - search by phone from SMS subscriptions only.**
- **By push data from subscriptions - search by the identifier of push subscription (device token).**
- **By profile identifier - search by system identifier profile_id.**
- **By custom field - search by custom database field, like customer ID in company CRM.**
- **By custom field from subscriptions- search by custom database field from subscriptions only.**
- **Multimatching - by Email or phone from profile and subscriptions (email_phone), or by email or phone from subscriptions (email_phone_sub).**

Matching features

There are several important nuances related to matchmaking:

- **The field by which the search is performed must be unique for each profile. The value of this field should not be the same for different profiles in the same database.**
- **It is not recommended to use different types of matching in the same database to avoid duplicate profiles.**
- **If you do not pass the search mode in the matching field when importing or updating profiles via API, the platform will search for the profile by email address.**

INFO

If you are going to use custom field matching, you should contact the platform administrator to create an index for this field. Thanks to the index, searching for a profile in the database will take much less time.

Types of fields in the database

Default database fields

This table provides a list of default database fields, their API matches, data types, and examples of values that can be used.

Field in the database	Field name in the API	Field type in the UI	The type of the field in the API	Example of the value
Email	**email**	**Line/Email**	**string**	**"example@example.com"**
Phones	**phones**	**Phone list**	**array**	**["+79001234567"]**
Name	**_fname**	**Line**	**string**	**"John"**
Last name	**_lname**	**Line**	**string**	**"Doe"**
Date of birth	**_bdate**	**Date**	**string**	**"2001-12-31"** **"2001-12-31T00:00+03:00"** **The time can only be transmitted via the API.**
Gender	**_sex**	**Enum**	**string**	**"male"** **"female"**
Date of reg.	**_regdate**	**Date**	**string**	**"2023-12-31"** **"2023-12-31T23:59:59+03:00"**
The IP address of the reg.	**_regip**	**IP**	**string**	**"127.0.0.1"**
City reg.	**_regcity**	**Line**	**string**	**"Moscow"**
Country reg.	**_regcountry**	**Line**	**string**	**"Russian Federation"** **"RU"**
Link reg.	**_regurl**	**Line**	**string**	**"https://example.com"**
IP address	**_ip**	**IP**	**string**	**"127.0.0.1"**
City	**_city**	**Line**	**string**	**"Moscow"**
Country	**_country**	**Line**	**string**	**"Russian Federation"** **"RU"**
Region	**_region**	**Line**	**string**	**"Europe/Moscow"**
Index	**_postal_code**	**Line**	**string**	**"190000"**
Time zone	**_tz**	**Line**	**string**	**"Europe/Moscow"**
Supplier	**_vendor**	**Line**	**string**	**"form #30"**

Custom database fields

The type of field determines which operators will be available in segments and in dynamic template variables, so we recommend carefully choosing the type of additional field in the database.

Field type in UI	Field type in API	Example of the value
Integer	**int**	**100**
Float	**float**	**100.5**
Line	**string**	**"string value"**
Date	**string**	**"2024-05-27T12:00:00Z"**
Boolean	**boolean**	**true** **false**
Enum	**string**	**"Option 1"** **You can also pass a value as a number, for example, 1. The number corresponds to the ordinal number of the item in the list.**
IP	**string**	**"127.0.0.1"**
Tags	**array**	**["Tag 1", "Tag 2", "Tag 3"]**
Object	**object**	**`{ "key_1": { "sub_key": "sub_value" } }`**
Array of objects	**array of objects**	**`[ { "key_1": "value_1" }, { "key_2": "value_2" } ]`**

INFO

When API importing or updating fields with the “Tags” type, their values will be overwritten, not added to the existing ones. To add new values without deleting old ones, use functional update.

Acceptable formats and restrictions

The platform expects certain value formats for each type of field. Make sure that the imported data meets the specified requirements, otherwise the system will not be able to process them correctly.

- **Email — the format must comply with RFC standards. A regular expression is used for validation:**

```
^[-a-z0-9!#$%&'*+/=?^_{|}~]+(\.[-a-z0-9!#$%&'*+/=?^_{|}~]+)*@((?:localhost|([a-z0-9]([-a-z0-9]{0,61}[a-z0-9])?\.)*([a-z]{1,24})))$
```

- **Phones — the number must correspond to the international format (without spaces, hyphens and other separators, except for the plus sign at the beginning). More information about the standard can be found in this article.**
- **Dates — are recorded in the format RFC 3339 (ISO 8601) indicating the time zone.**

- **Country — accepted as a two-letter code ISO 3166-1 alpha-2 or the full name (in any case).**
- **Gender — acceptable values are described in this article.**
- **Provider — the length of the string is no more than 26 characters.**
- **IP — IPv4 format.**
- **Line — it is not recommended to store more than 100 KB of text in one field.**
- **Objects and arrays of objects — you can write only peer-to-peer objects or peer objects nested in other objects. If you are using objects with a more complex structure, connect the SQL database and create queries to select profiles. In object field names, only Latin letters (in any case), digits from 0 to 9, the underscore `_`, and the hyphen `-` are allowed.Data type. You can store data of type `int`, `float`, `string`, `boolean` in the Object Array or Object fields.**
- **The limit on the amount of data of one profile in the database is 16 MB, including all metadata. However, it is not recommended to store a total of more than 2 MB in additional profile fields.**

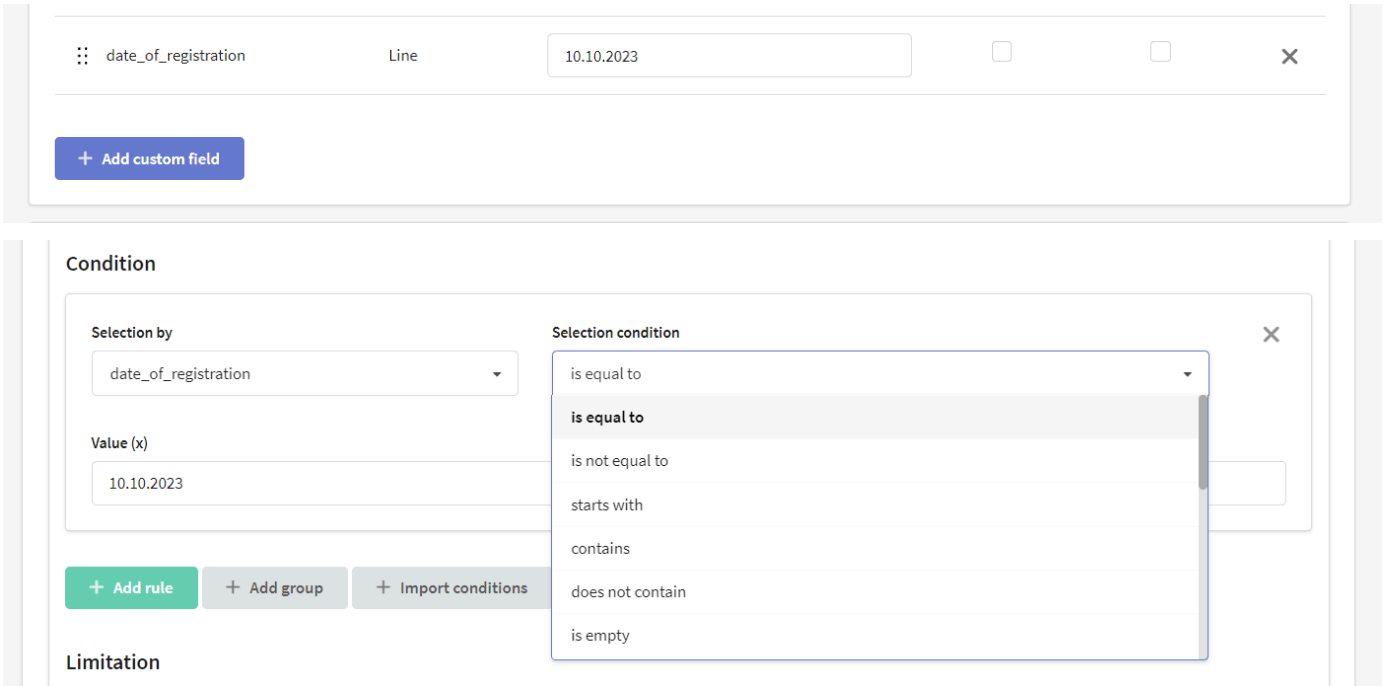
Matching of database fields and form elements

If you are configuring profile data import from form fields to database fields, make sure that you are using the correct field type for each form element.

Form element	Field in the database
Short text	**Custom field with the "String" type**
Email	**Default "Email" field**
Phone number	**Default field "Phones"**
Long text	**Custom field with the "String" type**
Number	**Custom field with the "Integer" type**
Yes/No	**Custom field with the "Boolean" type**
One choice	**Custom field with the "String" type**
Multiple choice	**Custom field with the "Tags" type**
Drop-down list	**Custom field with the "Tags" type**
Date	**Default or custom field with the "Date" type**
Range	**Custom field with the "Integer" type**

NOTE

Although you can write all user data in string fields, this will limit the possibilities for further segmentation. For example, if the date is written in a string field, you will not be able to use the comparison operators “date less” or “date more” in segments or templates.



Global control groups

What is a global control group

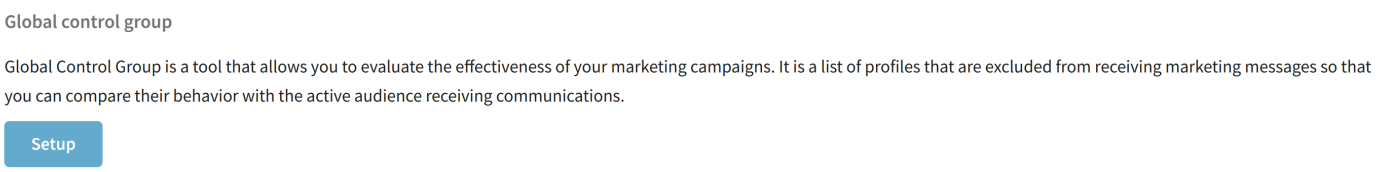
Global control groups are parts of the customer base that temporarily does not receive marketing communications in order to analyze the effectiveness of marketing strategies. After some time, the user can compare the key metrics of the main part of the customers and the control group. In case the metrics of the groups differ slightly, the marketer can conclude that his mailings are ineffective.

Having control groups allows to:

- **Evaluate the effectiveness of marketing strategies;**
- **Optimize marketing costs;**
- **Increase the conversion rate of marketing mailings.**

Configuring global control groups

You can customize the formation of a global control group for a profile database only after its creation. To do this, go to edit the created base and find the corresponding section at the bottom of the page:



The first step is to select an source audience for the GCG. This source can be either a profile database or a segment. If you have selected a segment as the source for the GCG, you will need to set up a segmentation query. You can read more about segmentation and selection conditions in this section.

The global control group based on the profile database has two ways of formation:

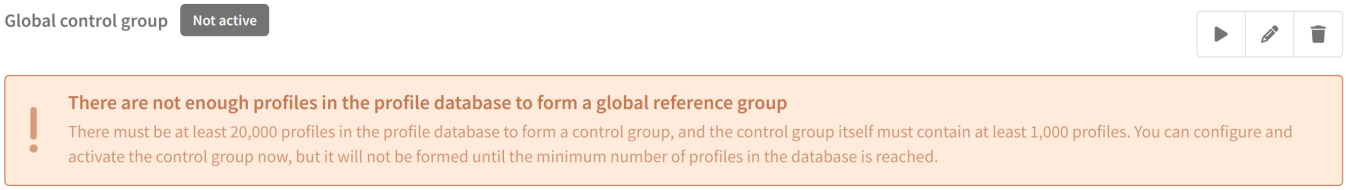
- **Random distribution — profiles for the control group are selected randomly according to the specified group size.**
- **From file — the control group will be formed from profiles in a file. The file must contain a column with an identifier to search for profiles for the control group. Keep in mind that the GCG will be generated from the entire file.**

TIP

If the random distribution method is selected for forming the audience of a global control group, the audience must meet the following requirements:

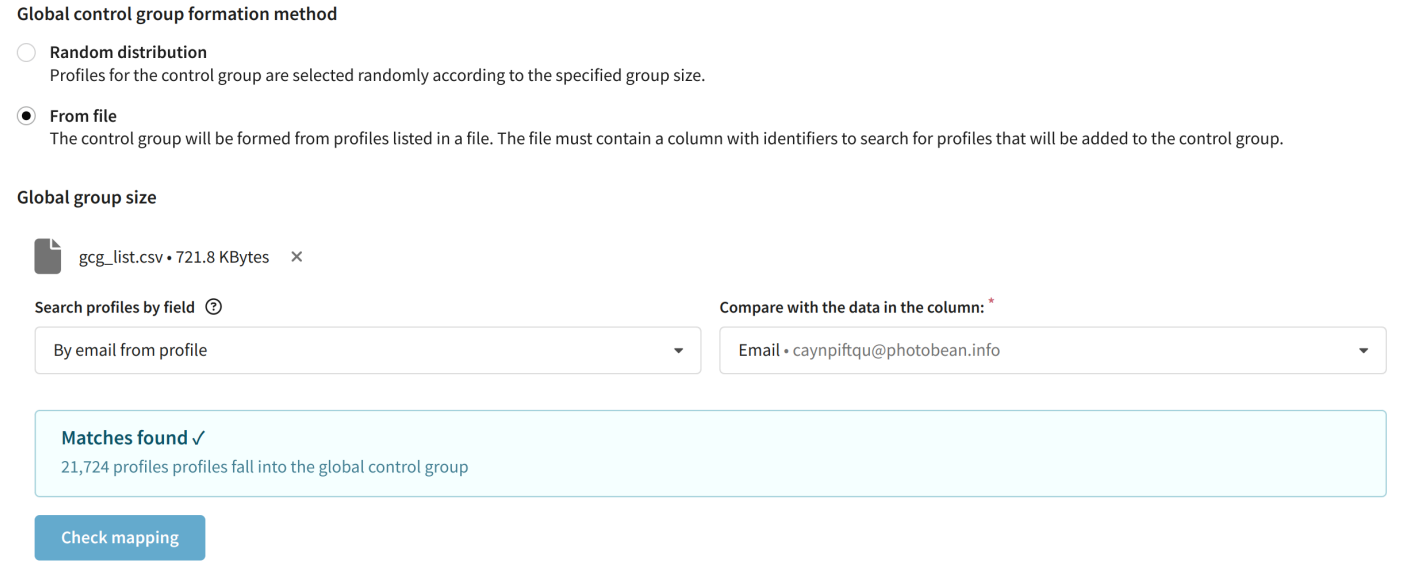
- **The GCG size must be greater than or equal to the `GCG_MIN_SIZE` parameter value — default 1,000;**
- **The GCG size must not exceed the share of the audience source defined by the `GCG_MAX_PERCENT` parameter — default 25%;**
- **The audience source size is limited by the `GCG_MAX_SOURCE_SIZE` parameter — default 50,000,000. If the source exceeds this value, the platform uses only the first records up to the limit and will not display an error;**
- **The minimum audience source size is calculated dynamically and depends on the selected GCG percentage: a smaller percentage requires a larger source. For example, with `GCG_MIN_SIZE` = 1,000 and a GCG share of 25%, the minimum source size would be 4,000 profiles, while at 5% it would be 20,000 profiles.**

You can set up and activate a control group at any time, but it will only be formed after all of the above criteria have been met. In case of any problems related to the number of profiles in the GCG, the platform will notify you:

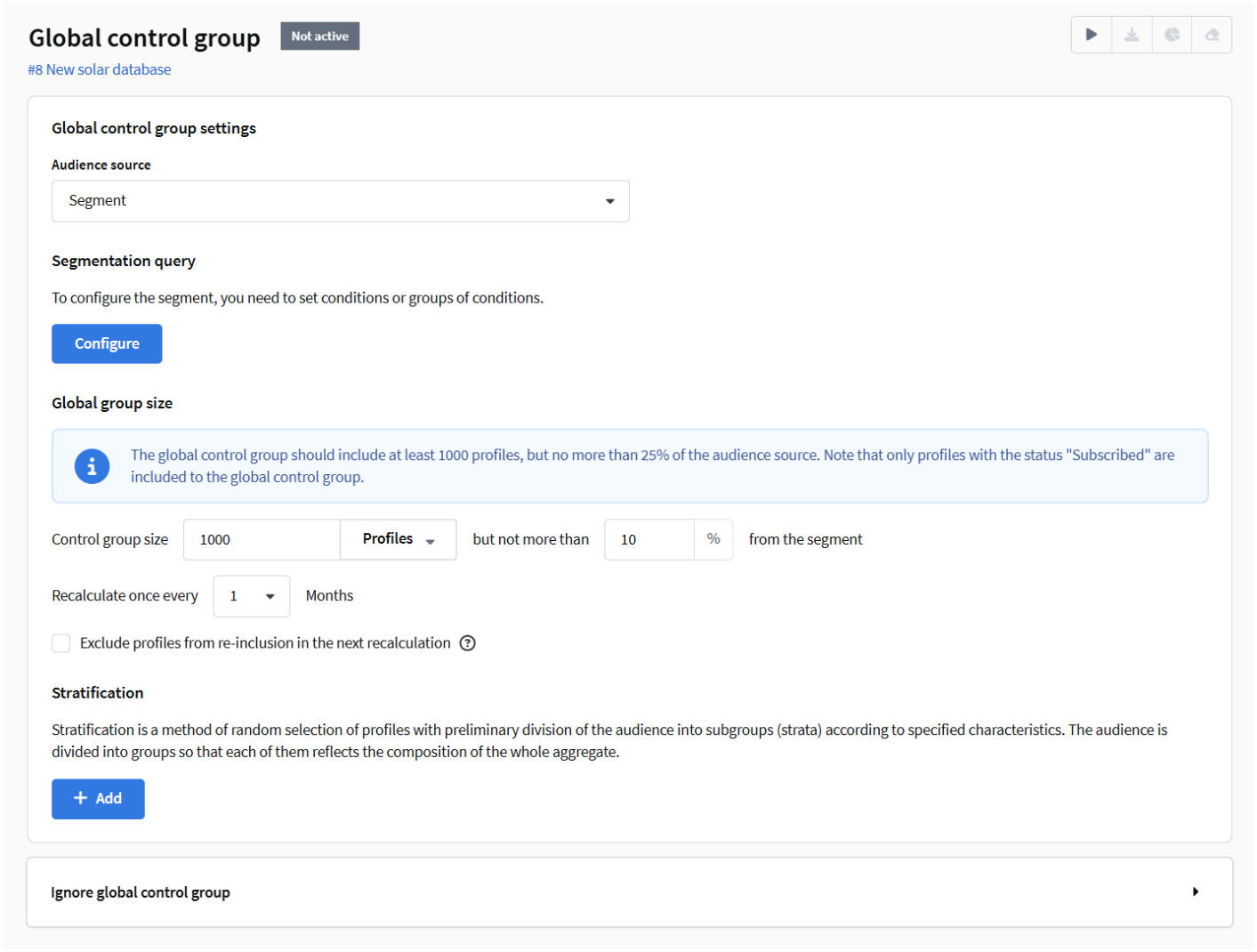


If the audience generation method from a file is used, the file contents will be used as the control group. Restrictions and additional requirements do not apply to this method. Profiles listed in the file will not receive communications.

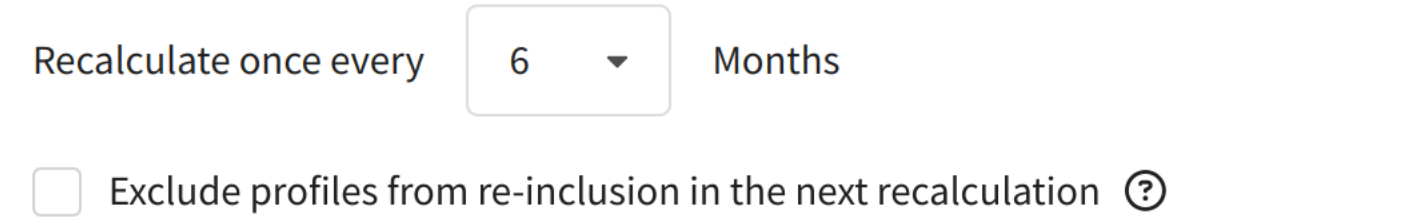
If you have chosen the option of forming from file, you will be offered to upload a file with the list of clients in .xls, .xlsx, .csv or .txt format, and then specify the field for searching profiles in the database and the column for comparison from the file. After clicking on the Verify compliance button, the platform will display a message indicating the number of profiles that will be included in the control group:



For random distribution, you can specify a specific number of profiles in the group, as well as the maximum percentage of GCGs to the entire database of profiles. The latter parameter is necessary to limit the size of the GCG in case the number of profiles in the database changes significantly:



Then you must specify the frequency of recalculation for the global control group, the minimum value is once every 12 months, the maximum value is once every 1 month. The recalculation takes place according to the specified schedule at 00:00 in the account's time zone:

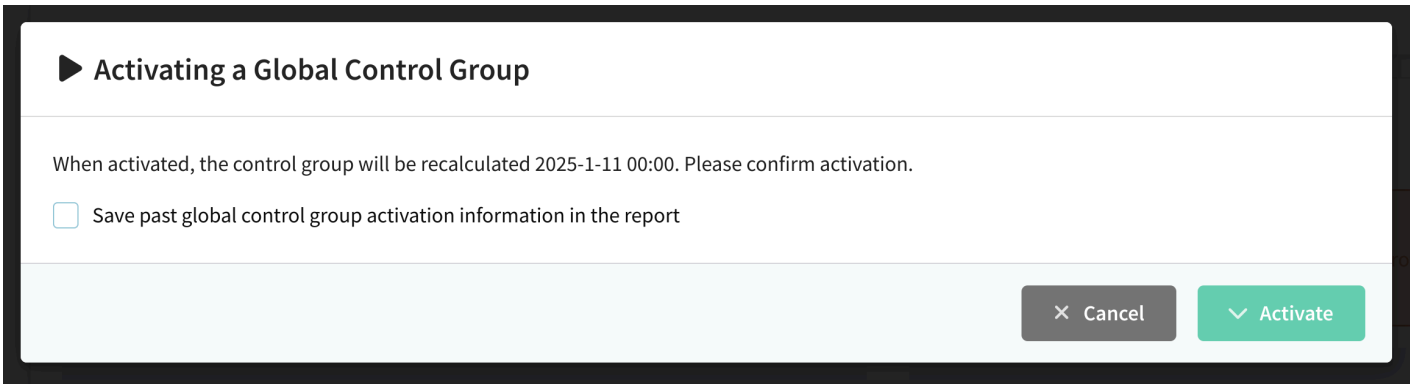


The parameter Exclude re-entry for a profile at the next recalculation allows to prohibit the profiles participating in the current control group to get there at the next recalculation. However, these profiles will be able to get to the GCG at the third and subsequent recalculations.

TIP

If the same profiles keep getting added to the control group, it can distort the results and lead to wrong conclusions. The "Exclude re-entry for a profile at the next recalculation" parameter helps change the control group, making the experiment's results more accurate.

In order for the global control group to start working, you need to activate it and save it. Before activation, the platform will inform you of the next recalculation date of the control group and offer to save information about the past activation:



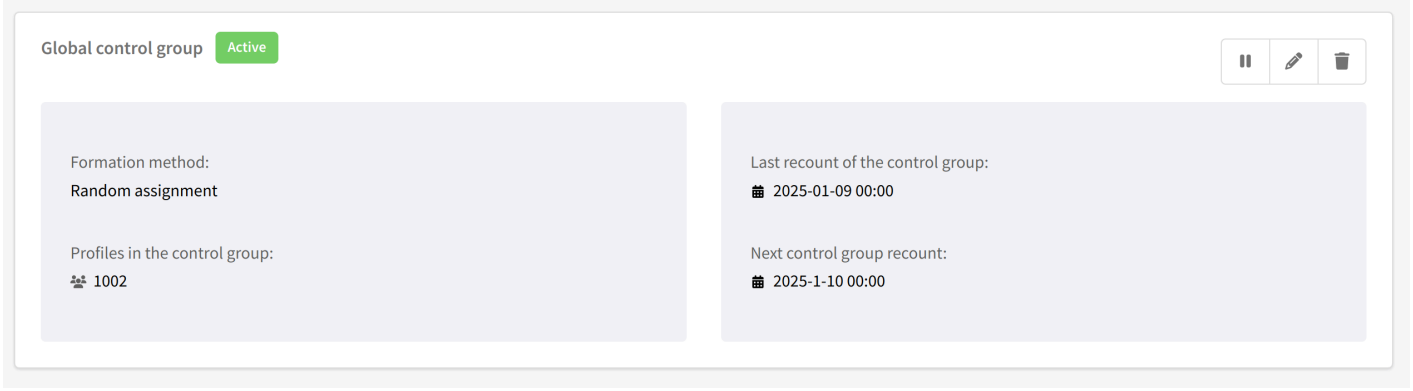
Once GCG is saved you will have access to quick-actions menu at the top of the screen:

![img](/img/attachments/gcg/en/quick-actions.png)	

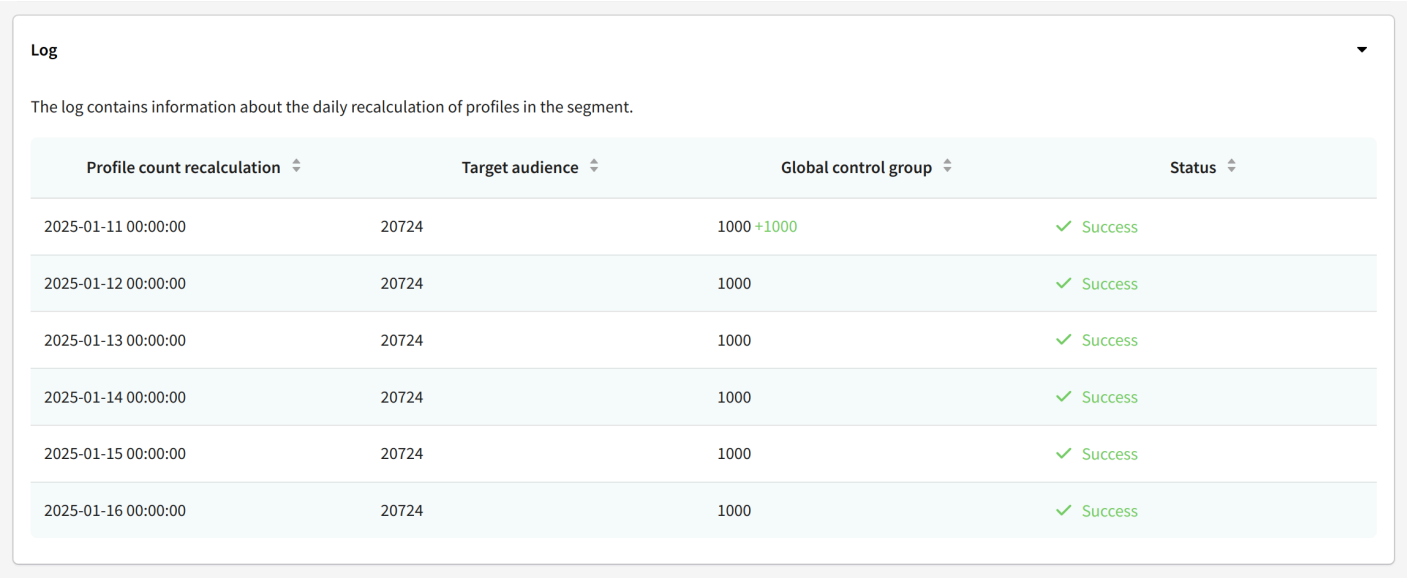
The following buttons are available to you in the menu:

- **Deactivate - stops the GCG from running**
- **Export Report - allows you to unload the list of profiles that are in the GCG**
- **Insights - opens appropriate page of analytics**
- **Reset Settings - resets GCG settings and clears all filled fields.**

Keep in mind that the first data and analytics on the participants of the GCG will be available after the first calculation. Once the CTG has been successfully set up and run, the Profile Database Edit screen will display information about the control group, the number of profiles in it, and the dates of the last and next recalculation:

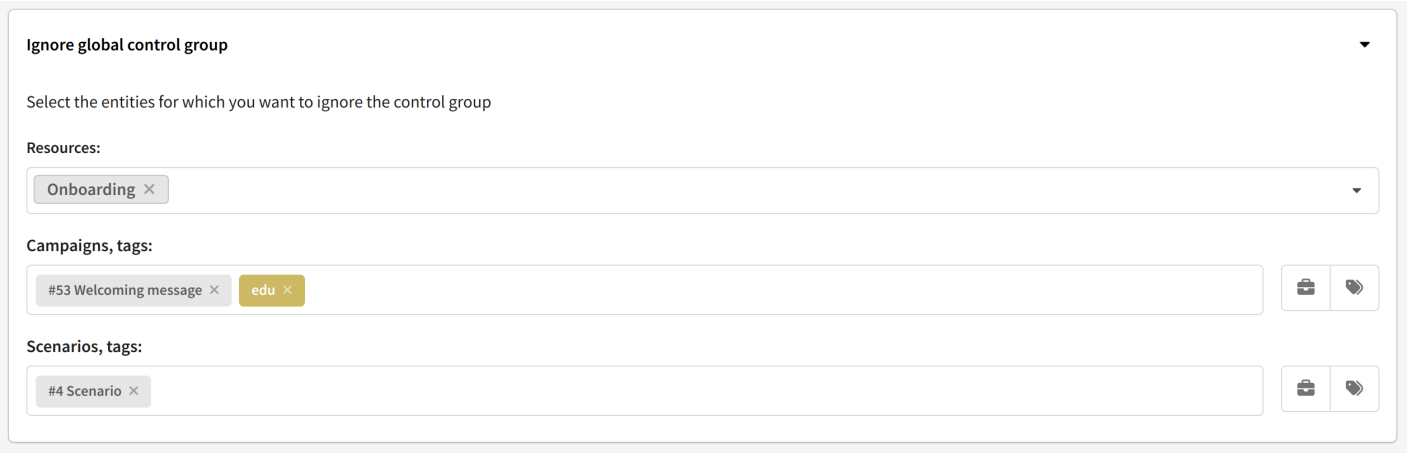


In order to view the log, which contains detailed information about GCG recalculations, you need to go to edit the control group and open the corresponding tab at the bottom of the screen:



Exclusions for the global control group

You can configure exclusions from the global control group in the Ignore global control group section. Within the selected resources, mailings and scenarios, marketing communications will be carried out even for those users who are in the global control group:



Here you can specify:

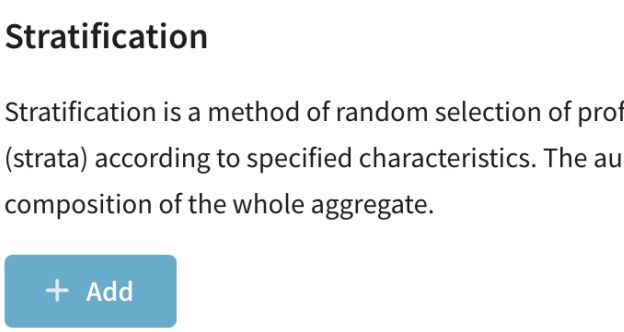
- **specific mailings and scenarios**
- **entities with certain tags**
- **entities with a specific attribute value.**

Stratification

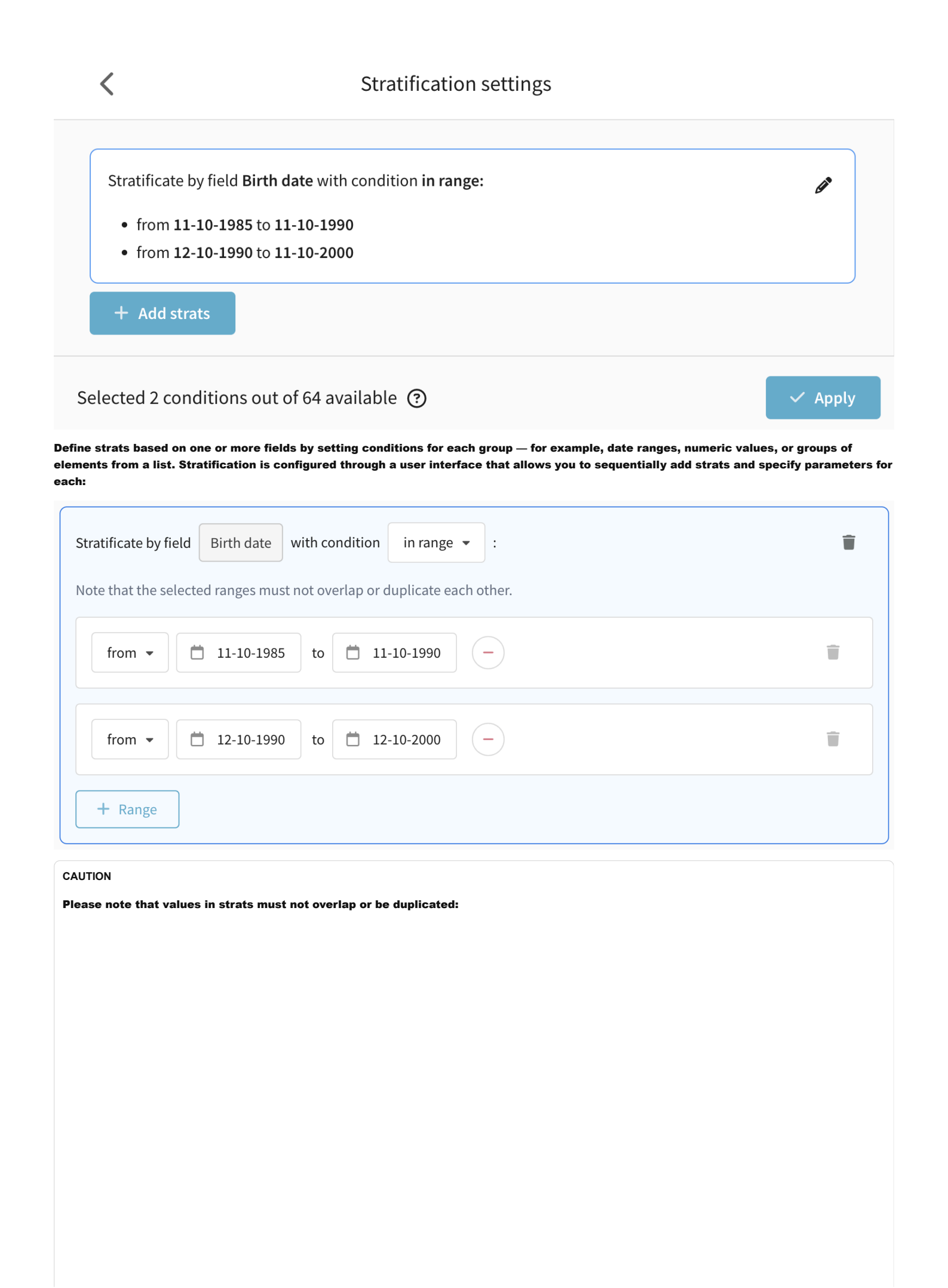
Stratification is a method of random profile selection that involves dividing the audience into subgroups (strats) based on specified characteristics. The audience is segmented in such a way that each group reflects the overall structure of the entire population.

A portion of profiles is then randomly selected from each group, proportionally to its size. This ensures that the control group has the same distribution of characteristics as the test group, making mailing results more accurate. The maximum number of strats depends on the number of selected characteristics and the number of groups defined within them. These limits can be configured in the system.

Add stratification by clicking the corresponding button:



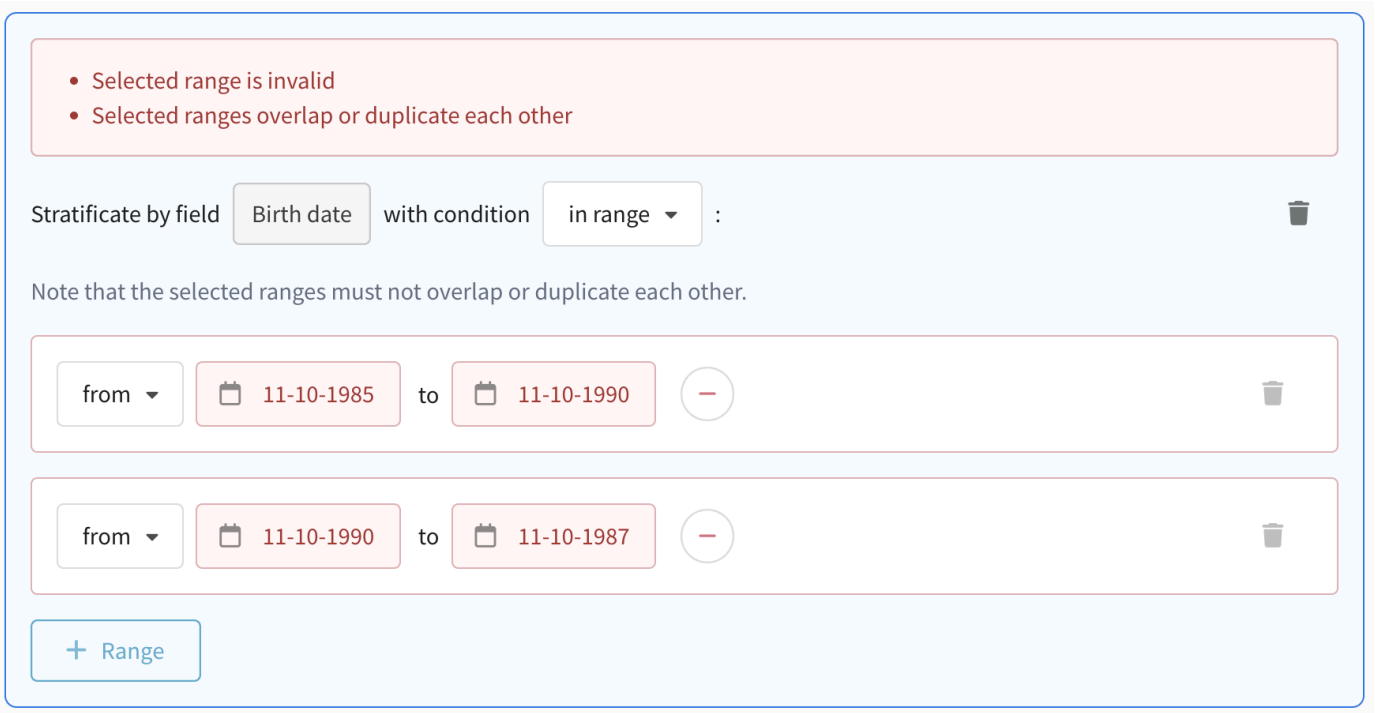
You will be taken to the stratification settings window, where you can create and edit strats:



Define strats based on one or more fields by setting conditions for each group — for example, date ranges, numeric values, or groups of elements from a list. Stratification is configured through a user interface that allows you to sequentially add strats and specify parameters for each:

CAUTION

Please note that values in strats must not overlap or be duplicated:



Supported field types for stratification:

- **boolean — logical values (e.g., yes/no);**
- **date — date values (e.g., birthdate, registration date);**
- **enum — predefined lists of values (e.g., region, category);**
- **float — decimal numbers;**
- **integer — whole numbers;**
- **unicode — string values (e.g., text labels or group names).**

Each of these field types supports appropriate condition formats: ranges, exact values, groups of values, etc.

TIP

The total number of condition combinations in stratification must not exceed 64. This value is calculated by multiplying the number of groups in each added stratum.

For example, if you select a text field and define 3 groups, then add a numeric field with 4 groups, the total number of combinations will be: 3 × 4 = 12 combinations.

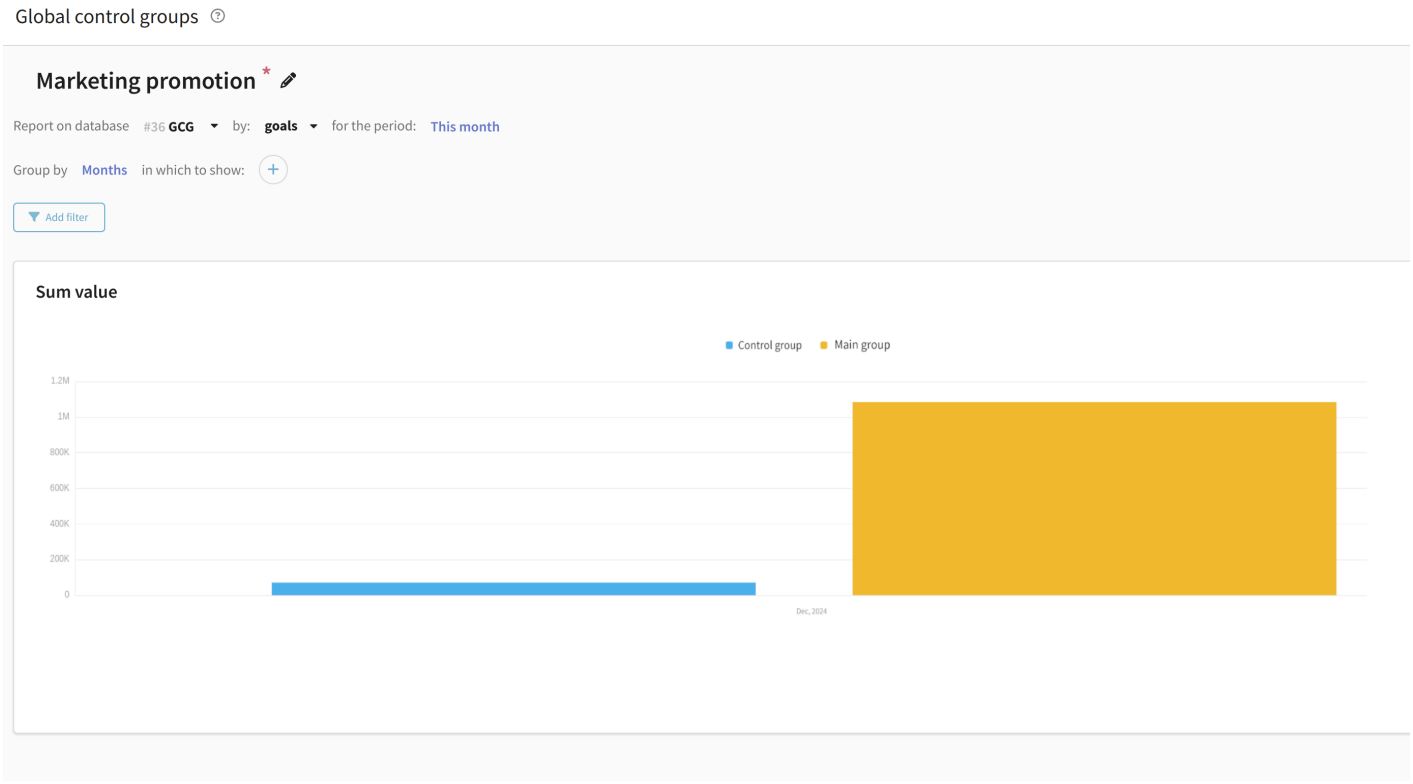
If the number of combinations exceeds 64, the system will prevent you from saving the configuration — this limitation ensures stable performance and accurate sample generation.

GCG statistics

A comparison of metrics for the audience receiving communications and the control group audience can be viewed in Analytics - Global Control Group:

![img](/img/attachments/gcg/en/8-analytics.png)	

A graph comparing the main group to the control group will appear in front of you. You can select pixel and target to compare metrics derived from tracking pixels and sales channel for Market metrics:



Read more about analytics on global control groups here.

Subscription Manager

Description

Altcraft Subscription Manager is a page that allows subscribers to manage their profile data and subscriptions: unsubscribe, resume, or suspend them for a specific period (14, 30, 60, or 90 days). If your company already has a subscription manager in use, it can be integrated into the platform.

> INFO
>
> **The built-in subscription manager only works with the subscription to which the message with the manager link was sent.**

Subscription Manager Configuration

The subscription manager is automatically activated when a profile database is created. In the required field Subscription Manager Title, specify the name that users will see when entering the interface. In the adjacent field, select the interface language for the manager.

> INFO
>
> **The language of the manager is set in the profile database settings and does not depend on the user's region or browser language. If English is selected, the interface will always display in English.**

When users follow the link, they are directed to the built-in subscription manager page. By default, the manager interface displays the main recipient details: First Name, Last Name, Email, and Phone:

Configuration of Displayed Fields

In the subscription manager, you can configure the visibility of profile fields. In addition to the required email and phone number, you can enable the display of standard or custom fields by simply ticking the corresponding options in the Subscription Manager column.

> CAUTION
>
> **Any changes made by the user to the displayed fields in the subscription manager will automatically update the corresponding data in their profile. Make sure that modifying these fields does not cause any errors.**

After saving changes, the selected fields will appear in the subscription manager:

Categories and Additional Settings

Subscription Categories

Categories help detail the type of content to which the user is subscribed. If a user unsubscribes from one category, they will continue to receive content from others.

Unlike the subscription to the resource, categories have only two statuses (Subscribed/Unsubscribed). The status can be manually changed in the profile or via API requests. To allow the subscriber to manage a category on their own, add it to the subscription manager:

After saving changes, the selected categories will appear in the Categories section of the subscription manager:

Learn more about resource categories in the article.

Custom Fields

Each resource channel can include additional fields, similar to those used in the profile database. These fields help store extra information about the subscription.

Creating Custom Fields

- **Go to resource settings**
- **Click Add Field and select its type:**
  - **Integer — positive and negative whole numbers. For example, you can specify order limits, bonus points, etc.**
  - **Float — positive and negative floating-point numbers. Examples: "Average Check", "Coefficient", etc.**
  - **String — text in any language, including numbers, special characters, and spaces. Unicode is used. Examples: "Order Comments", "Interests", etc.**

    - **Boolean — a field that allows you to choose between two options: True or False (Yes or No). In the subscription manager, it appears as a slider. Examples: “Notify about sale start”, “Receive exclusive offers”, etc.**

- **Select the fields that will be displayed in the subscription manager.**

> CAUTION
>
> **Recipients can modify the values of custom fields added to the subscription manager. Ensure that changing these fields does not lead to errors.**

In the subscription manager, the selected fields will appear in the Details section:

Learn more about custom fields in the article.

Subscription Manager Integration

Adding a Link to the Message Template

To add a link to the subscription manager:

- **Open the template editor**
- **Select Insert Link**
- **Choose the URL type Subscription Manager (preferences) link**

Using an Alternative Subscription Manager

To connect a third-party subscription manager:

- **Create a custom unsubscribe link**
- **In the Link field, enter the URL of your subscription manager**

Now, when using the unsubscribe link in the template, the recipient will be redirected to the subscription manager page.

> CAUTION
>
> **In this case, clicking the subscription manager link will automatically unsubscribe the profile from the resource. To avoid this, the chosen manager needs to be configured. Please contact your platform administrator for assistance.**

Communication channels

Client messaging

Altcraft Platform offers you 6 communication channels out of the box: Email SMS, Push (both browser and mobile app), as well as Telegram, MAX, WhatsApp*, Viber*™ and Notify. In various configuration options you will also get access to custom communication channels, like banners or social networks. This article provides you with instructions on their settings and debugging.

You import your customers' digital profiles into a Profiles database. They get subscribed to your messaging Resources — via one or several communication channels and contact data entries.

In Altcraft Platform each Mailing can be sent over one specific channel. If you need to orchestrate omnichannel messaging, you can organize cascade sending with Scenarios.

Content for all the available communication channels is created in a single Message template. Thus different mailings can use only one template that provides you with extra analytics intersections.

Email channel

![img] (/img/attachments/3114199/22708552.png)	Email allows you to send the largest amount of content, for a letter can contain different design elements and interactive sections, the ones to improve customer engagement and retention rates. To start messaging over SMTP Email, you will need a sender (From) domain to include in databases and resources settings.

INFO

This article will help you configure a sender domain for effective mailing.

INFO

And this one will provide recommendations on how to interact with email providers.

SMS channel

![img] (/img/attachments/3114199/22708553.png)	Short Message Services are just too great for transactional messaging to ignore them. All you need is your customer's phone number — and you will reach him or her even if no 3G+ network is available. No subscription confirmation required, no mass messaging restrictions. Most of the SMS messages are opened. To send SMS through Altcraft Platform, you need to integrate the platform with a telecom service. Such services often provide you with CALLER_ID — your "from name", which must be specified in the Resource when setting up an SMS channel.

INFO

Set up your first SMS mailing with the help of our Step-by-Step Guide!

This article contains information about ready-made telecom gateway integrations.

INFO

You can as well require a custom integration. Contact support@altcraft.com for details.

Push channel

![img] (/img/attachments/3114199/22708554.png)	Push notifications are one of the best ways to communicate with your subscribers as they are virtually free: no fee for sending a push compared to telecom operator fee with SMS.

INFO

For the push channel to work, first configure and activate it in Administrative panel.

INFO

Web browser push configuration is described in this article.

INFO

Mobile application push configuration is described in this article.

Telegram channel

![img] (/img/attachments/3114199/76809055.jpg)	There are 2 channels available for communication with subscribers in Telegram: **Telegram Bot** and **Telegram Group.** Telegram Bot will allow you to send direct messages to subscribers who have entered into a chat with the bot. With the help of Telegram Group, you will be able to publish posts in a Telegram group or channel through a bot.

INFO

This guide will help you organize sending direct messages to a subscriber in Telegram. Please note that when using the Telegram Bot channel, the platform only sends messages. Subscriber data is collected and transmitted to the platform on the client side.

INFO

To communicate with subscribers in a Telegram group or channel, use this guide.

WhatsApp channel

![img] (/img/attachments/3114199/76809056.png)	WhatsApp* is an effective communication channel that allows you to build a dialogue with the client in real time. Through this channel, you can inform your subscribers about upcoming events and changes in order status. Open rate in the messaging app reaches 90%.

INFO

This guide will help you start your first WhatsApp* mailing.

Viber*™ channel

![img] (/img/attachments/22708408/viber_icon.png)	Viber*™ is a fast and convenient communication channel. You can send promotional mailings, transactional and trigger messages to your subscribers with Viber*™. The cost of such sendings is cheaper than SMS.

INFO

The integration can be implemented through the providers Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto. This guide will help you create your first Viber*™ Mailing.

Notify channel

![img] (/img/attachments/22708408/notify.png)	Notify — an effective channel for business communication with the client. Send transactional and service messages via Vkontakte and Odnoklassniki.

INFO

For how to create a connection and how to set up integration, see the corresponding documentation articles.

Set up your first Notify mailing with the help of our Step-by-Step Guide!

MAX

CAUTION

When setting up integration with this communication channel, consider the internal rules of the MAX messenger.

![img] (/img/attachments/22708408/max.png)	Two channels are available in MAX for communication with subscribers: MAX Bot and MAX Group. MAX Bot is used to send personalized messages to subscribers who have started a chat with the bot. With MAX Group, you can publish posts in a MAX group or channel via a bot.

INFO

Step-by-step guide explains how to organize sending personalized messages to a subscriber in MAX.

Note that when using the MAX Bot channel, the platform only sends messages. Collecting and transferring data about users who started a chat with the MAX bot to the platform must be implemented on the client side.

INFO

To communicate with subscribers in a MAX group or channel, use this guide.

Communication channel limitations

Feature	Email	SMS	Push	WhatsApp™	Viber™	Notify™	Telegram	Max	Custom Channels
MVT testing	✅	✅	✅	❌	❌	❌	❌	❌	❌

Feature	Email	SMS	Push	WhatsApp™	Viber™	Notify™	Telegram	Max	Custom Channels
Web version saving	✅	✅	✅	❌	❌	❌	❌	❌	❌
Subscription from scenario	✅	✅	❌	❌	❌	❌	❌	❌	❌
Suppression lists	✅	✅	❌	❌	❌	❌	❌	❌	❌
Click map	✅	❌	❌	❌	❌	❌	❌	❌	❌

For more information on custom channel restrictions, check the administrator documentation.

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

***** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

First WhatsApp Mailing

![img] (/img/attachments/76809005/76809007.png)	**WhatsApp* is an effective communication channel that allows you to build a dialogue with the client in real time. Through this channel, you can inform your subscribers about upcoming events and changes in order status. Open rate in the messaging app reaches 90%.** This guide will help you start your first WhatsApp* mailing. </td> </tr>

Step 1. Connect WhatsApp Business API through a provider

To send bulk messages in WhatsApp*, you need to connect the WhatsApp* Business API. In Altcraft Platform, such a connection is possible through such official providers as Edna or 360dialog.

Register an account with one of the providers.To do this, you need a phone number and a verified Facebook* Business Manager account.

TIP

If you want to register a WABA account through Edna, leave a request at this link.

360dialog collects applications through this form.

TIP

A phone number must be clean. This is a number that has not been registered for WhatsApp* Business API in the past, and is currently not registered on the WhatsApp* mobile app.

CAUTION

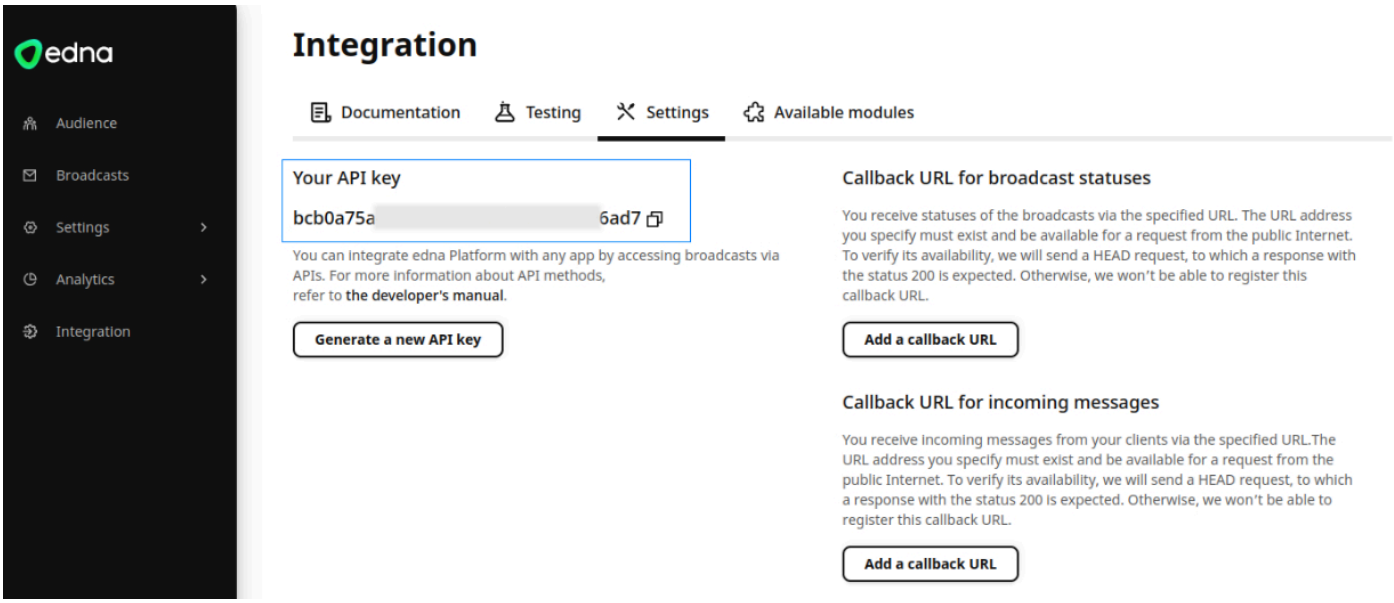
Integration with WhatsApp* Business API, as well as sending messages, are not charged additionally on the Altcraft side. However, you will need to pay for the services of your provider (360dialog or Edna). In addition, the cost of sending messages on WhatsApp* is also charged separately. You can find out more about the cost of WhatsApp* messages here.

Step 2. Obtain an API Key

The API key can be obtained in the provider's personal account:

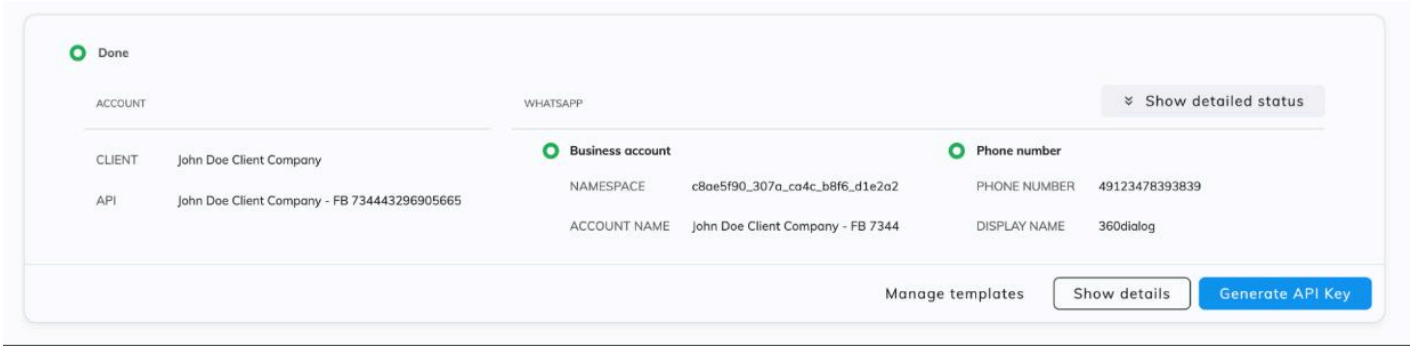
Edna

In your Edna account, go to Integration —> Settings:



360dialog

In 360 Client Hub go to *WhatsApp Accounts and click Generate API key:**

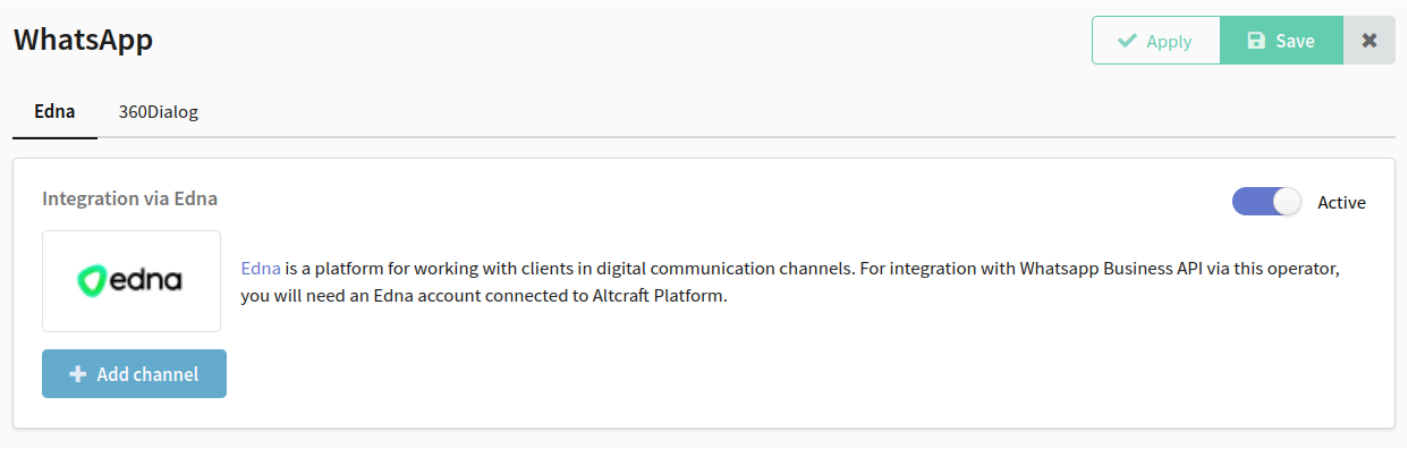


Step 3. Set up integration

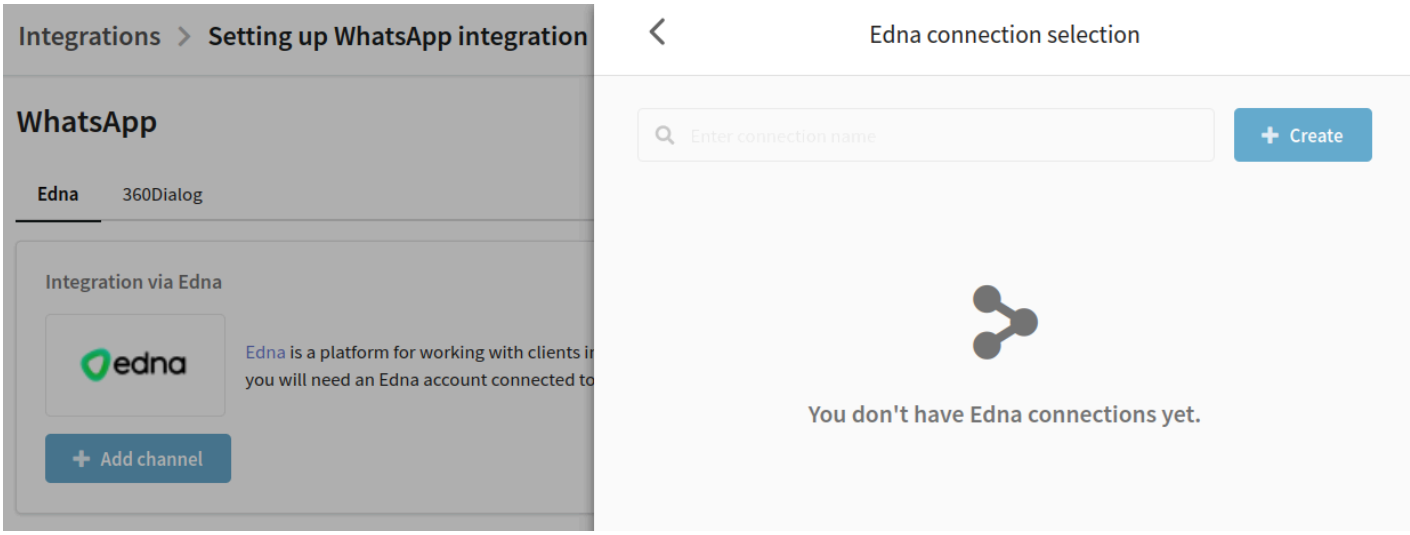
Go back to the platform and open Integrations. Select the type of integration — WhatsApp*. Switch to the tab of your provider.

Edna

Activate the integration using the toggle in the top right corner. Click Add channel:

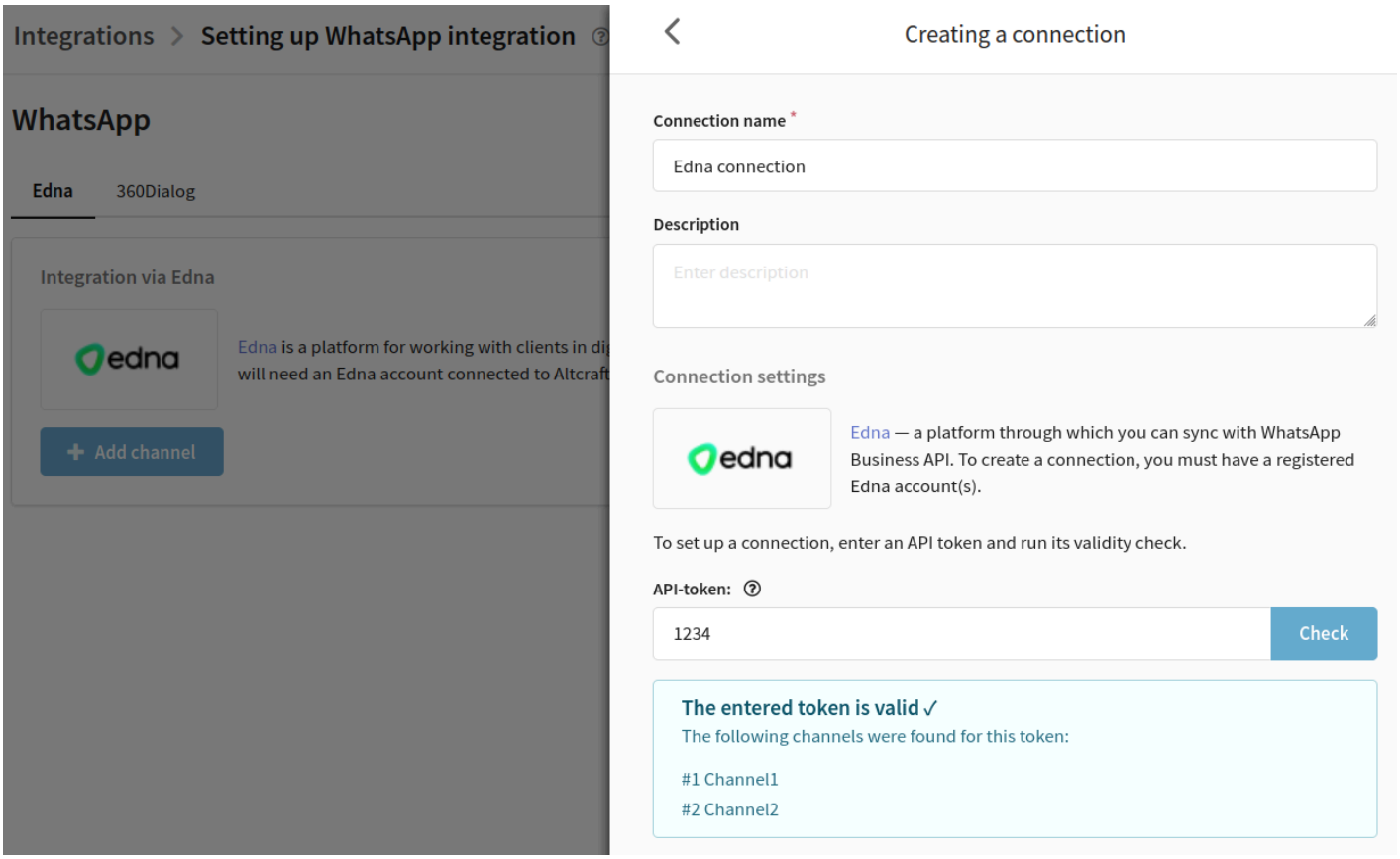


For integration, you need to link your WhatsApp* Business API account to Altcraft Platform. Linking accounts is done through connections. Click Create to set up a new connection:

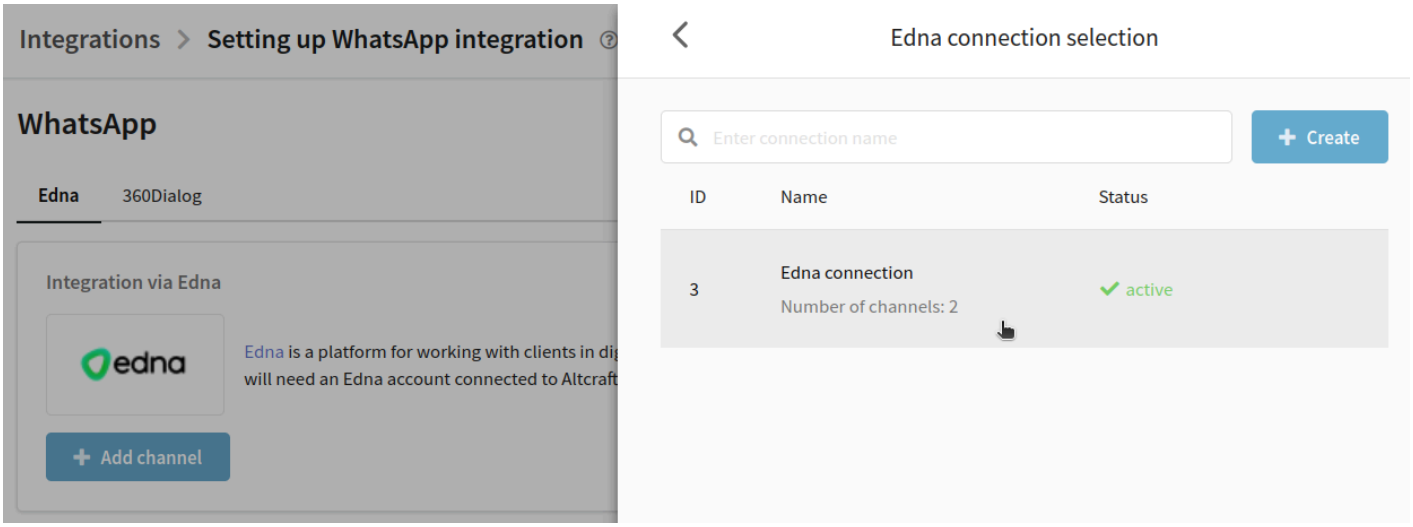


In the new window, enter a name and description for the connection.Enter the API token generated in the provider's personal account and run a check to confirm its validity.

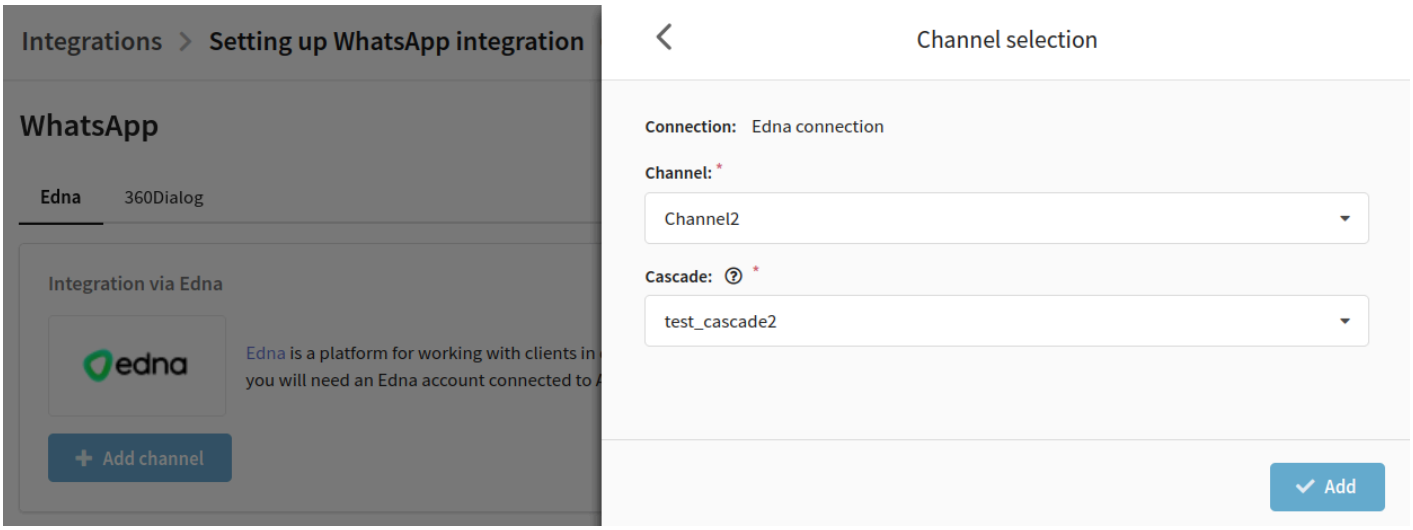
If the token is valid,the window will display data on the available channels in the account:



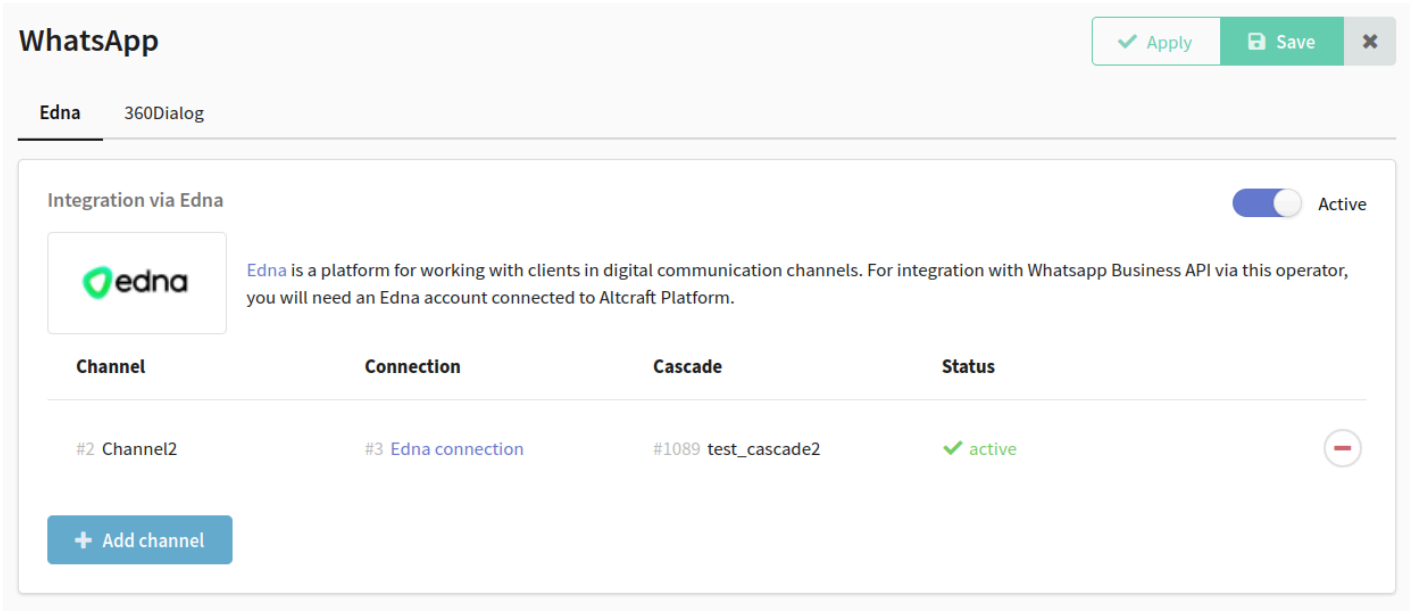
Next, click ✓Create in the lower right corner to save the connection. You will be taken back to the list of connections. Select the one you've just created:



Next step is to select a channel and cascade. The channel must be in the "Correct" status, and the cascade must use only one channel. Then click Add in the bottom right corner.



Data on the added Edna channel will be displayed in the integration:



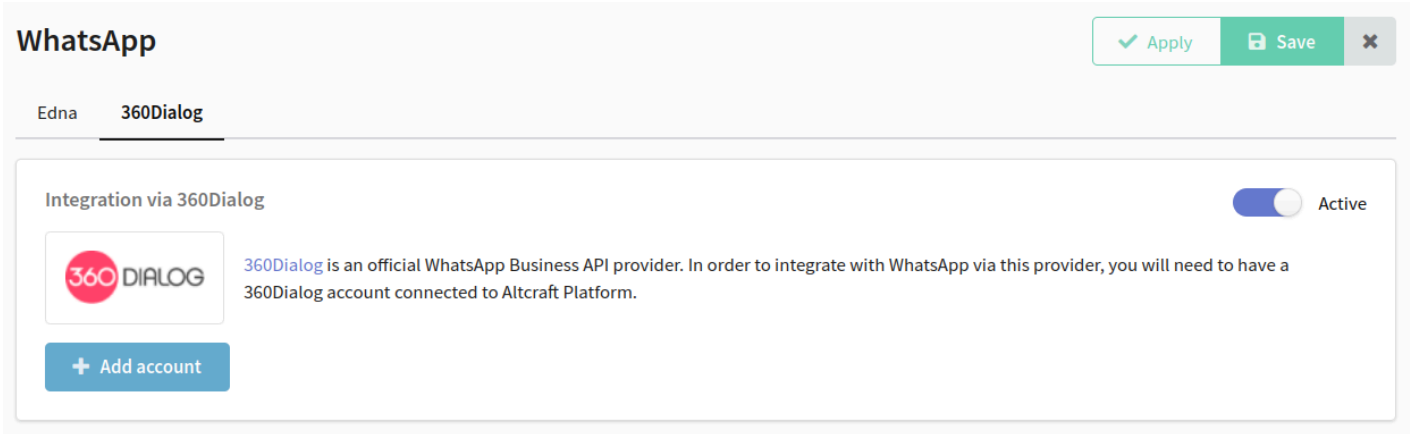
Save the integration using the buttons in the top right corner.

TIP

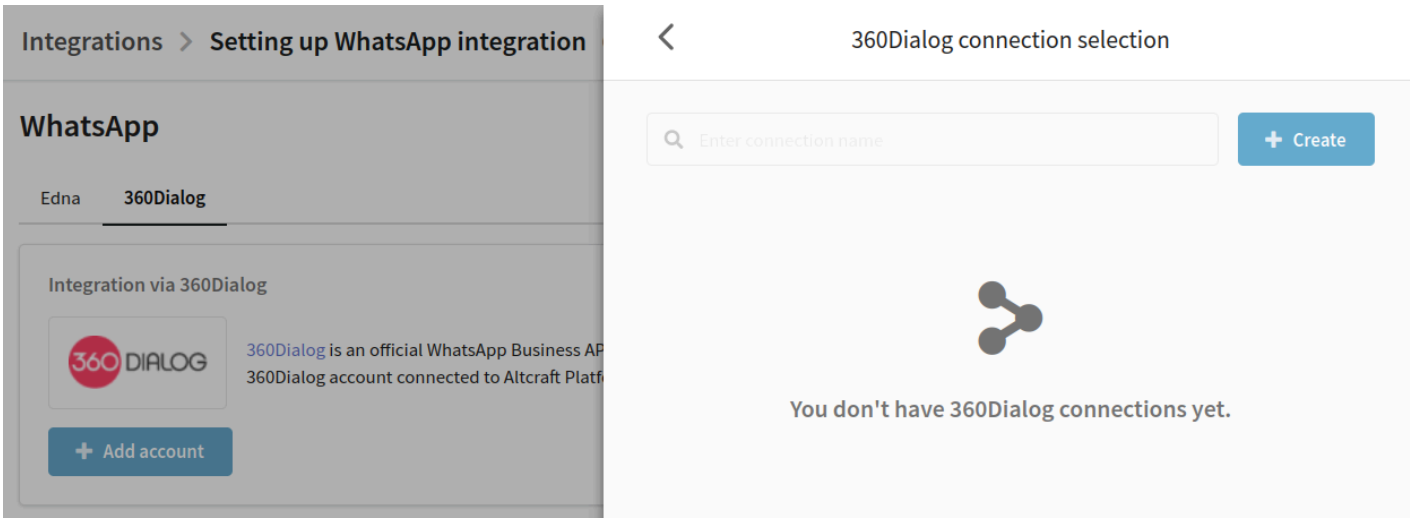
If necessary, you can add multiple channels for Edna integration.

360dialog

Activate the integration using the toggle in the top right corner. Click Add account:

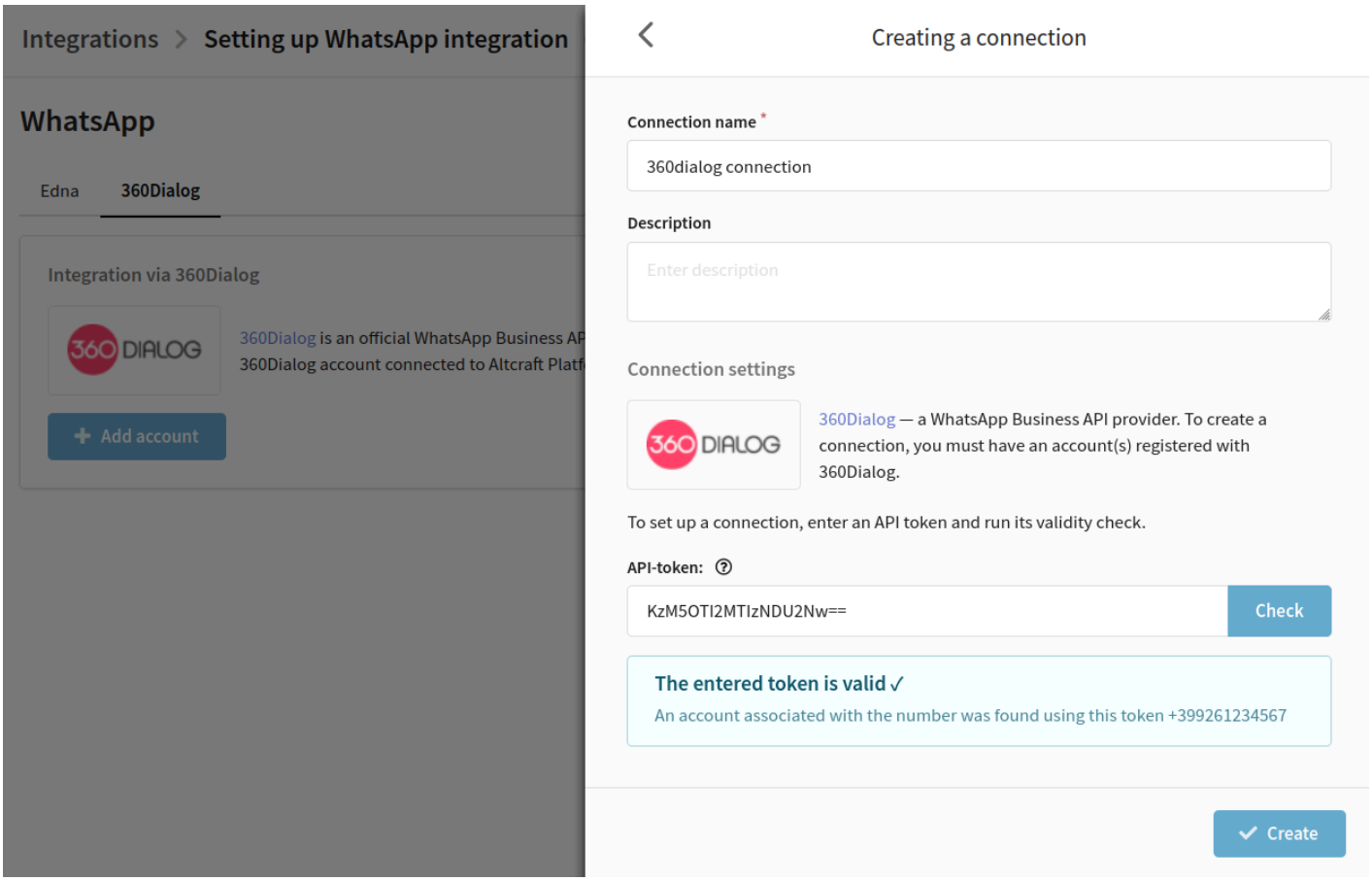


To set up the integration, you first need to link your WhatsApp* Business API account to Altcraft Platform. Linking accounts is done through connections. Click Create to set up a new connection:

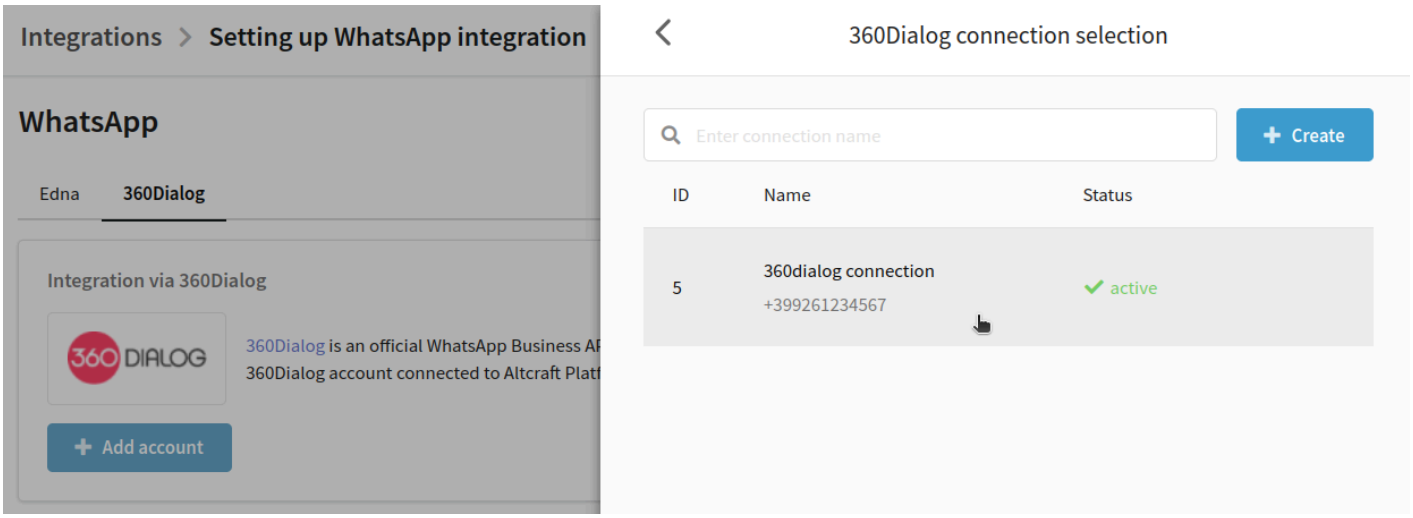


In the new window, enter a name and description for the connection.Enter the API token generated in the provider's personal account and run a check to confirm its validity.

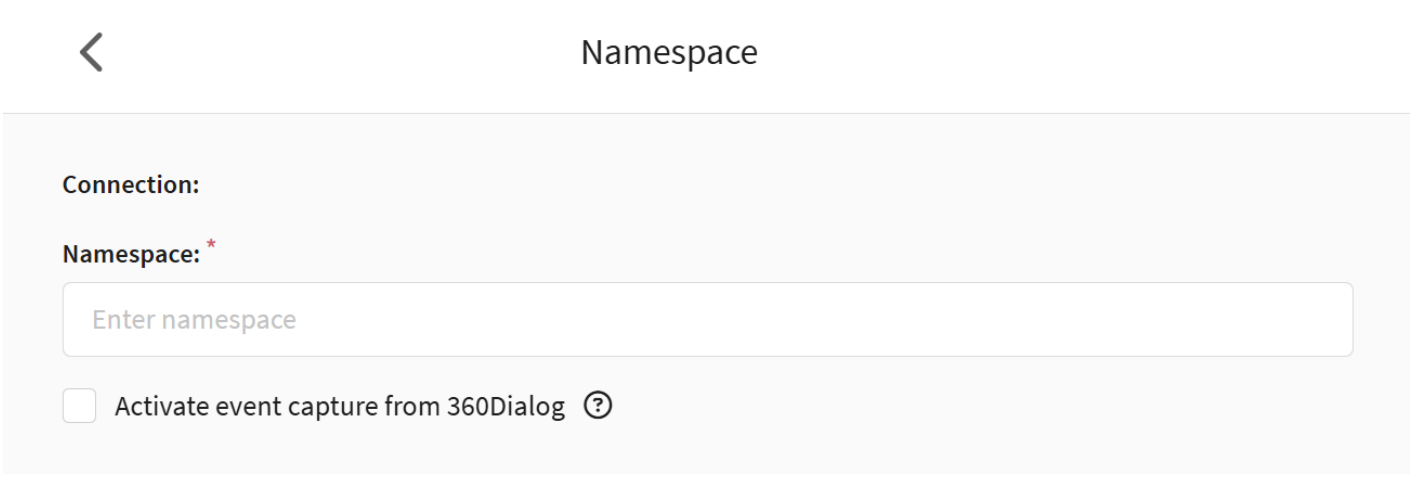
If the token is valid, you will see the phone number attached to the account you are connecting to:



Next, click ✓Create in the lower right corner to save the connection. You will be taken back to the list of connections. Select the one you've just created:

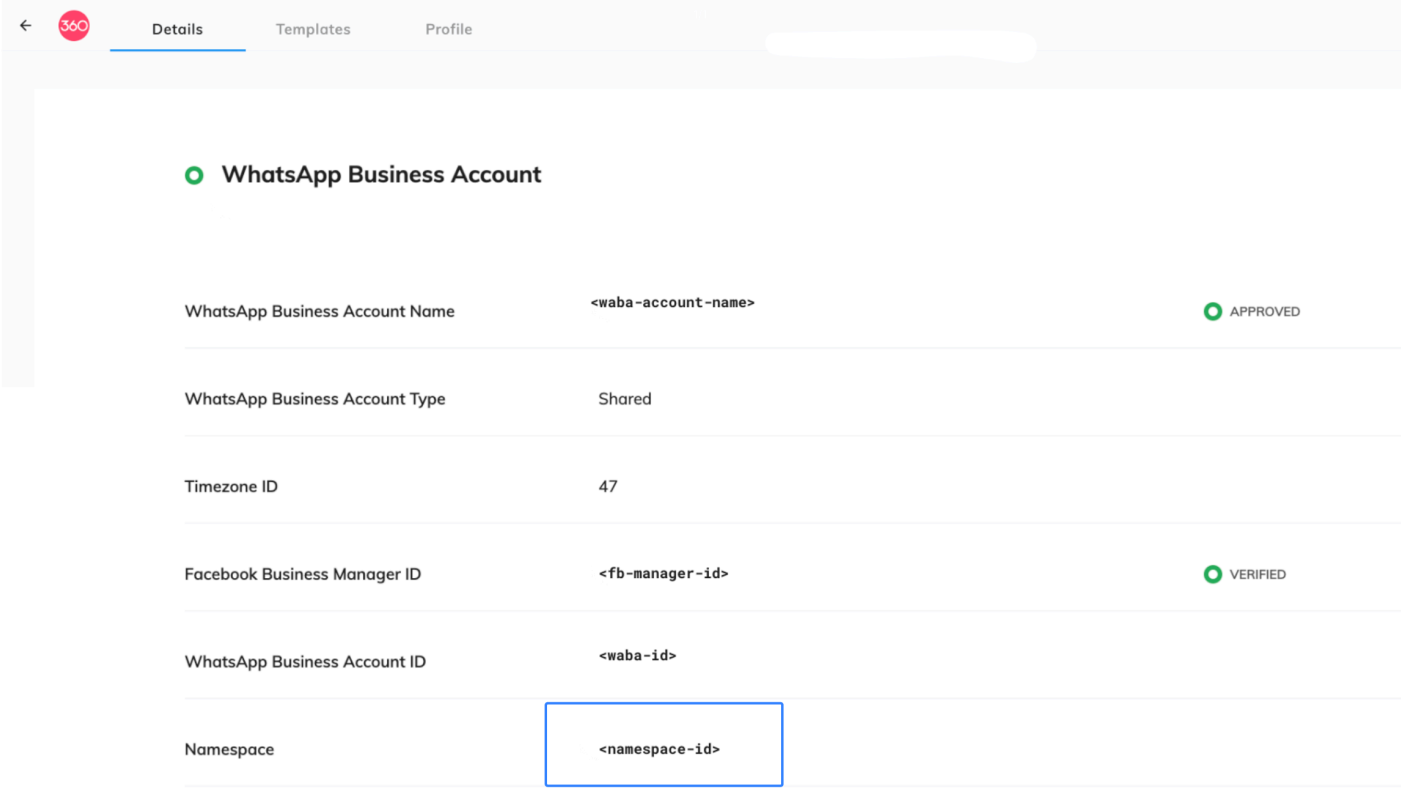


Fill in the Namespace field:

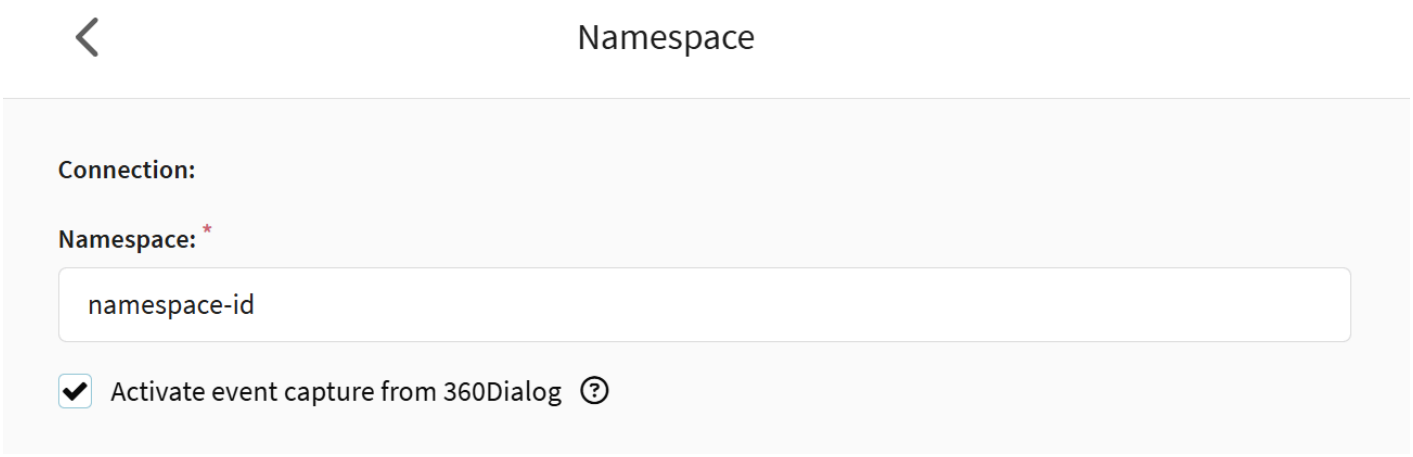


Namespace is a free string. Each 360dialog provider account stores the templates created for it and bound through this operator. Namespace can be viewed in the provider's personal cabinet in the account settings.

Go to the settings of your personal cabinet 360Dialog:



Copy the Namespace value from there, go back to the platform and paste it into the required field:



If you want, you can activate the Capture events from 360Dialog option.

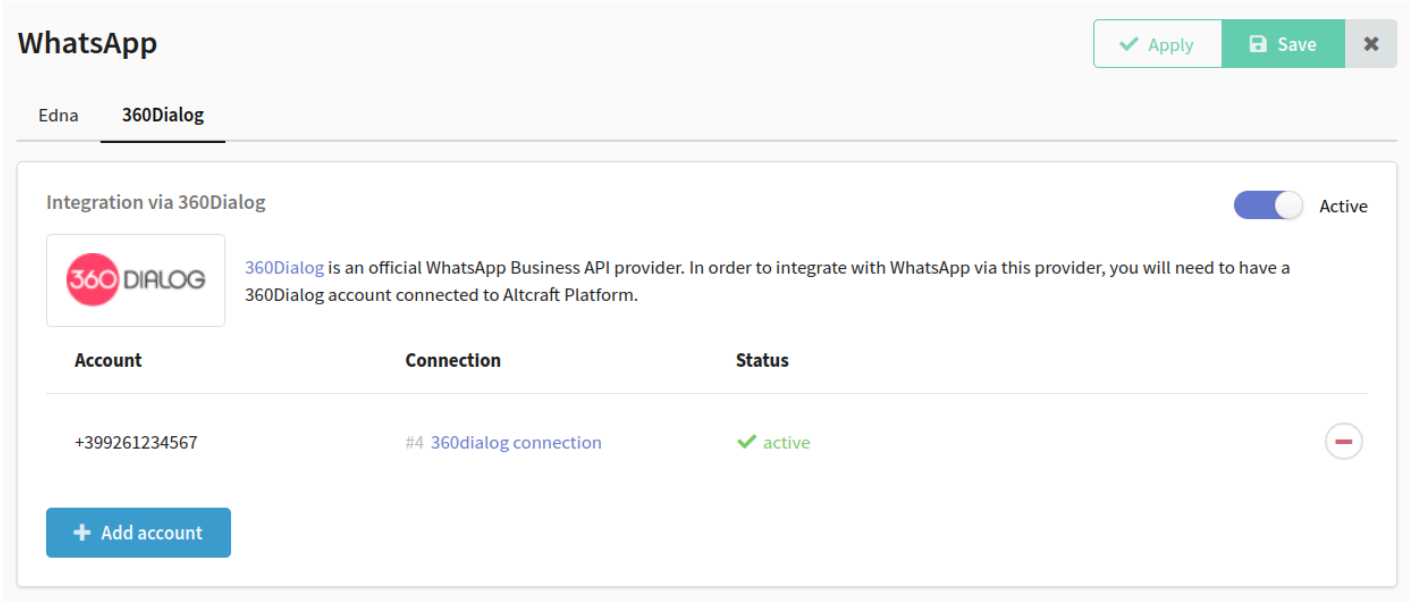
The Activate event capture from 360Dialog option enables sending WhatsApp event data to the Altcraft platform from 360Dialog.

If it is enabled, after saving the integration, a special link (Callback URL) will be automatically added to 360Dialog, through which Altcraft will receive information about message statuses, user actions and other events.

If the transmission of this data is not required, disable the setting.

Click Add in the lower right corner.

The integration is set up. You will see the data for the connected account:



Save the integration using the buttons in the top right corner.

TIP

If necessary, you can add multiple channels for Edna integration.

Step 4. Subscribe customers to WhatsApp channel

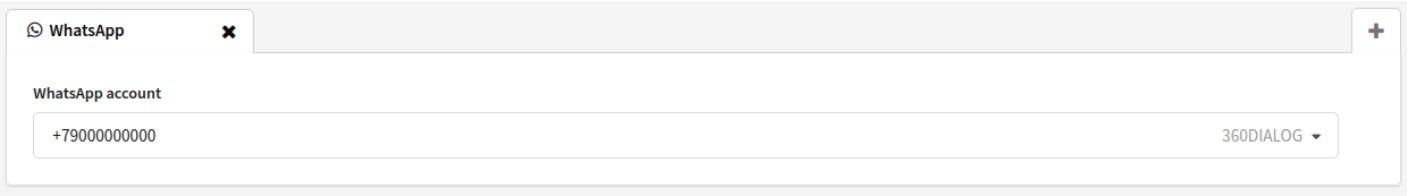
Sending a message to a user in WhatsApp* requires:

- **the user's phone number to be in the database;**
- **the user's profile to subscribe to the resource's WhatsApp* channel;**
- **the user's consent to receive messages from your company.**

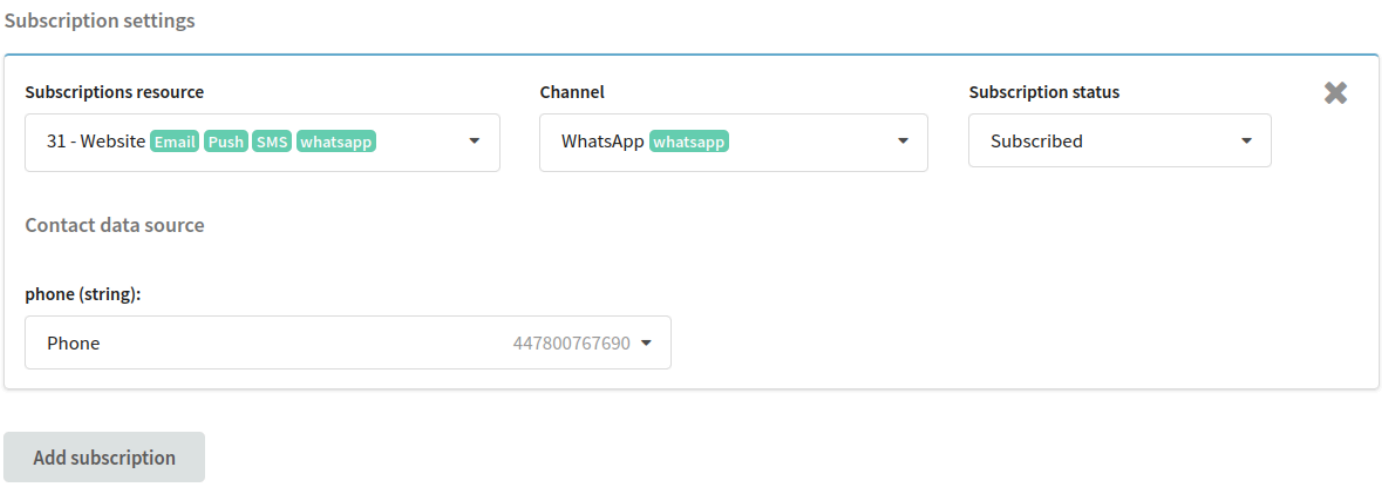
CAUTION

The profile phone number to which you send messages must be stored in international format. The following format must be used: +79001234567.

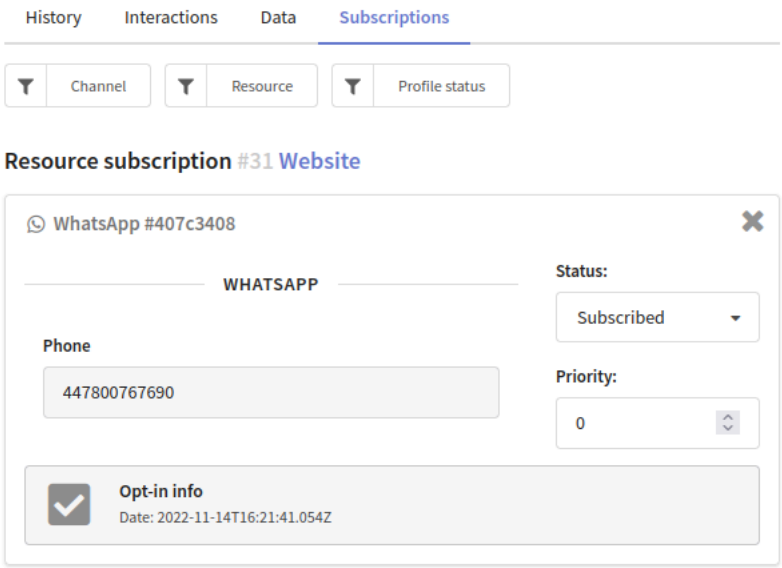
First, select or create a profile database that will store subscriber data, including phone numbers. Then create a new resource or edit an existing one by adding a WhatsApp* channel. To set up a channel, select the previously connected WhatsApp* account:



Next, [import subscriber data] into the platform, for example, via a file. When importing, add them a subscription to a resource with a WhatsApp* channel. Select the column in your file that contains phone numbers as the subscription source.



A [digital profile] will be created for each client. It combines personal data, activity history and subscriptions to resources. If a subscription to the WhatsApp* channel has been successfully added to the client during import, this data will be displayed in the client's history:



Step 5. Create a template in 360 Client Hub

There are two types of conversations in WhatsApp* Business:

- **User-initiated conversation. It is aconversation that starts in response to a user message. Whenever you reply to a user within the 24-hour service window, that message is associated with a user-initiated conversation. You can send free-form messages within this 24-hour customer service window.**
- **Business-initiated conversation. It is a conversation that starts from a business sending a user a message outside the 24-hour service window. Messages that initiate a business-initiated conversation require an [approved message template].**

Once the customer replies to the business's template message, the business can begin sending any type of message to the customer.

CAUTION

You can only use Altcraft Platform for business-initiated conversations. This means that you can only send approved templates messages (HSM).

A template message consists of fixed text and variables. Variables are text that can be changed before each mailing without the consent of Facebook*.

You need to create a template in your provider's account; in Altcraft Platform you can only add text into variables, set up buttons and add media files.

TIP

You can find more information about creating templates in your provider's documentation:

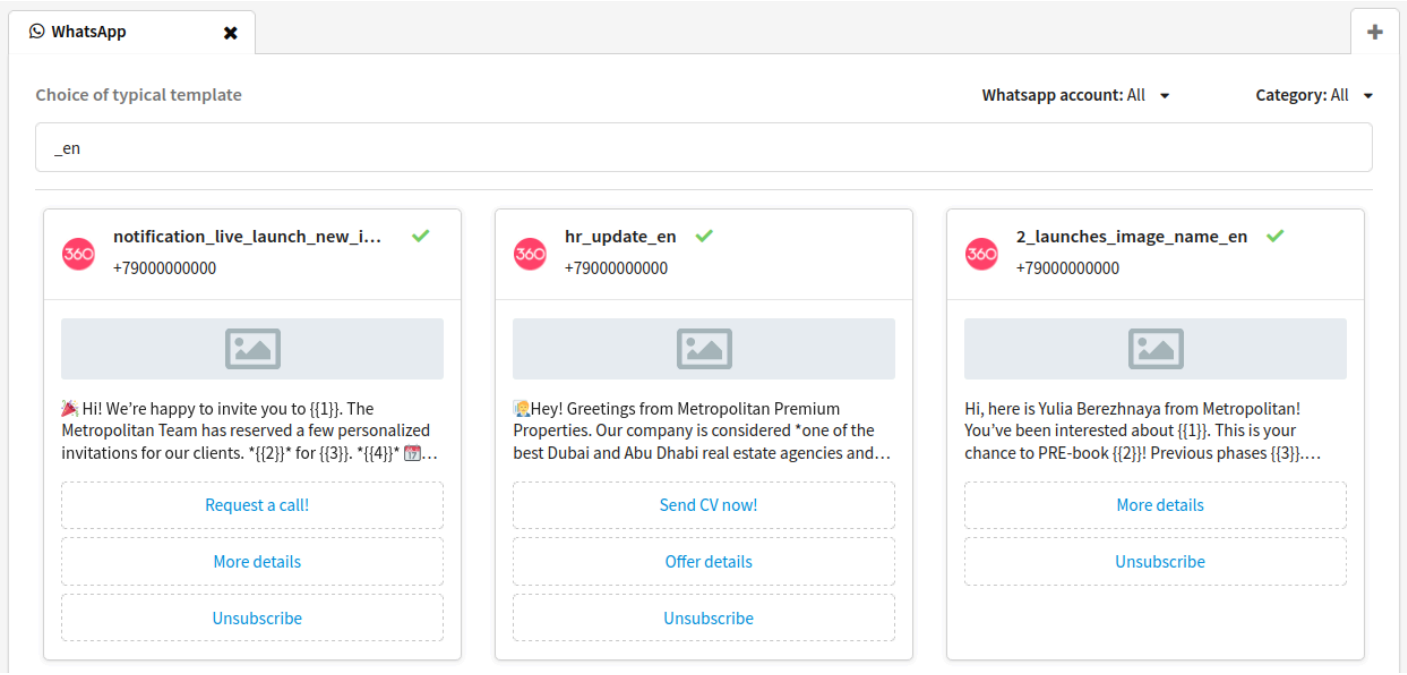
- **Edna: (https://docs-pulse.edna.ru/docs/template/whatsapp/operator-template)**
- **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

CAUTION

A template created in the provider's personal account must be verified by Facebook*. Mailings launched through Altcraft can only use templates approved by Facebook*.

Step 6. Create a message based on the template

Go back to Altcraft Platform, open the Templates section and create a new template for the WhatsApp* channel. All templates available in connected WhatsApp* accounts through edna or 360dialog are automatically available in the platform and displayed in the editor:



Choose a template and create a message based on it. To do this, fill in the variables, configure the buttons and add media files.

TIP

Here is a guide where you will find more details on how to create a template for the WhatsApp* channel.

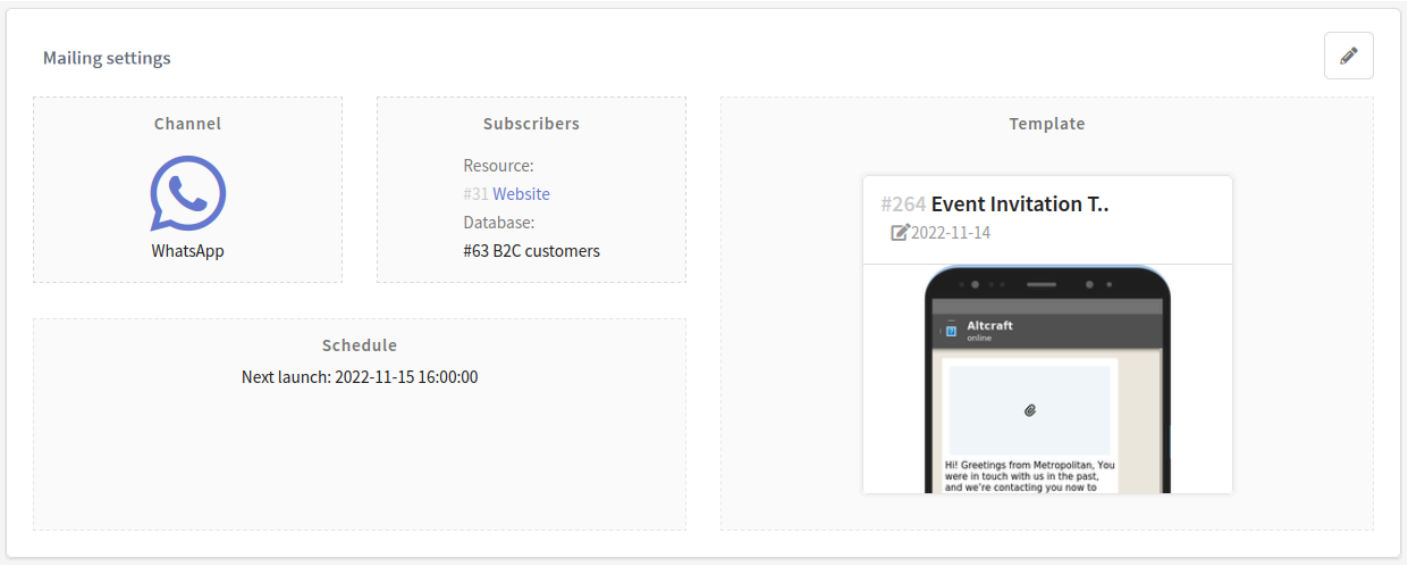
Step 7. Launch the mailing and evaluate its performance

Go to Mailings and select the mailing type:

- **Broadcast is a one-time ending of messages.**
- **Regular mailing isa periodic sending of messages according to a schedule.**

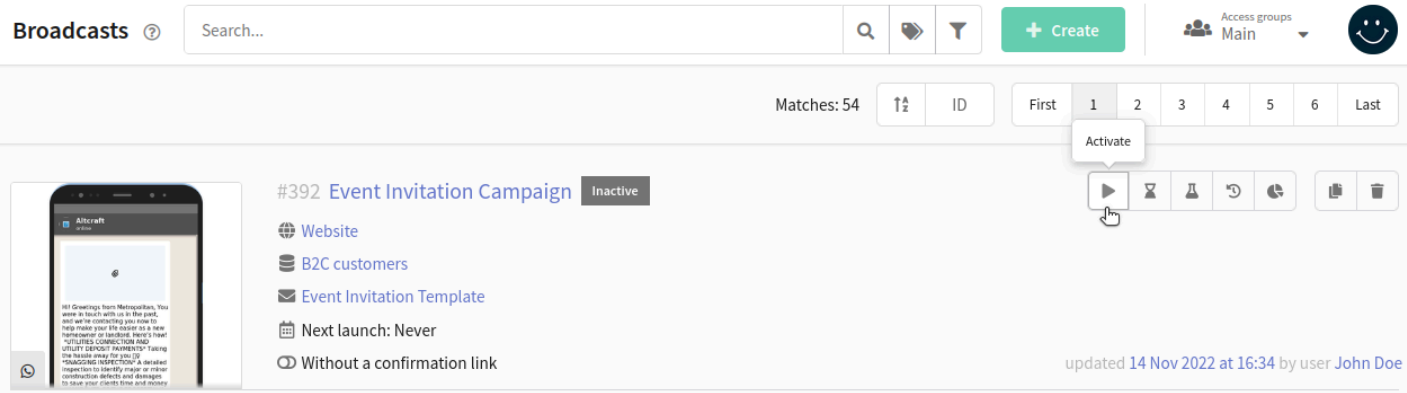
Set up a mailing:

- **select WhatsApp* as a communication channel**
- **select a previously configured resource and a recipient audience (database or segment);**
- **add a message template;**
- **set the schedule:**



Save the mailing. Find it in the list of mailings of the selected type and activate it by clicking

in the quick action bar.The active mailing will start sending messages when the scheduled launch time comes.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your mailings go to Analytics. The channel report will provide data for each mailing. You will receive information about the number of sent, delivered and undelivered messages, as well as the number of new subscriptions to the WhatsApp* channel. For the convenience of grouping, you can choose a preset of indicators for WhatsApp*.

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.*

First Viber*™ Mailing

![img](/img/attachments/76808932/76808934.png)	**This guide will help you launch your first Viber*™ mailing.** Integrating Altcraft Platform with Viber*™ is done through the **Devino Telecom** through the [**SMS Traffic**](https://www.smstraffic.ru/), [**MTS OmniChannel**](https://omnichannel.mts.ru/) or through the [**Rapporto**](https://rapporto.ru/). If your company doesn't have an account yet, please register one. </td> </tr>

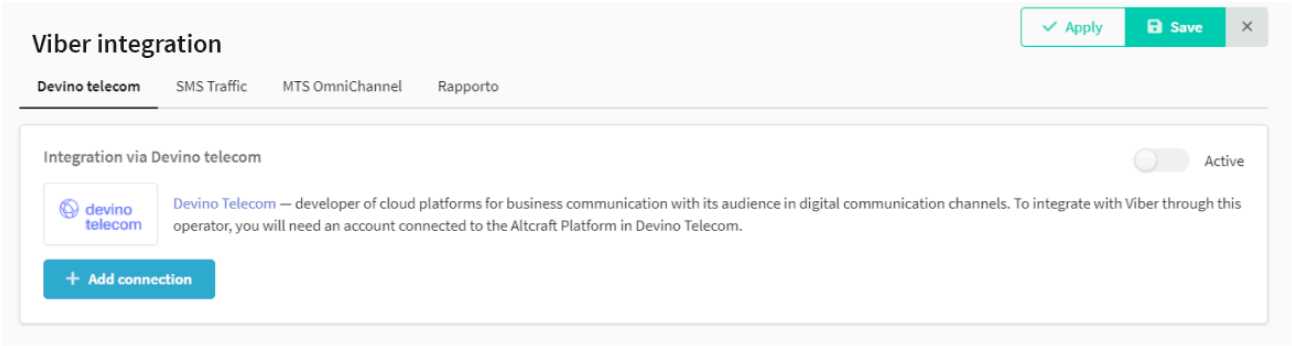
INFO

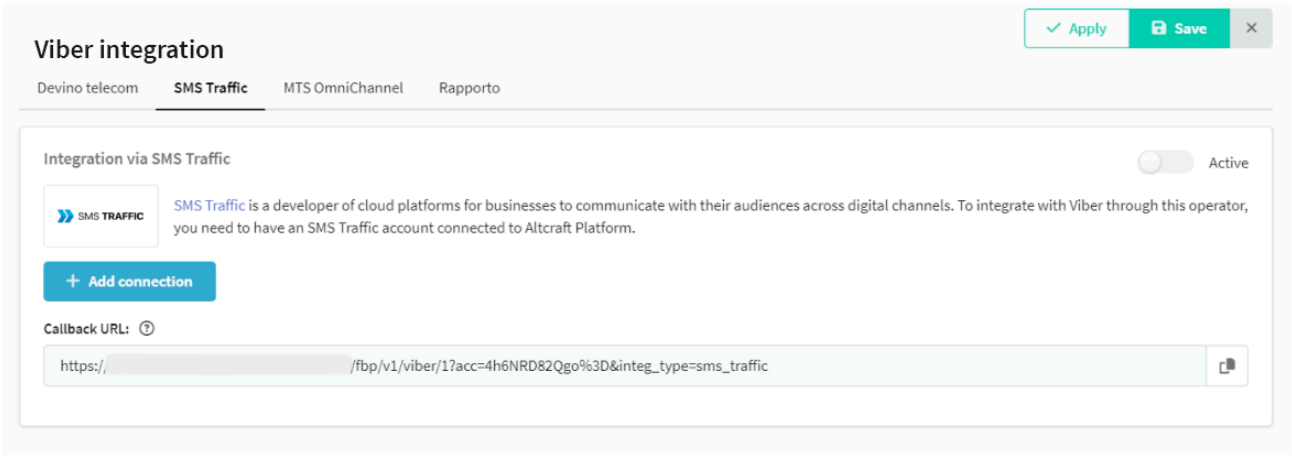
The cost of messages is determined by providers. Altcraft does not charge additional fees for sending messages to Viber*™.

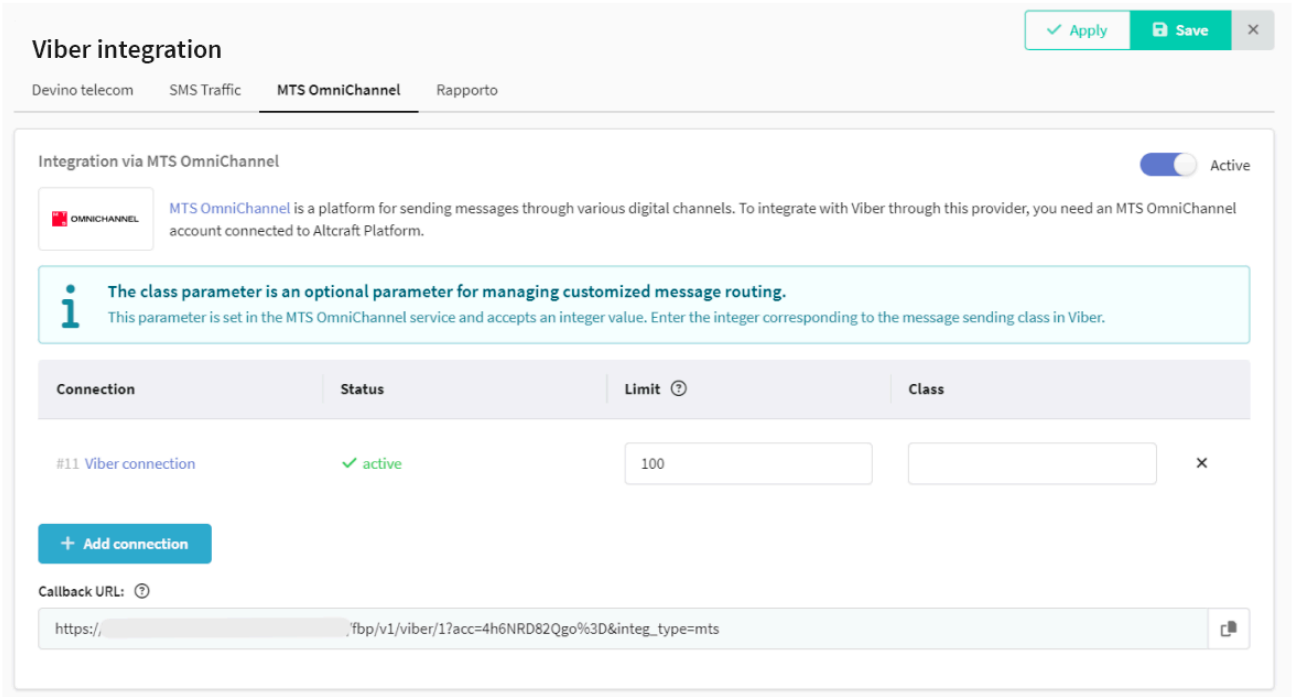
Step 1. Set up integration

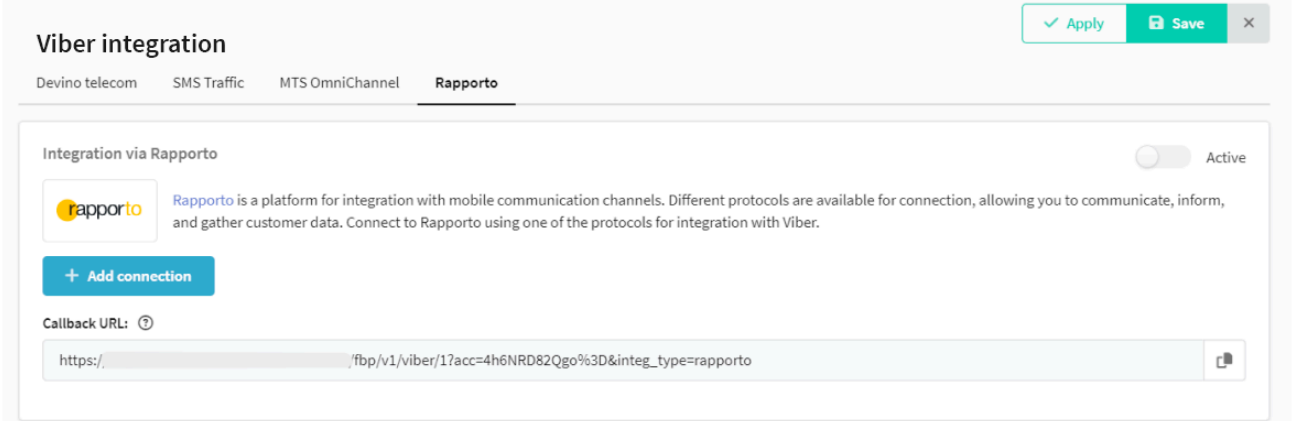
1. In the main menu, go to Integrations. Choose *Viber™ as the integration type.**

2. Choose the provider — Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto.


Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via Devino telecom
Active
Devino Telecom — developer of cloud platforms for business communication with its audience in digital communication channels. To integrate with Viber through this operator, you will need an account connected to the Altcraft Platform in Devino Telecom.
Add connection

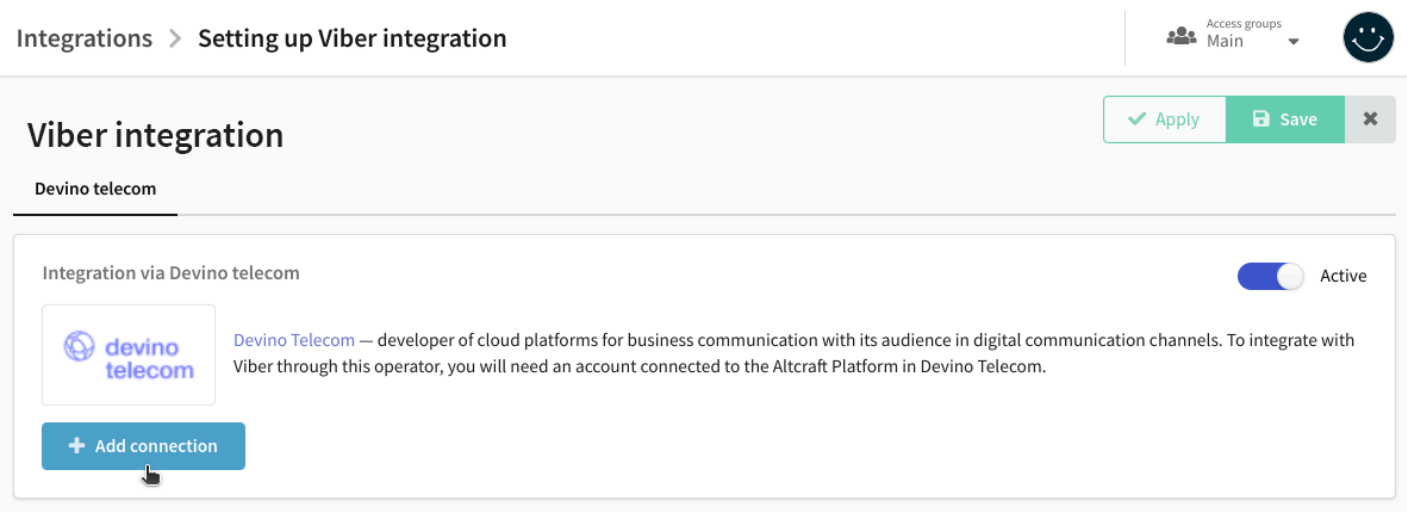

Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via SMS Traffic
Active
SMS Traffic is a developer of cloud platforms for businesses to communicate with their audiences across digital channels. To integrate with Viber through this operator, you need to have an SMS Traffic account connected to Altcraft Platform.
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=sms_traffic


Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via MTS OmniChannel
Active
MTS OmniChannel is a platform for sending messages through various digital channels. To integrate with Viber through this provider, you need an MTS OmniChannel account connected to Altcraft Platform.
The class parameter is an optional parameter for managing customized message routing.
This parameter is set in the MTS OmniChannel service and accepts an integer value. Enter the integer corresponding to the message sending class in Viber.
Connection
Status
Limit
Class
#11 Viber connection
active
100
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=mts

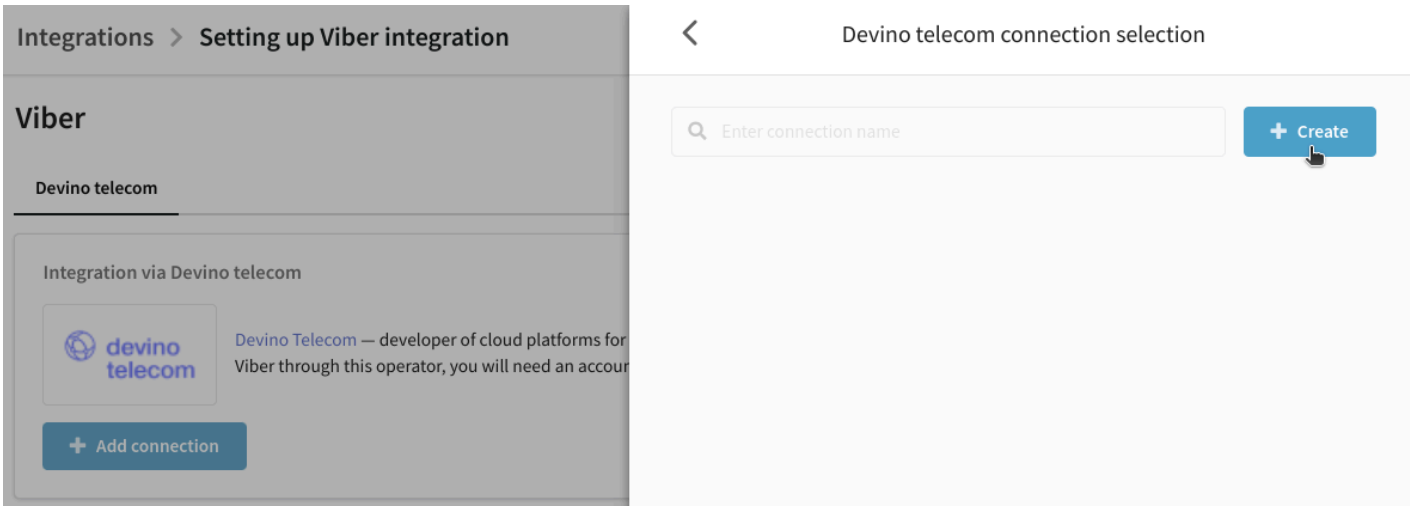

Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via Rapporto
Active
Rapporto is a platform for integration with mobile communication channels. Different protocols are available for connection, allowing you to communicate, inform, and gather customer data. Connect to Rapporto using one of the protocols for integration with Viber.
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=rapporto

3. Activate the integration using the toggle switch in the top right corner.

4. Click Add connection:



5. A new window will appear on the right, showing available connections. A connection links your provider account with Altcraft Platform. Click “Create” to set up a new connection:

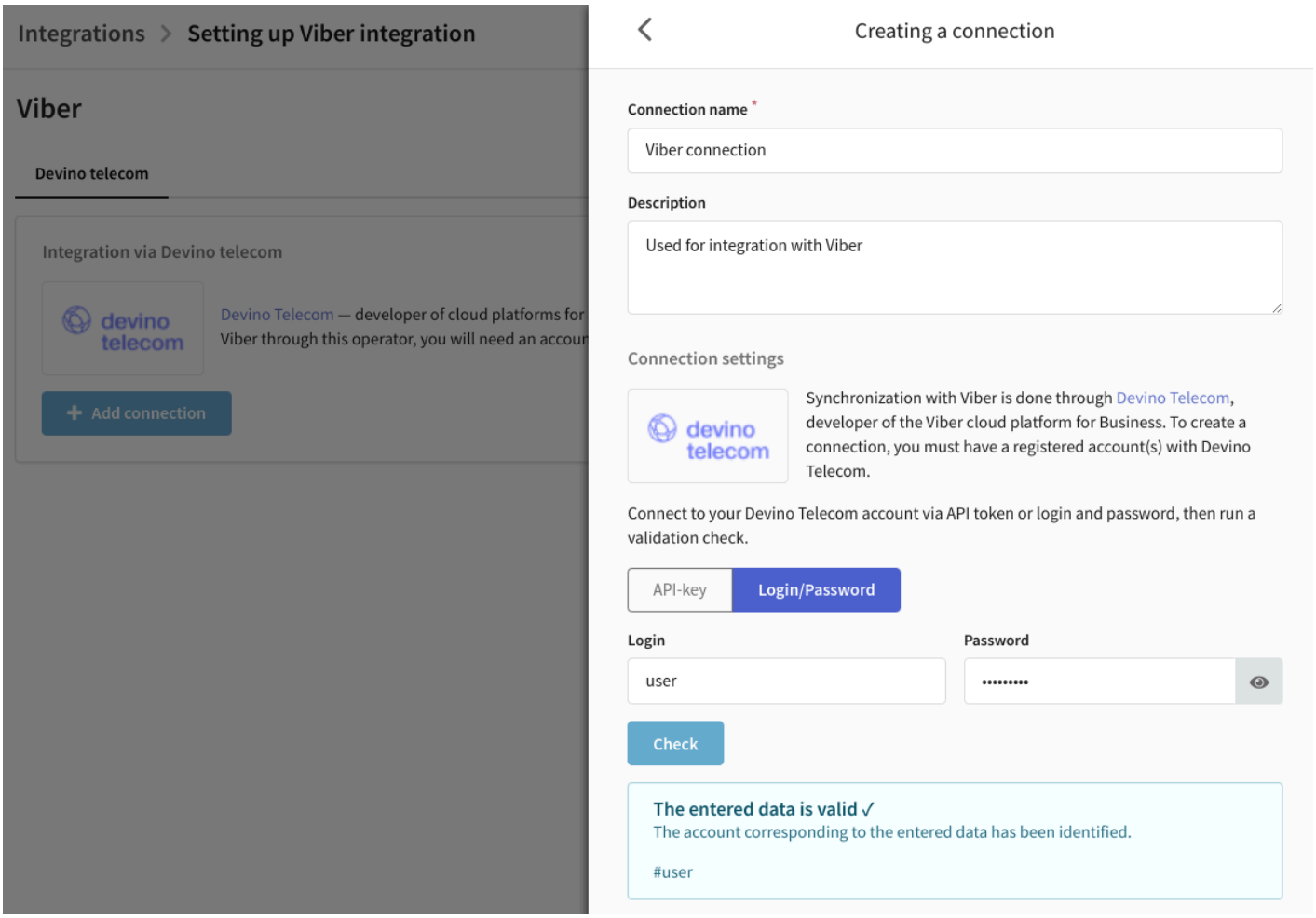


6. In the new window, enter the name and description of the connection.

7. Authorize in your provider account.

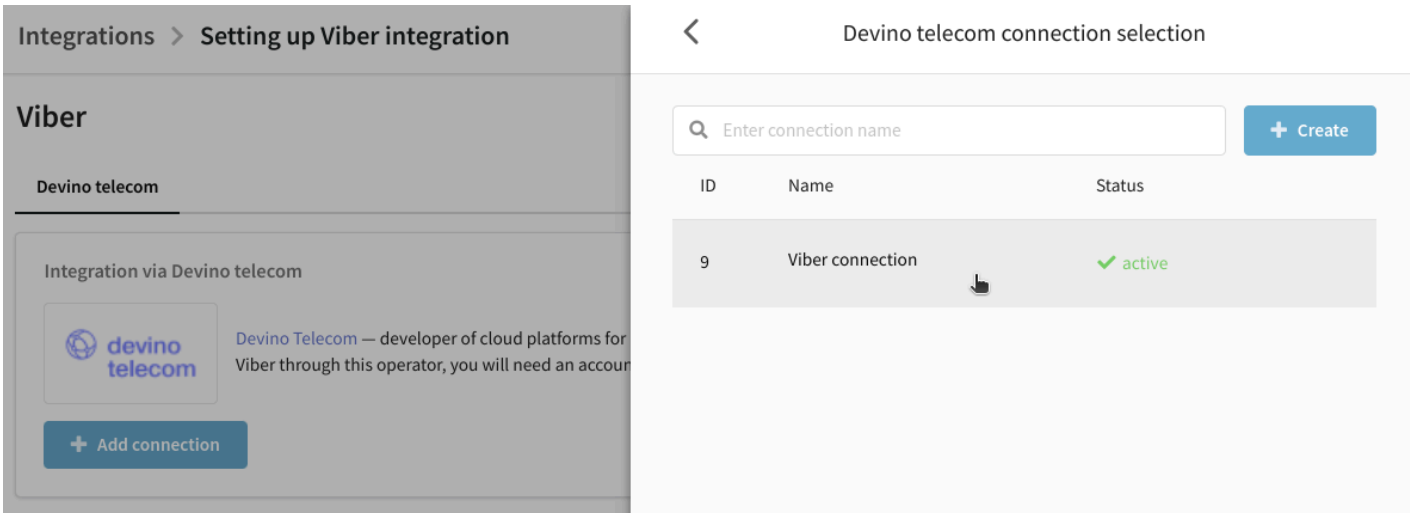
> TIP
>
> **For Devino Telecom there are two ways of authorization. You can learn more about creating of connection in this article.**

8. Enter the authentication details and click “Check” to verify their validity. If the data is correct, information about the successfully identified account will appear below.

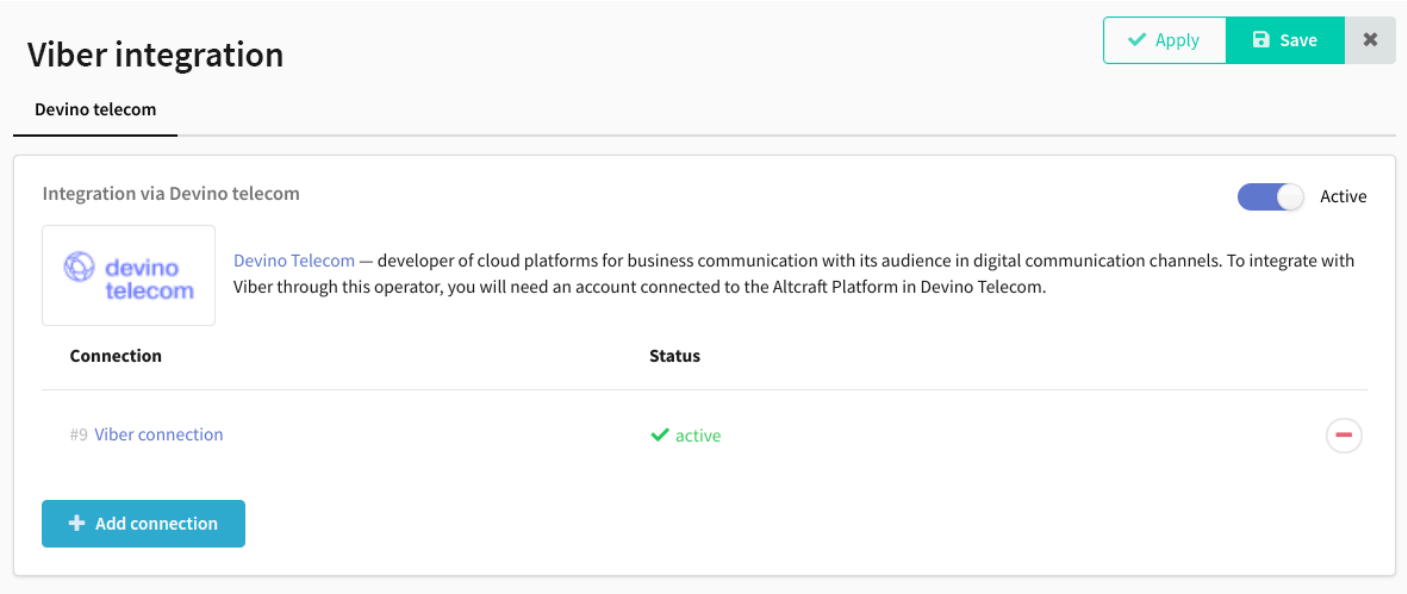


9. Click “Create” in the bottom right corner to save the connection.

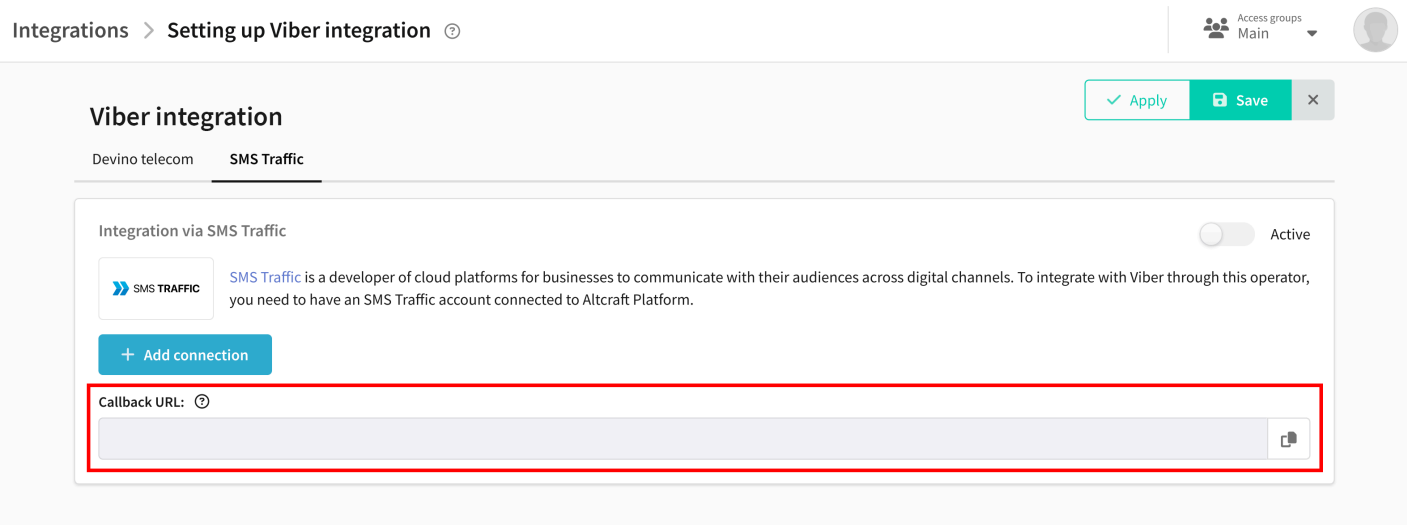
10. Go back to the list of connections, and select the one you just created.



11. Information about the added connection will be displayed in the integration section. Ensure that the connection status is “active.” If not, check your account status in the provider dashboard. Note that your account might be blocked if the subscription fee is not paid.



12. If you use SMS Traffic as a provider, you have to send your Callback URL from the integration to technical support



13. Save the integration settings using the buttons in the top right corner.

Step 2. Subscribe customers to the Viber*™ channel

To send a message to a user on Viber*™, you need:

- **The user's phone number in the database.**
- **The profile must have a subscription to the Viber*™ channel of the resource.**

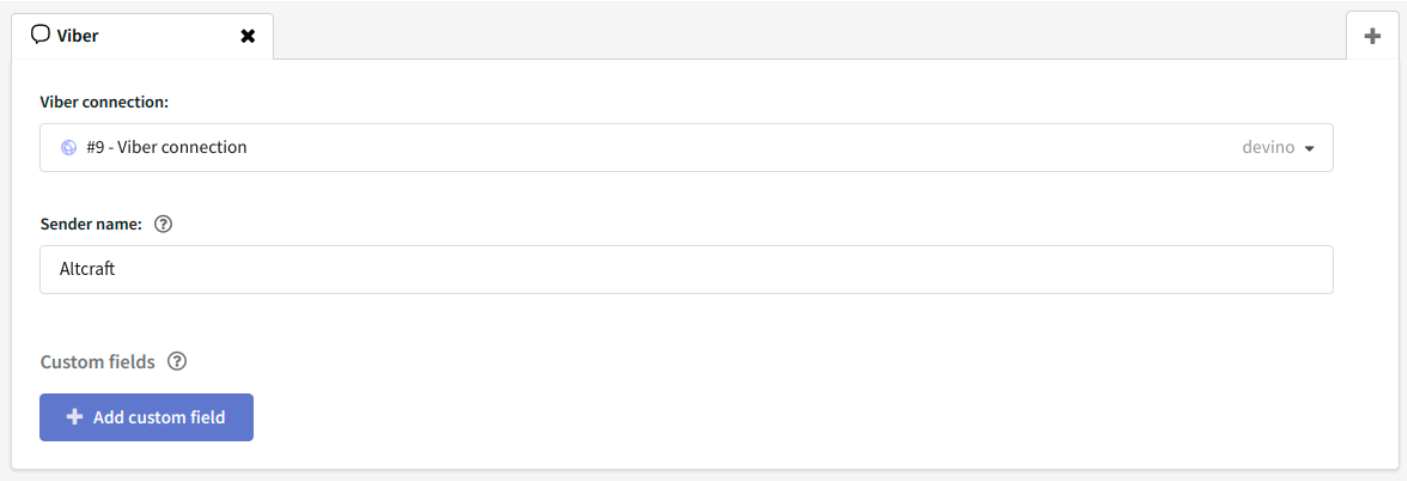
INFO

The profile phone number to which you send messages must be recorded in international format. Please use the following format: +79001234567.

To subscribe clients to a Viber*™ channel, follow these steps:

1. Choose or create a profile database where subscriber data, including phone numbers, will be stored.

2. Create a new resource or edit an existing one, adding a Viber*™ channel. To configure the channel, select the previously created Viber*™ connection and specify the sender's name. Subscribers will see this name when they receive a message. You can find the sender's name in your provider account, and it should be verified by your provider.

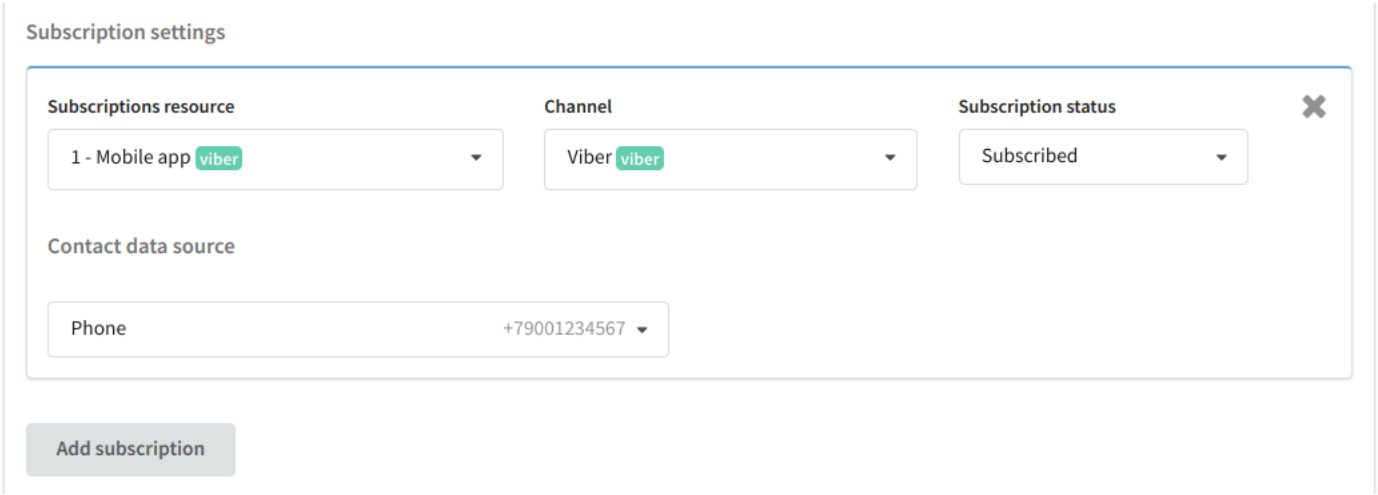


INFO

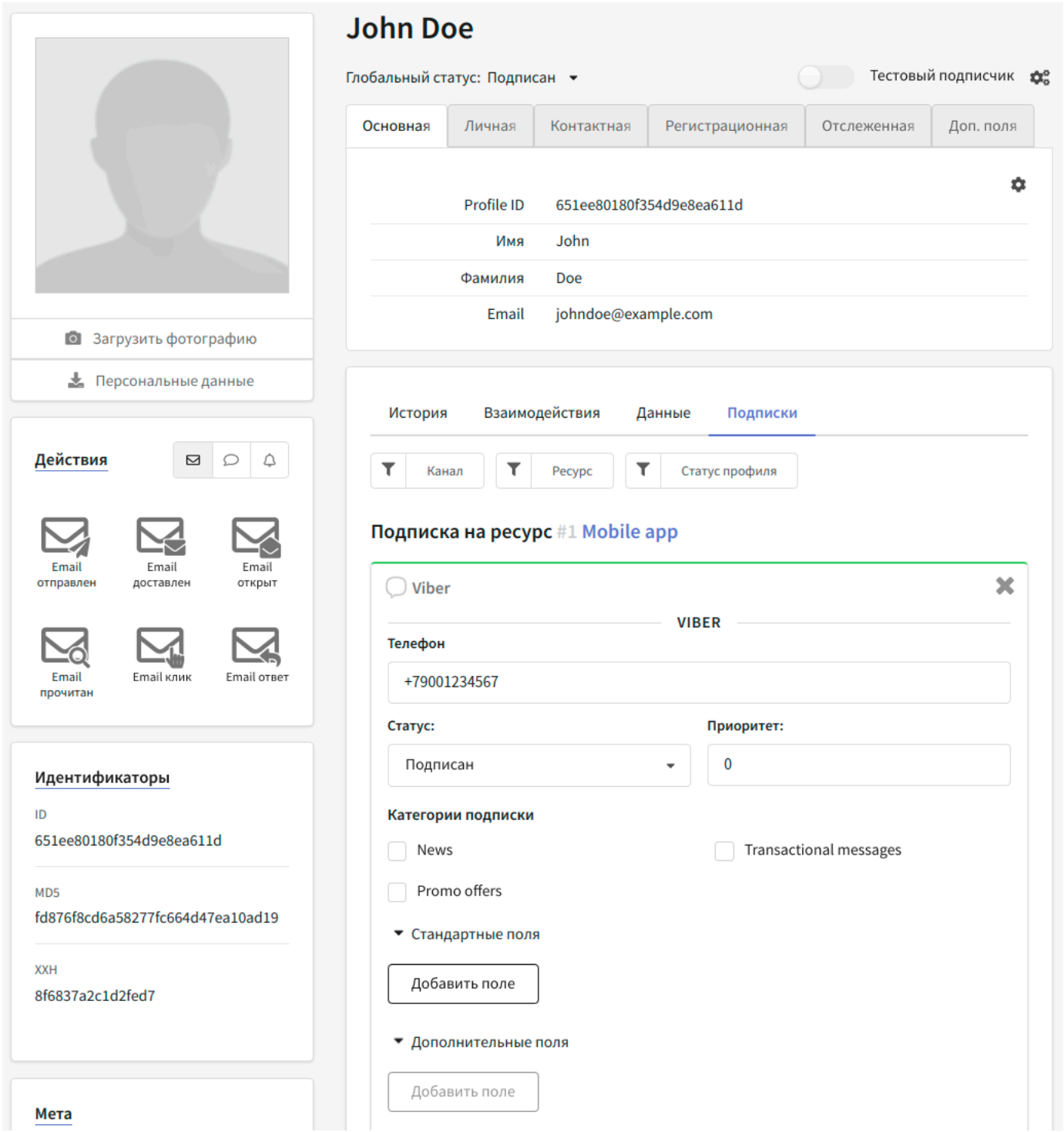
To set up callbacks with message statuses, please contact your account manager or [technical support] of provider:

- **Devino Telecom tech support**
- **SMS Traffic tech support**
- **MTS OmniChannel.**
- **Rapporto.**

3. Import subscriber data into the platform, for example, by using a file. During the import process, make sure to add a subscription to the Viber*™ channel in the resource for each subscriber. You should specify the column in your file that contains phone numbers as the source for subscriptions:



For each customer, a digital profile is created, which compiles their personal information, activity history, and subscriptions to various resources. If a client's subscription to the Viber*™ channel was successfully added during import, this data will be displayed in their history:



Step 3. Create a message template

1. In the main menu, open the Templates section and create a new template using the Create button on the top panel.

2. Choose the creation method — Editor, and the sending channel — *Viber™*.

3. Fill in the general template settings.

4. A Viber*™ template consists of a series of messages. To add the first message to the series, click the + Message button.

5. Select the message type:

Message type	Devino	SMSTraffic	MTS OmniChannel	Rapporto
Text	+	+	+	+
Text + button	+	+	+	+
Text + image + button	+	+	+	+
Image	+	+	+	+
Video	+	+	-	-
Video + text	+	+	-	-
Video + text + button	+	-	-	-
File	+	-	-	-

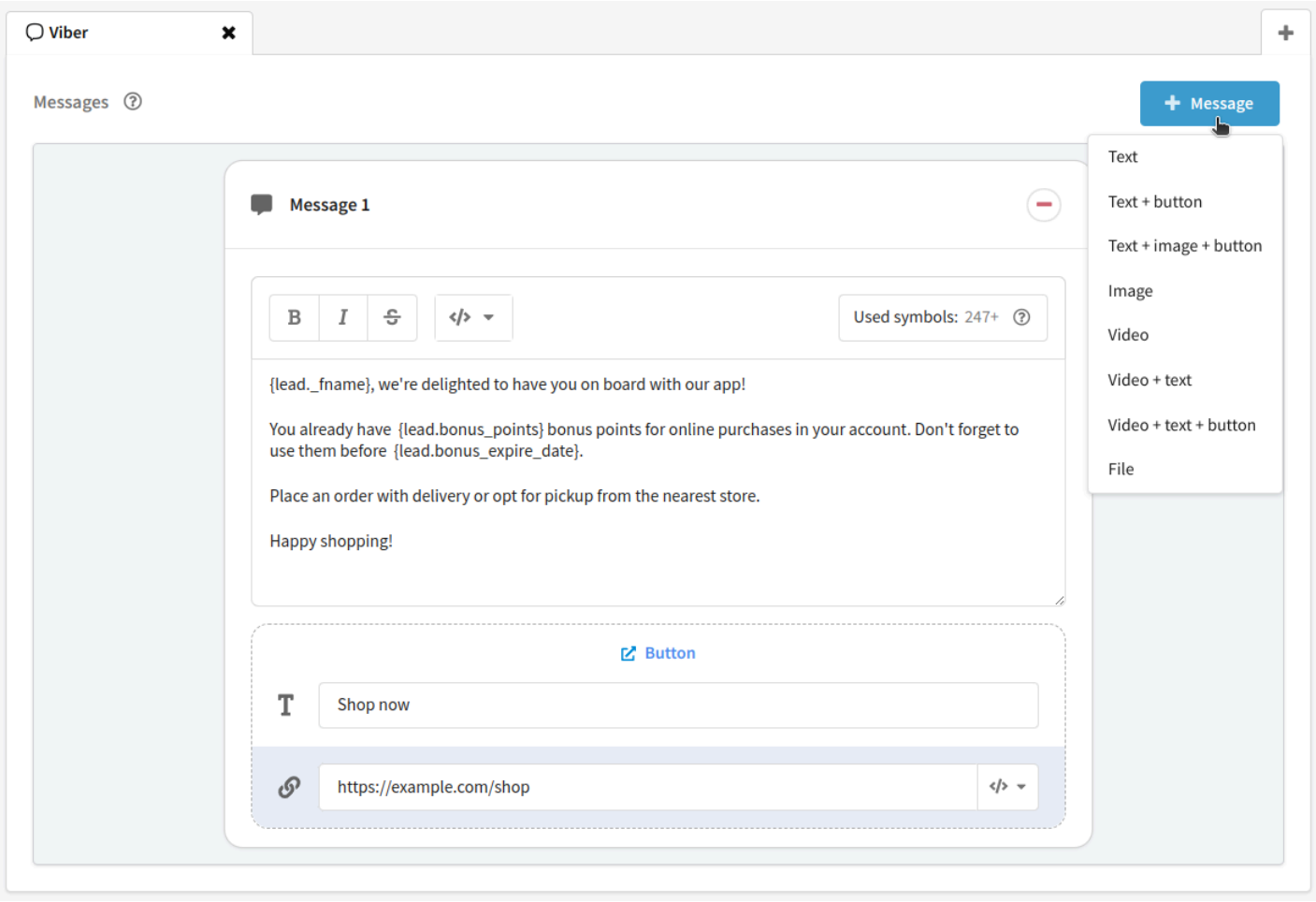
6. Fill in the content. To personalize the message, use variables in your messages

</>

.

INFO

The maximum number of messages in a series is 5. If multiple messages are used in the template, each will be sent with a 2-second interval.



7. Check how your recipients will see the message. To do this, click on the preview button on the top panel.

8. Save your message template.

TIP

You will find more information about setting a template for a Viber*™ channel in this article.

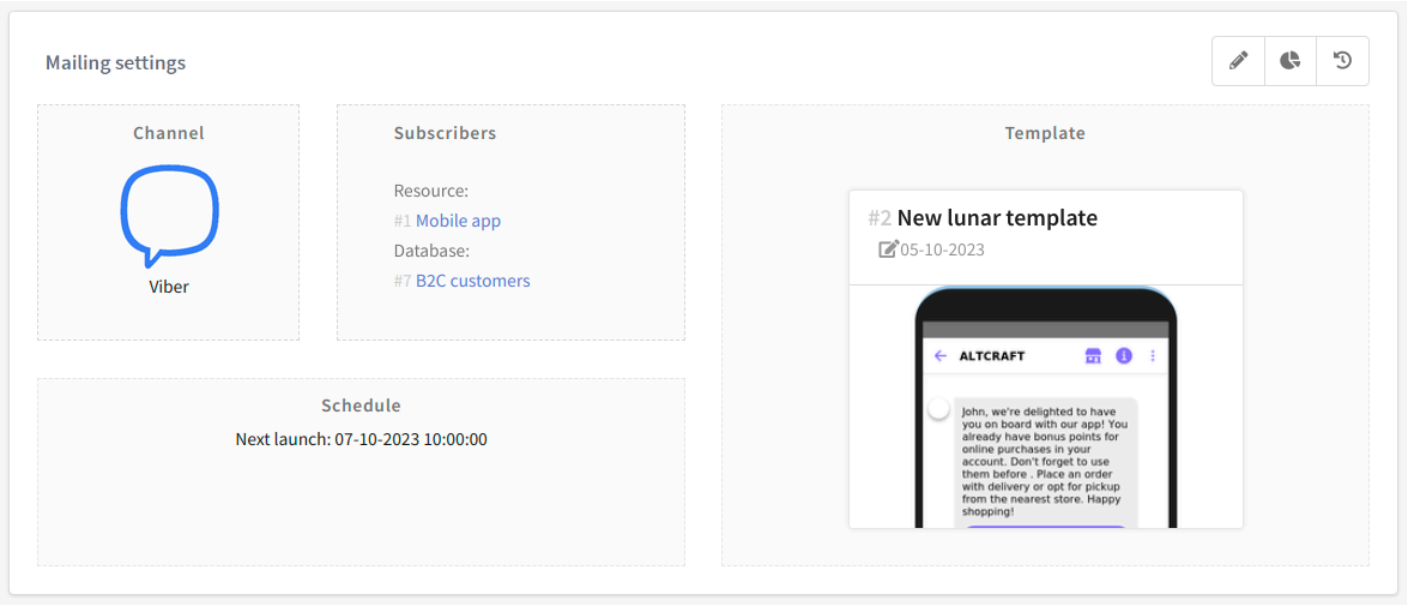
Step 4. Launch the mailing and evaluate its performance

1. Go to the Mailings section and choose the type of mailing:

- **Broadcast is a one-time ending of messages.**
- **Regular mailing is a periodic sending of messages according to a schedule.**

2. Configure the mailing:

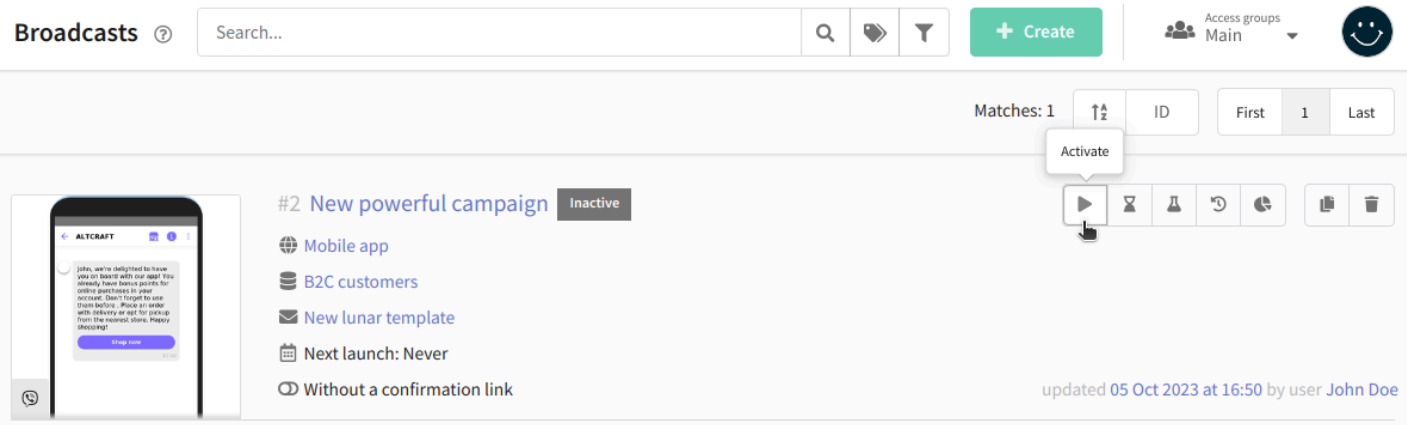
- **Select Viber*™ as the communication channel.**
- **Choose the previously configured resource and the recipient audience (database or segment).**
- **Add the message template to the mailing.**
- **Set the schedule.**



3. Save the mailing.

4. Find it in the list of mailings of the selected type and activate it by clicking the

button in the quick actions panel. An active mailing will start sending messages when the scheduled launch time arrives.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push mailings use the channel report. To access the report, click on the analytics button

in the quick actions panel. The report will provide the following information about your mailing:

- **The number of messages sent, delivered, and undelivered.**
- **The number of reads and clicks on links within the messages.**
- **The count of unsubscribes and subscriptions to the Viber*™ channel.**

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

First Notify mailing

INFO

According to the service policy, new access to the Notify service personal account is currently not provided.

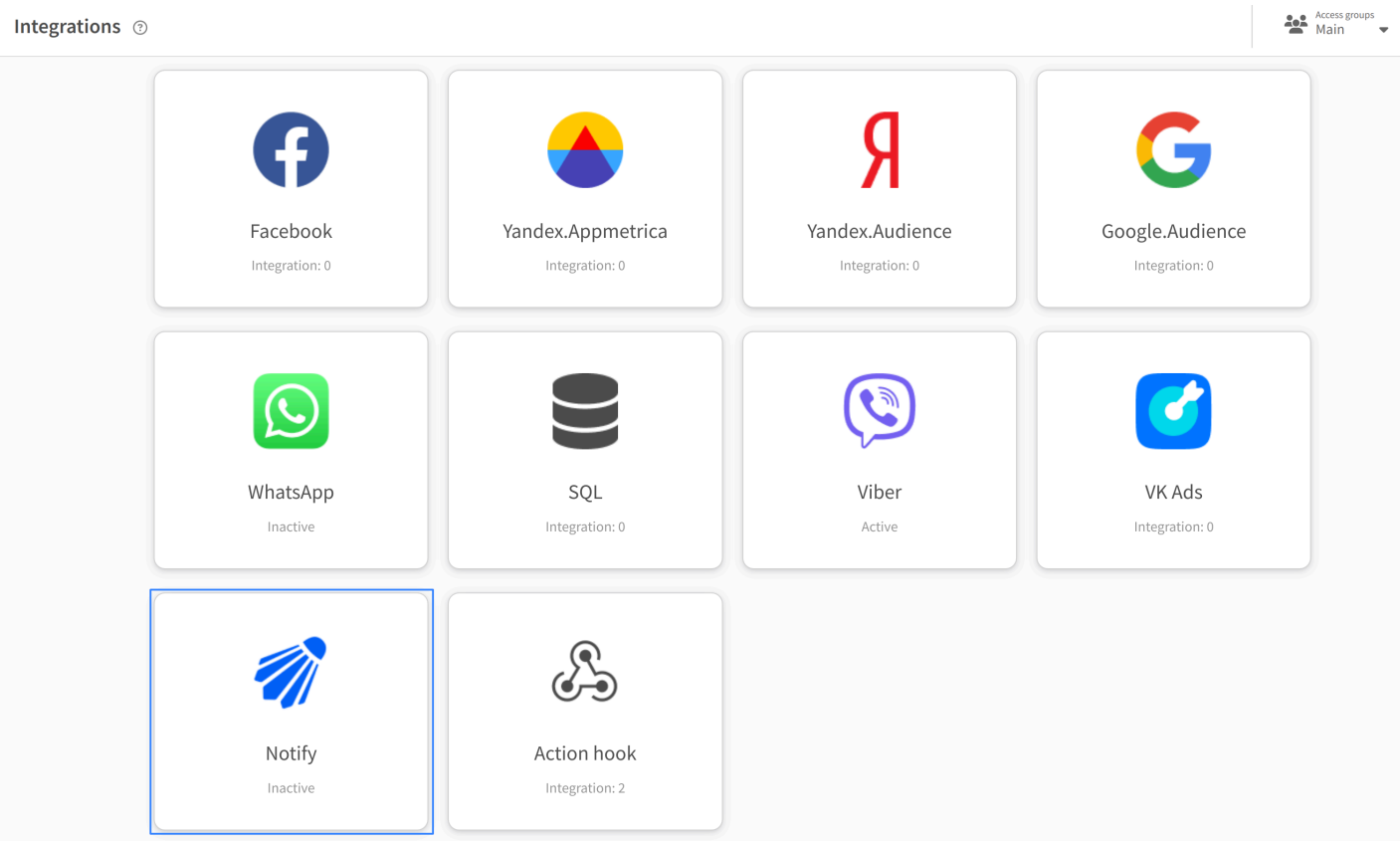
Support for current users is provided by MAX Business tech support.

Notify is a service that allows companies to send service and transactional notifications to VKontakte and Odnoklassniki users. Notifications are delivered from the company's official group, even if the user is not subscribed to it, and can be sent in the form of service messages that can be replied to, or business notifications with no reply option.

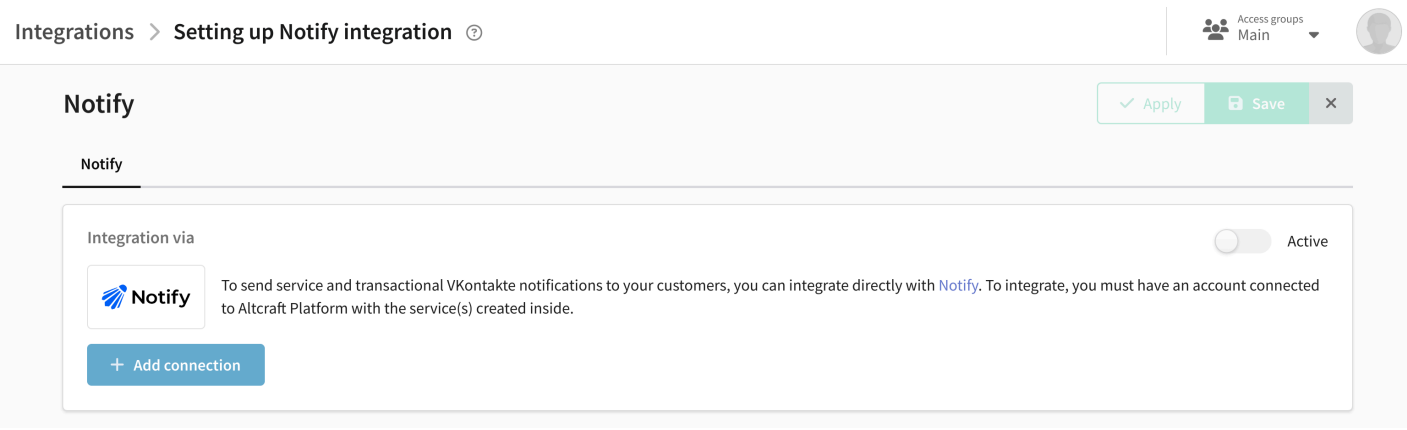
The service allows you to significantly reduce the cost of mailings, ensuring high delivery speed and a high percentage of messages read. In addition to business notifications for VKontakte and Odnoklassniki users, the Notify channel also supports sending SMS reminders.

Step 1: Configure the integration

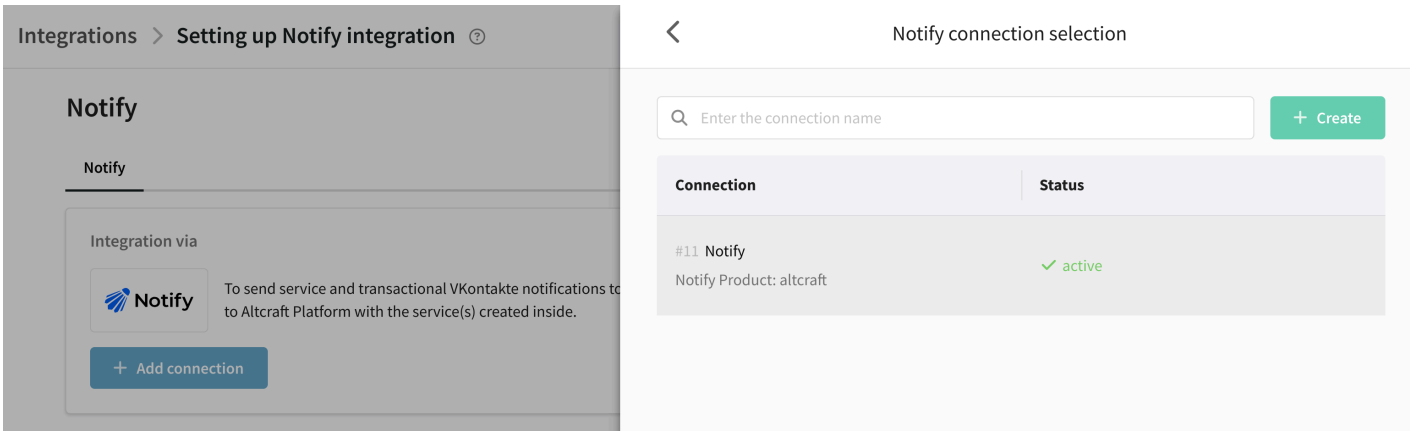
1. Open the Integrations section and select the integration type - Notify:



2. Click Add Connection:



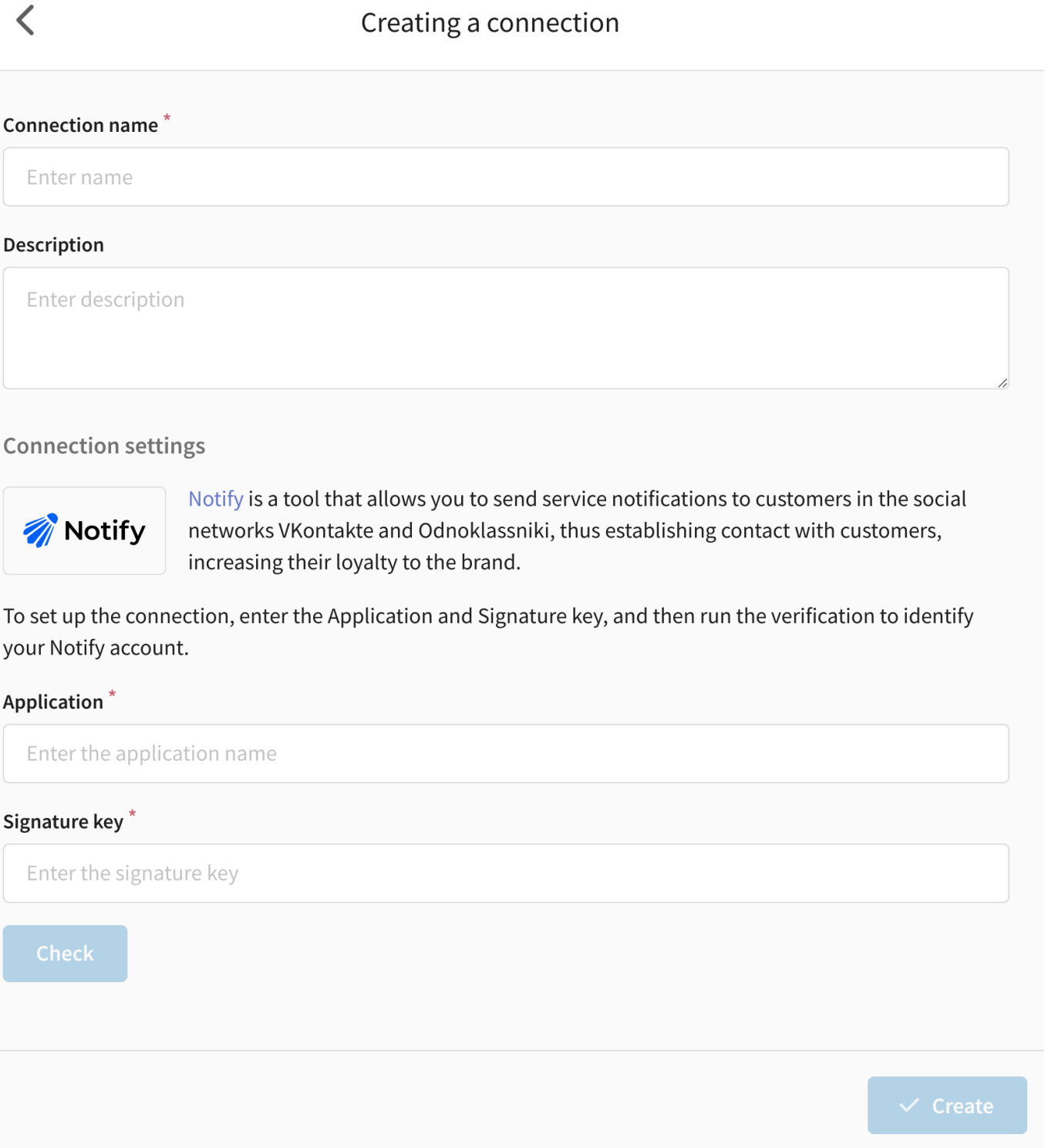
3. Select an available connection to Notify:



No available connections

If you haven’t set up a connection to Notify yet, you can create one right in this window.

Click Create. Enter a name and description for the connection. Next, enter the Application and Signature key parameters and click Check to verify that your data is correct. If the data has been verified, click the Create button in the lower right corner:

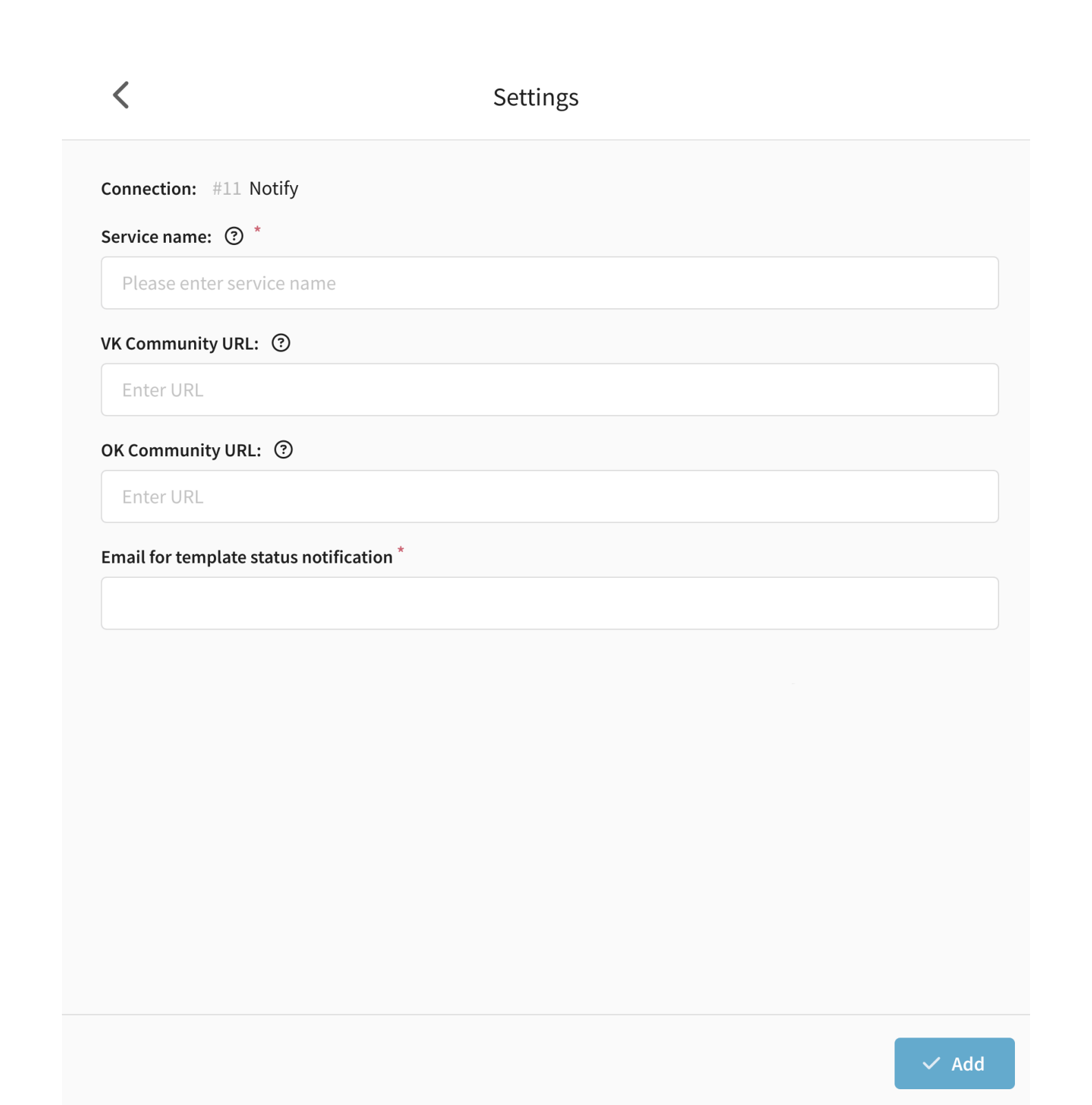


TIP

You can read more about creating a connection ih this article.

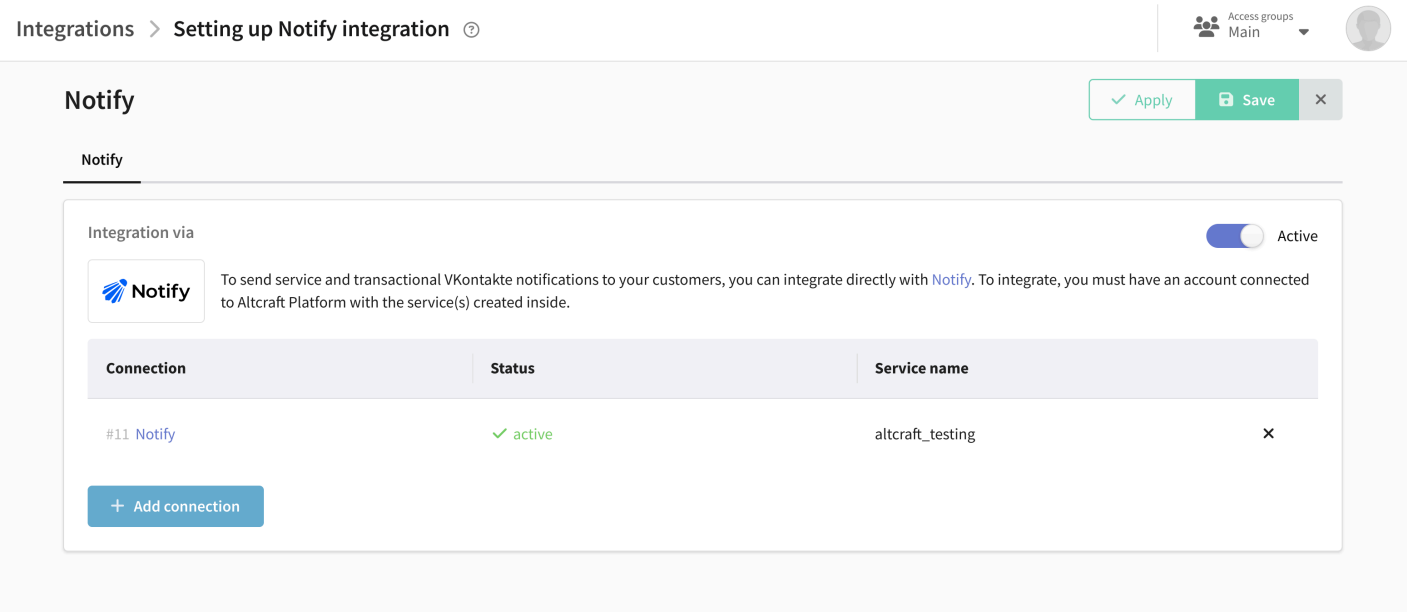
4. Next, specify integration parameters, for correct work they must exactly match the corresponding parameters from Notify account:

- **Service name - the name of the service for mailing, mandatory field.**
- **VK Community URL - link to the VK community on behalf of which the mailing will be made.**
- **OK Community URL - link to the community in OK on behalf of which the mailing will be sent.**
- **Email for template status notification - e-mail address where reports on template statuses will be sent. Required field.**



5. **After filling in the fields, click the Add button. If the entered data is incorrect, the platform will ask you to verify its correctness. If the data is correct, the parameters will be saved.**

6. **Activate the integration using the toggle and save it.**



.

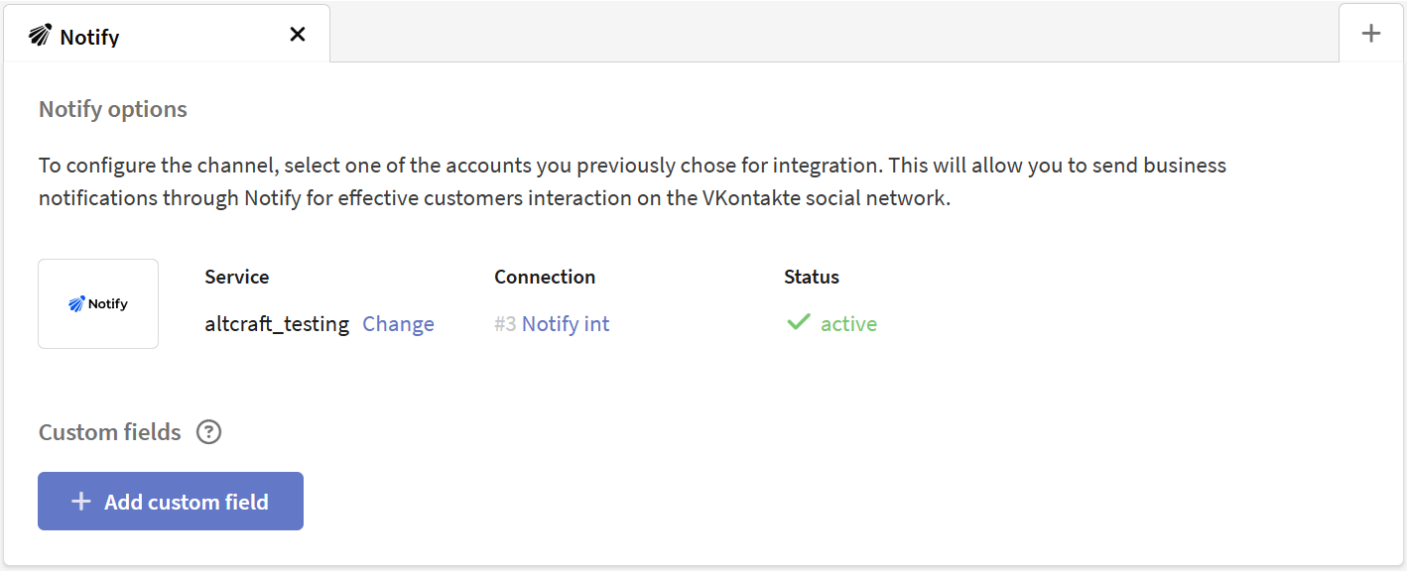
After customizing the integration, you can use it to add a Notify feed to resource.

Step 2: Sign customers up for Notify channel

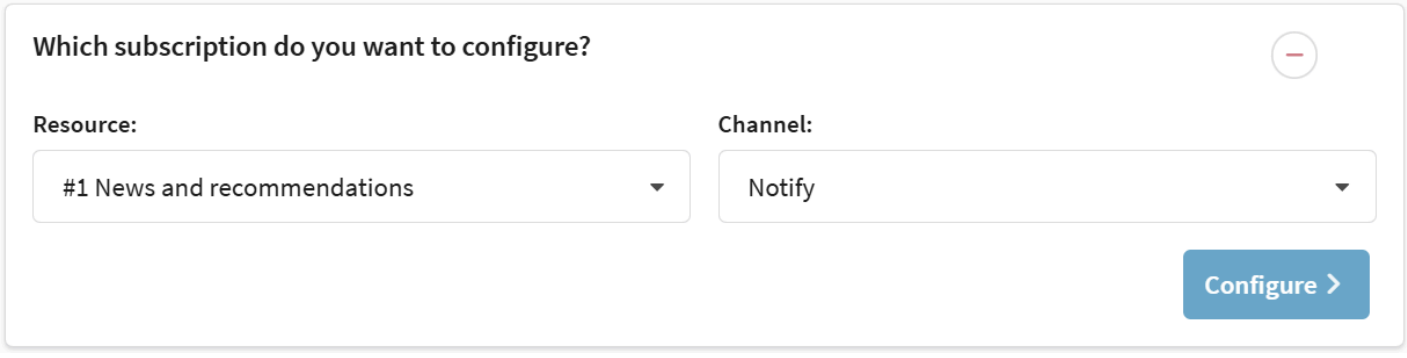
To subscribe clients to the Notify channel, follow the steps below:

1. Select or create a profile database where subscribers' data will be stored.

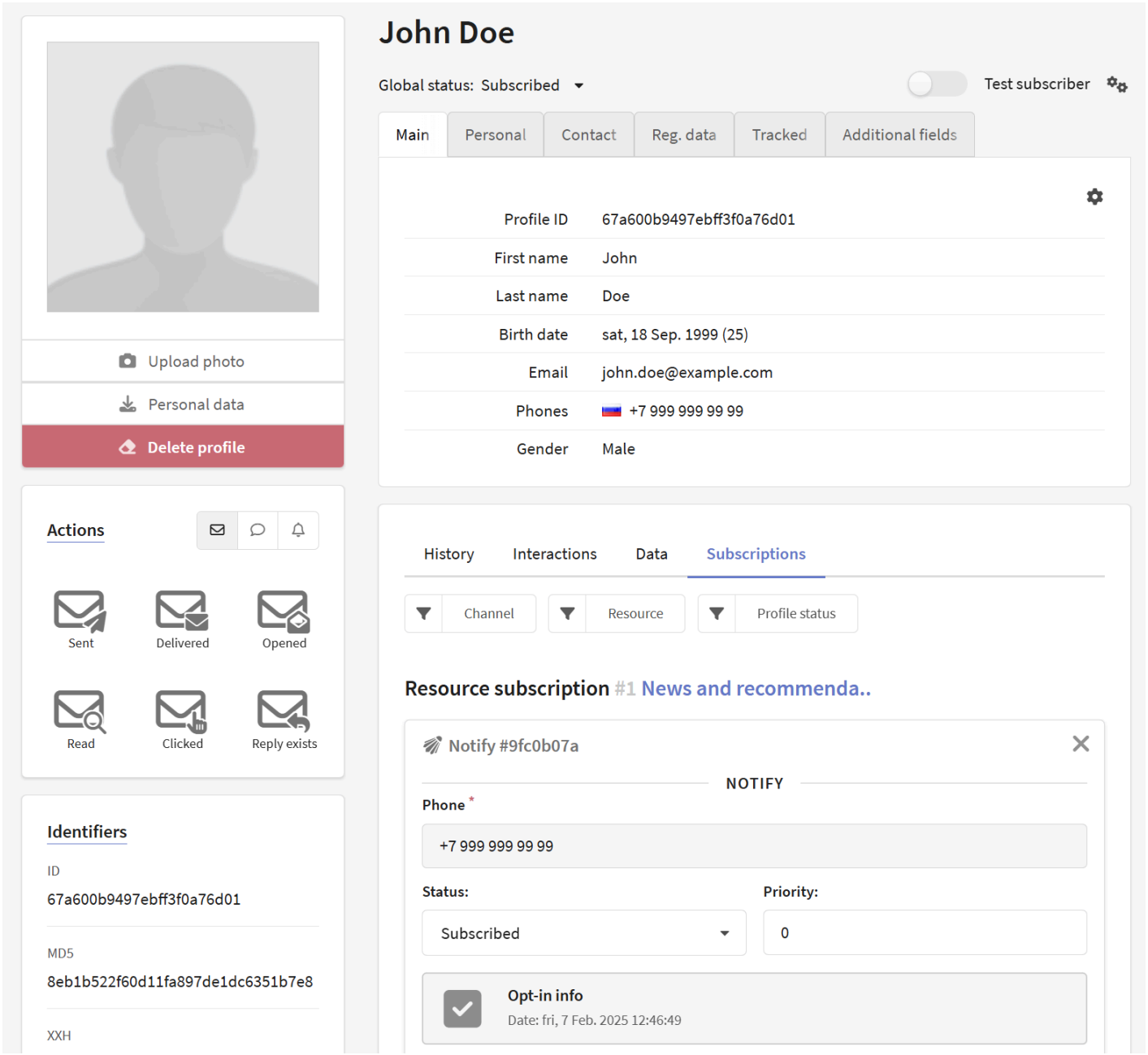
2. Create a new resource or edit an existing one by adding a Notify channel. To customize the channel, select one of the accounts you previously selected for integration.



3. Import subscriber data into the platform - e.g. via a file. When importing, add them a subscription to the Notify channel in the resource. In this case the phone number is the source of the subscription.



A digital profile will be created for each customer, combining personal data, activity history, and resource subscriptions. If a Notify channel subscription was successfully added to a customer during import, this data will be displayed in their history:



Step 3: Create a message template

INFO

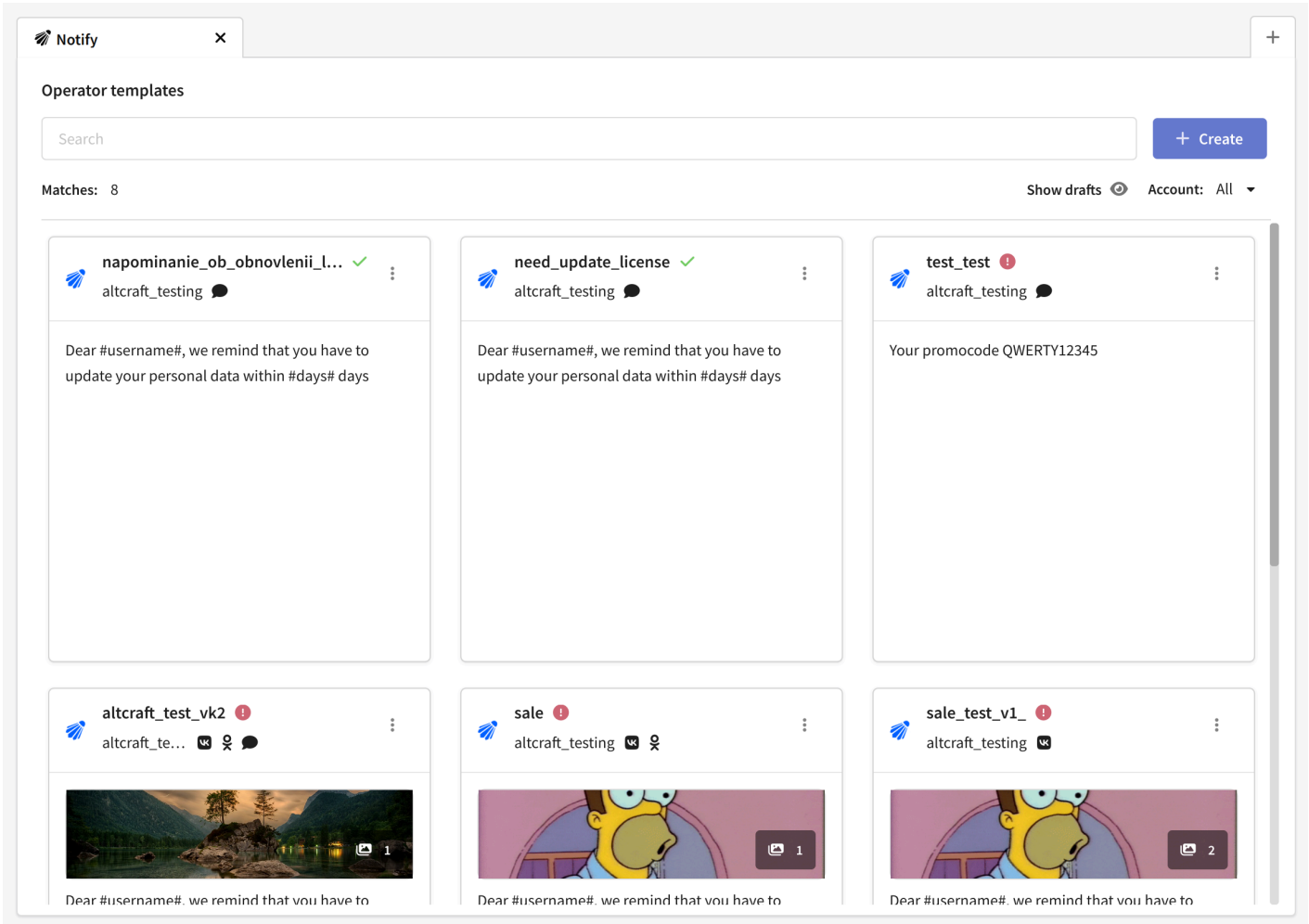
Only templates that have been moderated by Notify can be used to send messages.

1. In the main menu, open the Templates section and create a new template using the Create button on the top panel.

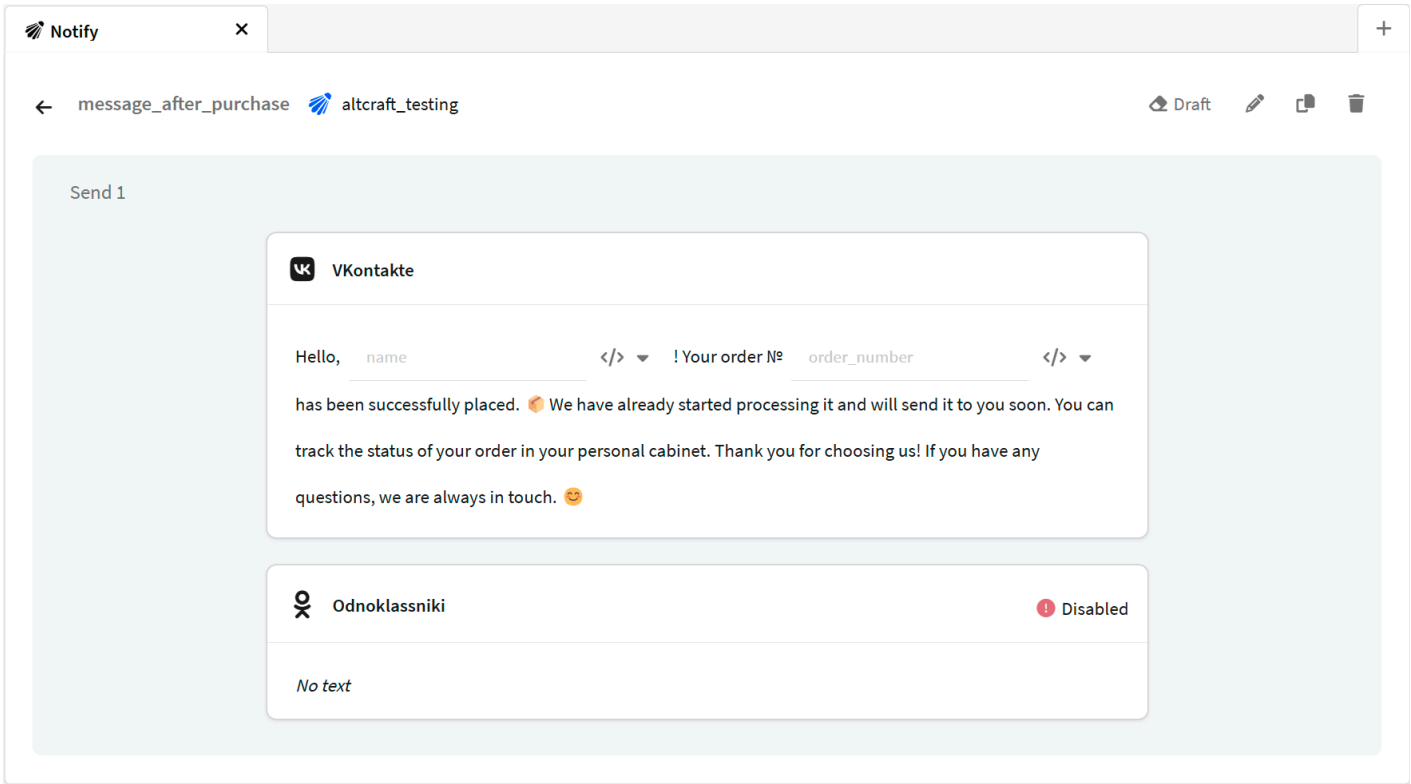
2. Select Editor as the creation method and Notify as the sending channel.

3. Fill in the basic settings of the template.

4. When you open the message editor for Notify, you will be shown the generic templates created in your Notify account. You can also create a new template by clicking the button.



5. Fill in the parameters inside the template. Parameters are text blocks that change for each new message (name, address, event name, date and time, link, etc.). You can manually prescribe parameter values or add Altcraft variables.



6. Check to see how your recipients will see the message. To do this, click the preview button on the top bar.

7. Save the message template.

TIP

You can also create a Notify template while in the platform. To create a template, click the corresponding button. The template creation window will open, in this window you should specify the name of the template and the service configured in your personal cabinet. More information about creating a template for Notify channel can be found in [this article](#).

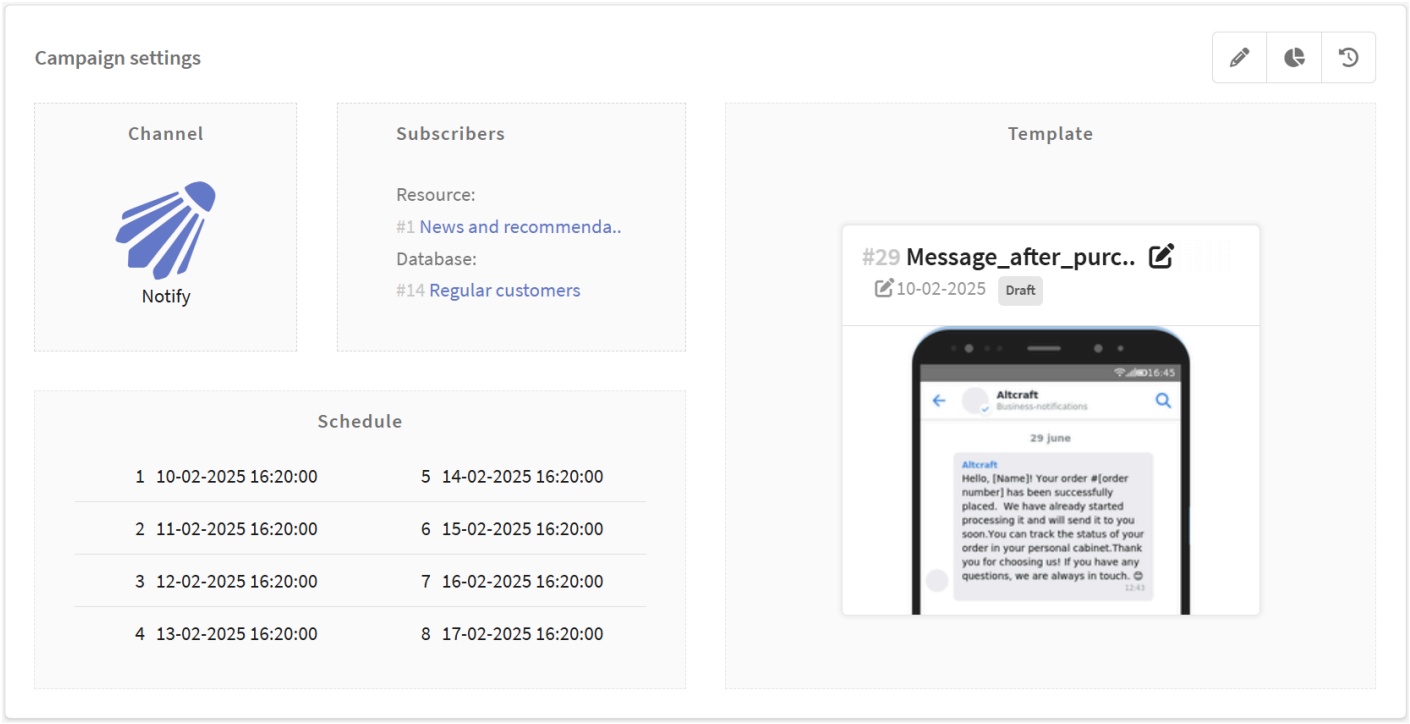
Step 4: Launch the mailing and evaluate the effectiveness of the mailings

1. Go to "Mailings" and select the type of mailing:

- **Broadcast - one-time mailing of messages.**
- **Regular Mailing is a periodic mailing.**

2. Customize the mailing:

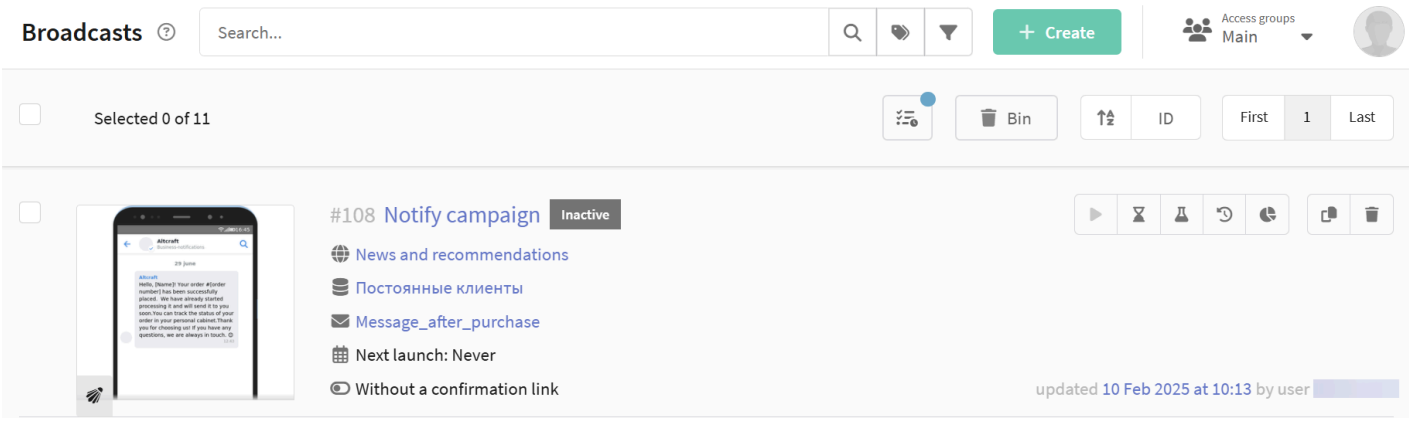
- **select Notify as the communication channel,**
- **select the previously configured resource and recipient audience (base or segment),**
- **add a message template to the mailing,**
- **set the schedule.**



3. Save the mailing.

4. Find it in the list of mailings of the selected type and activate it by clicking

in the Quick Actions panel. The active mailing will start sending messages when it's time to run on the schedule.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push mailings use the channel report to evaluate the results of your mailing. To access the report, click on the analytics button

in the quick action bar. The report will provide the following information about the mailing:

- **number of messages sent, delivered and undelivered;**
- **number of reads and clicks on links in messages;**
- **the number of unsubscribes and subscriptions to the Notify channel.**

Communication Channels Workflow

General Structure

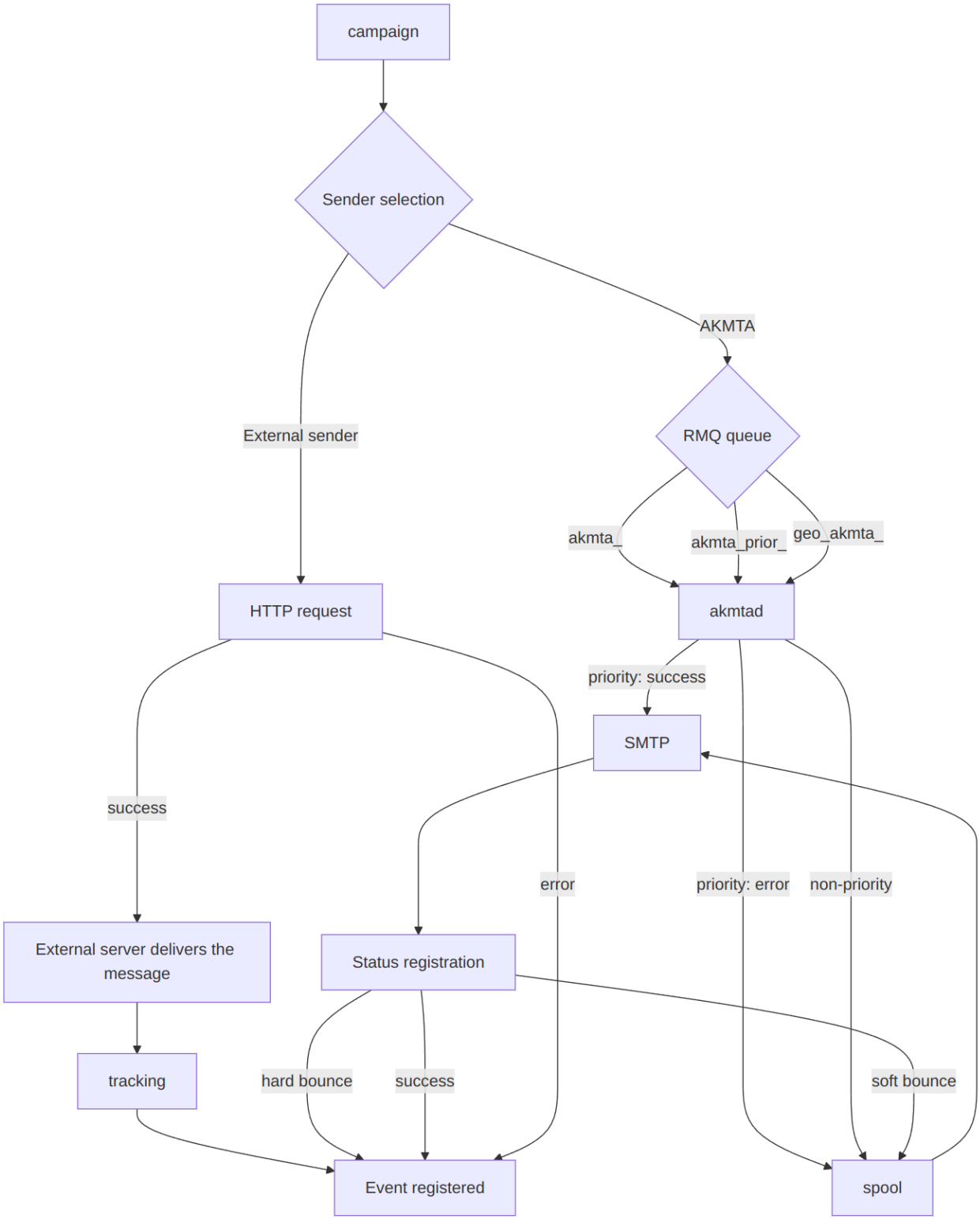
When a mailing is launched, the `mailing` binary is executed. It:

- **processes the template and business logic (blocklists, limits, subscriptions, filters, etc.);**
- **builds the message structure for delivery;**
- **sends the prepared messages to RabbitMQ queues;**
- **records errors in the mailing log.**

Each message then goes to the corresponding processing service depending on the channel.

Email

Several types of senders are supported for the email channel: AKMTA (the Altcraft sender) and external senders. The workflow may vary depending on the sender used.



Sending Process

AKMTA

The mailing sends messages to the following queues:

- `akmta_<sender_id>` — **standard queue;**
- `akmta_prior_<sender_id>` — **priority queue.**

Messages from the queues are processed by the `akmtad` **service. Its behavior depends on the queue type:**

- **For a non-priority queue, it writes messages to disk, periodically reads them, and sends them according to the sending strategy configured for the specific** `sender_id`**;**
- **For a priority queue, it makes an attempt to send immediately; if it fails, it writes the message to disk.**

The sending flow looks like this:

`mailing` → **RMQ Queue (** `akmta` / `akmta_prior` **)** → `akmtad` → `SMTP`

Other Senders

External senders receive messages via an HTTP request from the `mailing` **binary. If the platform cannot send a message, a mailing error is logged. Note that such an error is not considered a "Non-Delivery" and is not included in the non-delivery report.**

Status Registration

AKMTA

Delivery statuses are registered immediately using:

- **the SMTP server response code;**
- **bounce patterns (analysis of bounce templates).**

Other Senders

Tracking service of the platform handles event status.

Retry Logic

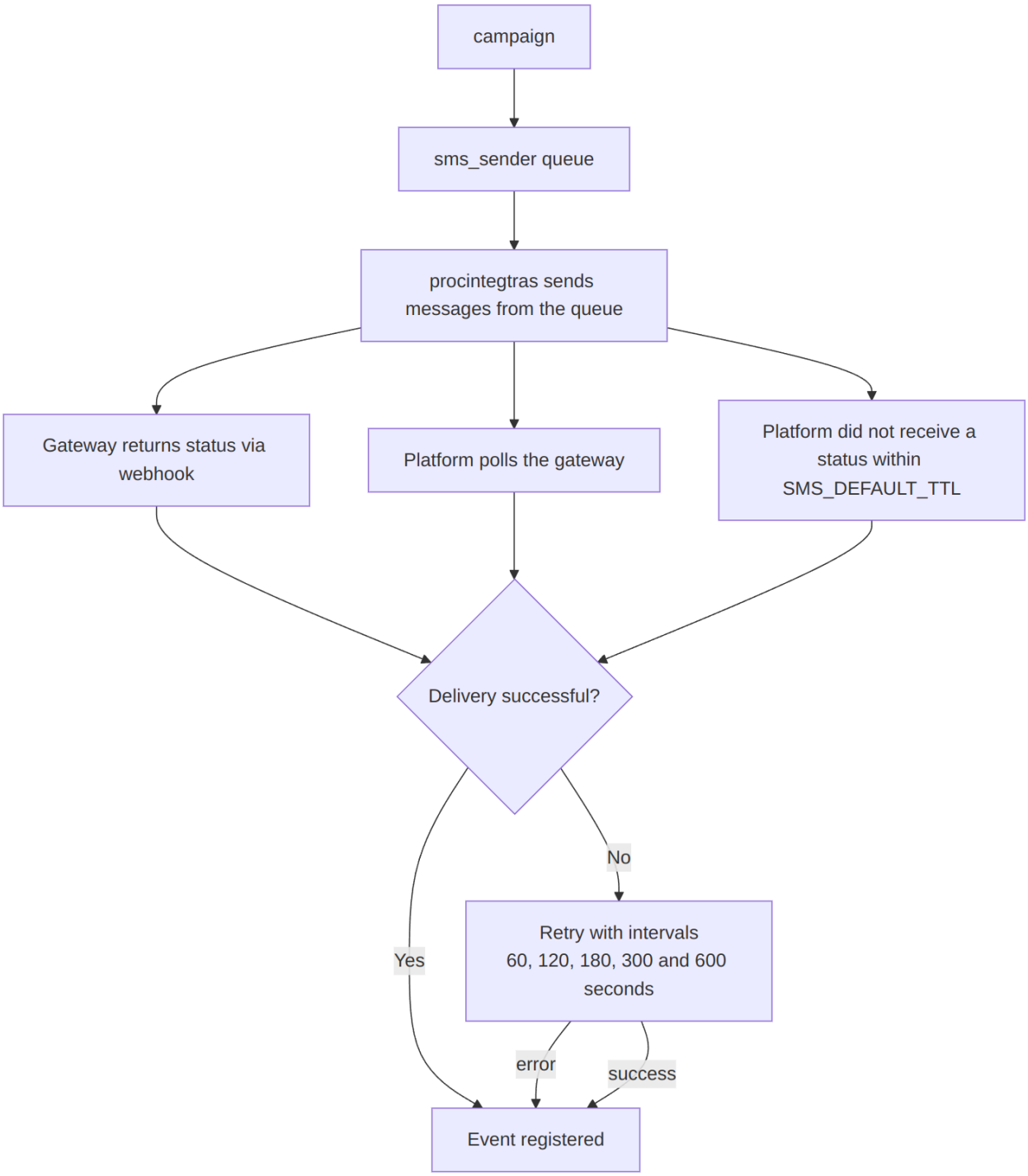
AKMTA

- **Retry rules are defined separately in the Admin Panel. Details are available in the administrator documentation;**
- **Based on the SMTP code, hard bounce or soft bounce statuses are registered;**
- **Retry logic differs for soft and hard bounce cases.**

Other Senders

External senders do not support retry attempts. If an HTTP request fails, the platform does nothing further.

SMS



Sending Process

The mailing sends messages to a single queue — `sms_sender`. The `procintegras` service then sends messages from the queue. Some gateways support batch sending.

Status Registration

Status registration works in one of the following ways:

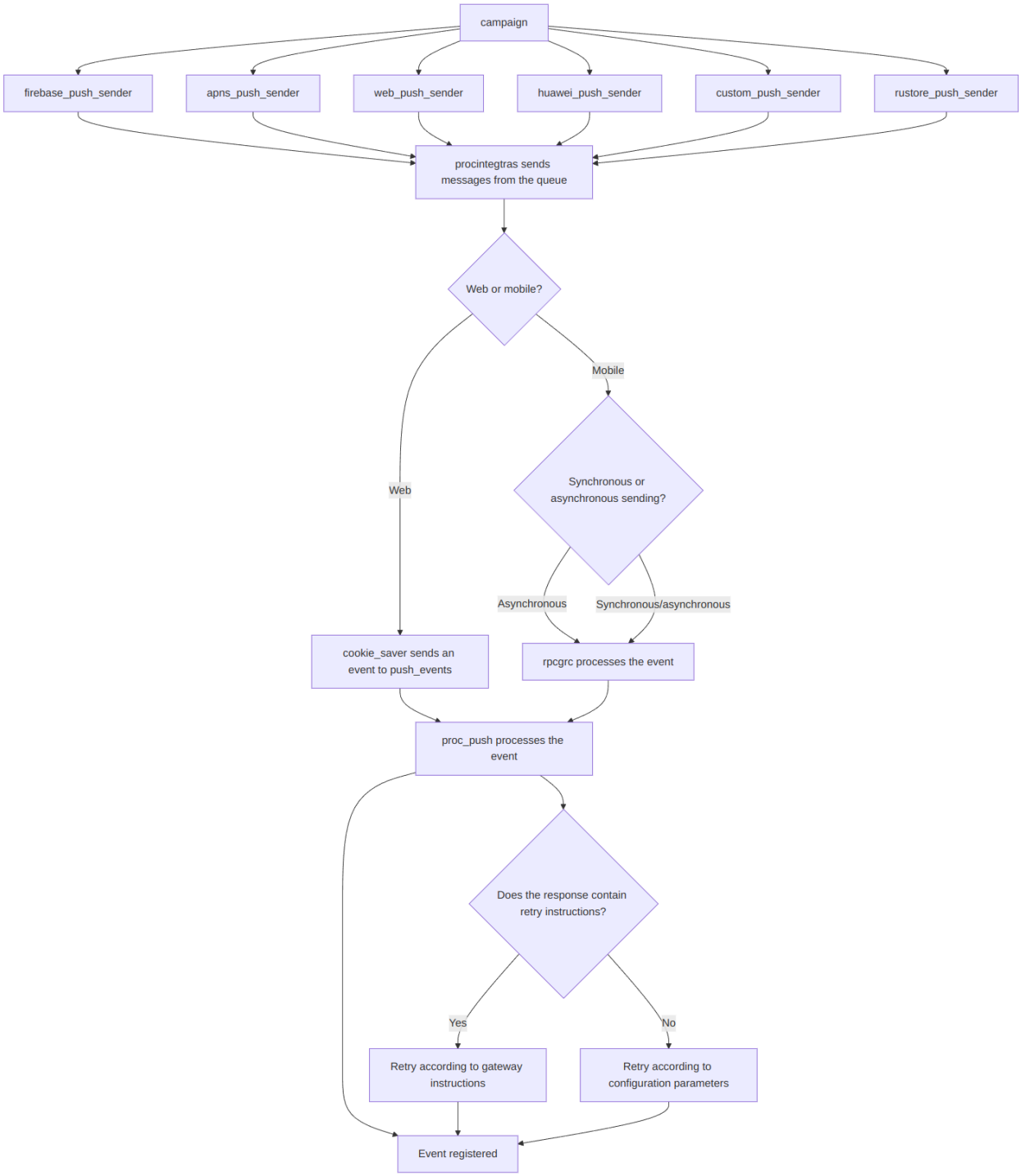
- **The gateway sends a message status via a webhook. The status is processed by `procsmslisten`;**
- **The platform polls the gateway. The status is processed by `procsmsev`.**

The lifetime of an SMS message is defined by the configuration parameter `SMS_DEFAULT_TTL` (default — 1 day). The platform registers a non-delivery if no status is received within this time.

Retry Logic

SMS retries happen after 60, 120, 180, 300, and 600 seconds from the last failed attempt. The maximum number of attempts is 5.

Push



Sending Process

The mailing sends messages to a queue. The specific queue depends on the provider:

- **APNs** — `apns_push_sender`
- **Firebase** — `firebase_push_sender`
- **Huawei** — `huawei_push_sender`
- **RuStore** — `rustore_push_sender`
- **Browser push (Firefox, Safari)** — `web_push_sender`
- **Other providers** — `custom_push_sender`

The `procintegras` service then sends messages from the queue. Some gateways support batch sending.

Status Registration

For browser push, the `cookie_saver` service handles status updates and sends events to `push_events`. Further processing occurs in `procpush`.

For mobile push, the logic is similar, but asynchronous sends are processed via `push_events`, while synchronous ones are immediately handled by `procrpc`.

The lifetime of a push message is defined by `MTA_DEFAULT_TTL` (default — 3 days). A non-delivery is registered if no status is received within this time.

> TIP: Synchronous and Asynchronous Requests
>
> **Synchronous requests — the platform registers the result based on the response, meaning it waits for a reply.**
> **Asynchronous requests — the platform registers the result when putting the message into the queue, meaning it does not wait for a reply.**

Retry Logic

The sender checks the gateway's response and headers to apply retry rules. If the gateway explicitly specifies retry instructions (for example, "try again in 5 minutes"), retries follow them. If not, the platform uses its configured parameters:

- **`PUSH_SENDER_RETRY_MAX_AMOUNT` — maximum retry count (default — 3);**
- **`PUSH_SENDER_RETRY_INTERVAL` — interval between retries (seconds) (default — 1800);**
- **`PUSH_SENDER_RETRY_SCAN_INTERVAL` — scan interval for retry records (seconds) (default — 60).**

Detailed information is available in the administrator documentation.

Messenger Channels

Sending Process

The mailing sends messages to a queue. RabbitMQ queues are processed depending on the channel:

- **Telegram Group, Telegram Bot, MAX Group, MAX Bot — queue `piper_messages`, processed by `procpiper`;**
- **Viber, WhatsApp, Notify — processed by `procintegras`.**

Status Registration

- **Telegram — status is determined based on the HTTP response from Telegram. Delivery is registered if no error occurs during the send request. A non-delivery is registered if an error is returned.**
- **Viber, WhatsApp, Notify — statuses arrive via webhooks. If an error occurs during sending, a non-delivery event is registered. Statuses are handled by `trklisten`.**

Retry Logic

Retries for messages in messengers other than MAX are not supported.
For MAX, retries occur after 15, 30, 60, 120, and 300 seconds; after that, an undelivered event is registered.

> INFO: Sending limits
>
> **Telegram imposes a limit of 28 requests per second per token.**
> **For Notify, the default limit is 20 requests per second, configurable via `VKNOTIFY_SEND_LIMIT_PER_SECOND`, with a maximum of 50 requests per second.**
> **For Viber, the default limit is 50 requests per second, configurable via `VIBER_DEVINO_SEND_LIMIT_PER_SECOND`.**
> **For WhatsApp, the default limit is 20 requests per second (`WHATSAPP_SEND_LIMIT_PER_SECOND`); individual providers may apply their own request limits.**

Email: sending domain configuration

NOTE: Requirements from major email providers

Major email providers (Gmail, Yahoo, Outlook, etc.) require that the email address specified in the `From` field actually exists on the mail server. Additionally, it is recommended that the domain of this address have an A-record pointing to a real resource.

For example, if `From: info@example.com`, then `example.com` must have an A-record, and the mailbox `info@example.com` must be created on the mail server for that domain.

SPF configuration

Sender Policy Framework (SPF) is a technology that verifies whether the sender domain has been spoofed. To make it work, you need to add a special TXT record to your sending domain (sender domain, from-domain), which defines the policy for allowing this domain to send emails from various hosts.

We recommend adding records for different SPF versions:

Domain	Record type	Contents
example.com.	**TXT**	`v=spf1 include:spf.akmta.net ~all`
example.com.	**TXT**	`spf2.0/pra include:spf.akmta.net ~all`

The domain spf.akmta.net contains the current list of Altcraft sender IP addresses.

NOTE

If you are using the Cloud version, SPF is covered automatically by the `include:spf.akmta.net` record.

If you are using the On-premises version, the domain used for MAIL FROM is determined by the sender and pool settings. Make sure the SPF record for your domain allows sending from that domain.

CAUTION: A domain can have only one SPF record

If an SPF record already exists on your domain, do not create a new one — it will overwrite the existing one. Instead, add `include:spf.akmta.net` to the existing record.

Check the current record with:

```
dig example.com TXT
```

Example output if a record already exists:

```
; <<>> DiG 9.18.28 <<>> example.com TXT
;; ANSWER SECTION:
example.com.            300     IN      TXT     "v=spf1 include:sendgrid.net ~all"
```

Good result: ANSWER SECTION contains one record starting with `v=spf1`. Bad result: `NXDOMAIN` (record not found) or multiple `v=spf1` records (this violates the standard).

If a record is found, combine mechanisms into a single line:

Was	Became
`v=spf1 include:sendgrid.net ~all`	`v=spf1 include:sendgrid.net include:spf.akmta.net ~all`

The directive `~all` (or `-all`) must remain single and be placed at the end of the record.

DKIM configuration

DomainKeys Identified Mail (DKIM) adds a digital signature to your email, linked to the from-domain. The signature is automatically verified on the recipient side, then used to refine sender reputation and displayed to the user. The email is signed with a private key installed on the sender side and unknown to anyone else. The public key is placed as a special TXT record on a subdomain of the from-domain.

You need to add the following records. Choose a selector — for example, `ak`. The selector is the prefix in the DKIM subdomain name. Selectors allow different senders to use the same domain: for example, `ak._domainkey.example.com` for Altcraft and `mail._domainkey.example.com` for another service.

Domain	Record type	Contents
ak._domainkey.example.com.	**TXT**	`v=DKIM1; k=rsa; p=<your personal key>`
_domainkey.example.com.	**TXT**	`o=-;`

NOTE

The selector in DNS must match the selector specified in the platform's sender settings. The from-domain is determined by the Sender Email field — set in the resource, template, or mailing. Priority order: mailing, template, resource.

TIP

For Altcraft cloud customers, the selector is always `akru2022`. For On-premises deployments, the selector can be any value — it is defined in the platform's sender settings.

DMARC configuration

DMARC technology allows the mail server to decide what to do with email if something is wrong with the DKIM and SPF records.

We recommend setting rules that reject messages received from third-party IP addresses or signed incorrectly.

Domain	Record type	Contents
_dmarc.example.com.	**TXT**	`v=DMARC1; p=reject; sp=reject; rua=mailto:report@example.com`

NOTE

DMARC verifies that the domain in `From` aligns with SPF and DKIM results. Make sure the from-domain (determined by the Sender Email field in the mailing, template, or resource) matches the domain for which SPF and DKIM are configured.

Tracking configuration

You need to add the following record on a subdomain of your from-domain:

Domain	Record type	Contents
trk.example.com.	**CNAME**	`trk.aksend.net`

Additionally, you can use as many tracking domains as needed; the configuration is the same.

NOTE

The record trk.aksend.net is used only for the Altcraft cloud solution. If you are using an On-premises solution, enter your own record. In the UI when creating a resource, the Tracking Domain field allows you to set a domain for a specific resource.

CNAME settings for processing bounces

Cloud clients must also configure an additional CNAME record pointing to the domain `akmta.net`:

Record example:

```
ac-bounces.example.com IN CNAME akmta.net
```

Here `ac-bounces` is the subdomain, and `example.com` is your from-domain. This subdomain will be used in the Return-Path, allowing SPF authentication and "alignment" to improve email deliverability.

A-record configuration for the from-domain

Major email providers (Gmail, Yahoo, Outlook, etc.) verify that the from-domain has an A-record pointing to a real IP address. Without this record, emails may not be delivered or may end up in spam.

Add an A-record for your from-domain pointing to the IP address of your mail server:

Domain	Record type	Contents
example.com.	**A**	`<IP address of your mail server>`

For example, if `From: info@example.com`, then `example.com` must have an A-record, and the mailbox `info@example.com` must exist on the mail server for that domain.

NOTE

The A-record must point to the IP address of the mail server where the mailbox from the `From` field actually exists. This can be the IP of your corporate mail server, Microsoft 365, Google Workspace, or another email hosting provider.

MX-record configuration

The MX record should point to the company's mail server — the organization sending emails on whose behalf.

It is required because spam filters check for the presence of an MX record on the sender's domain. The record must point to a real mail server, otherwise the email may be rejected or delivered to spam.

Additionally, replies from recipients are routed to the address specified in the MX record if `reply-to` is not set and reply capture is disabled in the platform account settings.

Domain	Record type	Contents
example.com.	**MX**	`<address of your mail server>`

Additional settings

BIMI (optional)

BIMI technology (Brand Indicators for Message Identification) is used together with SPF, DKIM and DMARC to display a branded image next to the email in the recipient's inbox. This setting is optional — it is needed if you want your company logo to appear next to emails in supporting mail clients.

> NOTE
>
> **BIMI email authentication is supported by email services such as Fastmail, Gmail, and Yahoo!**

To use this technology, you need to provide a path to a file in svg format. The image must be square and have no additional layers.

Add the following record in your domain settings:

Domain	Record type	Contents
default._bimi.example.com	**TXT**	`v=BIMI1; l=https://example.com/bimi/bimi.svg`

You can also use the record generator at https://bimigroup.org/bimi-generator/.

Troubleshooting

If DKIM, SPF, DMARC, or antispam headers show poor values after a test send, check your domain settings, sender configuration, and email content.

DKIM, SPF, or DMARC verification fails

If the email headers show `dkim=fail`, `spf=fail`, or `dmarc=fail`, the issue is most likely related to domain configuration.

Check:

- **DKIM selector in the admin panel;**
- **DKIM key in the domain DNS record;**
- **SPF record for the domain;**
- **DMARC record for the domain;**
- **domain in the `From` field;**
- **reverse address domain;**
- **domain alignment for DMARC to pass.**

If the domain uses a single DKIM selector, the admin panel must specify the same selector. For example, if the DNS publishes the record:

```
mail._domainkey.example.com
```

then the admin panel must specify the DKIM selector as:

```
mail
```

If a different selector is specified in the admin panel, the mail server will not find the correct DKIM key or will verify the email against a different DNS record.

Also verify the DKIM key itself. The public key value in DNS must match the key used to sign the email. If the key was replaced in DNS but not updated in the admin panel, DKIM may fail verification.

SPF verification fails

If the headers show `spf=fail`, check the SPF record for the domain used in the reverse address.

The reverse address is usually displayed in the header:

```
Return-Path: <bounce@example.com>
```

or in the SPF check result:

```
spf=fail smtp.mailfrom=bounce@example.com
```

SPF is checked not against the visible sender address in `From`, but against the domain from `smtp.mailfrom` / `Return-Path`.

Verify that the domain's SPF record allows sending from the required server or service. For example:

```
example.com. TXT "v=spf1 include:spf.example.net -all"
```

If sending through an external service, it must be added to the SPF record via `include` or another recommended mechanism.

Also check that the domain does not have multiple SPF records. There should be only one TXT record per domain starting with:

```
v=spf1
```

DMARC verification fails

If DKIM and SPF show `pass` but DMARC shows `fail`, the issue is most often related to mismatched domains.

Example of a problematic scenario:

```
From: News <news@example.com>
Return-Path: <bounce@mailer-service.com>

Authentication-Results: mx.example.com;
 dkim=pass header.i=@mailer-service.com;
 spf=pass smtp.mailfrom=mailer-service.com;
 dmarc=fail header.from=example.com
```

In this case, DKIM and SPF pass verification, but not for the domain `example.com` specified in `From`.

For DMARC to pass, at least one of the following conditions must be met:

- **DKIM passes verification and the DKIM domain aligns with the domain in `From`;**
- **SPF passes verification and the `smtp.mailfrom` domain aligns with the domain in `From`.**

If a different domain is used in the `From` field, it must also be configured: it needs its own DKIM, SPF, and DMARC records.

A different From-domain is used

Issues may arise if the email uses one sender domain while the configuration is set up for a different domain.

For example, the admin panel has the domain configured:

```
example.com
```

but the email is sent from the address:

```
From: News <news@promo.example.net>
```

In this case, DKIM, SPF, and DMARC must also be configured for the domain `promo.example.net`. If this domain is not configured, verification may fail.

Before a test send, make sure the domain in `From` matches the domain configured in the admin panel, or is a properly configured subdomain of it.

DNS records are not configured for the domain or IP address

For correct email delivery, DNS records for both the domain and the sending IP address are required.

Check:

- **`A` record for the domain;**
- **`MX` record for the domain;**
- **`PTR` record for the sending IP address;**
- **correctness of the reverse address;**
- **domain availability in DNS.**

The `PTR` record is especially important for the sending IP address. It links the IP address to a domain name and is used by mail systems as one of the reputation signals. The PTR record is configured with your hosting provider or IP management panel, not in the domain's DNS zone.

Verifying records with `dig`

Check basic records:

```
dig example.com A
dig example.com MX
dig example.com TXT
```

Example successful output for an A-record:

```
; <<>> DiG 9.18.28 <<>> example.com A
;; ANSWER SECTION:
example.com.            300     IN      A       192.0.2.4
```

Good result: ANSWER SECTION contains a record with an IP address. Bad result: `NXDOMAIN` (domain not found) or empty ANSWER SECTION.

Check the PTR record (reverse DNS) for the sender IP:

```
dig -x 192.0.2.4
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> -x 192.0.2.4
;; ANSWER SECTION:
4.0.2.192.in-addr.arpa. 300     IN      PTR     mta-1.example.com.
```

Good result: PTR points to a domain matching the A-record (`mta-1.example.com`). Bad result: `NXDOMAIN` (PTR not configured) or PTR points to a different name.

Check the DKIM record (replace `ak` with your selector):

```
dig ak._domainkey.example.com TXT
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> ak._domainkey.example.com TXT
;; ANSWER SECTION:
ak._domainkey.example.com. 300  IN      TXT     "v=DKIM1; k=rsa; p=MIGfMA0GCS..."
```

Good result: record found, starts with `v=DKIM1; k=rsa; p=`. Bad result: `NXDOMAIN` (selector not found) or `p=` key is empty.

Check the DMARC record:

```
dig _dmarc.example.com TXT
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> _dmarc.example.com TXT
;; ANSWER SECTION:
_dmarc.example.com.     300     IN      TXT     "v=DMARC1; p=reject; sp=reject; rua=mailto:report@example.com"
```

Good result: record found, contains `v=DMARC1` and policy `p=`. Bad result: `NXDOMAIN` (DMARC not configured) or policy `p=none` (not recommended for production).

The domain or IP address has not been warmed up yet

Even with correct DKIM, SPF, and DMARC, an email may receive a poor antispam score if sending started at a high rate to an unwarmed domain or IP address.

A new domain or new IP address requires gradual warm-up. Start with a small volume of sends and increase the rate gradually.

If you launch mass sending at a high rate immediately, mail systems may start rejecting emails, sending them to spam, or lowering the sender's reputation.

During warm-up, monitor:

- **deliverability;**
- **opens and clicks;**

- **spam complaints;**
- **bounces;**
- **spam placement;**
- **domain and IP address reputation.**

Email fails antispam filters

If DKIM, SPF, and DMARC pass verification but the email receives a high `SCL`, `BCL`, or ends up in spam, the issue is most likely not related to domain authentication but to the email content or sender reputation.

Different mail providers use different antispam filters with different headers and scores. For example, SpamAssassin uses `X-Spam-Status` and `X-Spam-Score`, Microsoft 365 uses `SCL` and `BCL`, and Gmail does not publish numeric scores. For universal checking, use online testers (see below).

Check:

- **email subject;**
- **email body text;**
- **HTML markup;**
- **text-to-image ratio;**
- **links;**
- **domains in links;**
- **attachments;**
- **presence of URL shorteners;**
- **domain reputation;**
- **IP address reputation;**
- **sending frequency;**
- **history of complaints and bounces.**

The email should appear acceptable to antispam filters: no suspicious links, aggressive wording, unnecessary attachments, broken markup, or phishing indicators.

Useful verification tools

For diagnostics, you can use external services:

- **Mail-Tester.com — send a test email to a unique address, receive a 0–10 score with detailed analysis of DNS, headers, SPF, DKIM, DMARC, and content.**
- **MXToolbox — DNS record, SPF, DKIM, DMARC, PTR, and blacklist checking.**
- **Google Postmaster Tools — monitoring domain reputation, IP addresses, delivery errors, and complaints for Gmail.**
- **Postmaster Mail.ru — monitoring reputation and send quality for Mail.ru.**
- **Microsoft SNDS — monitoring IP address reputation in Microsoft infrastructure.**

After making corrections, perform another test send and check the email headers again. A normal authentication result:

```
dkim=pass
spf=pass
dmarc=pass
```

If you use Microsoft 365 or Outlook, additionally check:

```
compauth=pass
SCL:0
BCL:0
```

Email: setting up and using postmasters

##Description {#id-postmasters-description}

Postmaster is a service of an email provider that tracks email distrubution statistics: how many messages were delivered, how many ended up in spam, sending failures, how many emails were complained about, etc.

Postmaster registration is required if you plan to deliver large volumes of messages, more than 10 thousand per day. A postmaster is also required when registering an FBL.

> TIP
>
> **FBL/CFL (FeedBack Loop) is a standard for providing information about spam complaints from an email service provider to the sender of emails. It is highly recommended to use this technology on all ISPs where it is available in one form or another. You can learn more about FBL in this article.**

The choice of postmaster depends on which email domains are most commonly used in your database. Each postmaster provides statistics only for their domains.

The most popular postmaster services in Russia:

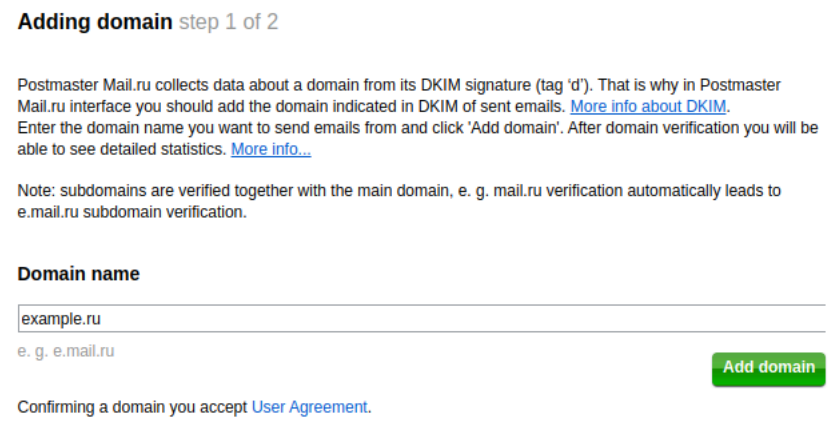
- **Mail.ru Postmaster**
- **Gmail Postmaster Tools**
- **Outlook.com Postmaster**
- **Yahoo Complaint Feedback Loop Service**

How to set up Postmaster Mail.ru

Step 1. Create a Mail.ru account or log in using one of the specified methods.

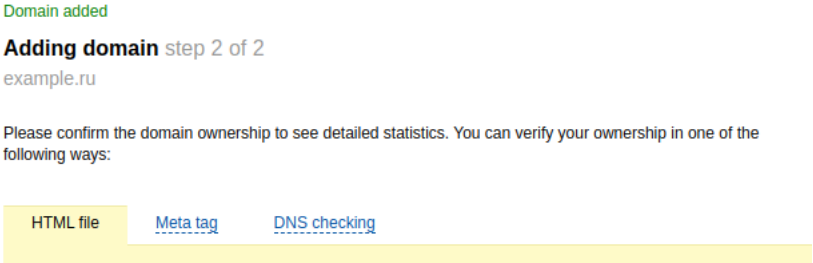
Step 2. Go to the domain adding page - https://postmaster.mail.ru/add/.

Step 3. Enter the name of the domain about which you want to collect statistics and click “Add Domain”:

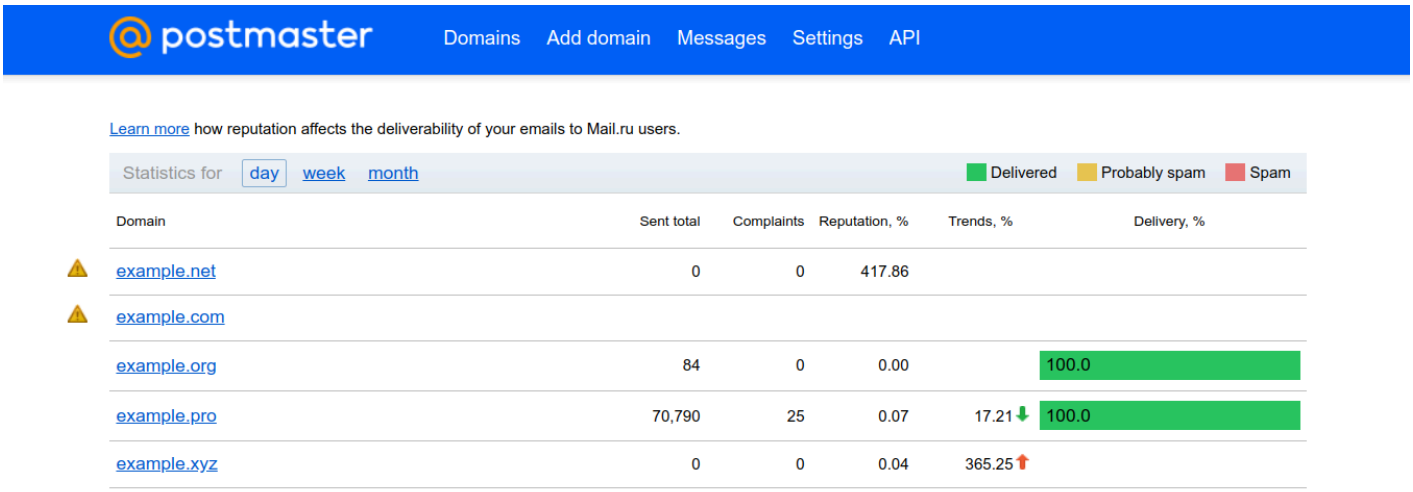


Step 4. Select one of the domain verification methods:

- **HTML file - this method can only be used if the domain has a web server. Follow the instructions described on the tab and click “Confirm”**
- **Meta tag - used only if you have your own website. Add a tag with a personal code to the `head` section of the main page of the site and click “Confirm”**
- **DNS check - allows you to use the postmaster even if you don’t have your own website. The DNS record value must be placed in the domain’s DNS records using the interface of the hosting provider.**



If all instructions are completed correctly, the domain will be added to the postmaster. In the described way, you can add an unlimited number of domains:

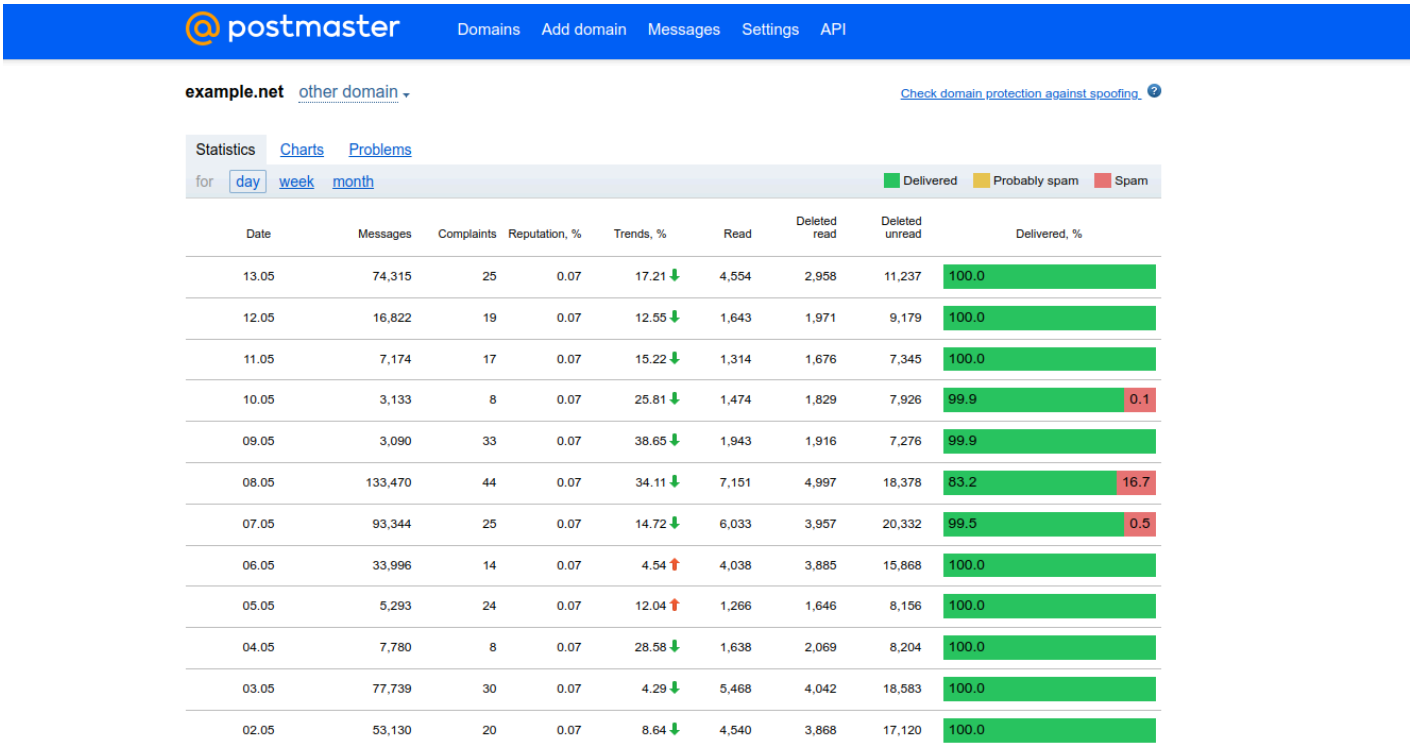


Setting up FBL forwarding

Postmaster Mail.ru supports FeedBack Loop technology. FeedBack Loop provides access to mailing statistics, grouping it by day, week and month.

This statistics allows you to track the following indicators:

- **Messages — number of messages sent;**
- **Complaints — number of clicks on the “Spam” button;**
- **Reputation, % - average percentage of complaints over 30 days. It is calculated as the ratio of the number of complaints over 30 days to the total number of submissions during this period of time and is expressed as a percentage;**
- **Trends, % - change in the percentage of complaints for 7 days in relation to the percentage for 30 days;**
- **Read — number of open letters;**
- **Deleted read — number of messages deleted after reading;**
- **Deleted unread — number of messages deleted before being read;**
- **Delivered, % — percentage of the following types of messages:**
  - **Delivered — the number of emails delivered to the Inbox;**
  - **Probably spam — the number of emails delivered to the Spam folder;**
  - **Definitly spam — the number of messages that were not accepted by Mail.ru servers. In response to such emails, you always receive a messages with the error “550 spam message discarded”.**



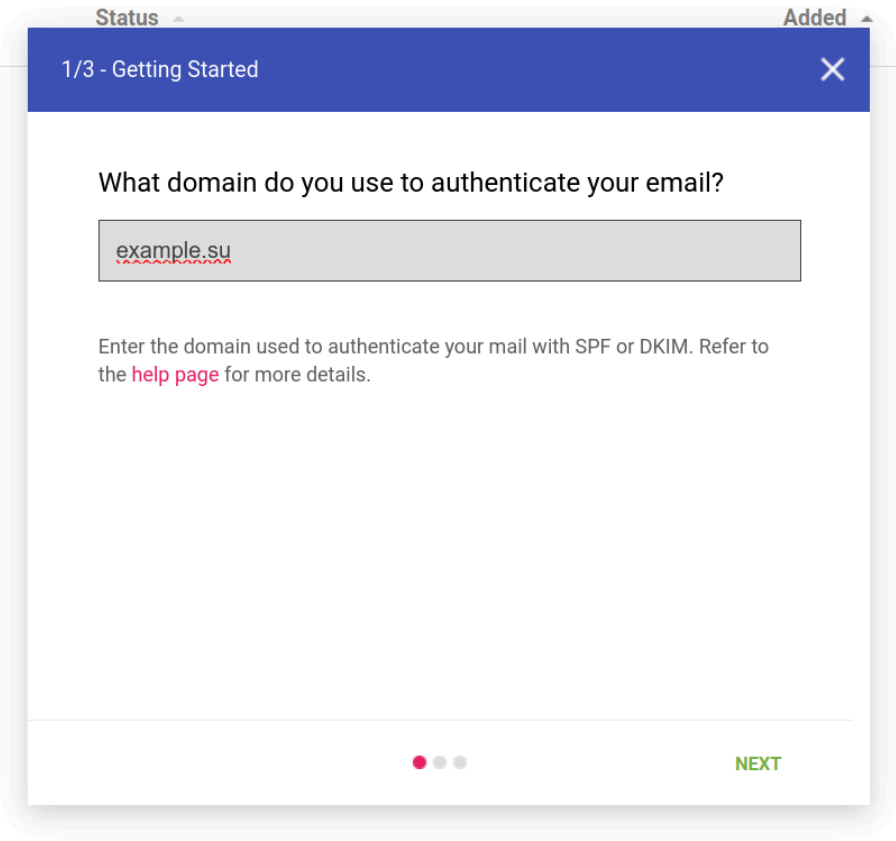
On the settings page for all domains, enter the FBL address. It will receive information about which recipients clicked the “Spam” button after receiving the message. After each mailing, Mail.ru Postmaster will display statistics on the number of emails sent and their status: Delivered, Probably spam, Definitely spam.

How to set up Postmaster Tools by Gmail

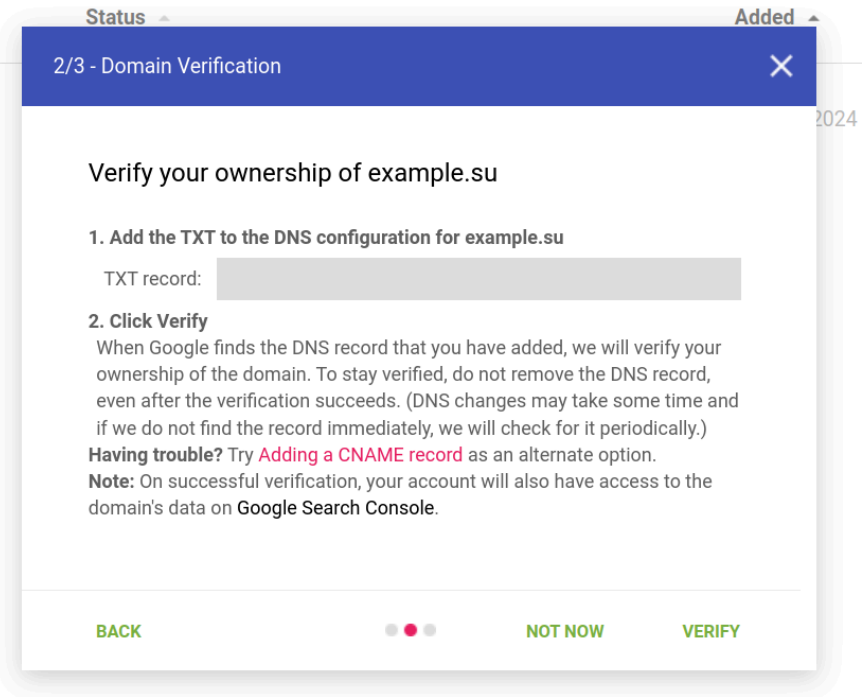
Step 1. Sign in to [Postmaster Tools] using your Google account.

Step 2. In the lower right corner, click the “Add” button.

Step 3. Enter your authentication domain (DKIM or SPF) and click “Next”:



Step 4. Confirm domain ownership. To do this, access your DNS record and add the TXT record provided by Google. Once complete, return to Google Postmaster and select “Verify.” If the entry is not found, check that the TXT was added correctly and try again. If you have trouble adding a TXT record, Google offers a second verification method that involves adding a “CNAME” record:

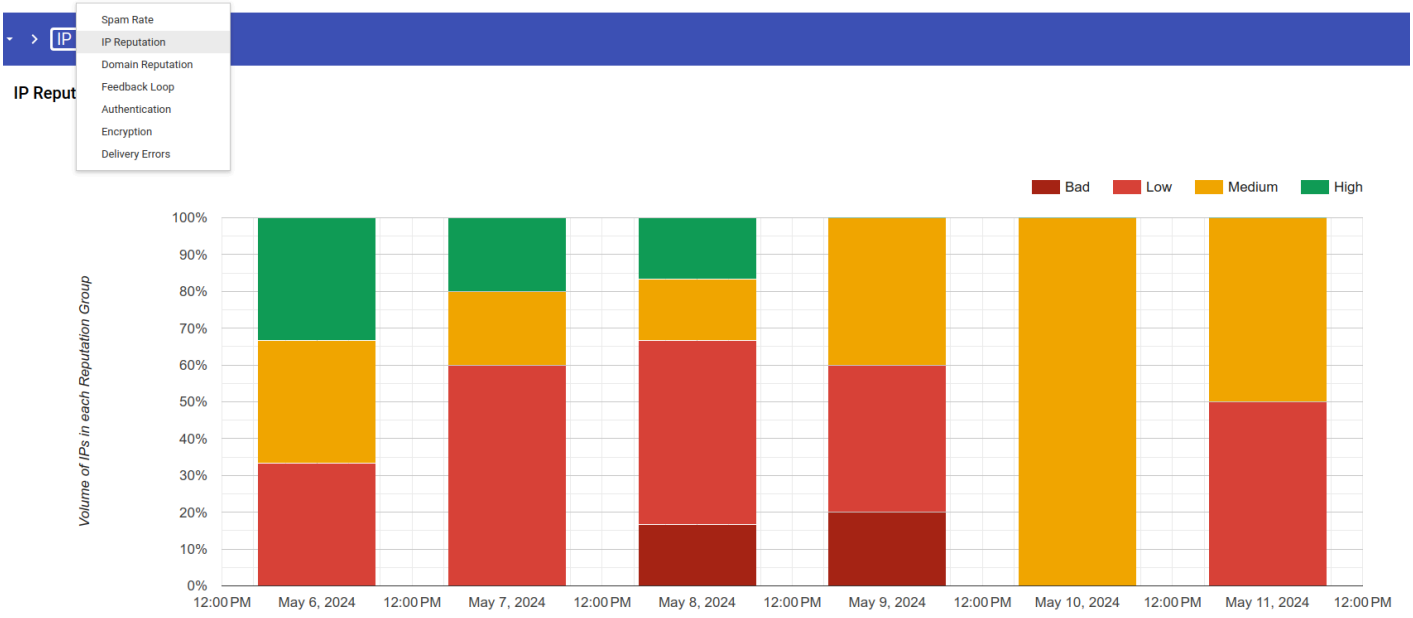


After confirming the rights to the domain, it will be displayed in the Postmaster Tools interface with the status “Verified”:

Domain	Status	Added
example.net	Verified	Aug 25, 2017
example.com	Verified	Aug 25, 2017
example.org	Verified	Dec 29, 2020
example.pro	Verified	Mar 2, 2023
example.xyz	Verified	Mar 14, 2023
example.io	Verified	Sep 25, 2023
example.su	Verified	Feb 26, 2024

Gmail Postmaster allows you to collect information about your mailings. The following data is tracked:

- **percentage of getting into spam;**
- **IP and domain reputation;**
- **information about authentication (passing DKIM, SPF and DMARC checks);**
- **information about traffic encryption (TLS share);**
- **information about delivery failures.**



All reports in Gmail Postmaster are generated in the form of graphs and tables with a selection of data for a period of 7 to 120 days. Information in reports may be missing if there is a small number of sends to Gmail mailboxes. Data will begin to be displayed when there is a sufficient amount of data (a hundred or more records) about daily traffic.

Setting up FBL forwarding

To set up FBL, the sender must add a new Feedback-ID header, consisting of parameters that uniquely identify the mailing. Such parameters are called identifiers. IDs whose messages often end up in spam or have delivery problems are displayed in the FBL panel in Postmaster Tools.

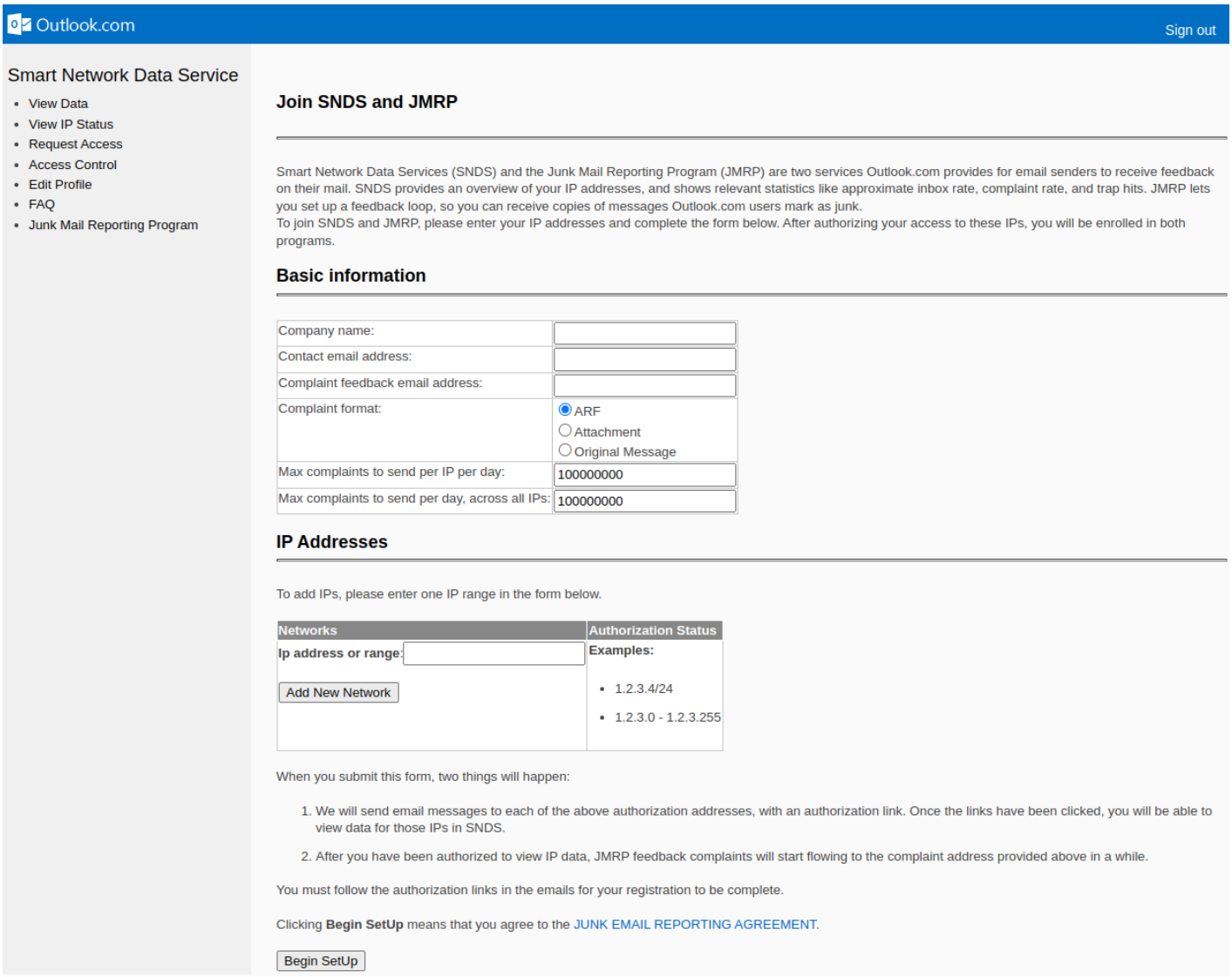
The header format is `Feedback-ID: a:b:c:SenderId`**,**

where `Feedback-ID:` **is the name of the header to add, and** `a, b, c:` **are optional fields with which the sender can specify up to three identifiers (for example, with the name of a company or client).** `SenderId` **– mandatory unique identifier selected by the sender (5–15 characters). It must be the same in all emails.**

How to set up Outlook.com Postmaster

Step 1. Sign in to your Microsoft account and go to service.

Step 2. Complete and submit the form on the [Junk Mail Reporting Program] page. Here you need to enter all IP addresses for mailings:



Step 3. After entering the IP address, the service asks you to select mailboxes to which, after submitting the form, you will receive a message with a confirmation link. You will need to open this link.

After all requirements are met, notifications about complaints about mailings will begin to arrive at the specified mailbox.

[Outlook.com] Postmaster Services allows the user to:

- **Receive reports on spam incidents reported by [Outlook.com] users**
- **recognize potential problems with email distribution methods and content**
- **improve sender reputation by removing unwanted subscribers from lists**
- **provide general information about how users rate the email they receive and the reliability of a range of sender IP addresses based on [Outlook.com] system metrics**
- **see how your email is rated by spam filters**
- **track the number of users complaining about your email**

How to set up Yahoo FBL

Yahoo provides a tool for senders to unsubscribe users who complain about spam - Yahoo Complaint Feedback Loop Service (YFBL).

Step 1. For YFBL to work, you will need a mailbox like postmaster@(your_domain). Create it on the service where your domain mail is configured.

Step 2. Follow the link and fill out the required fields:

- **Company name — name of the organization;**
- **Contact Name — name of the contact person;**
- **Contact Email — email of the contact person;**
- **Reporting Email - address for receiving FBL reports;**
- **Selector — specify “*”;**
- **Request type — request type: adding, changing or deleting. Specify “Add”;**
- **Domain — the domain for which you are setting up FBL. After filling out the required fields, click the “Get Verification Code” button.**

Step 3. Find the email from the service in the created mailbox and copy the confirmation code. Paste the copied code into the appropriate field on the YFBL service page and click “Create Request”.

Please note that it takes up to 48 hours to confirm your settings and enable reporting. After successful setup, the service will work as follows: if the recipient clicks the “Spam” button in a letter from your mailing list, an FBL report will be sent to your account, and the recipient will be automatically unsubscribed from mailing lists.

How email tracking works

The Altcraft platform tracks recipient engagement with email messages using invisible tracking pixels. Based on pixel loading data, the platform generates metrics for opens, glances, skims, and reads.

TIP: Tracking Pixel Placement

Whenever possible, place tracking pixels closer to the beginning of the email. This ensures that the email client loads them sooner and reduces the risk that Gmail clipping will prevent an open from being recorded. However, the pixels must be carefully embedded in the HTML template so that they do not affect the email layout.

Tracking mechanism

Two invisible 4×4 pixels are automatically added to each email:

- **Open pixel — placed at the end of the email. When this pixel loads, the platform records an email open event. If the pixel does not load (for example, images are blocked), the open may be registered on the first click on a link within the email.**
- **Read pixel — placed after the main content. When this pixel is requested, the server keeps the HTTP connection open for up to 12 seconds and measures how long the mail client holds the connection.**

Based on the HTTP connection duration, the event type is determined:

Event type	Name	Duration
Glance	**Glanced**	**0–4 seconds**
Skim	**Skimmed**	**4–8 seconds**
Read	**Read**	**8+ seconds**

If the connection is closed before 12 seconds (for example, the recipient closed the email), the actual viewing time is recorded and the corresponding event type is registered. The maximum wait time is 12 seconds: if the email stays open longer, the server closes the connection and registers the event as a read.

Email client limitations

Not all email clients load tracking pixels correctly. Below are the main scenarios where metrics may be incomplete or missing.

Remote image blocking

Some email clients block remote content by default, including tracking pixels:

- **Outlook — remote content blocking is enabled by default. The user must manually click “Download Images” for each email.**
- **Thunderbird — image blocking is enabled by default.**
- **Yahoo Mail — may block remote content.**

Consequence: pixels do not load, opens and reads are not recorded.

Solution: add the platform’s tracking domain to the image blocking exceptions.

Image proxy servers

Some email services do not load images directly from the sender’s server, but through their own proxy:

- **Gmail Image Proxy — all images in emails are loaded through Gmail’s proxy. The pixel is opened, but the HTTP connection is closed immediately after loading, so the viewing time is zero.**
- **Apple Mail Privacy Protection (MPP) — images are loaded through Apple’s proxy with randomized timing. The open is recorded, but the viewing time does not reflect the user’s actual behavior.**
- **Corporate proxies (Proofpoint, Mimecast, etc.) — similar to Gmail’s proxy, load pixels through an intermediate server.**

Consequence: the open is recorded, but glanced/skimmed/read metrics do not work — the proxy does not keep the HTTP connection open, so `timeload ≈ 0` and only a glanced event with zero is registered.

Gmail prefetching

Gmail may pre-load images from emails even if the recipient has not opened the email — for example, when displaying in the preview pane or when indexing content.

Consequence: a false open is recorded, even though the email was not actually viewed.

Gmail clipping

Gmail may truncate long emails, removing part of the content, including tracking pixels, if they are located beyond the truncated area.

Consequence: the pixel does not load, the open is not recorded.

Why glanced/skimmed/read may not be recorded

The glanced/skimmed/read metrics are based on measuring the HTTP connection duration when loading the read pixel. This mechanism works only with a direct connection between the mail client and the tracking server.

When the request goes through a proxy server:

1. **The proxy loads the pixel on behalf of the client**
2. **The proxy saves the image and delivers it to the client**
3. **The HTTP connection between the proxy and the tracking server closes immediately after loading**
4. **The server records `timeload ≈ 0` and registers a glanced event**

Thus, glanced/skimmed/read metrics work only with a direct connection without a proxy. Through a proxy, the only reliable signal is the open event (request received or not).

Recommendations

How to check if a client blocks images

Send a test email to your own address and open it in the problematic mail client. If the report does not record an open, the client blocks remote images or uses a proxy.

How to add the tracking domain to exceptions

Find out from the platform administrator which domain is used for tracking. Add this domain to your mail client's exceptions list:

- **Outlook: allow automatic image loading or add the domain to trusted senders (the exact path depends on your Outlook version).**
- **Thunderbird: Settings — Privacy — Uncheck "Warn me when web sites try to load images".**
- **Apple Mail: Settings — Privacy — Uncheck "Load remote images through proxy".**

Which metrics to rely on when working through proxies

If most of your audience uses email services with image proxying (Gmail, Apple Mail), the glanced/skimmed/read metrics will be incomplete. In this case, it is recommended to focus on:

- **Opens — a "yes/no" metric that works even through proxies.**
- **Clicks (CTR) — do not depend on tracking pixels, reflect real engagement with content.**
- **CTOR — the percentage of clicks from opens, shows engagement among those who opened the email.**

Quick Start

How to send your first email mailing

This article shows the basic workflow in Altcraft Platform. The goal is to send your first email and understand how the platform works.

Sequence of steps: Profile Database — Resource — Profile — Template — Mailing — Reports

TIP: Domain configuration

If you have not yet configured a domain for use with the platform, first read the article Configuring your own from-domain. Without proper DNS records, emails may not be delivered or may end up in spam.

Step 1. Create a profile database

Profile Database is where customer data is stored. In the documentation, you may see terms like “customer”, “subscriber”, or “user”, but in all cases this refers to a profile.

1. Go to **Data — Profile Databases** and click: ![img](/img/attachments/quick-start/en/creation-button.png)	![img](/img/attachments/quick-start/en/database-navigation.png)

---

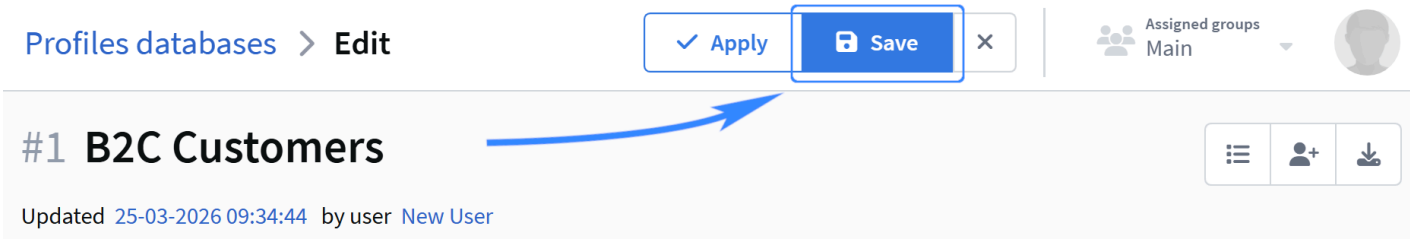
2. Enter the **Profile Database Name**	![img](/img/attachments/quick-start/en/name.png)

![img](/img/attachments/quick-start/en/subscr-manger.png)

Subscription Manager Title — text on the subscription management page (for example, your company name).

---

4. Save the database



For more details on working with profile databases, see this article.

Step 2. Create a resource

Resource is the source of communications: a website, service, brand, or project. Subscriptions are collected and messages are sent through a resource. You cannot create a mailing without a resource.

A resource must be linked to a profile database.

INFO

For email sending, DKIM must be configured for the sender domain. It is not provided automatically. Contact technical support chat to enable it.

1. Go to the **Resources** section	![img](/img/attachments/quick-start/en/resource-creation.png)

2. Select a subscription channel. **For your first mailing, we recommend using Email**	![img](/img/attachments/quick-start/en/resource-channel.png)

3. Enter the **Resource Name**

---

4. Select the previously created Profile Database

---

5. Enter the Website URL and Tracking Domain

```
</td>
  <td width="55%">
  ![img](/img/attachments/quick-start/en/resource-main-settings.png) </img>
  </td>
</tr>
```

Website / form / application URL — the address where users submit their contacts. You can specify your company’s main website

Tracking Domain — technical domain for tracking link clicks from emails. If the field is already filled, leave the value unchanged

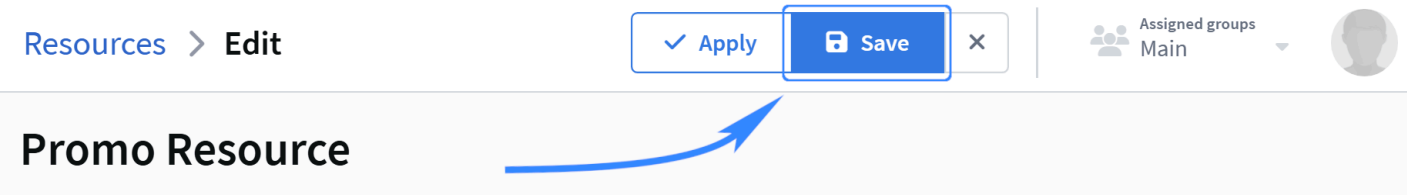
6. Enter the **Sender Name** and **Sender Email** 7. Enter the List ID	![img](/img/attachments/quick-start/en/resource-name-email-and-listid.png)

- **Sender Name** — the name shown in emails - **Sender Email** — the sender address - **List ID** — identifier of the subscription list within the resource. Specify any clear value you understand, you can use the sender email in format `example.domain.com`

INFO

Sender Name and Sender Email are set at the resource level. When an email is sent, priority is determined as follows: if the message template specifies its own values, they override the resource settings. If the mailing has an explicit override for the template, it takes the highest priority. Priority order: mailing, template, resource.

8. Save the resource



For more details on working with resources, see this article.

Step 3. Add a profile

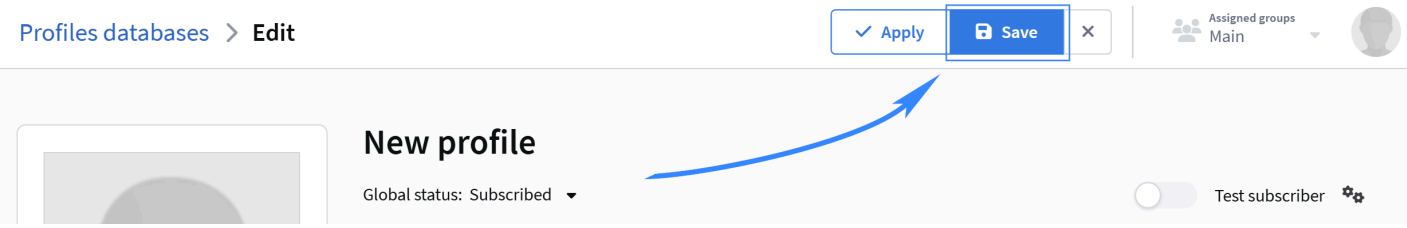
Create a profile to send the email to. To do this, go back to Data — Profile Databases.

1. Click Import customer profiles — Create profile manually

![img](/img/attachments/quick-start/en/create-profile.png)

2. Enter the profile name and click **Add subscription**	![img](/img/attachments/quick-start/en/add-subs.png)
3. Select the created resource and the Email channel	![img](/img/attachments/quick-start/en/add-subscriptions.png)
4. Enter the email address to send the message to	![img](/img/attachments/quick-start/en/email.png)

5. Save the profile



For more details on profiles, see this article.

Step 4. Create a message template

Template is the email content.

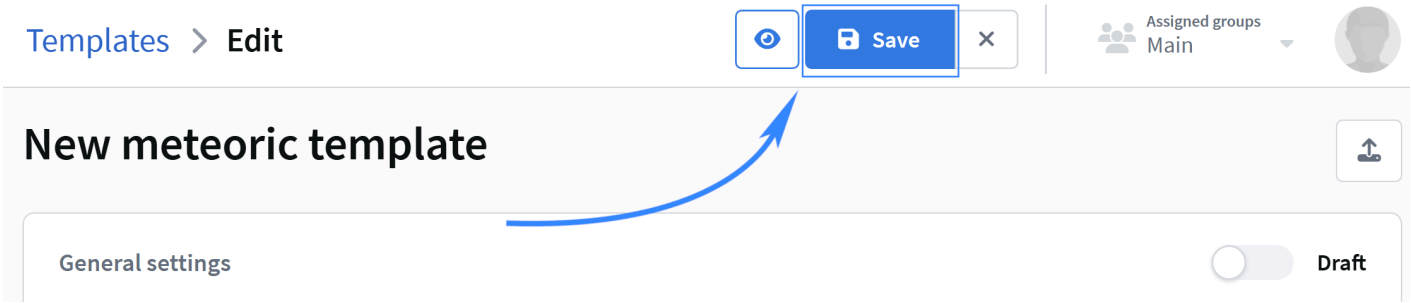
1. Go to **Templates** and click: ![img](/img/attachments/quick-start/en/creation-button.png)	![img](/img/attachments/quick-start/en/temp-nav.png)

2. Select creation method Editor and channel Email

![img](/img/attachments/quick-start/en/editor.png)	![img](/img/attachments/quick-start/en/edit-email.png)

3. Open the Code Editor:

and save the template:



In this case, the default (placeholder) message template will be used. To configure or import your own templates, we recommend reading the Message Templates section.

Step 5. Create a mailing

1. Go to **Mailings — Broadcasts** and click: ![img](/img/attachments/quick-start/en/creation-button.png)	![img](/img/attachments/quick-start/en/mailing-navigation.png)

2. Enter the **Mailing Name** and select the **Email** channel	![img](/img/attachments/quick-start/en/mailing-name-and-channel.png)

---

3. Configure recipients

Go to the Subscribers tab and specify:

- **Resource — select the created resource**
- **Subscriber source — leave Database**
- **Profile Database — select the database created in step 1**

![img](/img/attachments/quick-start/en/mailing-recipients.png)

---

4. Select content

In the Templates tab, select the created Message Template.

![img](/img/attachments/quick-start/en/mailing-content.png)

---

5. Set the sending time

Go to the Schedule tab:

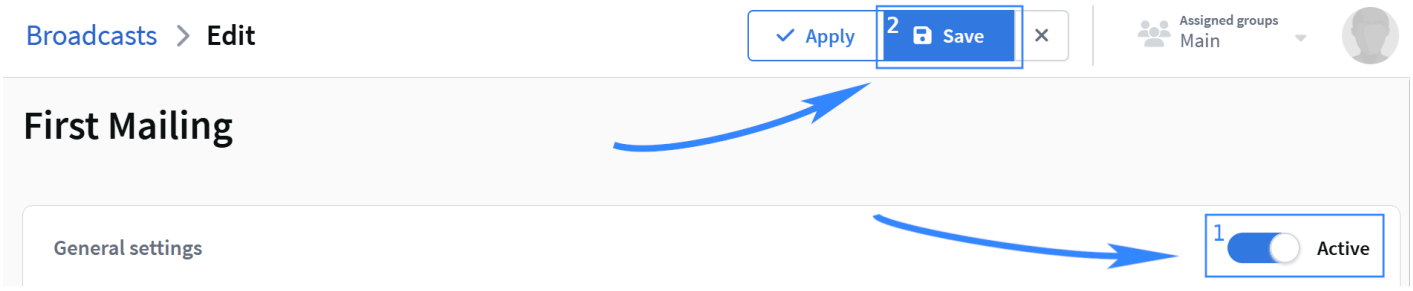
- **You can set a specific start time**
- **To send immediately after saving, enable Start mailing now (in addition to schedule)**

![img](/img/attachments/quick-start/en/mailing-schedule.png)

---

6. Save and start

Activate the mailing and click Save. After saving, the mailing will start sending according to your settings.



The email will be sent to the created test profile.

INFO

For more detailed information on sending email mailings, you can refer to this article.

Step 6. Check the result

You can check the results in two ways: in the analytics section or in the profile card.

Method 1. In the “Insights” section

1. Select the completed mailing and click

![img](/img/attachments/quick-start/en/reports-navigation.png)

2. Review the statistics

![img](/img/attachments/quick-start/en/reports-list.png)

Method 2. In the profile

1. Go to **Data — Profile Databases**	![img](/img/attachments/quick-start/en/database-navigation.png)

2. Open the created profile card and go to the History tab — it contains all information about the profile:

![img](/img/attachments/quick-start/en/profile-history.png)

INFO

For more detailed information on reports, you can refer to this article.

First Email Mailing

Email remains one of the most effective methods of marketing communication with customers. You can send subscribers more content than through other channels: images, links to various products and services, interactive elements that attract attention and increase customer engagement.

This guide will help you set up an Email mailing from scratch: from configuring the sending domain to viewing mailing statistics.

> INFO
>
> **Altcraft Platform tracks subscriber actions in Email messages, such as opens, reads, and link clicks.**

Step 1: Configure Email Sender

The Altcraft Platform sends email using the SMTP protocol. For this purpose, we developed our own transport module AKMTA (Altcraft Mail Transfer Agent), which optimizes email delivery — even when the mailing audience reaches tens of millions of recipients.

Sending and tracking domains

To send email messages, you will need a sending domain. The domain will be used in the sender address (from email), which recipients see in their mail client.

To track recipient actions in emails, you will also need a tracking domain. This can be a subdomain of your main domain or a separate domain. Link clicks in the message are redirected to the tracking domain — this is how the platform tracks clicks. Images are also loaded through it for registering email opens.

Once you have acquired a domain, you need to add configuration records on the side of the domain name registrar that provided it, to ensure efficient delivery.

Custom domain settings are described in a separate article.

For cloud clients:

- **SPF — authorizes sending emails from your domain**
- **DKIM — digital signature of emails**
- **DMARC — policy for handling emails that fail SPF/DKIM**
- **CNAME tracking — subdomain for tracking clicks from emails**

For On-premises clients:

- **A-record — specifies the IP address of the mail server (AKMTA)**
- **MX-record — directs mail to the sender's server**
- **PTR-record — reverse DNS for the sender's IP address**
- **SPF — authorizes sending emails from your domain**
- **DKIM — digital signature of emails**
- **DMARC — policy for handling emails that fail SPF/DKIM**
- **CNAME tracking — subdomain for tracking clicks from emails**

Installing AKMTA

For cloud clients:

When using the cloud service, we provide our own sending infrastructure with all necessary settings. The only thing you need to start sending is your configured domain (see the previous section).

For On-premises clients:

When installing the Altcraft Platform platform on your equipment, you can configure one or several sending servers with AKMTA transport modules. To do this, you need to configure the following in the platform's Admin Panel:

- **Nodes — network sending nodes. The module uses the IP addresses available to you for sending.**
- **Senders — configuration records that combine sender identification parameters, sending speed, and sender behavior settings in case of sending errors.**
- **Virtual senders — rules for sending from multiple senders. Can be configured in both the Admin Panel and the User Interface.**

Step 2: Run a test check

Before sending at scale, send a test email to a single mailbox and check the results.

Check the headers of the received email:

- `dkim=pass` **— DKIM signature verified**
- `spf=pass` **— SPF record confirms the sender**
- `dmarc=pass` **— DMARC policy passed**

If you are using Microsoft 365 or Outlook, also check for `compauth=pass`, `SCL:0`, `BCL:0`.

TIP: Online testers

For a comprehensive antispam score and content check, use online testers such as Mail-Tester.com. Antispam filter headers vary depending on the mail provider and are not always available for manual inspection.

If any check fails, resolve the issue before launching the mailing. For detailed diagnostic instructions, see the Troubleshooting section of the from-domain configuration article.

```
X-Spam-Score: 2.331
X-Spam-Level: **
X-Spam-Status: No, score=2.331 required=6.6 tests=[BAYES_50=0.8,
 DKIM_SIGNED=0.1, DKIM_VALID=-0.1, DKIM_VALID_AU=-0.1, DKIM_VALID_EF=-0.1,
 DMARC_PASS_REJECT=-1.2, HTML_MESSAGE=0.001, HTML_TITLE_SUBJ_DIFF=2.036,
 MIME_HTML_ONLY=0.1, RDNS_NONE=0.793, SPF_HELO_NONE=0.001, SPF_PASS=-0.001,
 URIBL_BLOCKED=0.001] autolearn=no autolearn_force=no
Authentication-Results: mail.altcraft.com (amavis); dkim=pass (1024-bit key)
```

Step 3: Subscribe profiles to the Email channel

Profiles Databases

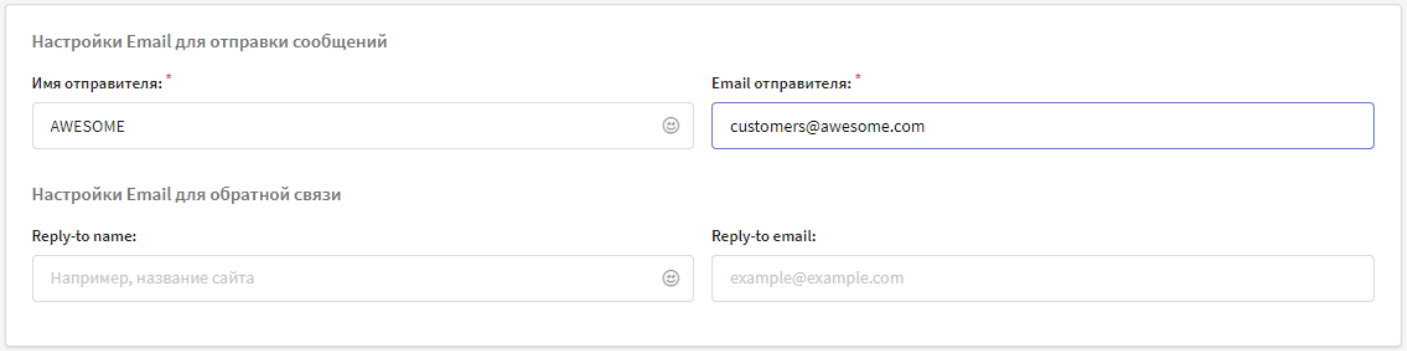
Altcraft Platform mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

Each customer profile has a system field Email: it is used to identify profiles and in cases where your customers only have one email address.

TIP

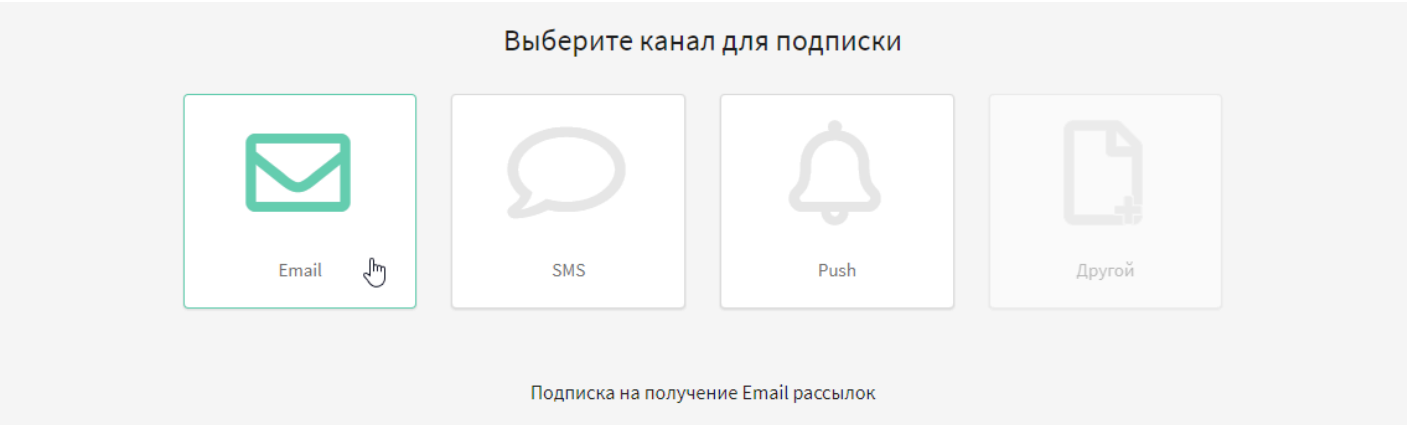
If you use different contact details for the same customer — or send content of different types — set up multiple subscription resources.

When creating a database, the Sender Email is a required parameter. Sending will be performed on its behalf without using resources. Enter the sending address in the corresponding field. Also specify the Sender Name, and if needed — the reply recipient details (Reply-to name and Reply-to email):

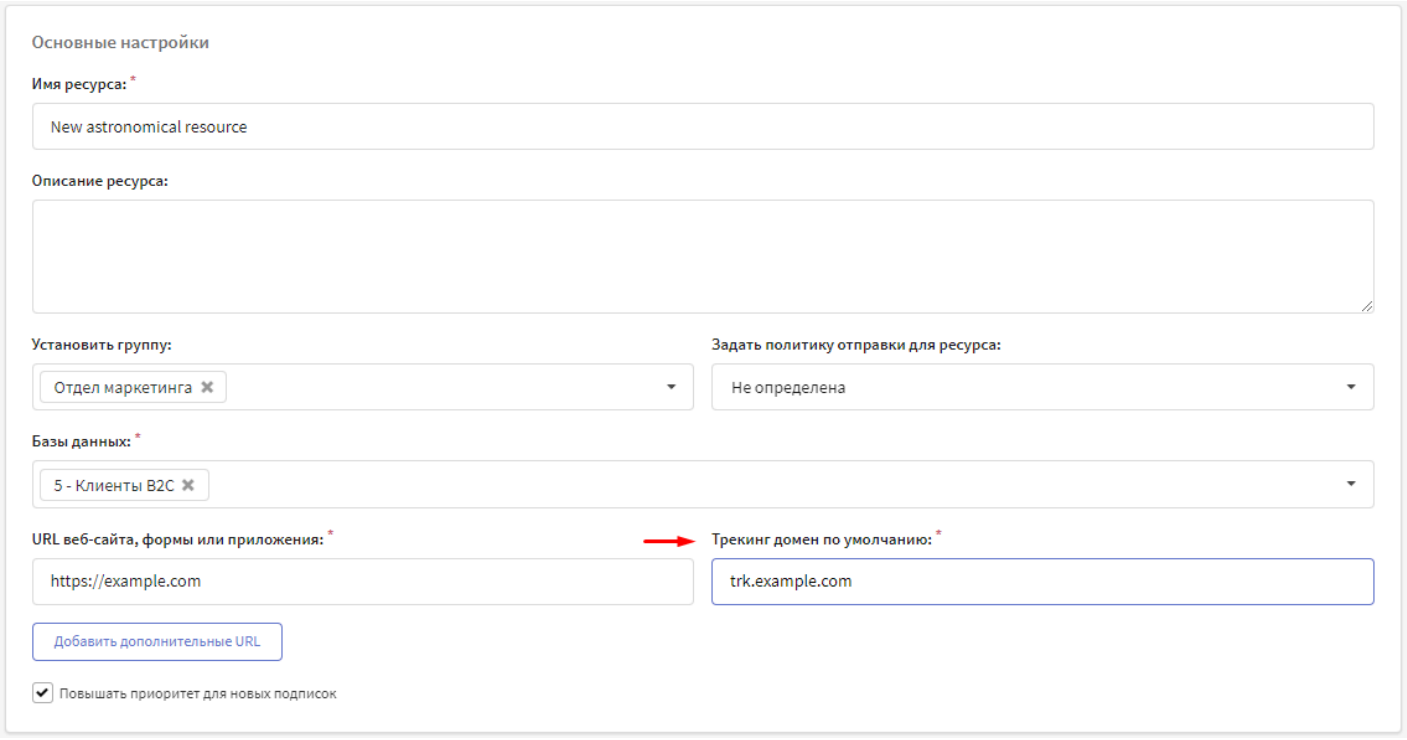


Subscription resources

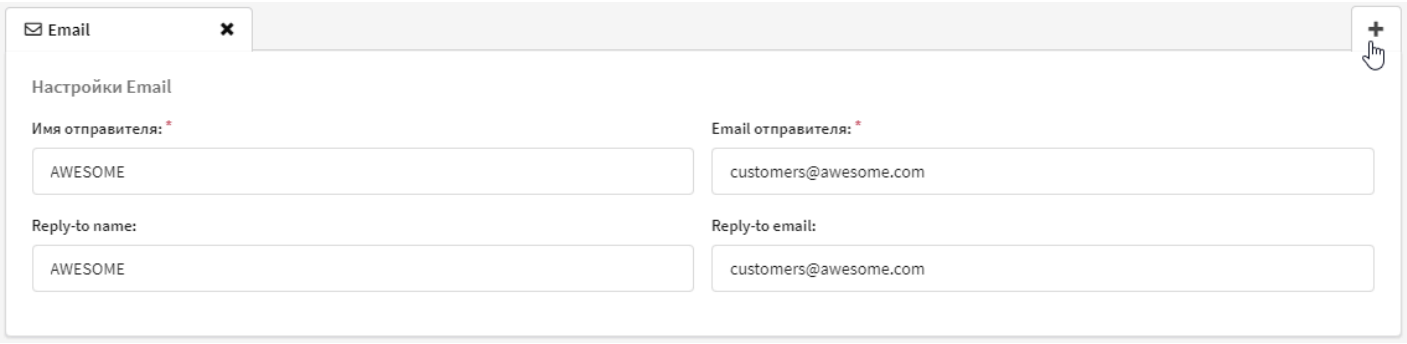
Each client profile can be subscribed to one or more resources. A resource can be a separate marketing direction, a website, an application, or a type of content. Select the corresponding item from the main menu, create a new resource or select an existing one. Add the Email channel to an existing resource or select it as the primary channel for a new one:



In the general settings, add the required databases. If needed, assign a sending policy. Specify the Default tracking domain configured earlier:



In the Email channel settings of the resource, specify the Sender Name and Sender Email, as well as — if needed — the name and address for receiving replies. These values will be used by default if they are not overridden at the message template or mailing level. Priority order: mailing, template, resource.



Customer profiles and subscription to the resource

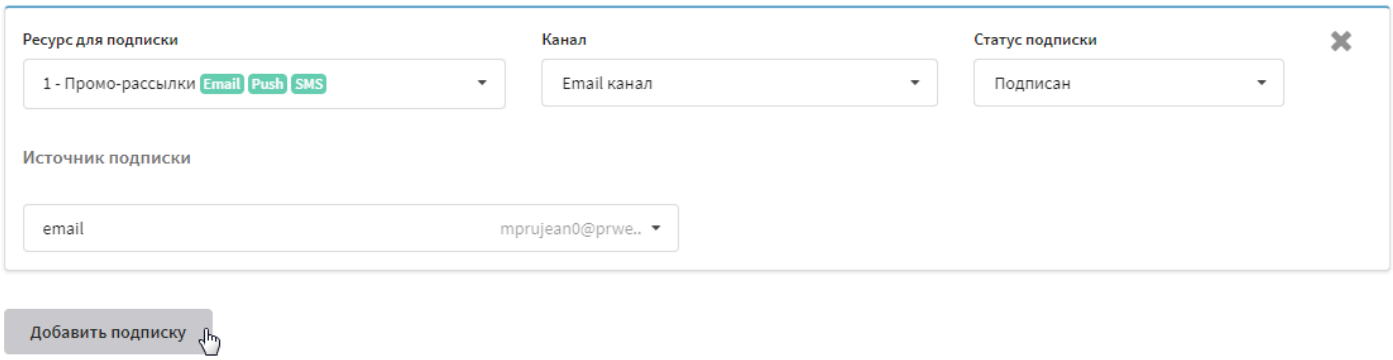
After configuring the database and subscription resource, you can import customer data into the platform. For each client, Altcraft Platform creates a single digital profile combining personal data, action history, and subscriptions to your resources.

You can upload customer data manually: one at a time, in a list, or from a spreadsheet file. Various import methods are described in this article.

You can also set up automatic import of client profiles:

- **on a schedule from your SQL database**
- **via integration with the platform API**

When importing data, subscribe clients to the previously prepared resource. Select Email as the communication channel and specify the column with subscribers' addresses:



TIP

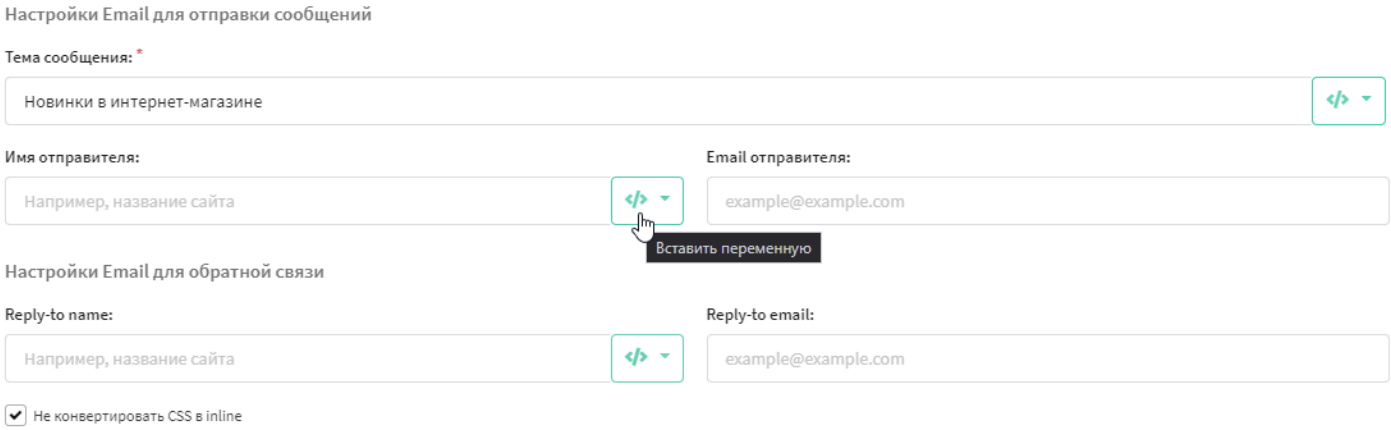
If you already have a profiles database, add subscriptions to the new resource. There are two options:

- **if customers have subscriptions to another resource with up-to-date email addresses — use bulk profile update.**
- **if you need to upload email addresses to the platform for subscription — use profile import in "update only" mode.**

Step 4: Prepare content for sending

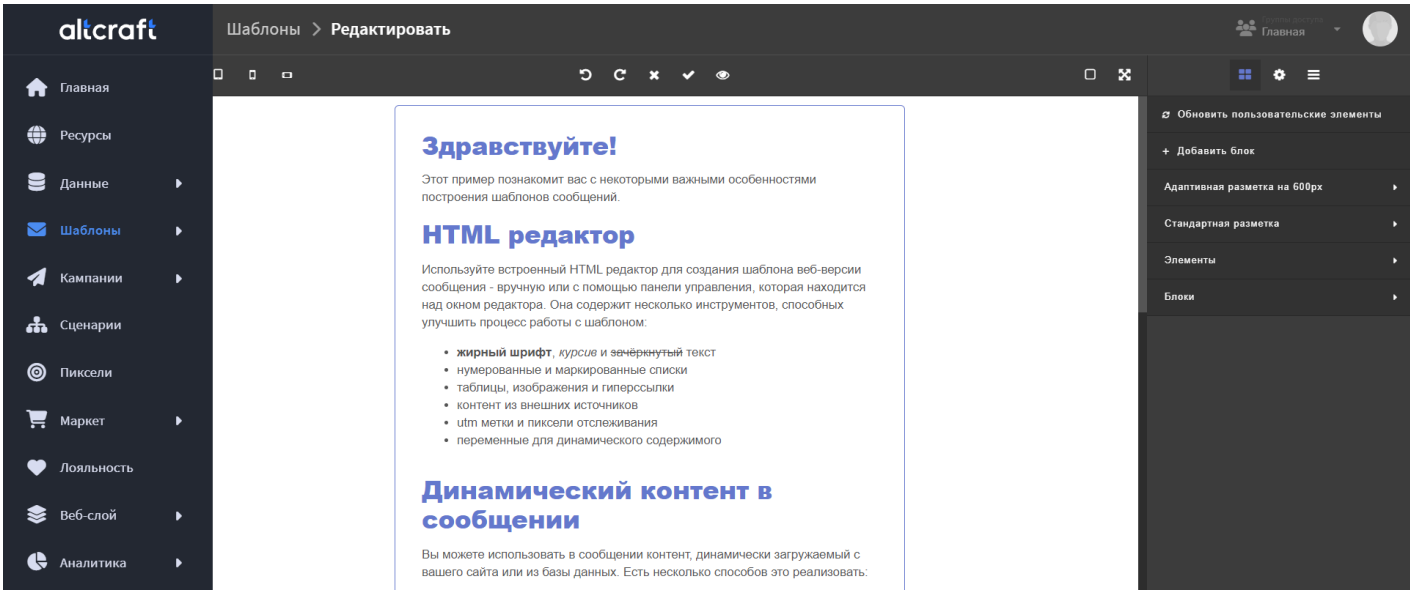
Content for your Email mailings is created in Message Templates. Templates contain content for all available channels. When creating a new template, select the Email channel or add it to an existing one.

For the Email version of the template, you can set a Subject — it will be displayed in the correspondence. You can also set the sender name and address — the template settings will override the settings of both the resource and the database.



Templates in Altcraft Platform offer many personalization options for content tailored to a specific recipient. This uses various types of variables, logical expressions, loops, and functions, as well as dynamic content from various external sources.

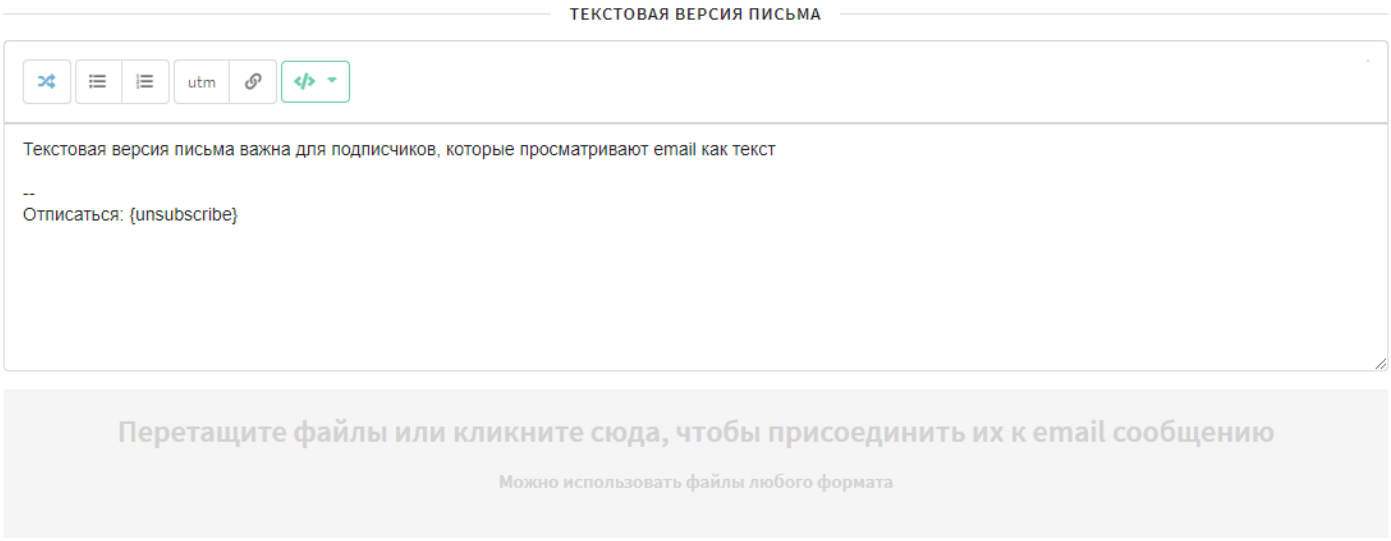
To create Email message templates, use the Code Editor or the Visual Editor:



Some of your subscribers may use a mail client where image loading is unavailable — for example, due to corporate policy. For such cases, the template includes a Text version.

CAUTION

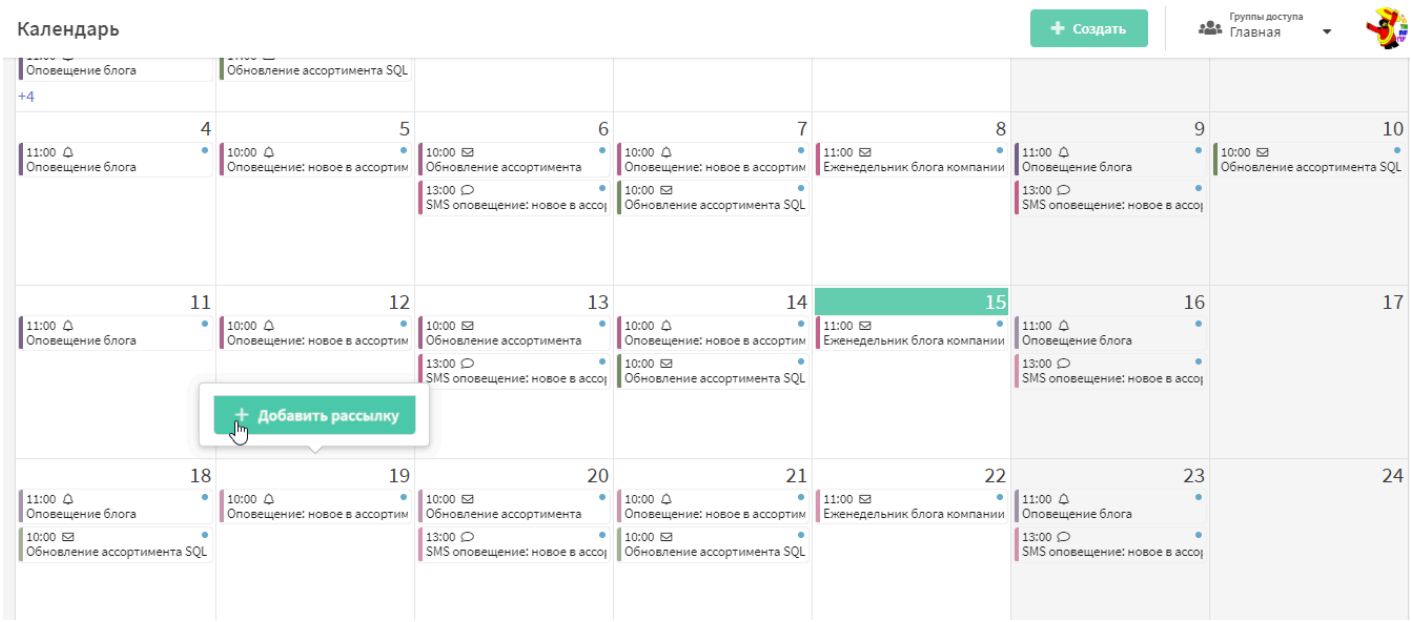
The text version is sent together with the HTML version of the email. If it is left empty — or does not match the HTML — this may negatively affect the sender's reputation.
Conversely, having a text version significantly increases trust metrics for the mailing.



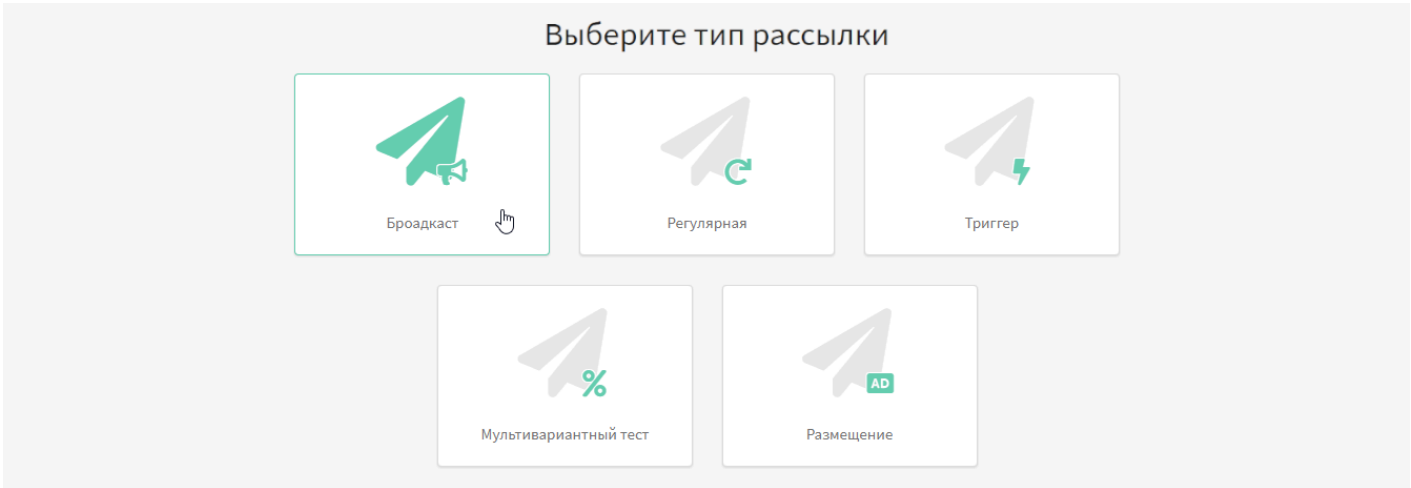
Below the text version editor is a form that allows you to attach files in any format to the message.

Step 5: Select the type of mailing

![img] (/img/attachments/22708692/164429.png)	**To send Email messages, use the Mailing tool.** **If you need to send a one-time mailing to an entire profiles database or a specific audience segment — a broadcast is the right choice.** **If the mailing will be periodic — choose a regular mailing.** **In the main menu, select Mailings and open the Calendar. It displays all mailings sent on a schedule.**



Decide on a date and create a mailing of the required type:



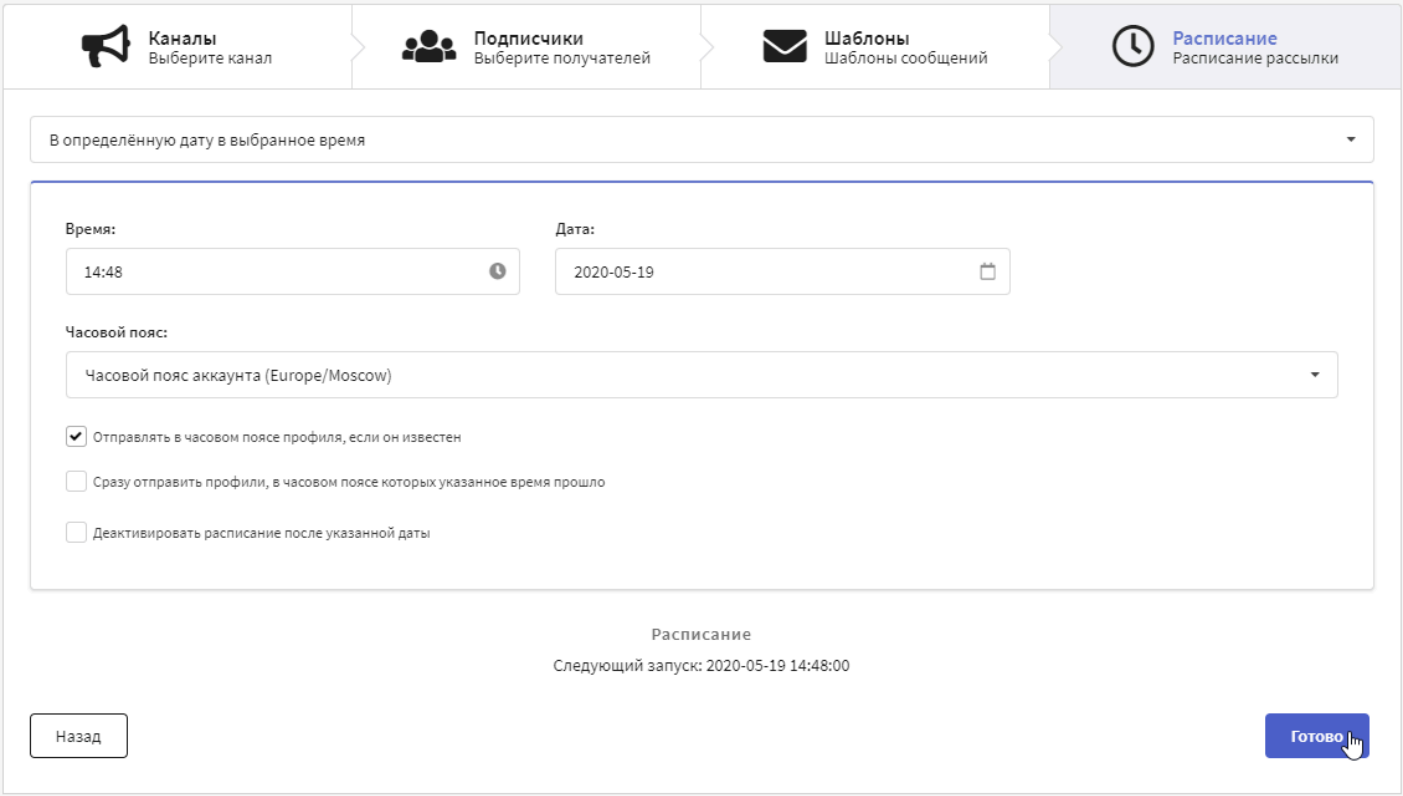
TIP

Detailed instructions for setting up a Broadcast can be found on this page.

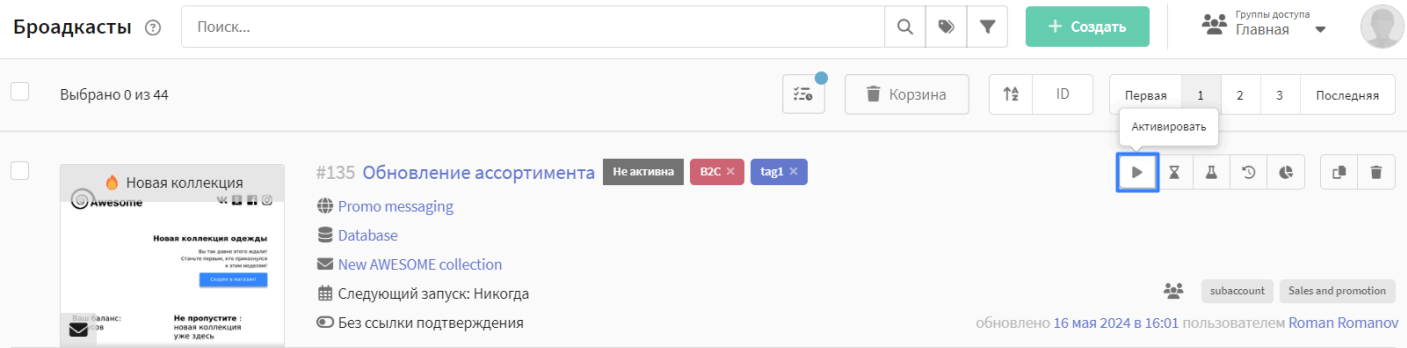
TIP

Detailed instructions for setting up a Regular mailing can be found on this page.

Once you have filled in the general mailing settings, select Email as the communication channel, choose the subscription resource configured earlier, and select the recipient audience. Add a message template to the mailing and set a schedule. For example, for a broadcast, the option "On a specific date at a selected time" works well:



When the mailing is configured, save it. It will appear in the list of mailings of the corresponding type. On the right side of each entry there is a quick actions menu. Use it to send a test — and if everything is successful, activate the mailing:



CAUTION

The active mailing will start sending emails as soon as it will be time to launch as scheduled. Messages are sent to the sender. You can also open the sender queue from the quick action menu. For more details on queue management, see Managing the Sender Queue.

Sending speed depends on the sender domain reputation.

Step 6: Evaluate mailing effectiveness

Mailing Log

The Mailing Log is a tool that provides a summary of the mailing execution progress. It records key events, mailing parameters, and statistics, allowing you to identify potential issues.

To open the mailing log, click the Journal button in the quick actions menu:

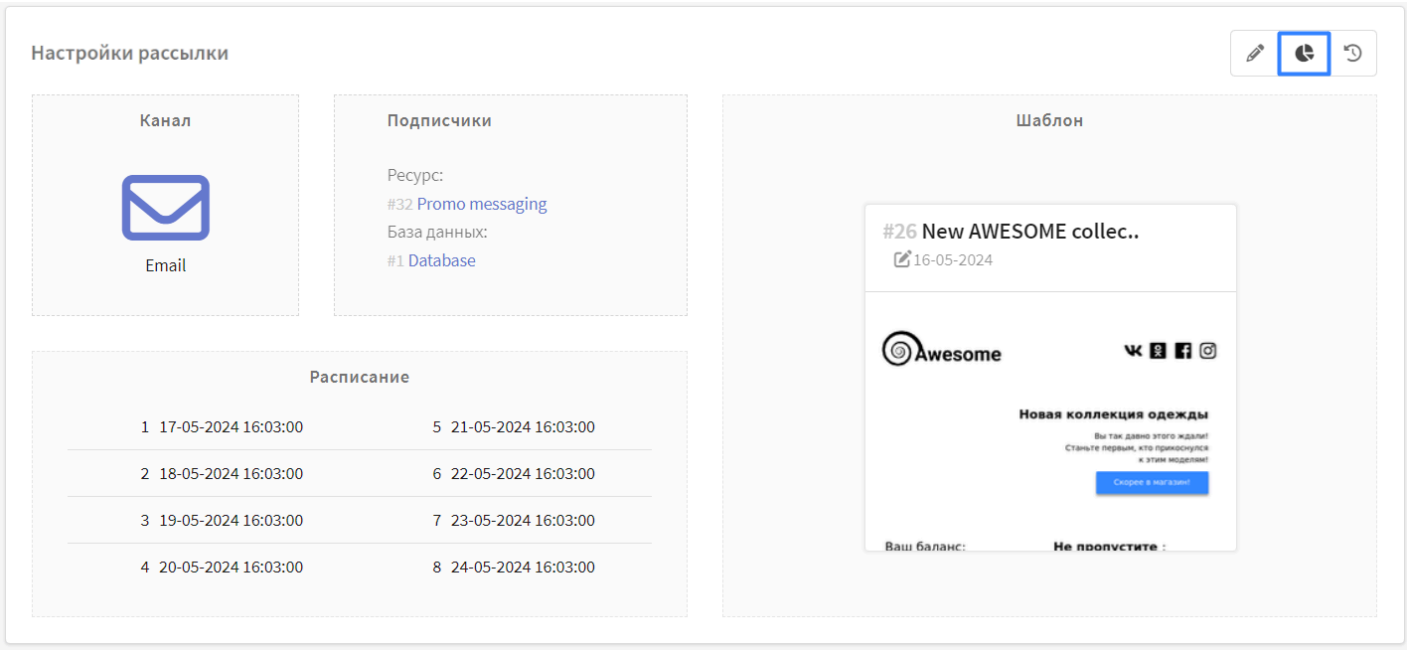
The mailing log contains the following information:

Type	Description
Mailing started	**Start time of the email mailing execution**
Limit	**Shows whether a sending limit is set**
Audience source	**Audience source:** **List — Database** **Segment — Segment**
Segment ID / Database ID	**Segment identifier / Database identifier**
Time retrieving segment	**Time spent loading segment data from the database (in ms)**
Number of profiles received from source	**Total number of profiles found in the specified segment/database**
Number of subscribers found	**Number of profiles in the specified segment/database subscribed to the selected resource**
Number of hard bounces	**Emails that could not be delivered (non-existent addresses, etc.)**
Number of subscribers who received email	**Actual number of successful sends**
Mailing finished	**Mailing completion time**
Total time spent	**Total duration of the entire mailing from start to completion**

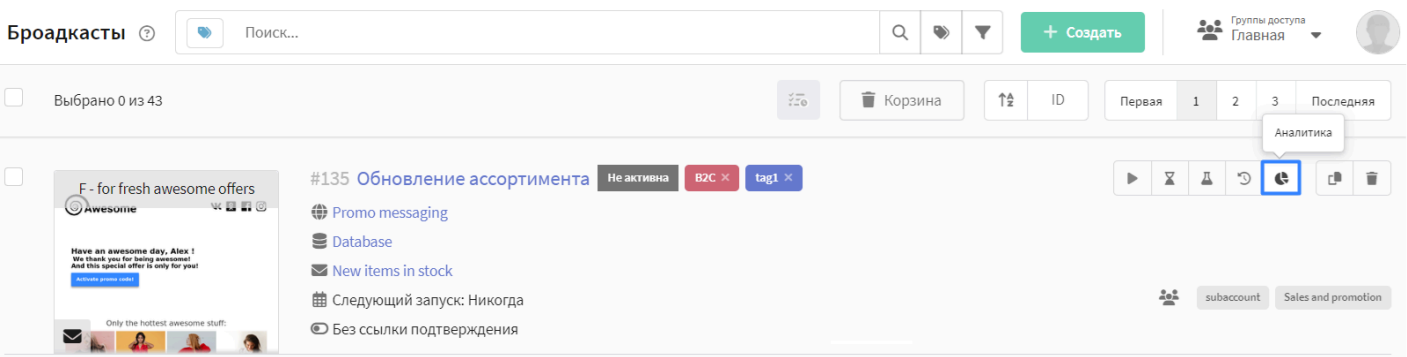
Reports and Analytics

To evaluate the results of a current mailing, you can view a channel report. The [Channel report] provides analytics for one or several mailings, for a selected communication channel.

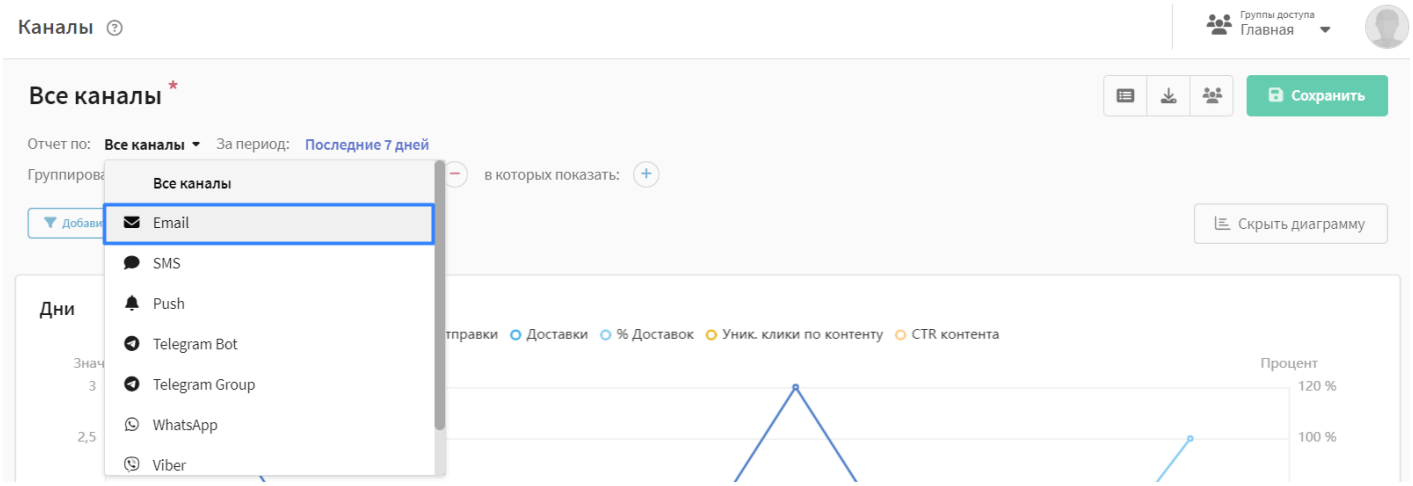
You can access the report for a specific mailing via the link in the mailing editor:



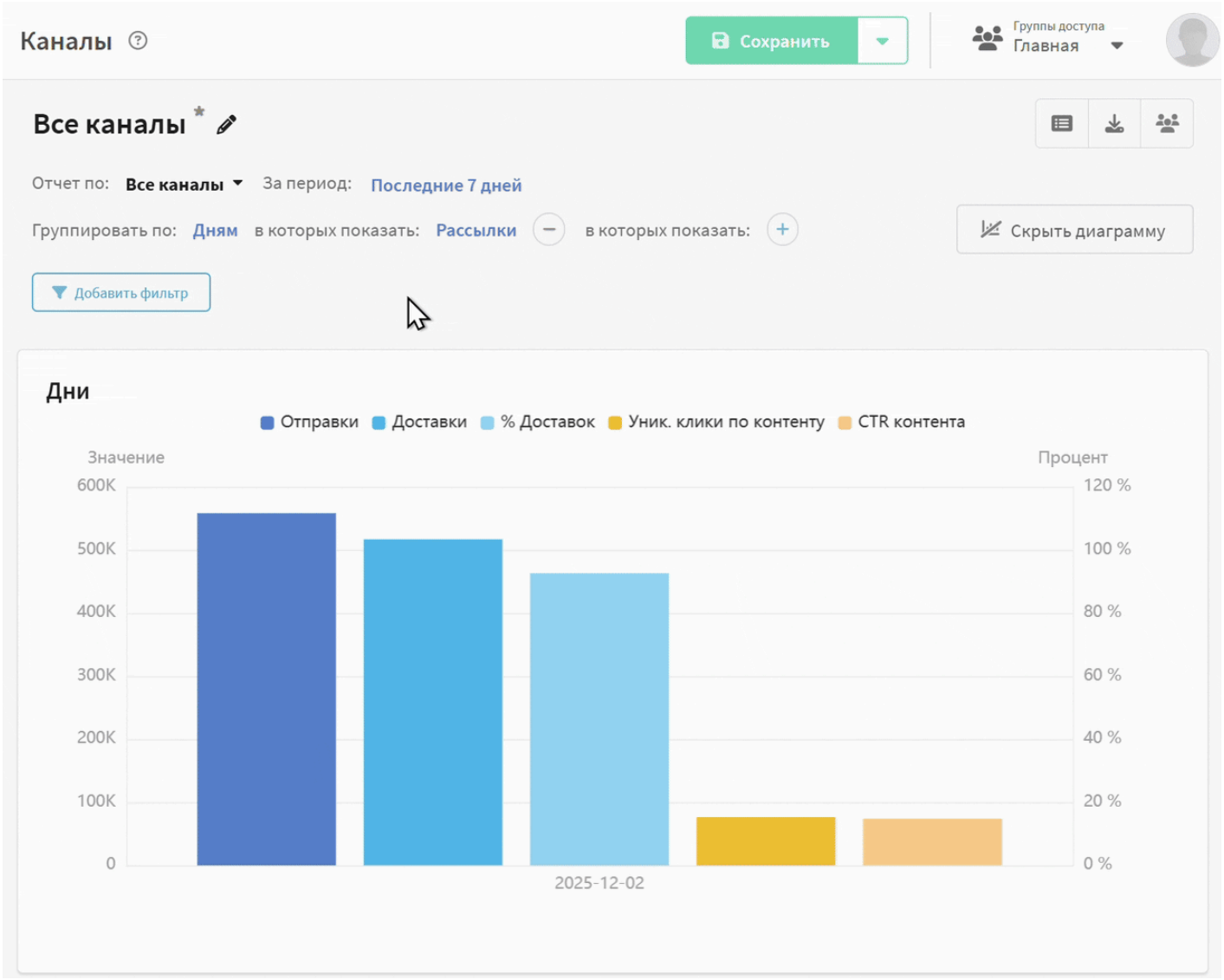
Or select the Analytics button from the quick actions panel:



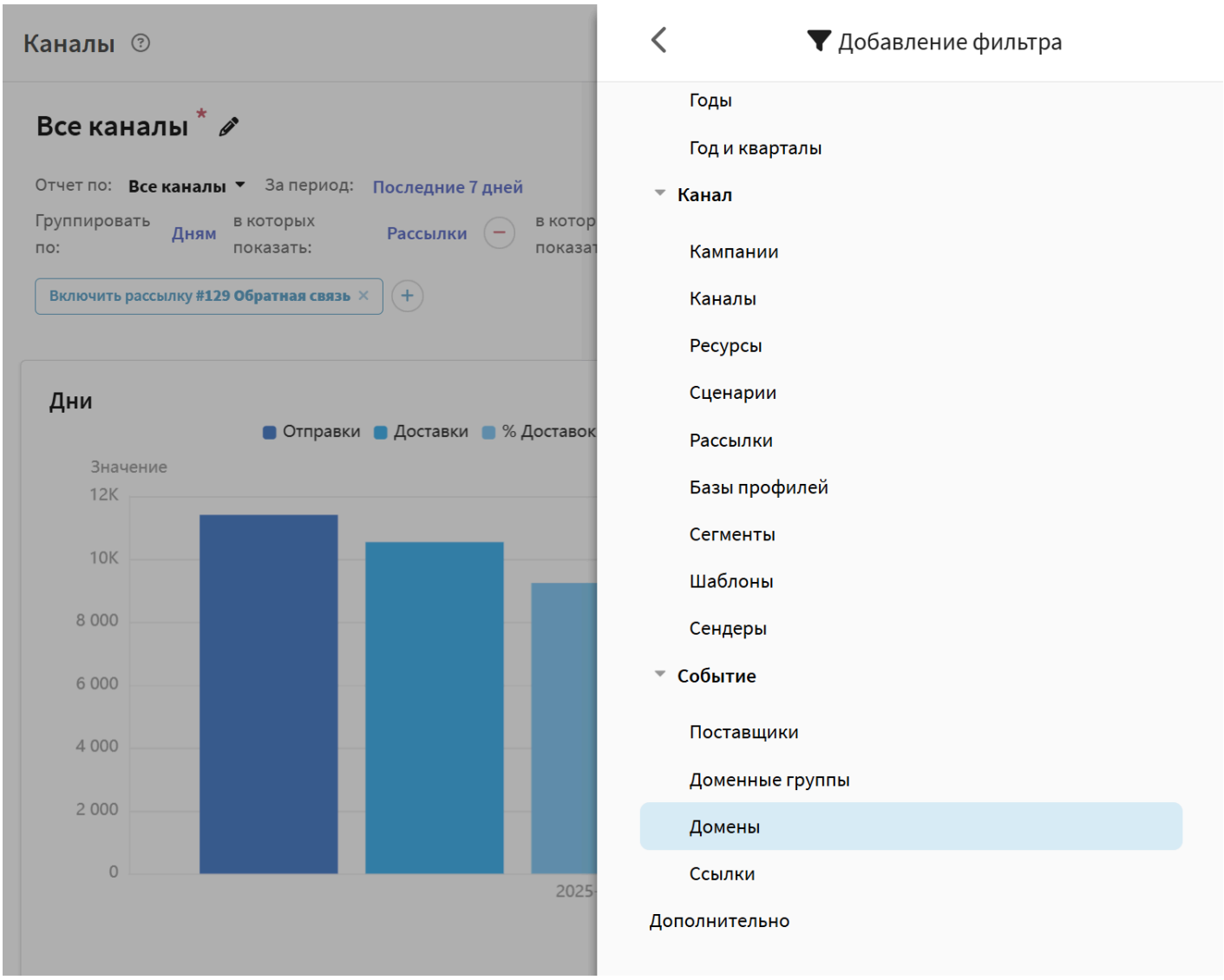
There is a third way to get mailing statistics. Select the Analytics section by Channels in the main platform menu. By default, a report for All channels for the last 7 days will be displayed. To group by the Email channel, select “Report by”:.



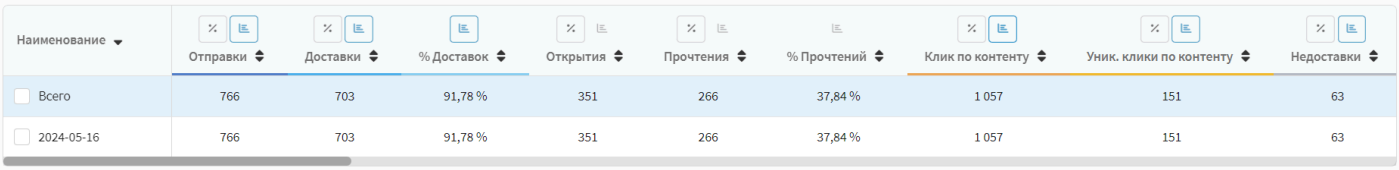
Next, Add a filter by Mailings. From the displayed list, select the required mailing.



To analyze message deliverability to email mailboxes of different providers, you can group the report by recipient domain group:

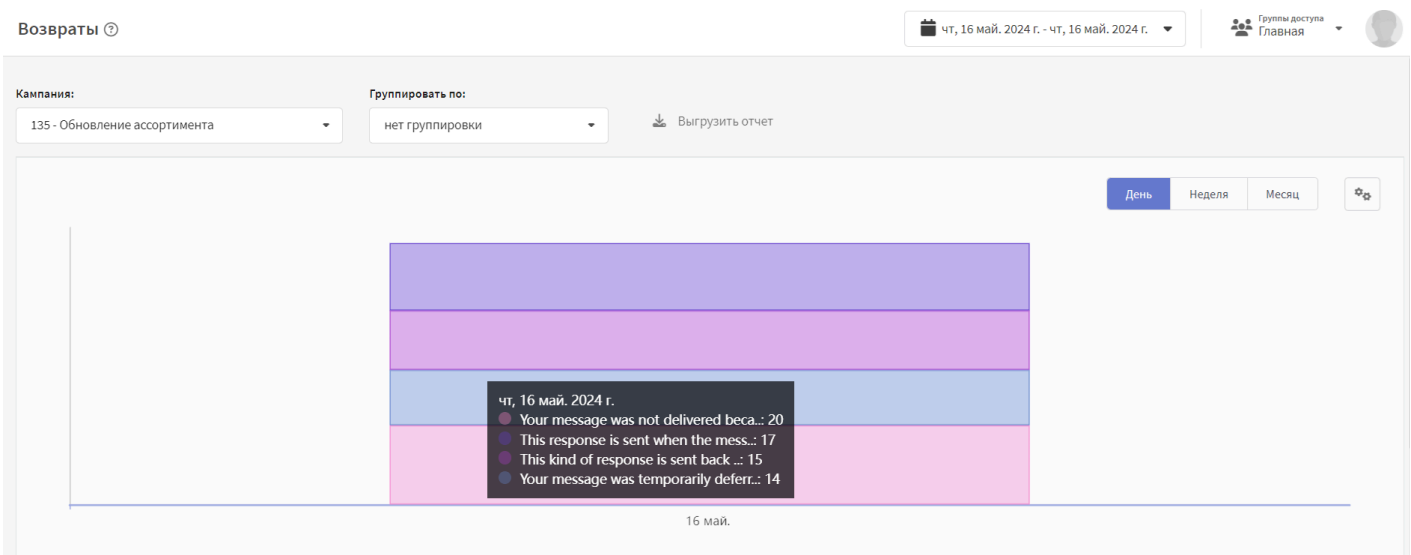


Provider information will appear in the table below the graph:

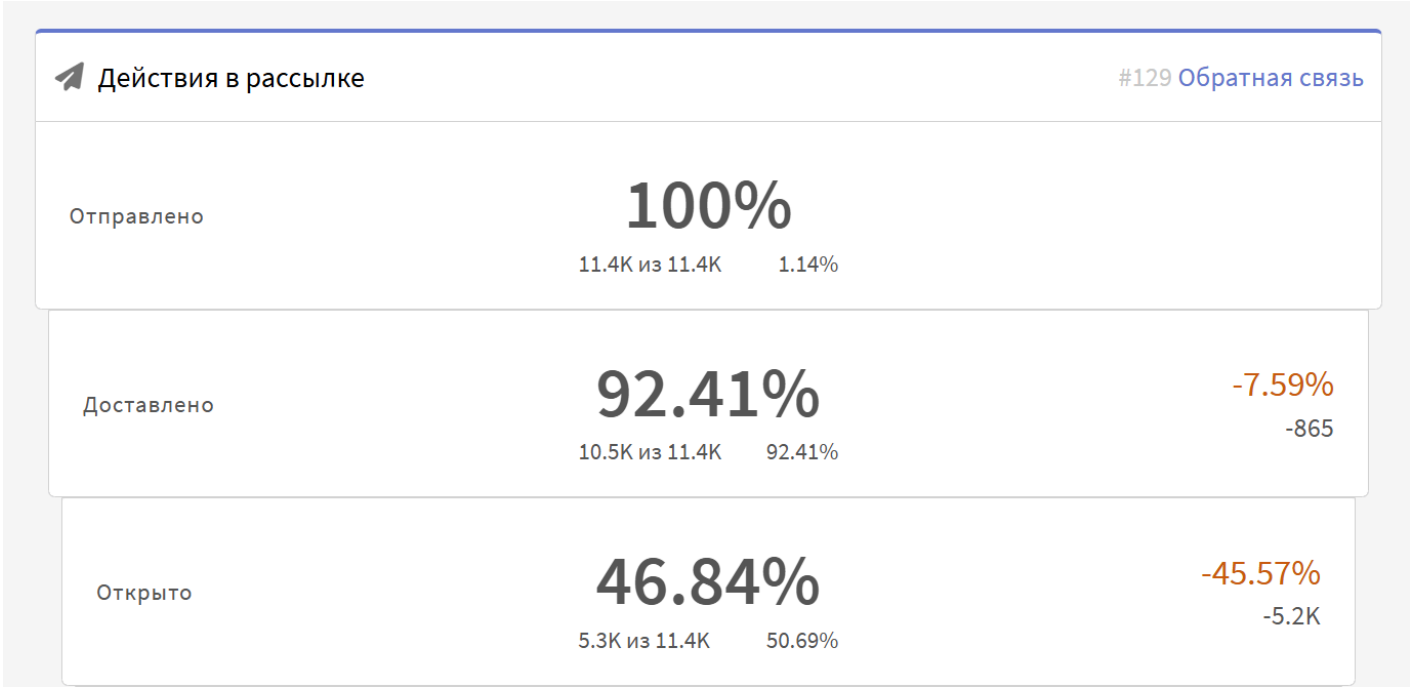


Наименование	Отправки	Доставки	% Доставок	Открытия	Прочтения	% Прочтений	Клик по контенту	Уник. клики по контенту	Недоставки
Всего	766	703	91,78 %	351	266	37,84 %	1 057	151	63
2024-05-16	766	703	91,78 %	351	266	37,84 %	1 057	151	63

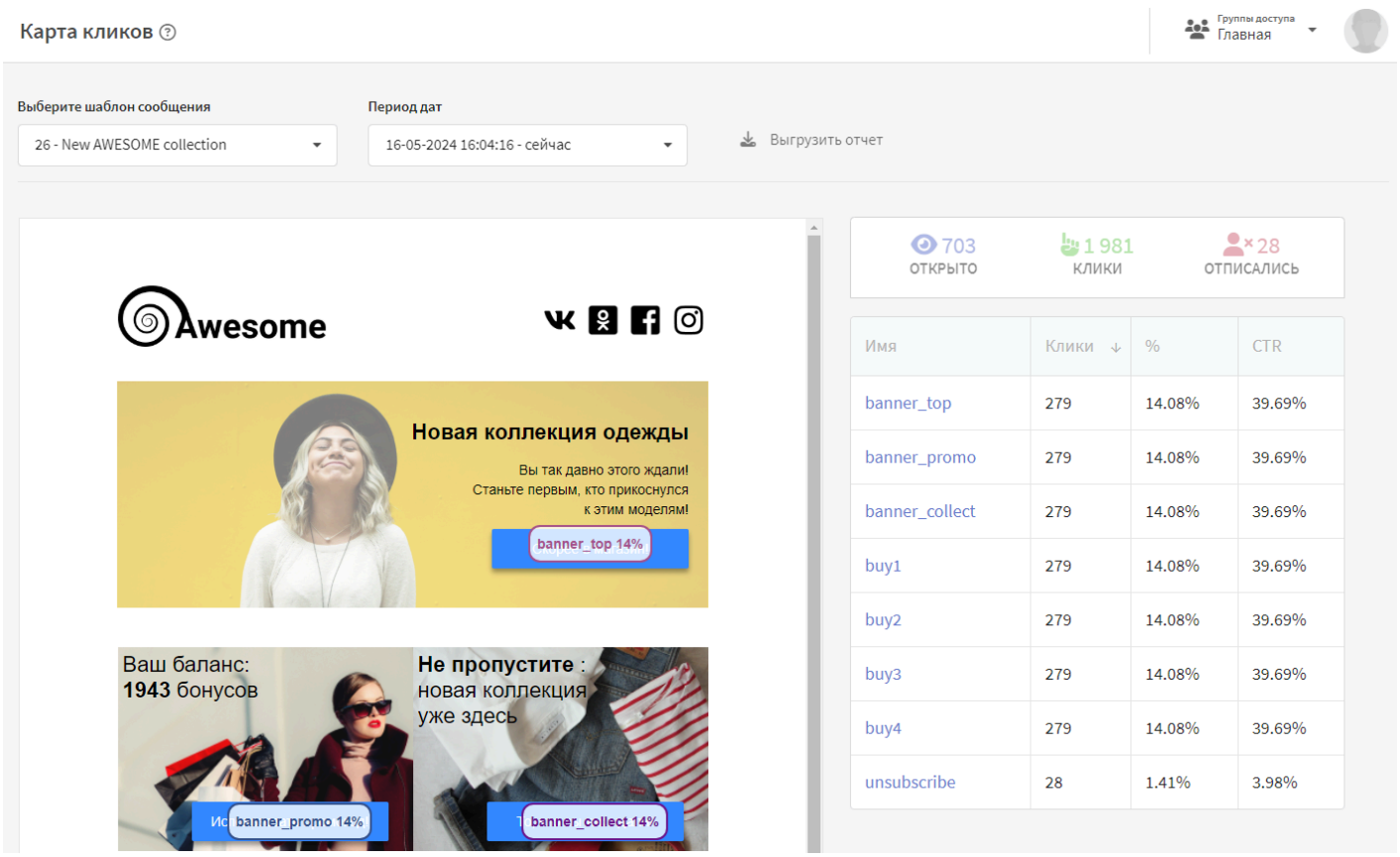
To analyze Email deliverability issues, use the Bounces and Undeliveries reports:



Actions in an Email mailing or template can become steps in an interactive conversion funnel:



For Email, a [click map]() is also available — it is compiled for templates with an Email version that have been sent at least once. When you make changes to the template and send again, a new version of the click map is created, allowing you to evaluate the dynamics of subscriber behavior:



Push Notifications

Push notifications are short messages sent to the screens of user devices. They are delivered directly and displayed even when the application or website is inactive.

Web Push and Mobile Push

Push notifications are divided into two types depending on the sending source.

Web Push

Web Push notifications are sent through a web browser. They work through browser technologies (Chrome, Safari, Firefox) and do not require installing an application. Notifications are delivered to desktop and mobile browsers.

When to use Web Push?

- **reach website visitors without installing an application**
- **cross-browser audience**
- **quick launch without developing a mobile application**

For more details, see the article Web Push.

Mobile Push

Mobile Push notifications are sent from mobile applications. They appear in the notification center of a mobile device — in the notification shade of Android and iOS. To receive notifications, the user installs the application.

When to use Mobile Push?

- **working with mobile application users**
- **higher engagement level**
- **access to device features (deep links, rich media)**

For more details, see the article Mobile Push.

Rich Push

Rich Push is an extended push notification format with additional elements: images, buttons, actions, deeplink. Element availability depends on the platform, browser, OS, and provider.

To create a Rich Push template, use the Rich Push editor.

Quick Start

To set up Web Push:

1. **Set up the resource and website**
2. **Set up providers**

To set up Mobile Push:

1. **Select providers**
2. **Integrate the application**

Mobile Push

TIP

The setup of browser push notifications is described in the corresponding article.

![img](/img/attachments/37847505/pic.jpg)	**Mobile Push** are notifications sent from mobile applications. They appear in the notification center of a mobile device — in the notification shade of Android and iOS.

Unlike Web Push, Mobile Push notifications are delivered to an installed application, not to a browser.

The delivery method of push notifications depends on the platform and application implementation:

- **On Android, FCM is most commonly used, but it is not the only option: an application can work with another push service if supported on both the application and server side.**
- **On iOS, push notifications are delivered through APNs. The user must grant permission to receive them.**

Push Tokens

A push token is a unique identifier linking a user's device to a push service. The token is assigned when the device registers in the application.

The token may change when:

- **the application is reinstalled**
- **the operating system is updated**
- **notification settings on the device are changed**

Altcraft mSDK integrated into the app automatically updates the push token.

Providers

Provider	Platforms	When to Use
Firebase Cloud Messaging	Android, iOS	Main provider for Android
APNs	iOS	Direct integration with Apple
AppMetrica	Android, iOS	When using AppMetrica SDK
Huawei Mobile Services	Android, iOS	Huawei devices without Google
RuStore	Android	RuStore ecosystem

CAUTION

Yandex.AppMetrica uses Google Firebase SDK for sending. The Firebase SDK must be installed for sending.

Application Integration

For Mobile Push to work, Altcraft mSDK integration into the mobile application is required. mSDK handles push provider connection, subscription token submission, notification display, and event registration.

Select your platform and follow the guide:

- **Android — Quick Start**
- **iOS — Quick Start**
- **Flutter — Quick Start**
- **React Native — Quick Start**

For more details, see the section Application Integration with Altcraft.

Rich Push

Rich Push allows sending notifications with extended elements: banners, buttons, deeplink. Setup is available when creating a template.

For more details, see the article Push Templates.

Quick Start

1. **Provider Overview — selecting providers for your application**
2. **First Mobile Push Mailing — step-by-step guide**
3. **Application Integration — mSDK technical integration**

First Mobile Push Mailing

![img] (/img/attachments/37849061/37849080.png)	**This tutorial will help you to create and launch your Mobile Push mailing in the Altcraft Platform from scratch.** Mobile push notifications are displayed in your customers' mobile device message centre – in the upper part of the screen most commonly. They are allowed by default on application installation. That's why you can start engaging your customers right away and increase your app conversion. Push notifications are much more profitable than SMS, because sending them practically costs nothing. </td> </tr>

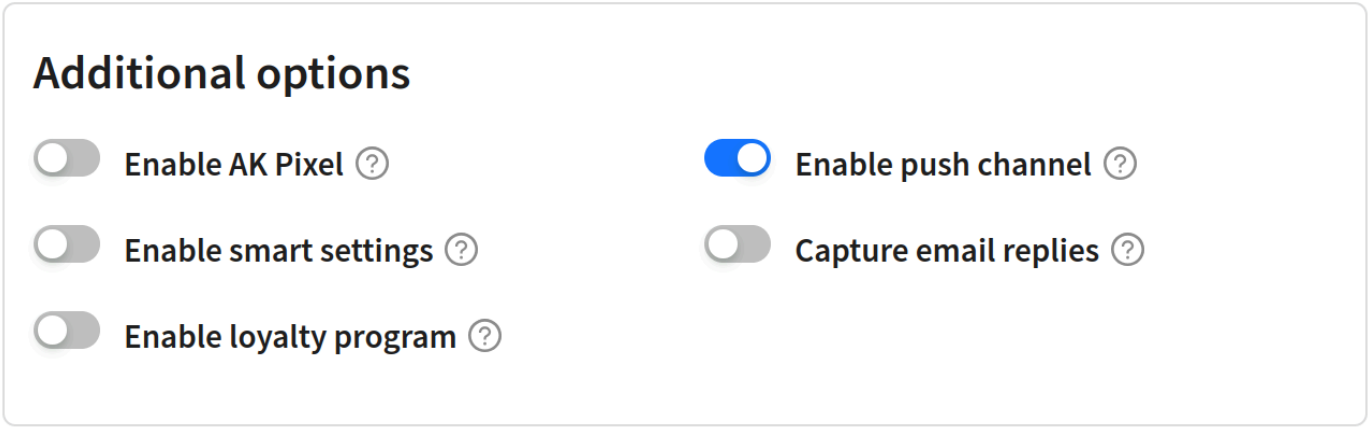
Step 1: Configure a Domain to Work with Push

CAUTION

If you use an Altcraft cloud account, please give the information to the support team at support@altcraft.com. They will help you to connect the integration.

If the Altcraft platform is installed on your equipment (On-premises), contact your platform administrators. Or just contact our support team!

Make sure your website is running over HTTPS. In addition, web pixels must be configured in the Altcraft system. Also, push notifications must be activated in the Admin Panel for accounts from which you plan to send:



Step 2: Set Up Your Mobile Provider Project

Depending on the provider you choose, create a project on the provider's side and obtain the data needed to configure the resource in the platform:

- **Firebase Cloud Messaging — project in Firebase Console.**
- **APNs — key or certificate in Apple Developer.**
- **AppMetrica — application in AppMetrica.**
- **Huawei Mobile Services — project in AppGallery Connect.**
- **RuStore — push notifications project in RuStore Console.**

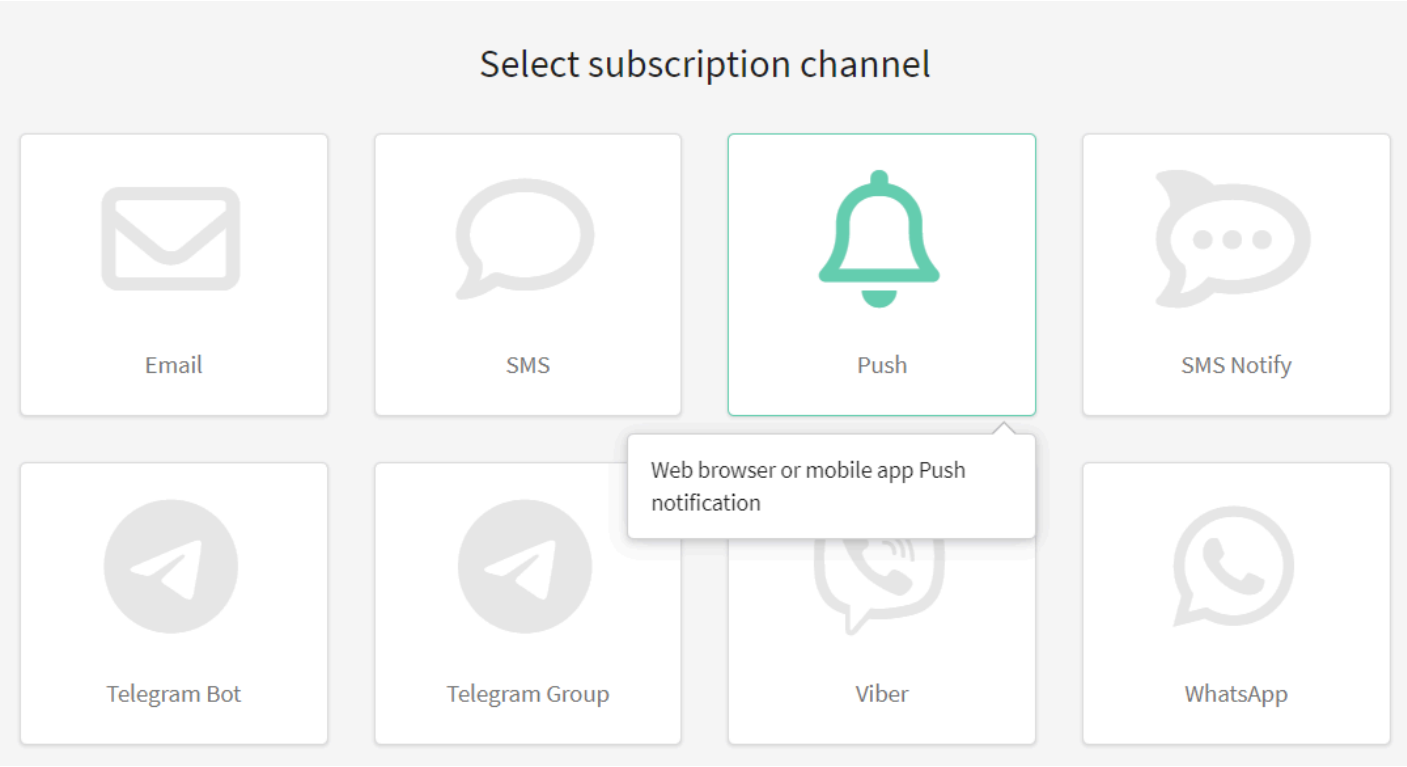
Step 3: Create a Database and Subscription Resource

Altcraft mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

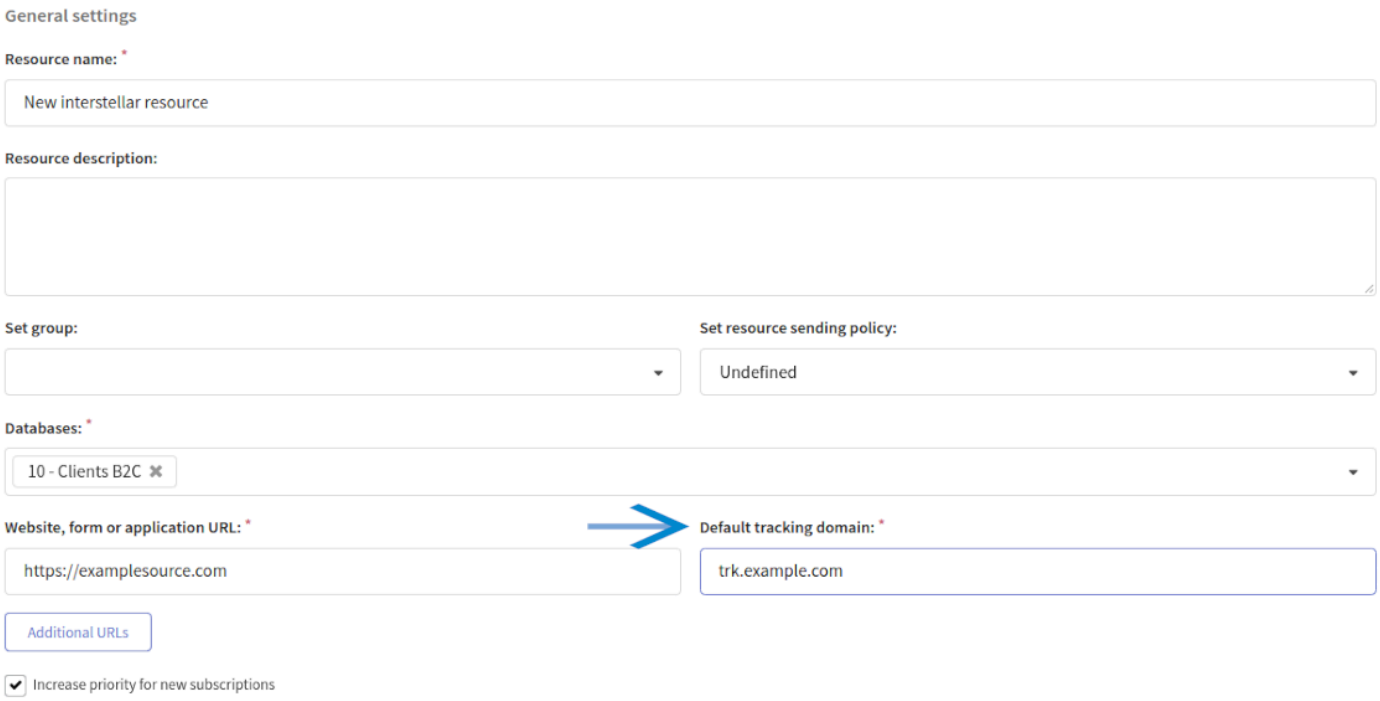
When you start your Push mailing, you use profile subscriptions to resources. This way, you can send notifications to different devices or to different browsers. Each client profile can be subscribed to one or more resources.

What can be a resource? A specific marketing area, a site, an application, or the type of content.

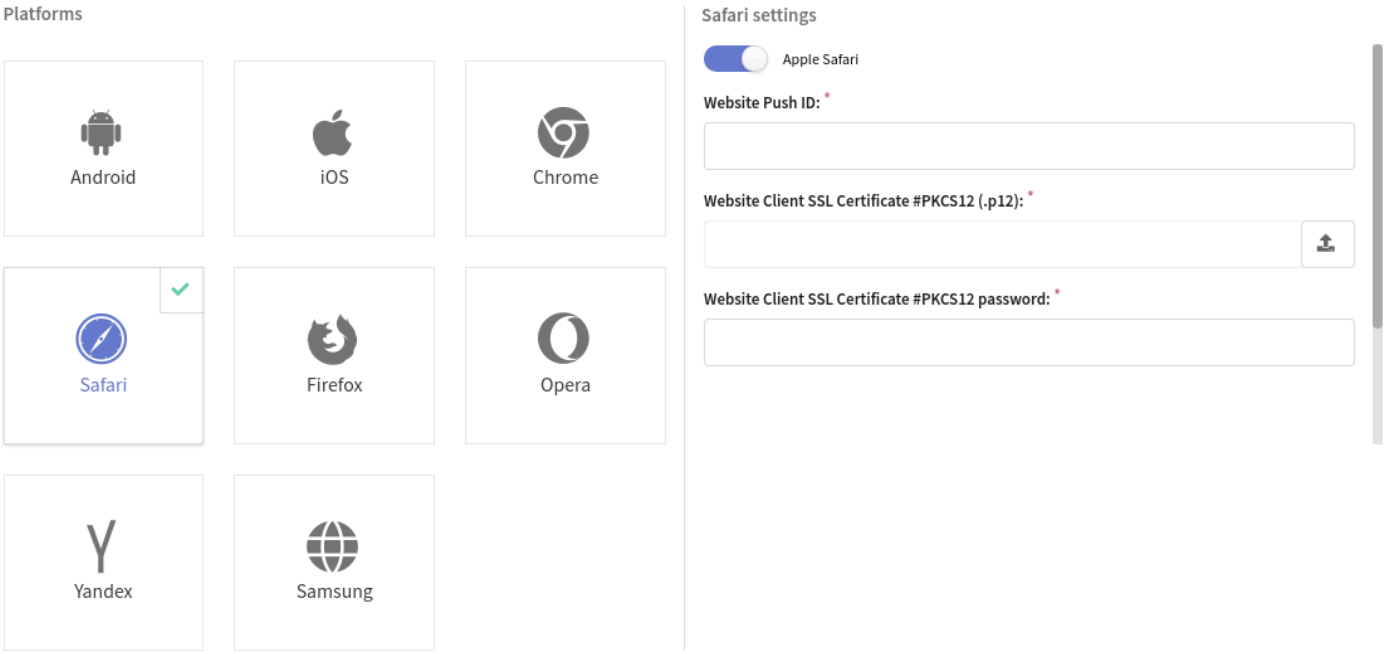
Create a profiles database if you don't have one yet. Then create a new resource with a Push channel:



In the general settings, add the database. If needed, assign a sending policy and specify the Default tracking domain. This domain will process your customers' clicks on links in messages:



In Push settings, select the platforms you are going to send notifications to and enter the provider settings obtained in the previous step:



Step 4: Integrate mSDK into Your App

To send Mobile Push notifications, you need to integrate Altcraft mSDK into your mobile application. mSDK handles push provider connection, subscription token submission, notification display, and event registration.

JWT Token Authorization

When configuring the resource, use a JWT token for mSDK authorization. The token is generated and signed on the app's server, providing security and allowing profile lookup by any identifiers (email, phone, custom ID). It supports multiple users on one device and profile recovery after app reinstallation.

For JWT to work, you need to prepare your app's backend to generate and sign tokens. For more details, see Role Token and JWT Token.

> INFO
>
> **As an alternative, rToken (role token) is available, where profile matching is done only by push token. This option is less reliable and limits capabilities — the profile may be "lost" when the device changes or the app is reinstalled.**

Share the resource token with your developers and follow the quick start guide for your platform:

- **Android — Quick Start**
- **iOS — Quick Start**
- **Flutter — Quick Start**
- **React Native — Quick Start**

> TIP
>
> **A subscription to Yandex.AppMetrica can be added automatically to users of the app that has the connected metric service. Read more in the article "Integration with Yandex.AppMetrica".**

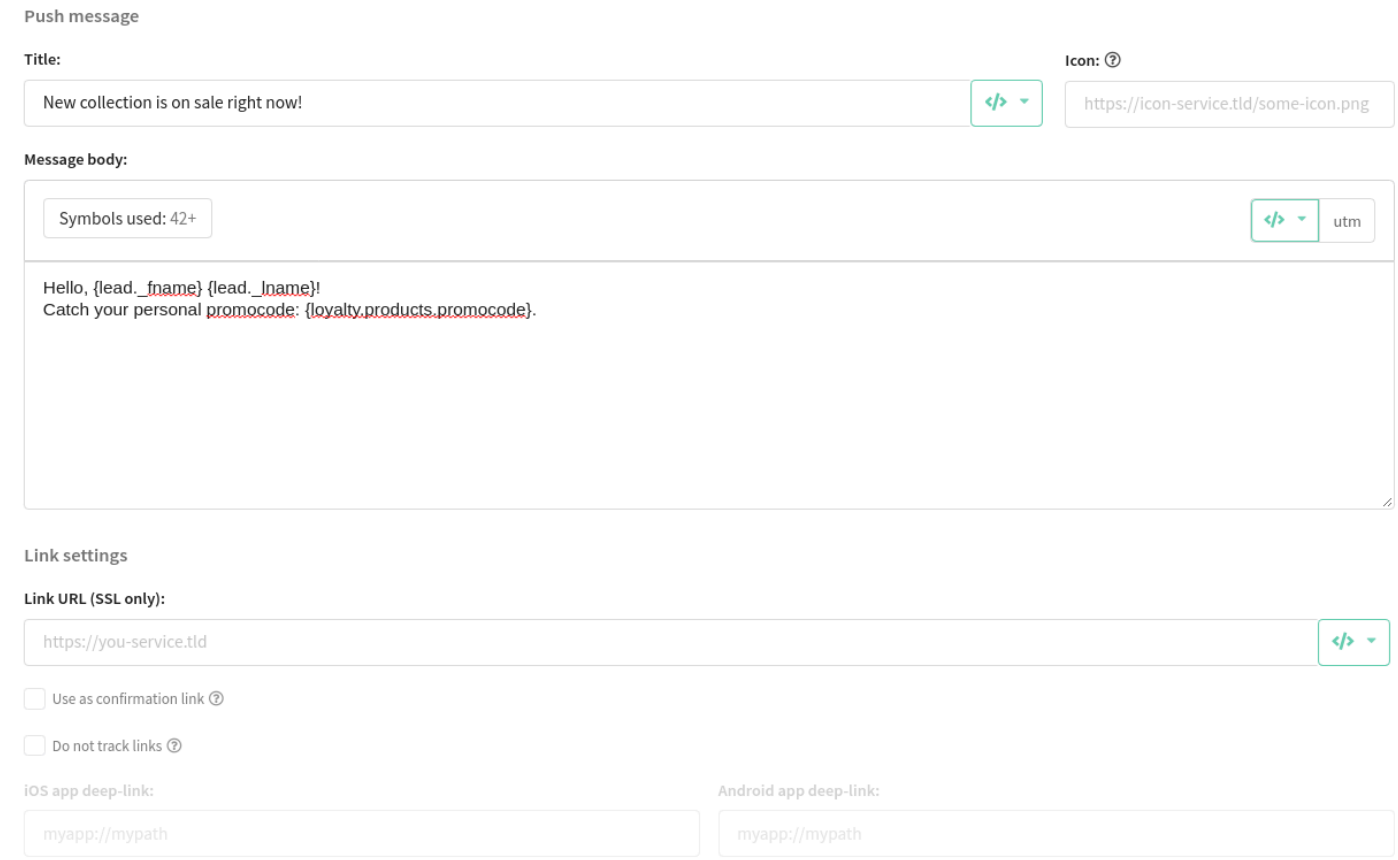
Step 5: Prepare Content for Sending

The content for your Push mailings is created in Message Templates. The templates contain content for all available channels. When you create a new template, select the Push channel or add it for an existing one.

The template of the push notification can contain:

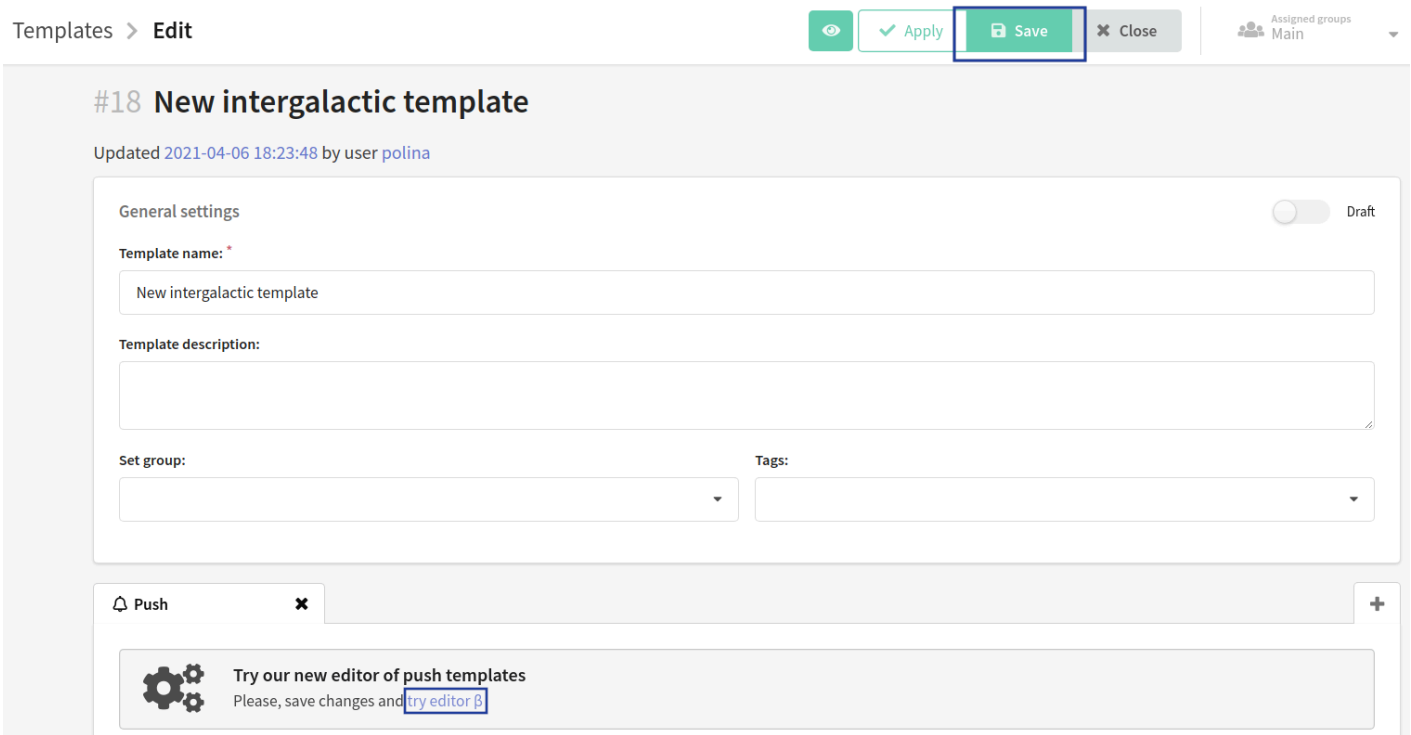
- **A title that is similar to email subject**
- **Notification body is the main text part of the message**
- **An image or an icon that will be displayed in the notification**
- **A clickable link is the link that the subscriber will follow when he clicks on the push notification.**

To the title, link and body, you can add the data of a specific client, as well as dynamic text content from external sources:



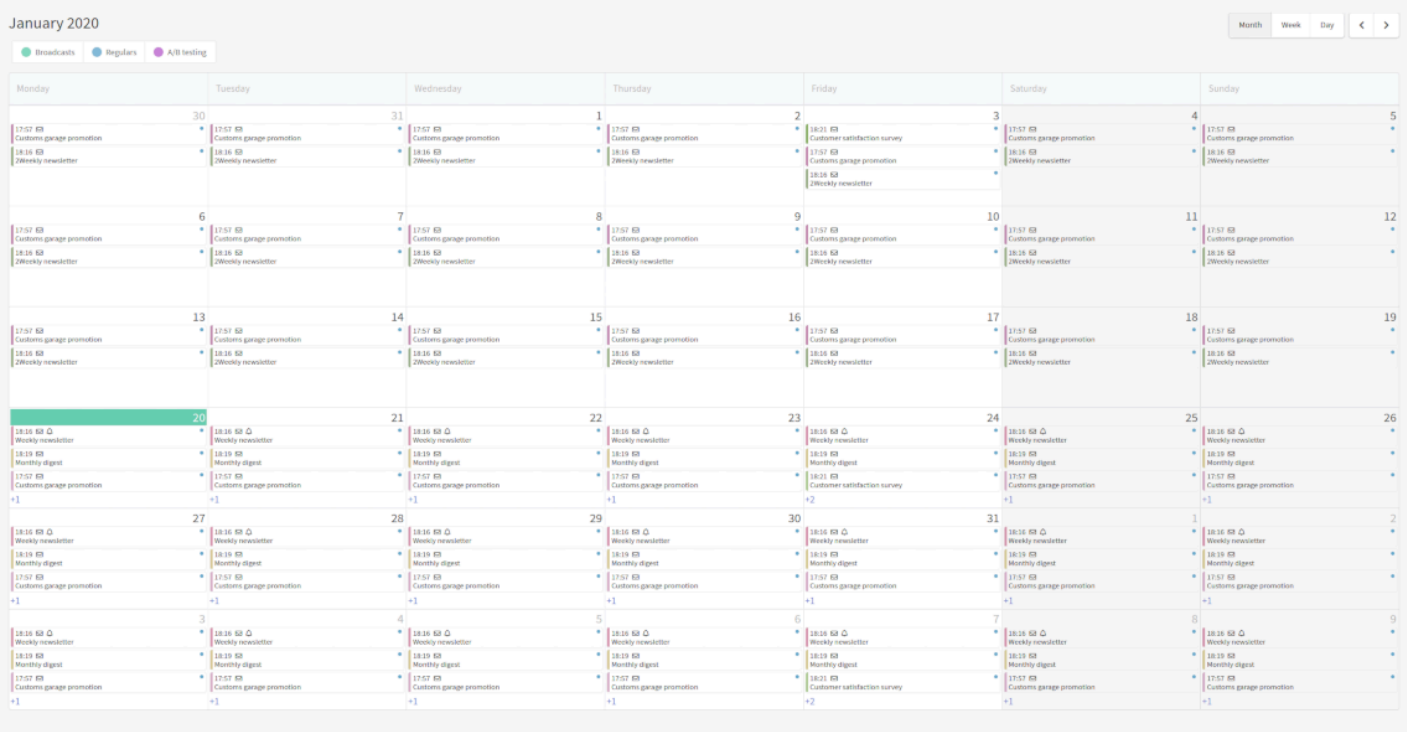
If you send not just push, but Rich push notifications, you'd better take a look at the special editor to create a template. In Rich push notifications, you can add not only a text and an icon, but also buttons, images, banners, and even audio and video files. What's more, the new editor will allow you to individually customize the notification for a specific OS, as well as add adaptive links and deeplinks.

To open the Rich push editor, you need to make an initial version of a push message in the usual template editor at first. We've just described how to do this above. Then click Save. Now go back to the created template and click try editor β to launch the new editor.

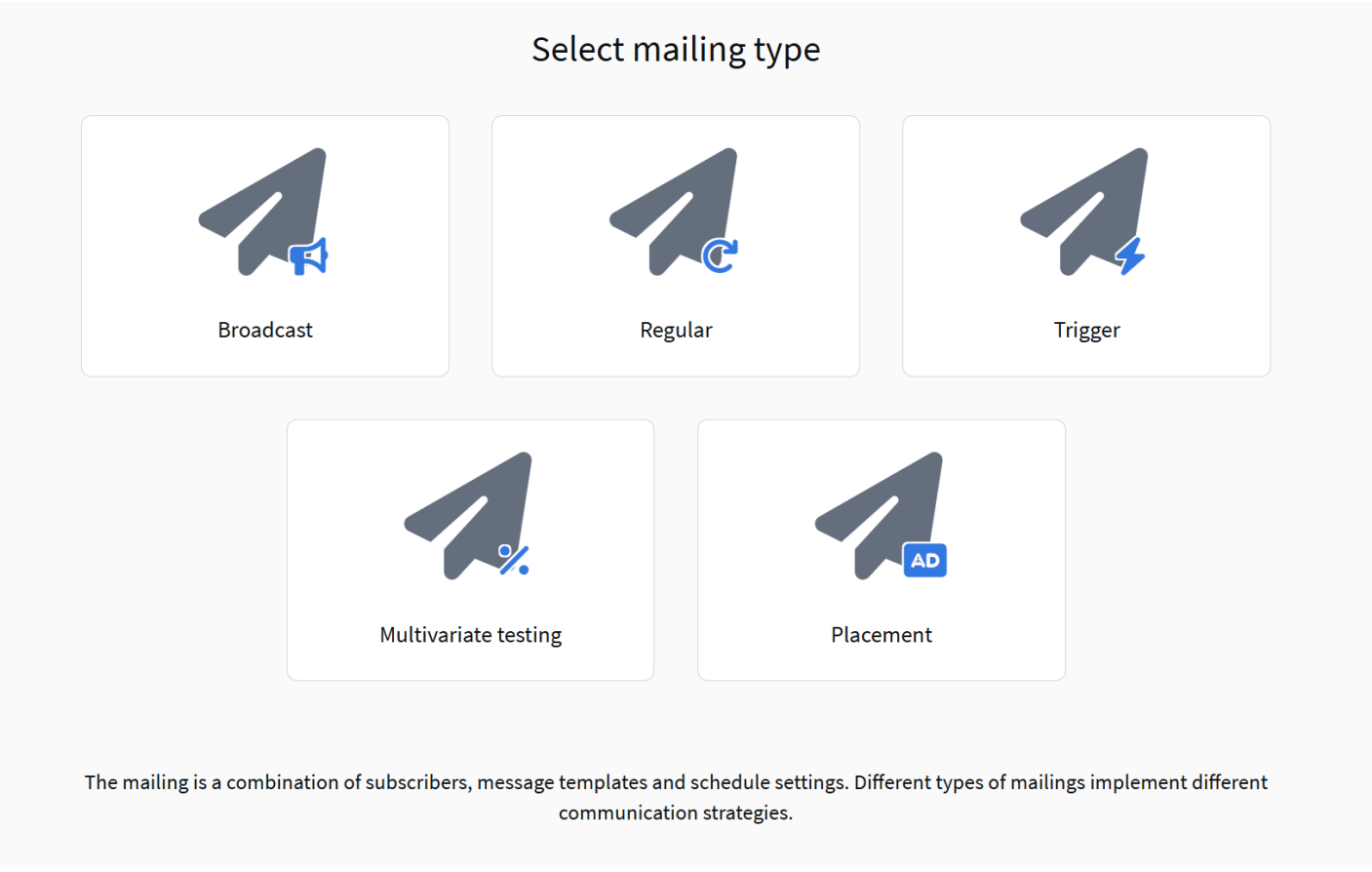


Step 6: Select a Mailing Type

![img] (/img/attachments/37849061/37849103.png)	**If you want to send a Push message, you need to use such a tool as a Mailing.** Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose ** ** Broadcasts. If the mailing is periodic, your choice is a ** ** Regular mailing. In the main menu select Mailing and open ** ** Calendar. This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



TIP

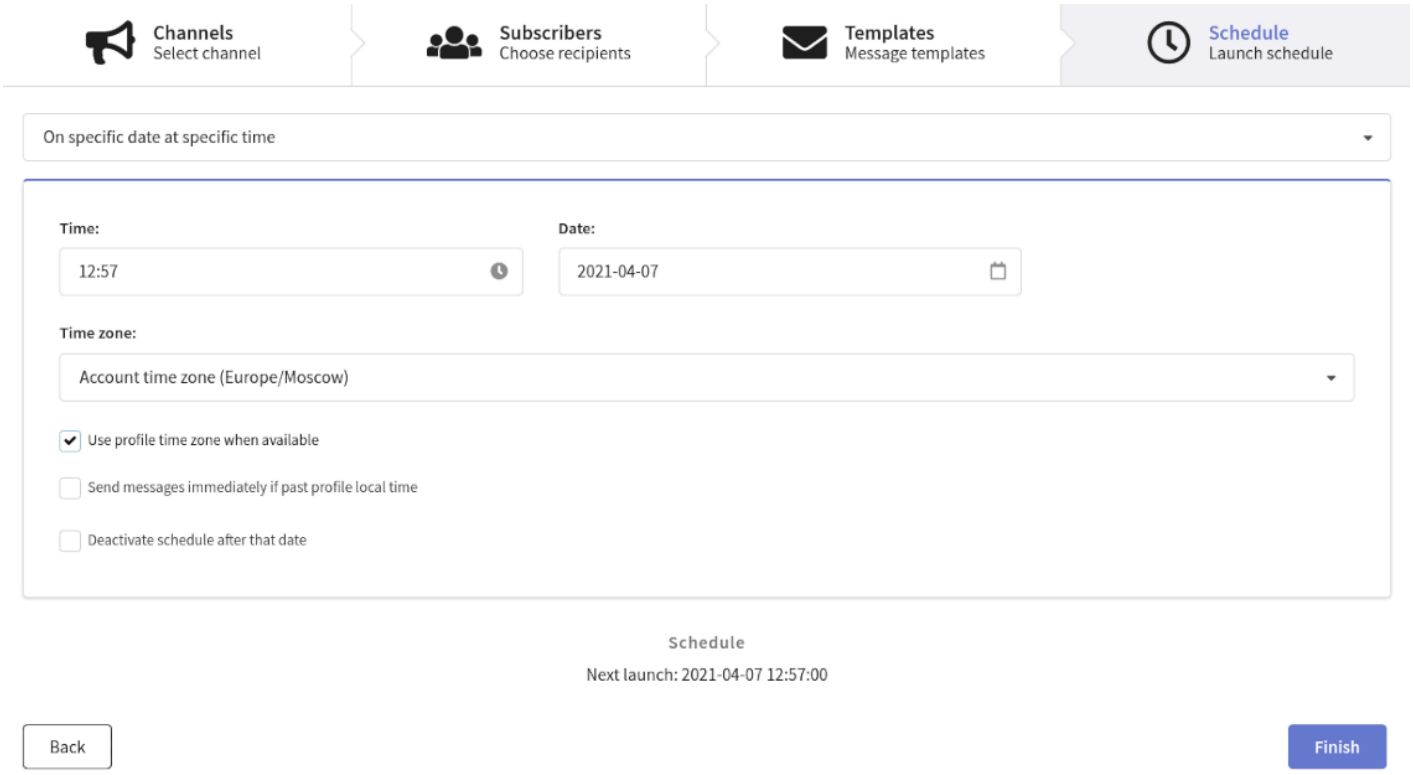
You can find details about how to set up Broadcast on this page.

TIP

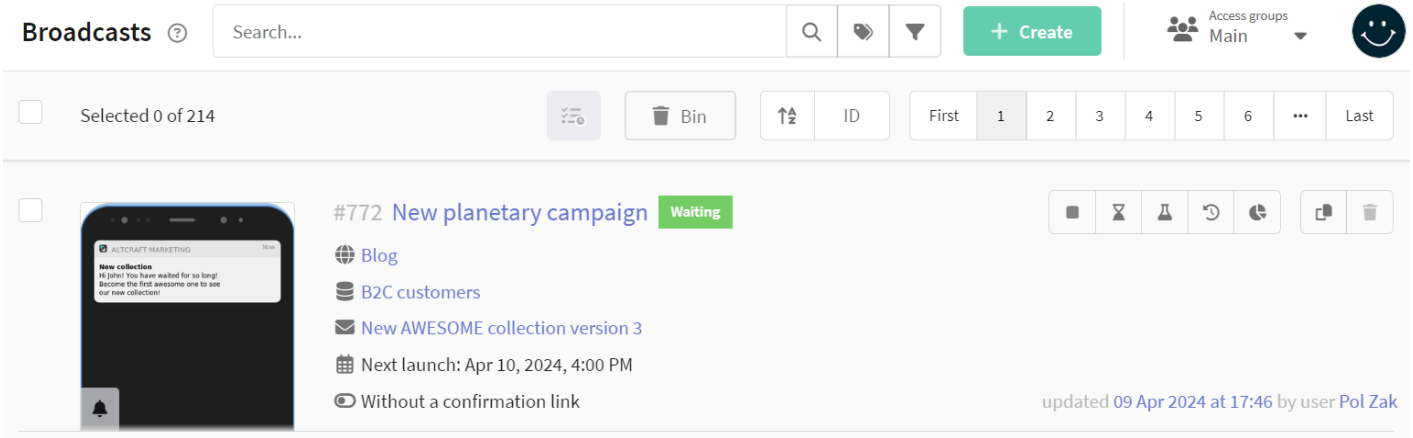
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select Push as the communication channel**
2. **Select providers and sending mode. There are two sending modes - **fanout**, sending to all subscriptions, or **last**, sending to the subscription with the most recent creation date. If there are several of these (created at the same time), then one subscription will be selected randomly. If mailings have filters by provider, custom fields or categories, then the filters are first applied, and then the most recent subscription is chosen;**
3. **Select the subscription resource you've just configured and the audience of recipients;**
4. **Add a message template to your message;**
5. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is available.**



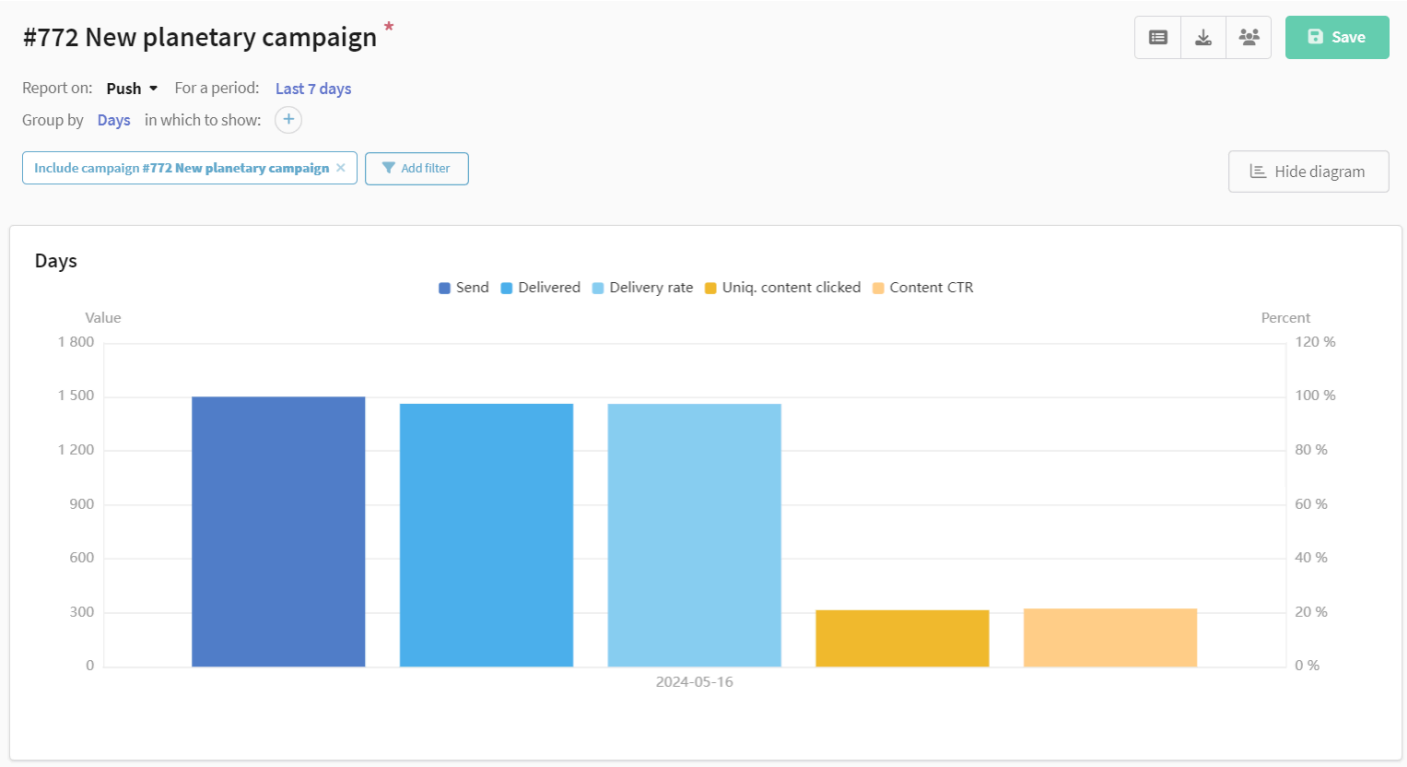
Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. There is a quick action menu on the right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



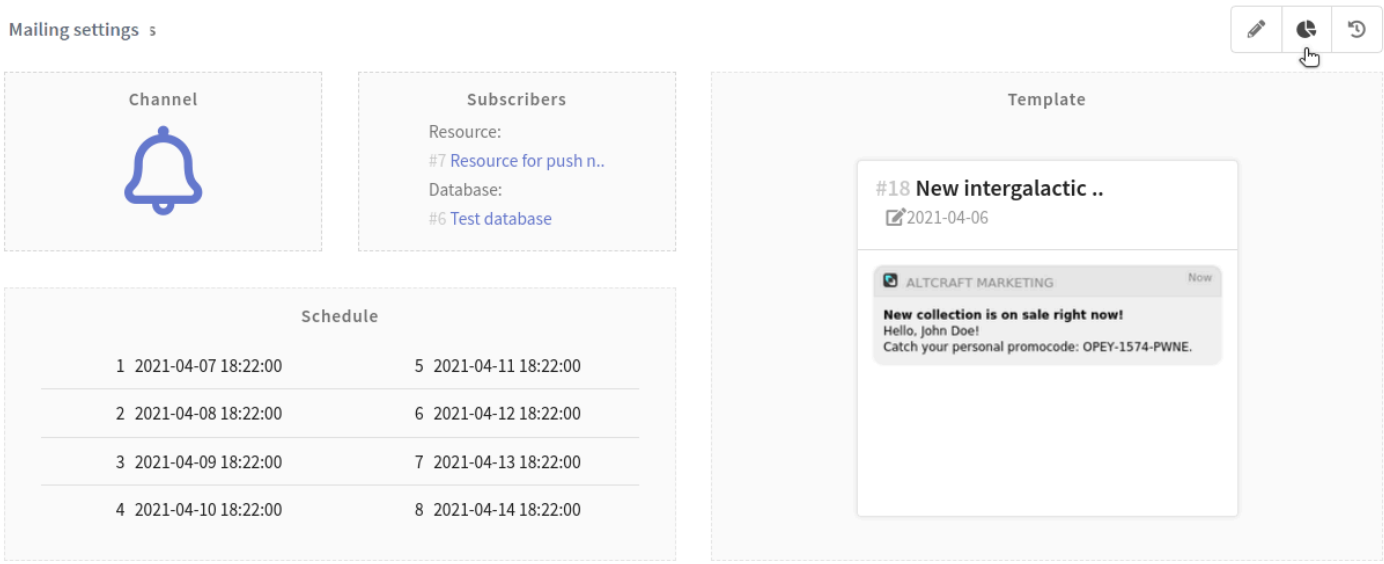
Step 7: Assess the Mailing Effectiveness

Do you want to assess the effectiveness of your mobile push mailings? Go to the section Analytics in the main menu of the platform.

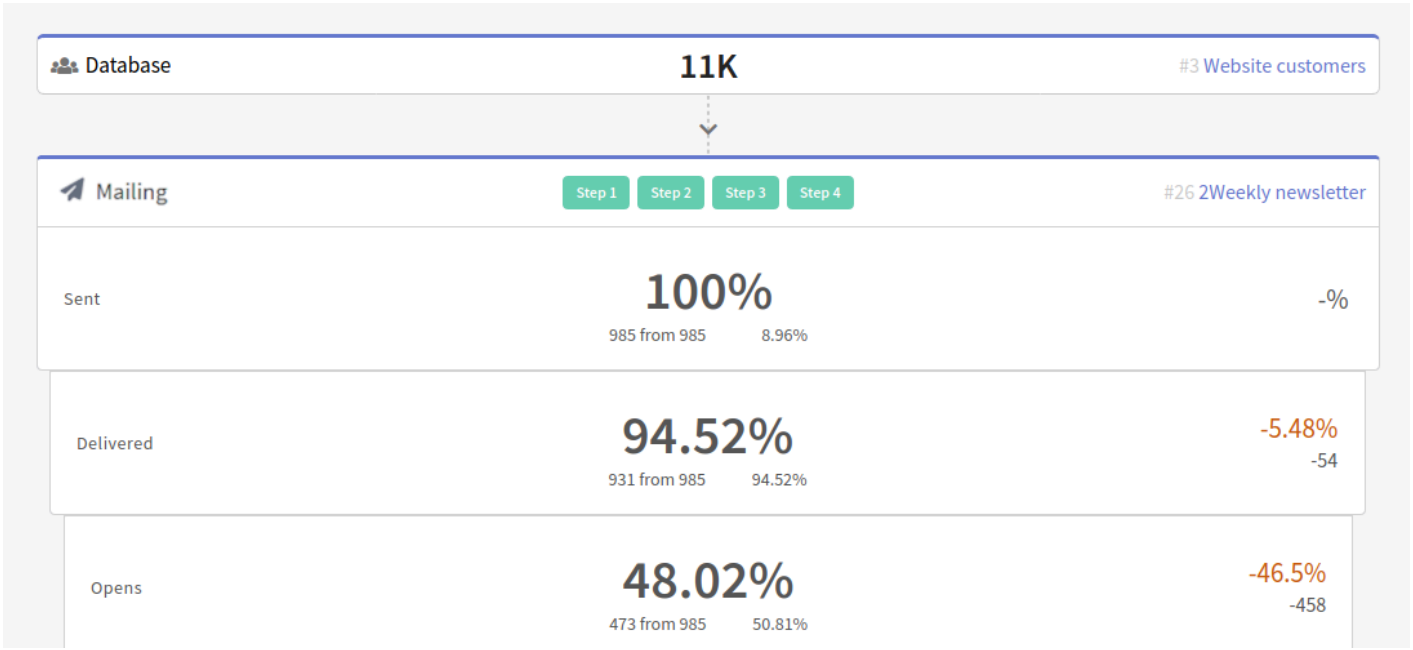
The channel report provides analytics for one or several mailings, for one or several communication channels. For grouping, you can select Report on Push:



You can also go to the channel report for a specific mailing by following the link in the mailing editor:



Actions in the Push mailing or in the template can turn into steps in the interactive conversion funnel:


Database
11K
#3 Website customers
Mailing
Step 1
Step 2
Step 3
Step 4
#26 2Weekly newsletter
Sent
100%
985 from 985
8.96%
-%
Delivered
94.52%
931 from 985
94.52%
-5.48%
-54
Opens
48.02%
473 from 985
50.81%
-46.5%
-458

Mobile Push Providers

To send push notifications to mobile devices, you need to set up integration with a push provider. A provider is a service that delivers notifications to the user's device.

The choice of provider depends on:

- **target platform (Android, iOS)**
- **device ecosystem (Google, Huawei, RuStore)**
- **tools already in use (AppMetrica)**

Available providers

Provider	Platforms	When to use
Firebase Cloud Messaging (FCM)	Android, iOS	Primary provider for Android apps and part of iOS scenarios
Apple Push Notification Service (APNs)	iOS	Sending notifications to iOS devices
Yandex.AppMetrica	Android, iOS	If your app uses AppMetrica Push SDK
Huawei Mobile Services	Android, iOS	Huawei devices without Google services
RuStore	Android	Android apps in the RuStore ecosystem

TIP

Yandex.AppMetrica uses Google Firebase SDK for delivery. To send notifications, you need to install the Firebase SDK in your app.

Next steps

Detailed setup for each provider is described in the corresponding articles. Select the provider you need above, or start with Firebase Cloud Messaging if you're setting up push for an Android app.

For technical information about push message format, see Integrate your app with Altcraft.

Firebase Cloud Messaging

To send Mobile Push to Android and iOS devices, you need to set up a project in Firebase Console in advance.

Step 1. Creating a project in Firebase

![img](/img/attachments/sdk-v2/android/fcm/fcm_1.png)

![img](/img/attachments/sdk-v2/android/fcm/fcm_2.png)

![img](/img/attachments/sdk-v2/android/fcm/fcm_3.png)

![img](/img/attachments/sdk-v2/android/fcm/fcm_4.png)

Step 2. Configuring in the Altcraft platform

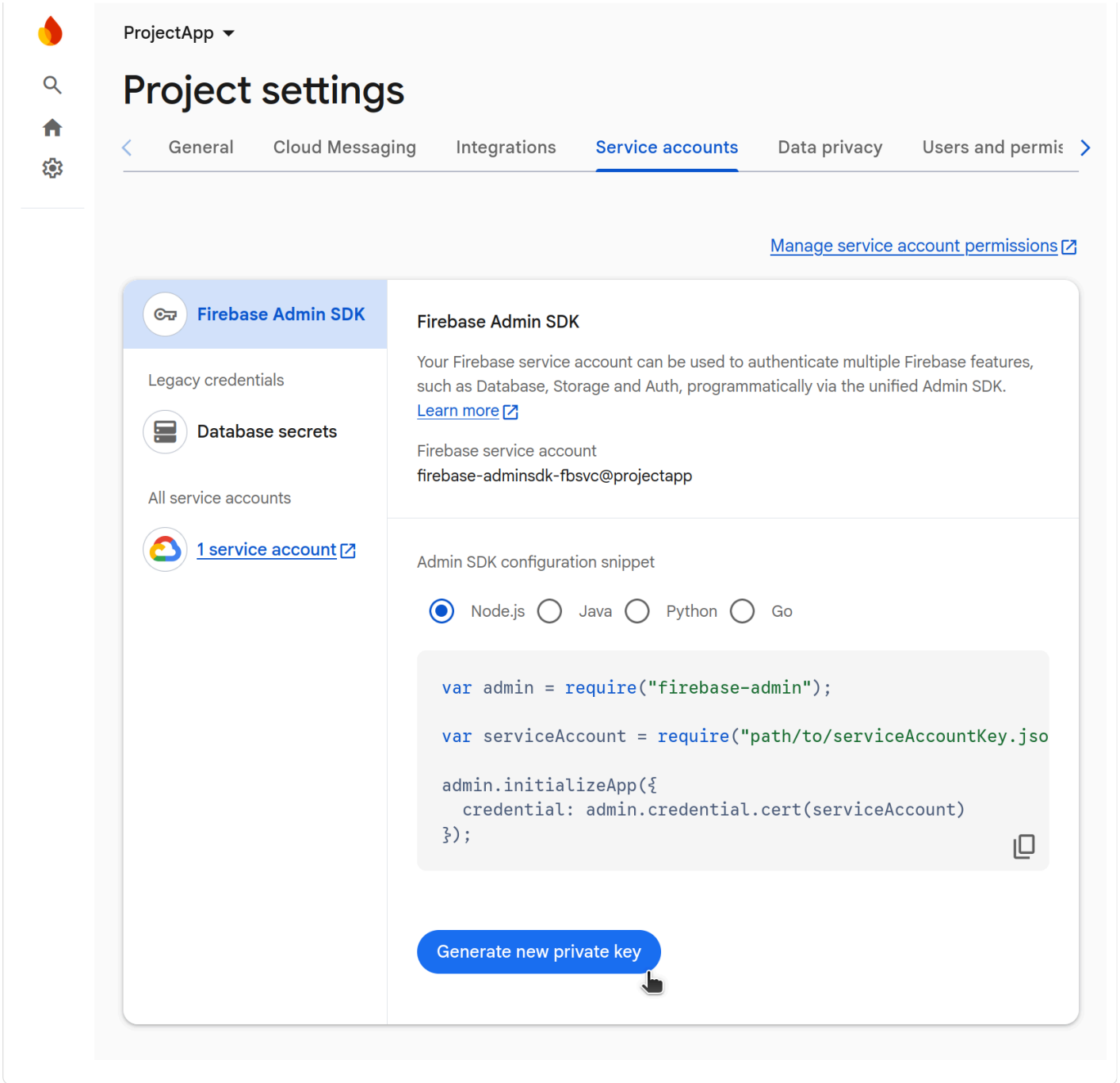
Android

![img](/img/attachments/sdk-v2/android/fcm/resource_fcm_en.png)

![img](/img/attachments/sdk-v2/android/fcm/settings_fcm_en.png)

TIP: Private key

Private key generates in this settings section:



![img](/img/attachments/sdk-v2/ios/providers/ios_fcm_4.png)

![img](/img/attachments/sdk-v2/ios/providers/ios_fcm_5.png)

![img](/img/attachments/sdk-v2/ios/providers/ios_fcm_2_en.png)

App integration with Altcraft — adding SDK and registering subscriptions
Push message structure — message format for Firebase

Apple Push Notification Service

To send Mobile Push to iOS devices, you need to set up Apple Push Notification Service in advance.

> **CAUTION**
>
> **All actions are performed on a macOS device. An Apple Developer account is required for setup.**

APNs supports two types of authentication: authentication key (recommended) and certificate.

Option 1: authentication key

Step 1. Creating an Auth Key

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-2.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-3.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-4.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-5.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-6.png)

Step 2. Configuring in the Altcraft platform

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-13.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-14.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-7.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-8.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-9.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-10.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/apns-11.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-12.png)

![img](/img/attachments/22708437/7export_en.png)

Step 2. Creating pem files

Navigate to the folder with the exported certificate via terminal.

Create the pem key:

```
openssl pkcs12 -legacy -in filename.p12 -out filename.key.pem -nocerts -nodes
```

Create the pem certificate:

```
openssl pkcs12 -legacy -in filename.p12 -out filename.crt.pem -clcerts -nokeys
```

Open both files in a text editor — you will need to paste their contents into the platform fields:

Step 3. Configuring in the Altcraft platform

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-13.png)

![img](/img/attachments/sdk-v2/ios/providers/apns/en/apns-15.png)

App integration with Altcraft
[APNs push message structure]

Huawei Mobile Services

To send Mobile Push to Huawei devices, you need to set up a project in AppGallery Connect in advance. H

Step 1. Creating a project in AppGallery Connect

![img](/img/attachments/sdk-v2/android/hms/en/hms_1.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms_2.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms_3.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms_4.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms_5.png)

![img](/img/attachments/sdk-v2/android/hms/en/rost.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms_6.png)

Step 2. Configuring in the Altcraft platform

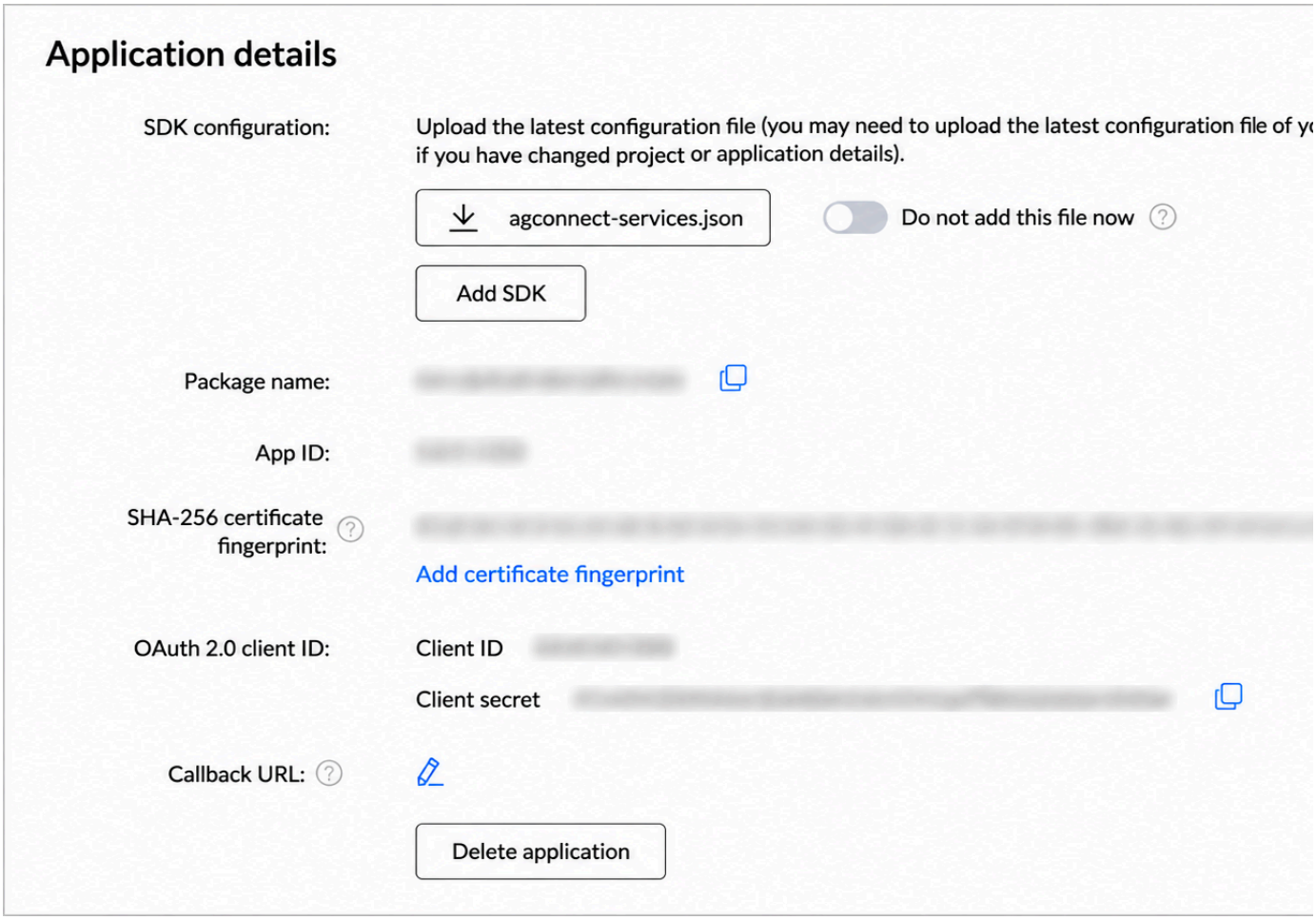
Android

![img](/img/attachments/sdk-v2/android/hms/en/resource_hms.png)

![img](/img/attachments/sdk-v2/android/hms/en/settings_hms.png)

TIP

All of this information can be found in the App Information section of Huawei Mobile Services:



![img](/img/attachments/sdk-v2/android/hms/en/ios-add-app-hms.png)

![img](/img/attachments/sdk-v2/android/hms/en/ios-plist.png)

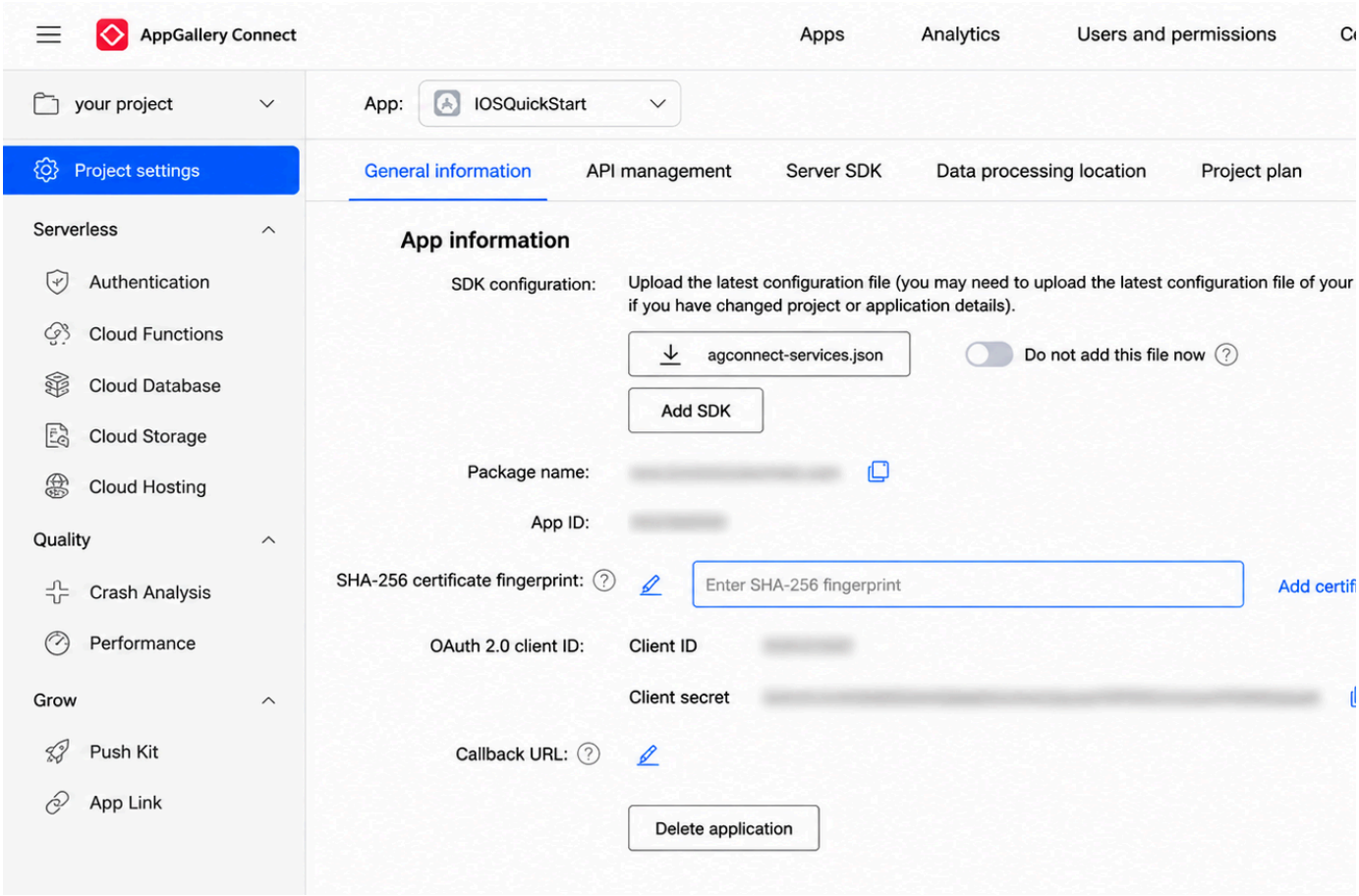
![img](/img/attachments/sdk-v2/android/hms/en/ios-config.png)

![img](/img/attachments/sdk-v2/android/hms/en/hms-platform-ios.png)

![img](/img/attachments/sdk-v2/android/hms/en/settings_hms.png)

TIP

All of this information can be found in the App Information section of Huawei Mobile Services:



App integration with Altcraft

[HMS push message structure]

RuStore

To send Mobile Push via RuStore, you need to create a push notifications project in the RuStore develope

INFO: Note

The RuStore app must be installed on the device.

Step 1. Creating a project in RuStore Console

![img](/img/attachments/sdk-v2/android/rustore/ru_1.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_2.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_3.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_4.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_5.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_6.png)

![img](/img/attachments/sdk-v2/android/rustore/ru_7.png)

Step 2. Configuring in the Altcraft platform

![img](/img/attachments/sdk-v2/android/rustore/resource_ru.png)

![img](/img/attachments/sdk-v2/android/rustore/settings_ru.png)

TIP: RuStore notification requirements

- **The RuStore app is installed on the device**
- **The RuStore app has notification permission granted**
- **The RuStore app has permission to disable battery saver in background mode**

What's next

- **App integration with Altcraft**
- **RuStore push message structure**

Yandex.AppMetrica

When to use

Yandex.AppMetrica is an analytics platform that also supports push notification delivery. Use AppMetrica

- **your app already uses AppMetrica for analytics**
- **you want to combine analytics and push campaigns**
- **you prefer managing settings in a single service**

> CAUTION
>
> **Yandex.AppMetrica uses Google Firebase SDK for delivery. To send notifications, you need to install the**

Prerequisites

- **AppMetrica Push SDK installed in the app. More details in the AppMetrica documentation.**
- **An AppMetrica account with a created app.**

Subscription sources

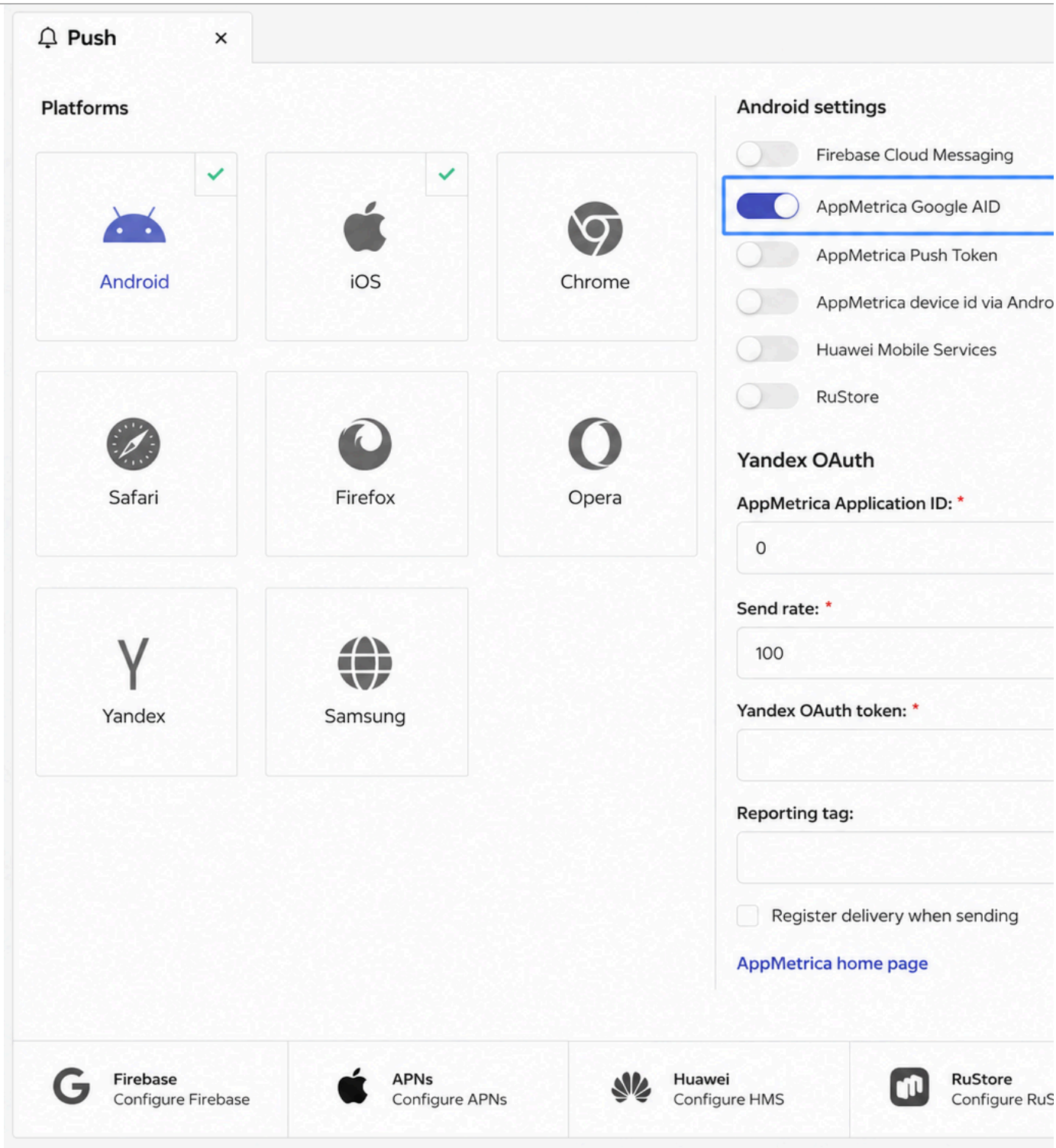
The following identifiers can be used as subscription sources:

- **AppMetrica device ID — device identifier in AppMetrica (for both OSes)**
- **iOS IFA — advertising identifier (Apple only)**
- **Google AID — advertising identifier (Android only)**
- **Android Push Token — token from AppMetrica SDK**
- **iOS Push Token — token from AppMetrica SDK**

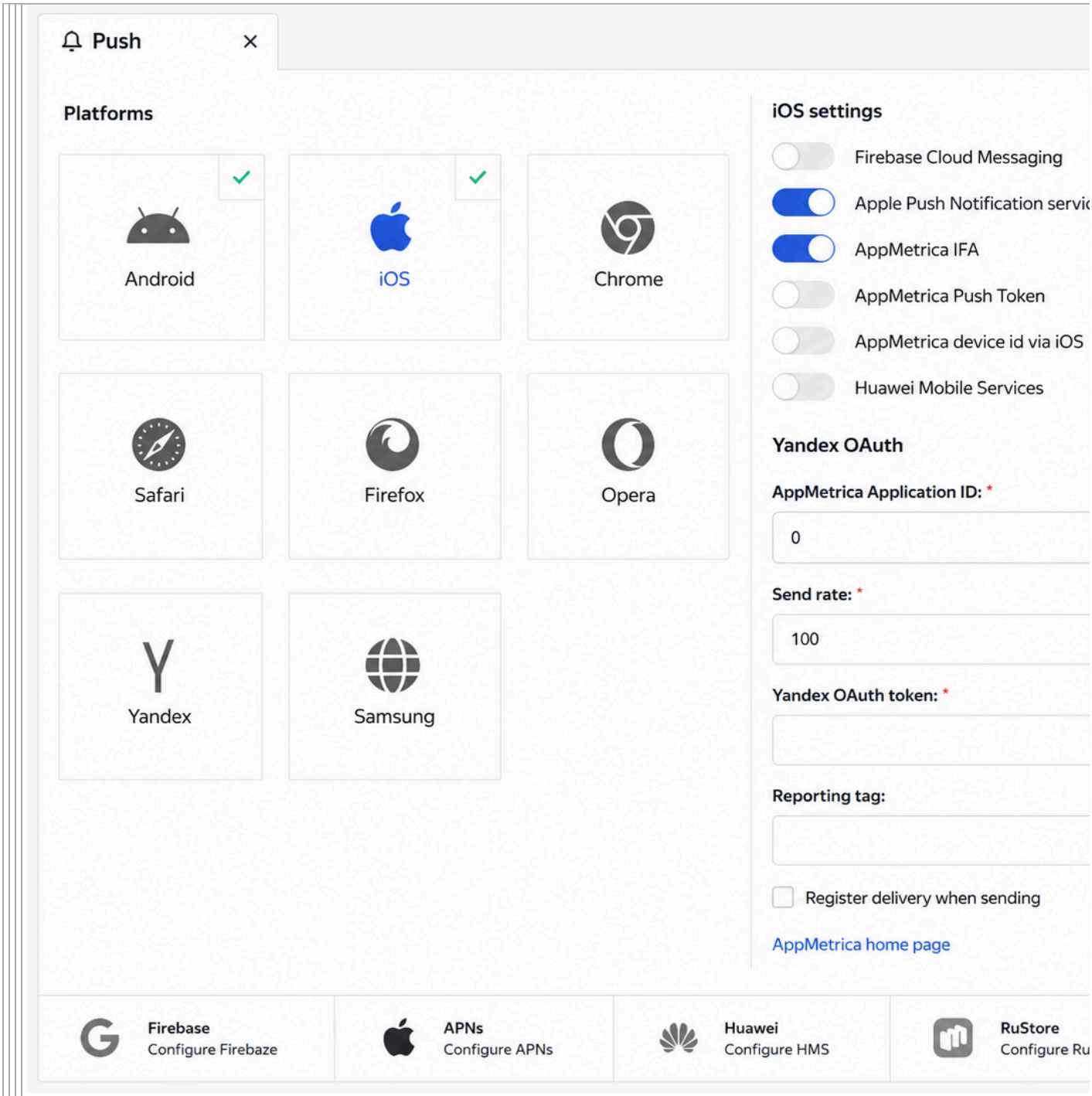
Setting up in Altcraft

You can configure delivery to both operating systems within a single resource.

1. **On the Push tab in the resource, select the AppMetrica platform.**
2. **Fill in the settings:**



- **AppMetrica API token — token for accessing the AppMetrica API**
- **AppMetrica Application ID — app identifier in AppMetrica**
- **Send rate — maximum notification delivery rate per second**
- **Yandex OAuth token — authorization token for Yandex services**
- **Reporting tag — tag for AppMetrica reports**



TIP

AppMetrica subscriptions can be added automatically to app users with the connected analytics servic [with Yandex.AppMetrica] article.

What's next

- **[App integration with Altcraft]**
- **[AppMetrica push message structure]**
- **[Integration with Yandex.AppMetrica] — for importing profiles and events**

Mobile Push: Integrate Your App with Altcraft

To send Mobile Push notifications, you need to integrate Altcraft mSDK into your mobile application. mSI subscription token submission, notification display, and delivery/open event registration.

Quick Start

Select your app platform and follow the mSDK integration guide:

- **Android — Quick Start — add SDK via Gradle, implement JWT interface, initialize and subscribe to pu**
- **iOS — Quick Start — add SDK via Swift Package Manager, configure APNs and Firebase, initialize an**
- **Flutter — Quick Start — install the package, register providers natively, initialize via Dart API.**
- **React Native — Quick Start — install the npm package, configure providers natively, initialize SDK fr**

Additional Resources

- **mSDK Setup — detailed setup instructions for each platform**
- **Role Token and JWT Token — choosing and configuring an authorization type**
- **mSDK Features — full list of SDK capabilities**
- **Deprecated: Manual Push Setup — the old approach with manual subscription management, event re structure without mSDK**

Deprecated: Setup and Connection

> **CAUTION**
>
> **This article contains deprecated instructions. Provider setup is described in individual articles in the M integration is done via Altcraft mSDK.**

Adding push app subscriptions

Push notification subscriptions are assigned to users via an API request from the app: Add a push subsc

> **TIP**
>
> **Subscriptions to Yandex.AppMetrica can be added automatically for users of apps with the metrics ser see Integration with Yandex.AppMetrica.**

Event tracking in app push

Push notifications for apps include `ack`, `open` and `read` links. GET requests to these links create push de events. They will appear in the user history and channel reports.

```
{
    "hub_link": {
        "ack": "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/2ab9162159...",
        "open": "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9...",
        "read": "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9..."
    }
}
```

Event registration in app push

You can add custom events via an API call: Add events from app push.

Processing and adding a subscription

Once the app received a push token, the app must transfer it to Altcraft Platform in order to create a sub it.

With this request, you can:

- **Add new subscriptions.**
- **Change the status of existing subscriptions.**

The profile search mode can be set using the `matching` field in the request (see specification below). If th be created.

Moreover, if several databases are specified in the resource, then all of them will be processed separate create and update profiles in several databases at once. An example of such a case can be seen below.

If there are no databases in the resource, then an error will be returned.

URL

Method: POST

`https://<cookiesaver_host>/push_subscribe`

Request parameters

Parameter	Type	Example	Required	Description
id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**Profile ID**
resource_token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**Unique resource ID**
db_ids	**[ ] int**	**[2,3]**	**No**	**Resource database filter.**
matching	**string**	**“custom”**	**No**	**Profile search mode. If the parameter is not passed or is empty, profile search by ID will be used. If ID is als For the `push_subscribe` method, this parameter can only accept the following values: • custom • profile_id**
subscriptions	**JSON array SubscriptionObject**	`[{ “provider” : “android-firebase”, “subscription_id” : “%subscriptionid%” }]`	**Yes**	**Push subscriptions that should be added to profile or should be updated. “subscription_id” is unique for each device.**

SubscriptionObject

Parameter	Type	Example	Required	
provider	**string**	**“android-firebase”**	**Yes**	**Altcraft Platform Provider ID. The available values: • android-firebase • ios-firebase • ios-apns • yandex-Appmetrica-device-id-ios • yandex-Appmetrica-device-id-android • yandex-Appmetrica-ios-ifa • yandex-Appmetrica-google-aid • yandex-Appmetrica-android-token • yandex-Appmetrica-ios-token • android-huawei • ios-huawei Each token type in AppMetrica has its own pr**
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Yes**	**Subscription push token**
unsubscribe	**bool**	**false**	**No**	**Unsubscribe flag. Dismisses subscription if tr**

Response parameters

Parameter	Type	Description
data	**JSON array DataObject**	**Information about the created subscriptions. The array can contain several objects in the following cases: • Several SubscriptionObject were passed in the request. • The Altcraft Platform resource uses several databases. In this case, a new profile with subscriptions will be added for each database f**
error	**int**	**Error code**
error_text	**string**	**Error text**

DataObject

Parameter	Type	Description	
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Subscription push token**
unsubscribe	**bool**	**false**	**Flag whether the subscription was added or canceled**
profile_id	**string**	**“5fb62e815ae47a4aa04dfadd”**	**ID of the profile (transferred or newly created) for which the subscription v**

Parameter	Type	Description	
provider	**string**	**“android-firebase”**	**Altcraft Platform Provider ID.** **The available values:** • **android-firebase** • **ios-firebase** • **ios-apns** • **yandex-Appmetrica-device-id-ios** • **yandex-Appmetrica-device-id-android** • **yandex-Appmetrica-ios-ifa** • **yandex-Appmetrica-google-aid** • **yandex-Appmetrica-android-token** • **yandex-Appmetrica-ios-token** • **android-huawei** • **ios-huawei** **Each token type in AppMetrica has its own provider ID.**
db_id	**int**	**2**	**Altcraft Platform Database ID. The profile is stored in this database.**
resource_id	**int**	**1**	**Altcraft Platform resource ID.**
subscription_hash_id	**string**	**“92807e89”**	**Subscription hash**
status	**string**	**“error”**	**Operation status** • **ok** • **error**
description	**string**	**“Duplicate push subscription”**	**Error description**

File import

Push tokens can be imported not only via API, but also from a CSV file through the platform interface. Th need to bulk-add already collected tokens without modifying the mobile app or server integration.

When importing, you specify the push notification provider, for example `android-firebase`, `ios-apns`, or anot tokens will be added as push subscriptions to profiles.

Before importing, prepare a CSV file with data for profile matching and subscription tokens. Profiles can such as system profile ID or an additional field.

> **INFO**
>
> **Before importing, verify that the required databases are configured in the resource and the correct sub token is submitted with an incorrect provider, push notifications for that subscription will not be deliver**

Examples

Request with matching by system profile ID

Request

```
{
  "id": "59db33d269d426509e9c4bf9",
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bf9",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request with matching by a custom profile field

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfc",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request with matching by push subscriptions

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfa",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request to a resource with several databases

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response

```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "provider": "android-firebase",
      "resource_id": 22,
      "status": "ok",
      "subscription_hash_id": "340ca49c",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    },
    {
      "db_id": 2,
      "description": "matching error: unknown field",
      "resource_id": 22,
      "status": "error"
    }
  ]
}
```


Request with matching by a custom profile field in a certain database

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "db_ids": [1],
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 1,
      "description": "",
      "profile_id": "637f1a284fda7c5c08be62f",
      "provider": "android-firebase",
      "resource_id": 31,
      "status": "ok",
      "subscription_hash_id": "f8af057de",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    }
  ]
}
```


Request did not update or add any profile subscriptions

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response

```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "resource_id": 22,
      "status": "ok"
    }
  ]
}
```

Event registration

There are two schemes for registering push message delivery and opening events in Altcraft:

1. **Directly from the app.**
2. **Through an intermediate service.**

Registering events directly from the app

By default, when sending a push message, the `hub_link` object is included in the payload:

```
{
    "hub_link" : {
        "ack" :  "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/2ab9162159..." ,
        "open" :  "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9...",
        "read" :  "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9..."
    }
}
```

The app must execute a GET request to the `ack`, `open` and `read` links to register delivery, push opening a
The `read` event is registered when a recipient reads a push notification in the notification center. It is us
notification content is duplicated for deferred viewing.

Registering events through an intermediate service

> **CAUTION**
>
> **This method requires Extra UID in the push message user data (configurable on request)**

The app reports the event to the intermediate service, passing the Extra UID from the push.
The gateway must send this event to Altcraft Platform via API.

URL

Method: POST

`https://example.com/api/v1.1/pushes/post_events`

Request parameters

Parameter	Type	Example	Required	Description
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
events	**int**	**1 — delivered** **2 — opened** **3 — delivered and opened**	**Yes**	**Events**
uid	**string**	**“uid”**	**Yes**	**Comes along with the push message from Altcraft in the Data paylo**

Request example

```
{
    "token" :  "abcdefghijklmnqrstuvwxyz" ,
    "events" : 1,
    "uid" :  "e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns"
}
```

Response example

```
{
    "error" : 0,
    "error_text" :  "Successful operation"
}
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**

Providers: push message structure

Altcraft Platform supports the following services for sending mobile push messages:

- **Google Firebase Cloud Messaging — for Android and iOS apps**
- **Apple Push Notification Service — for iOS apps only**
- **Yandex.AppMetrica — for Android and iOS apps**
- **Huawei Mobile Services — for Android and iOS apps**
- **RuStore — for Android apps**

> NOTE
>
> **Connecting the corresponding SDKs in the app is required.**

> TIP
>
> **Yandex.AppMetrica uses Google Firebase SDK for delivery. You will need to install it in your app.**
> **Integration with Yandex.AppMetrica is available in Altcraft Platform for importing user profiles, register value (cost).**

You need to verify that the app supports the Altcraft Platform push notification structure. If the format is the message structure from its side.

Push message format

Android Firebase

> NOTE
>
> **It's possible to edit the JSON object (on request).**
> **Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.**

```
{
        "message": {
                "token": "{{.SubscriptionID}}",
                "android": {
                        "priority": "{{.Priority}}",
                        "ttl": {{.TTL}},
                        "data": {
                                "_ac_push": "Altcraft",
                                "_uid": "{{.UID}}",
                                "_launch_id": "{{.LaunchID}}",
                                "_title": "{{.Title}}",
                                "_provider": "{{.Provider}}",
                                "_body": "{{.Body}}",
                                "_icon": "{{.Icon}}",
                                "_click_action": "{{.ClickURL}}",
                                "_color": "{{.ImageBackgroundColor}}",
                                "_image": "{{.Image}}",
                                "_vibration": "{{.Vibration}}",
                                "_soundless": "{{.Soundless}}",
                                "_hub_link": "{\"open\": \"{{.URLOpenEvent}}\", \"ack\": \"{{.URLDelivEvent}}\", \"read\": \"{{.URLReadE
                                "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}
                        }
                }
        }
}
```

Field	Description
`token`	**Device push token from FCM. Unique for each device and app.**
`android.priority`	**Delivery priority: `HIGH` or `NORMAL`. Determines whether the device is woken up to receive the message.**
`android.ttl`	**Message lifetime in seconds (TTL — Time To Live). The message is not delivered after expiration. Set in account settings.**
`data._ac_push`	**Send marker. Always `"Altcraft"`. Used by the app to identify pushes from Altcraft Platform.**
`data._uid`	**Unique message ID (SendMessageID). Contains encoded data: account, campaign, resource, channel, profile, subscription, timestamp, and checksum. Ex**
`data._launch_id`	**Send ID within the mailing. Contains accountID, campaignID, and launch timestamp, encoded in base58. Example: `123_456_aBcDeFgHiJkLmNoPqRsTuVwXyZ`. All mes**
`data._title`	**Push notification title. Displayed in the notification shade.**
`data._provider`	**Provider identifier. For this provider: `Android Firebase`.**
`data._body`	**Push notification body. Main message text.**
`data._icon`	**Notification icon URL. Displayed next to the text in the notification shade.**
`data._click_action`	**URL opened on tap. Can be a regular link or a deeplink.**
`data._color`	**Icon background color in hex format (e.g., `#FF5722`).**

Field	Description
`data._image`	**Banner image URL. Displayed as a large image in the expanded notification.**
`data._vibration`	**Forced vibration flag: `true` or `false`.**
`data._soundless`	**Silent notification flag: `true` or `false`. When `true`, sound is disabled.**
`data._hub_link`	**JSON object with tracking URLs: `open` — URL to register the open event, `ack` — URL to register the delivery event, `read` — URL to register the read event**
`data._buttons`	**JSON array of action buttons. Each button contains `label` (text) and `link` (URL). Generated from buttons added in message settings.**

TIP

`_uid` — unique message ID (SendMessageID). Contains encoded data about account, campaign, resourc and timestamp. Each push has its own unique `_uid`.

`_launch_id` — send ID within the mailing. All messages in the same mailing share the same `launch_id`. Co launch timestamp.

iOS Firebase

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "message": {
                "token": "{{.SubscriptionID}}",
                "apns": {
                        "headers": {
                                "apns-expiration": "{{.TTL}}"
                        },
                        "payload": {
                                "aps": {
                                        "category": "Altcraft",
                                        "sound": {{.Sound}},
                                        "mutable-content": 1,
                                        "alert": {
                                                "title": "{{.Title}}",
                                                "subtitle": "{{.SubTitle}}",
                                                "body": "{{.Body}}"
                                        }
                                },
                                "_uid": "{{.UID}}",
                                "_provider": "{{.Provider}}",
                                "_launch_id": "{{.LaunchID}}",
                                "_ac_push": "Altcraft",
                                "_click-url": "{{.ClickURL}}",
                                "_media": "{{.Media}}",
                                "_soundless": "{{.Soundless}}",
                                "_hub_link": "{\"open\": \"{{.URLOpenEvent}}\", \"ack\": \"{{.URLDelivEvent}}\", \"read\": \"{{.URLReadE
                                "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}
                        }
                }
        }
}
```

Field	Description
`token`	**Device push token from APNs via FCM. Unique for each device and app.**
`apns.headers.apns-expiration`	**Message lifetime in seconds. APNs does not deliver the message after expiration.**
`apns.payload.aps.category`	**Notification category. Always `"Altcraft"`. Used to determine available actions on tap.**
`apns.payload.aps.sound`	**Notification sound. Default: `"default"`. Empty string in silent mode. Can contain a sound file name from the app.**
`apns.payload.aps.mutable-content`	**Always `1`. Enables Content-Extension for notification modification (e.g., image download).**
`apns.payload.aps.alert.title`	**Push notification title.**
`apns.payload.aps.alert.subtitle`	**Notification subtitle. Displayed below the title.**
`apns.payload.aps.alert.body`	**Push notification body. Main message text.**
`apns.payload._uid`	**Unique message ID (SendMessageID). Same as `_uid` in Android.**
`apns.payload._provider`	**Provider identifier. For this provider: `iOS Firebase`.**
`apns.payload._launch_id`	**Send ID within the mailing.**
`apns.payload._ac_push`	**Send marker. Always `"Altcraft"`.**
`apns.payload._click-url`	**URL opened on tap.**
`apns.payload._media`	**Image URL for rich notification.**
`apns.payload._soundless`	**Silent notification flag: `true` or `false`.**
`apns.payload._hub_link`	**JSON object with tracking URLs: `open` (open), `ack` (delivery) and `read` (read).**
`apns.payload._buttons`	**JSON array of action buttons with `label` and `link`.**

TIP

`_uid` — unique message ID (SendMessageID). `_launch_id` — send ID within the mailing.

iOS APNS

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "aps": {
                "category": "Altcraft",
                "mutable-content": 1,
                "sound": {{.Sound}},
                "alert": {
                        "title": "{{.Title}}",
                        "subtitle": "{{.SubTitle}}",
                        "body": "{{.Body}}"
                }
        },
        "_ac_push": "Altcraft",
        "_click-url": "{{.ClickURL}}",
        "_soundless": "{{.Soundless}}",
        "_media": "{{.Media}}",
        "_uid": "{{.UID}}",
        "_provider": "{{.Provider}}",
        "_hub_link": {
                "open": "{{.URLOpenEvent}}",
                "ack": "{{.URLDelivEvent}}",
                "read": "{{.URLReadEvent}}"
        },
        "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}\",\"link\":\"{{$button.Link
}
```

Field	Description
`aps.category`	**Notification category. Always `"Altcraft"`.**
`aps.mutable-content`	**Always `1`. Enables Content-Extension.**
`aps.sound`	**Notification sound. Default: `"default"`.**
`aps.alert.title`	**Notification title.**
`aps.alert.subtitle`	**Notification subtitle.**
`aps.alert.body`	**Notification body.**
`_ac_push`	**Send marker. Always `"Altcraft"`.**
`_click-url`	**URL to open on tap.**
`_soundless`	**Silent notification flag.**
`_media`	**Image URL for rich notification.**
`_uid`	**Unique message ID (SendMessageID).**
`_provider`	**Provider identifier: `iOS APNS`.**
`_hub_link.open`	**URL to register the open event.**
`_hub_link.ack`	**URL to register the delivery event.**
`_hub_link.read`	**URL to register the read event.**
`_buttons`	**JSON array of action buttons.**

AppMetrica iOS

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "push_batch_request": {
                "group_id": 97737,
                "tag": "tag",
                "batch": [
                        {
                                "messages": {
                                        "iOS": {
                                                "silent": true/false,
                                                "content": {
                                                        "title": "Title",
                                                        "text": "Body",
                                                        "badge": 0-999,
                                                        "expiration": 0,
                                                        "data": "{\"hub_link\":
{\"ack\":\"URLDelivEvent\",\"open\":\"URLOpenEvent\",\"read\":\"URLReadEvent\"},\"icon\":\"Icon\",\"media\":\"Media\"}"
                                                },
                                                "open_action": {
                                                        "url": "URL",
                                                        "deeplink": "Deeplink"
                                                }
                                        }
                                },
                                "devices": [
                                        {
                                                "id_values": ["SubscriptionID"],
                                                "id_type": "ios_push_token"
                                        }
                                ]
                        }
                ]
        }
}
```

Field	Description
`push_batch_request.group_id`	**Group ID in AppMetrica. Set when configuring the channel.**
`push_batch_request.tag`	**Mailing tag. Used to group sends in AppMetrica.**
`push_batch_request.batch`	**Array of messages to send.**
`messages.iOS.silent`	**Silent push flag: `true` — background delivery without notification, `false` — standard notification.**
`messages.iOS.content.title`	**Notification title.**
`messages.iOS.content.text`	**Notification body.**
`messages.iOS.content.badge`	**App badge counter. Range: 0–999.**
`messages.iOS.content.expiration`	**Message lifetime in seconds. `0` — message is not stored if the device is unavailable.**
`messages.iOS.content.data`	**JSON string with additional data: `hub_link` (tracking), `icon` (icon), `media` (image).**
`messages.iOS.open_action.url`	**URL to open on tap (regular link).**
`messages.iOS.open_action.deeplink`	**Deeplink to open on tap.**
`devices.id_values`	**Array of device push tokens.**
`devices.id_type`	**Identifier type: `ios_push_token` for iOS.**

NOTE

In AppMetrica iOS, `_uid` and `_launch_id` fields are not passed directly in the message structure. Trackin the `data` field.

AppMetrica Android

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "push_batch_request": {
                "group_id": 97737,
                "tag": "tag",
                "batch": [
                        {
                                "messages": {
                                        "android": {
                                                "silent": true/false,
                                                "content": {
                                                        "title": "Title",
                                                        "text": "Body",
                                                        "time_to_live": 0,
                                                        "image": "Icon",
                                                        "icon_background": "IconBackground",
                                                        "banner": "Banner",
                                                        "led_color": "LEDColor",
                                                        "data": "{\"hub_link\":{\"ack\":\"URLDelivEvent\",\"open\":\"URLOpenEvent\",
                                                },
                                                "open_action": {
                                                        "url": "URL",
                                                        "deeplink": "Deeplink"
                                                }
                                        }
                                },
                                "devices": [
                                        {
                                                "id_values": ["SubscriptionID"],
                                                "id_type": "google_aid"
                                        }
                                ]
                        }
                ]
        }
}
```

Field	Description
`push_batch_request.group_id`	**Group ID in AppMetrica.**
`push_batch_request.tag`	**Mailing tag.**
`push_batch_request.batch`	**Array of messages to send.**
`messages.android.silent`	**Silent push flag.**
`messages.android.content.title`	**Notification title.**
`messages.android.content.text`	**Notification body.**
`messages.android.content.time_to_live`	**Message lifetime in seconds.**
`messages.android.content.image`	**Notification icon URL.**
`messages.android.content.icon_background`	**Icon background color in hex format.**
`messages.android.content.banner`	**Banner image URL.**
`messages.android.content.led_color`	**LED indicator color in hex format.**
`messages.android.content.data`	**JSON string with tracking data: `hub_link` contains URLs for `ack` (delivery), `open` (open) and `read` (read) events.**
`messages.android.open_action.url`	**URL to open on tap.**
`messages.android.open_action.deeplink`	**Deeplink to open on tap.**
`devices.id_values`	**Array of device identifiers (Google AID).**
`devices.id_type`	**Identifier type: `google_aid` for Android.**

NOTE

**In AppMetrica Android, `_uid` and `_launch_id` fields are not passed directly in the message structure. Tra
inside the `data` field.**

HMS Android

NOTE

It’s possible to edit the JSON object (on request).

**Adding additional data to the `data` object is supported. This can be done for each sending individually i
settings.**

```
{
        "message": {
                "token": ["{{.SubscriptionID}}"],
                "data": {
                        "_ac_push": "Altcraft",
                        "_title": "{{.Title}}",
                        "_uid": "{{.UID}}",
                        "_provider": "{{.Provider}}",
                        "_body": "{{.Body}}",
                        "_launch_id": "{{.LaunchID}}",
                        "_vibration": "{{.Vibration}}",
                        "_soundless": "{{.Soundless}}",
                        "_hub_link": "{\"open\": \"{{.URLOpenEvent}}\", \"ack\": \"{{.URLDelivEvent}}\", \"read\": \"{{.URLReadEvent}}\"}
                        "_icon": "{{.Icon}}",
                        "_click_action": "{{.ClickURL}}",
                        "_color": "{{.ImageBackgroundColor}}",
                        "_image": "{{.Image}}",
                        "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}\",\"link
                },
                "android": {
                        "urgency": "HIGH",
                        "ttl": {{.TTL}}
                }
        }
}
```

Field	Description
`message.token`	**Array of HMS device push tokens. Unlike Firebase, the token is passed as an array.**
`data._ac_push`	**Send marker. Always `"Altcraft"`.**
`data._title`	**Notification title.**
`data._uid`	**Unique message ID (SendMessageID).**
`data._provider`	**Provider identifier: `Android HMS`.**
`data._body`	**Notification body.**
`data._launch_id`	**Send ID within the mailing.**
`data._vibration`	**Forced vibration flag.**
`data._soundless`	**Silent notification flag.**
`data._hub_link`	**JSON object with tracking URLs: `open`, `ack` and `read`.**
`data._icon`	**Notification icon URL.**
`data._click_action`	**URL to open on tap.**
`data._color`	**Icon background color in hex format.**
`data._image`	**Banner image URL.**
`data._buttons`	**JSON array of action buttons.**
`android.urgency`	**Delivery urgency. Always `HIGH`.**
`android.ttl`	**Message lifetime in seconds.**

HMS iOS

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "message": {
                "token": ["{{.SubscriptionID}}"],
                "apns": {
                        "headers": {
                                "apns-expiration": "{{.TTL}}"
                        },
                        "payload": {
                                "aps": {
                                        "category": "Altcraft",
                                        "mutable-content": 1,
                                        "sound": {{.Sound}},
                                        "alert": {
                                                "title": "{{.Title}}",
                                                "subtitle": "{{.SubTitle}}",
                                                "body": "{{.Body}}"
                                        }
                                },
                                "_ac_push": "Altcraft",
                                "_provider": "{{.Provider}}",
                                "_click-url": "{{.ClickURL}}",
                                "_soundless": "{{.Soundless}}",
                                "_uid": "{{.UID}}",
                                "_media": "{{.Media}}",
                                "_hub_link": {
                                        "open": "{{.URLOpenEvent}}",
                                        "ack": "{{.URLDelivEvent}}",
                                        "read": "{{.URLReadEvent}}"
                                },
                                "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}
                        },
                        "hms_options": {
                                "target_user_type": 1
                        }
                }
        }
}
```

Field	Description
`message.token`	**Array of HMS device push tokens.**
`apns.headers.apns-expiration`	**Message lifetime in seconds.**
`apns.payload.aps.category`	**Notification category. Always `"Altcraft"`.**
`apns.payload.aps.mutable-content`	**Always `1`. Enables Content-Extension.**
`apns.payload.aps.sound`	**Notification sound.**
`apns.payload.aps.alert.title`	**Notification title.**
`apns.payload.aps.alert.subtitle`	**Notification subtitle.**
`apns.payload.aps.alert.body`	**Notification body.**
`apns.payload._ac_push`	**Send marker.**
`apns.payload._provider`	**Provider identifier: `iOS HMS`.**
`apns.payload._click-url`	**URL to open on tap.**
`apns.payload._soundless`	**Silent notification flag.**
`apns.payload._uid`	**Unique message ID (SendMessageID).**
`apns.payload._media`	**Image URL for rich notification.**
`apns.payload._hub_link.open`	**URL to register the open event.**
`apns.payload._hub_link.ack`	**URL to register the delivery event.**
`apns.payload._hub_link.read`	**URL to register the read event.**
`apns.payload._buttons`	**JSON array of action buttons.**
`apns.hms_options.target_user_type`	**Target user type in HMS. Always `1`.**

RuStore Android

NOTE

It’s possible to edit the JSON object (on request).

Adding additional data to the `data` object is supported. This can be done for each sending individually i settings.

```
{
        "message": {
                "token": "{{.SubscriptionID}}",
                "data": {
                        "_ac_push": "Altcraft",
                        "_title": "{{.Title}}",
                        "_uid": "{{.UID}}",
                        "_provider": "{{.Provider}}",
                        "_body": "{{.Body}}",
                        "_icon": "{{.Icon}}",
                        "_image": "{{.Image}}",
                        "_launch_id": "{{.LaunchID}}",
                        "_vibration": "{{.Vibration}}",
                        "_soundless": "{{.Soundless}}",
                        "_color": "{{.ImageBackgroundColor}}",
                        "_click_action": "{{.ClickURL}}",
                        "_hub_link": "{\"open\": \"{{.URLOpenEvent}}\", \"ack\": \"{{.URLDelivEvent}}\", \"read\": \"{{.URLReadEvent}}\"}
                        "_buttons": "[{{ range $index, $button := .Actions }}{{if $index}},{{end}}{\"label\":\"{{$button.Label}}\",\"link
                },
                "android": {
                        "priority": "HIGH",
                        "ttl": {{.TTL}}
                }
        }
}
```

Field	Description
`message.token`	**RuStore device push token.**
`data._ac_push`	**Send marker. Always `"Altcraft"`.**
`data._title`	**Notification title.**
`data._uid`	**Unique message ID (SendMessageID).**
`data._provider`	**Provider identifier: `Android RuStore`.**
`data._body`	**Notification body.**
`data._icon`	**Notification icon URL.**
`data._image`	**Banner image URL.**
`data._launch_id`	**Send ID within the mailing.**
`data._vibration`	**Forced vibration flag.**
`data._soundless`	**Silent notification flag.**
`data._color`	**Icon background color in hex format.**
`data._click_action`	**URL to open on tap.**
`data._hub_link`	**JSON object with tracking URLs: `open`, `ack` and `read`.**
`data._buttons`	**JSON array of action buttons.**
`android.priority`	**Delivery priority. Always `HIGH`.**
`android.ttl`	**Message lifetime in seconds.**

TIP

`_uid` — unique message ID (SendMessageID). `_launch_id` — send ID within the mailing.

Common fields across all providers

The fields below are present in Altcraft Platform push message structures regardless of the provider.

Identifiers

Field	Description
`_uid`	**SendMessageID — unique identifier for each sent message. Contains encoded data: account, campaign, resource, channel, profile, subscription IDs, send timesta (Unique), `E` (Email). Example: `T2k3mN8vR4pQ1xY7zL5wJ9hF6dA0bC`.**
`_launch_id`	**Send ID within the mailing. All messages in the same mailing share the same `launch_id`. Contains accountID, campaignID, and launch timestamp, encoded in base Example: `123_456_aBcDeFgHiJkLmNoPqRsTuVwXyZ`.**
`_ac_push`	**Send marker from Altcraft Platform. Always `"Altcraft"`. Used by the app to identify push notifications from the platform.**
`_provider`	**Provider identifier through which the message was sent. Examples: `Android Firebase`, `iOS Firebase`, `iOS APNS`, `Android HMS`, `Android RuStore`.**

Notification content

Field	Description
`_title` / `title`	**Push notification title. Displayed in the device notification shade.**
`_body` / `text` / `body`	**Push notification body. Main message text.**
`_subtitle` / `subtitle`	**Notification subtitle. iOS only. Displayed below the title.**
`_icon` / `image`	**Notification icon URL. Displayed next to the text in the notification shade.**
`_image` / `banner` / `_media`	**Banner image URL. Displayed as a large image in the expanded notification.**
`_click_action` / `_click-url` / `url`	**URL opened on tap. Can be a regular link or a deeplink.**
`_color` / `icon_background`	**Icon background color in hex format (e.g., `#FF5722`).**

Delivery control

Field	Description
`ttl` / `apns-expiration` / `time_to_live` / `expiration`	**Message lifetime in seconds (TTL — Time To Live). The message is not delivered after expiration. Set in acc**
`priority` / `urgency`	**Delivery priority: `HIGH` or `NORMAL`. Determines whether the device is woken up to receive the message.**
`_soundless` / `silent`	**Silent notification flag: `true` — sound disabled, `false` — default sound.**
`_vibration`	**Forced vibration flag (Android): `true` or `false`.**
`sound`	**Notification sound (iOS). Default: `"default"`. Can contain a sound file name from the app.**

Event tracking

Field	Description
`_hub_link`	**JSON object with encrypted URLs for event tracking: `open` (open), `ack` (delivery), `read` (read). Each URL is unique for a specific message and subscripti**
`_hub_link.open`	**URL to register the open event. Fires when the user taps the notification.**
`_hub_link.ack`	**URL to register the delivery event. Fires when the push service confirms delivery to the device.**
`_hub_link.read`	**URL to register the read event. Fires when the recipient reads the notification in the notification center.**

Action buttons

Field	Description
`_buttons`	**JSON array of action buttons. Each button contains `label` (button text) and `link` (URL opened on tap). Generated from buttons added in Altcraft Platform messa**

Additional data

Field	Description
`ak_uid`	**Additional field added to the `data` object alongside `_uid`. Contains the same SendMessageID. Used for compatibility with some SDKs.**
`category`	**Notification category (iOS). Always `"Altcraft"`. Used to determine available actions on tap.**
`mutable-content`	**iOS flag. Always `1`. Enables Content-Extension for notification modification (e.g., image download).**

Differences between Legacy and v1 formats

Altcraft Platform supports two formats for sending via Firebase:

Legacy API (`https://fcm.googleapis.com/fcm/send`):

- **Uses the deprecated FCM endpoint.**
- **Parameters are passed at the root level of the JSON object.**
- **Custom data is added to the `data` field at the top level.**

API v1 (`https://fcm.googleapis.com/v1/projects/{project_id}/messages:send`):

- **Modern FCM endpoint.**
- **Message structure is wrapped in a `message` object.**
- **Custom data is added to `message.data`.**
- **Uses JWT authentication instead of server key.**
- **Supports batch sending and multicast.**

The current version of Altcraft Platform uses API v1 for all Firebase sends.

How to add custom data to the `data` object

Altcraft Platform supports adding user-defined fields to the `data` object of a push message. Custom data merged with system fields.

Processing order:

1. **Base custom data from `push.CustomJSON` (global for the message).**
2. **Platform-specific custom data from `push.Android.CustomJSON` or `push.IOS.CustomJSON` (overrides base data).**
3. **Altcraft system fields are added independently.**

Example of custom data:

```
{
  "custom_field_1": "value1",
  "custom_field_2": 42,
  "custom_object": "{\"nested_key\": \"nested_value\"}"
}
```

NOTE

Custom data is added to the `data` object alongside system fields. Field names should not conflict with s `_ac_push`, etc.).

To add custom data, contact Altcraft Platform support.

Web Push

> **TIP**
>
> **Setting up mobile push notifications is described in the corresponding article.**

![img](/img/attachments/web-push/37847523.png)	**Firebase Cloud Messaging**	**Chrome, Firefox, Opera, Yandex Brov Internet**
Mozilla Services	**Firefox**	**Direct sending to Firefox without Fir**
Safari / APNs	**Safari**	**Safari browser on macOS**

> **INFO**
>
> **Sending to Firefox is possible both directly via Mozilla Services and through Firebase Cloud Messaging.**

Rich Push

Rich Push allows adding extended elements to a notification: banners, buttons, adaptive links, and deepl a template.

For more details, see the article Push Templates.

Quick Start

1. **Resource and website setup — preliminary settings, resource, SDK, subscription code**
2. **Provider setup — selecting providers for browsers**
3. **Subscriber data transfer — profile and event import**

Additional Resources

- **Migration to Altcraft — transferring subscriptions from other services**
- **PWA Push — push notifications for PWA applications**
- **First Web Push Mailing — step-by-step guide**
- **SDK Methods Reference — Web Push SDK methods**

First Web Push Mailing

![img] (/img/attachments/37849016/37849018.png)	**Web push** notifications (also known as browser push notifications) are small pop-ups in the browser. These messages are cc engage, and retain website visitors. You can send such notifications to your clients' desktop and mobile browsers. When a user subscribes to Web push notifications, every time you send a push notification it is displayed on the user's device at the top of the screen. Moreover, it is displayed even if the window is inactive. </td> </tr>

This tutorial will help you to create and launch your Push mailing in the Altcraft Platform from scratch.

TIP

Push notifications are much more profitable than SMS, because sending them practically costs nothing.

Step 1: Configure a Domain to Work with Push

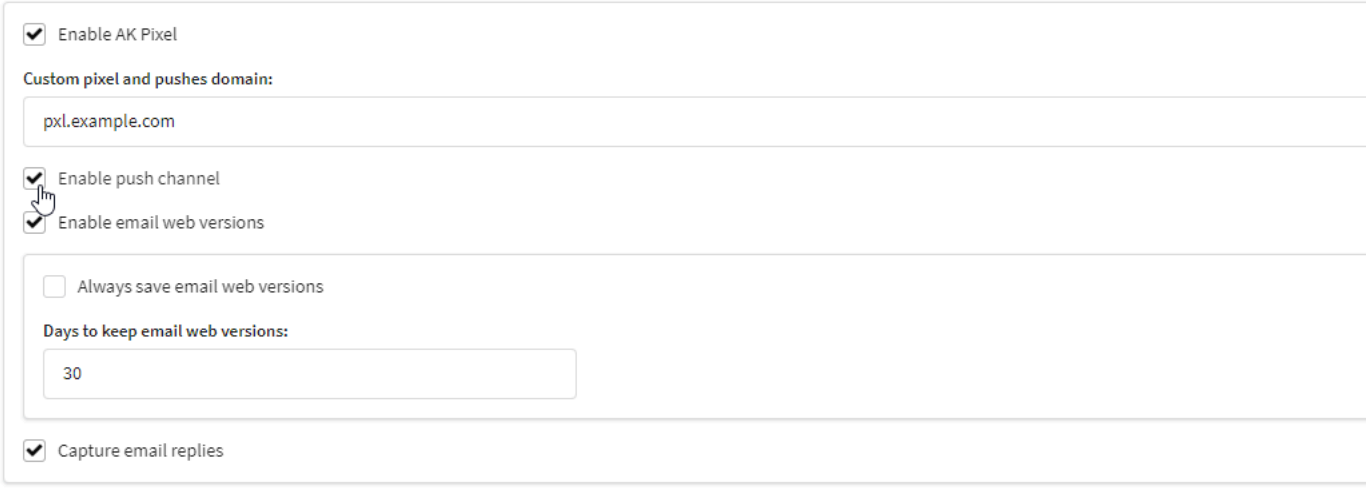
CAUTION

If you use an Altcraft cloud account, please give the information to the support team at support@altcraf the integration.

If the Altcraft platform is installed on your equipment (On-premises), contact your platform administrato team!

Make sure your website is running over HTTPS. In addition, web pixels must be configured in the Altcraft

Go to Administration Panel — Account Management.



Step 2: Subscribe customers to the Push channel

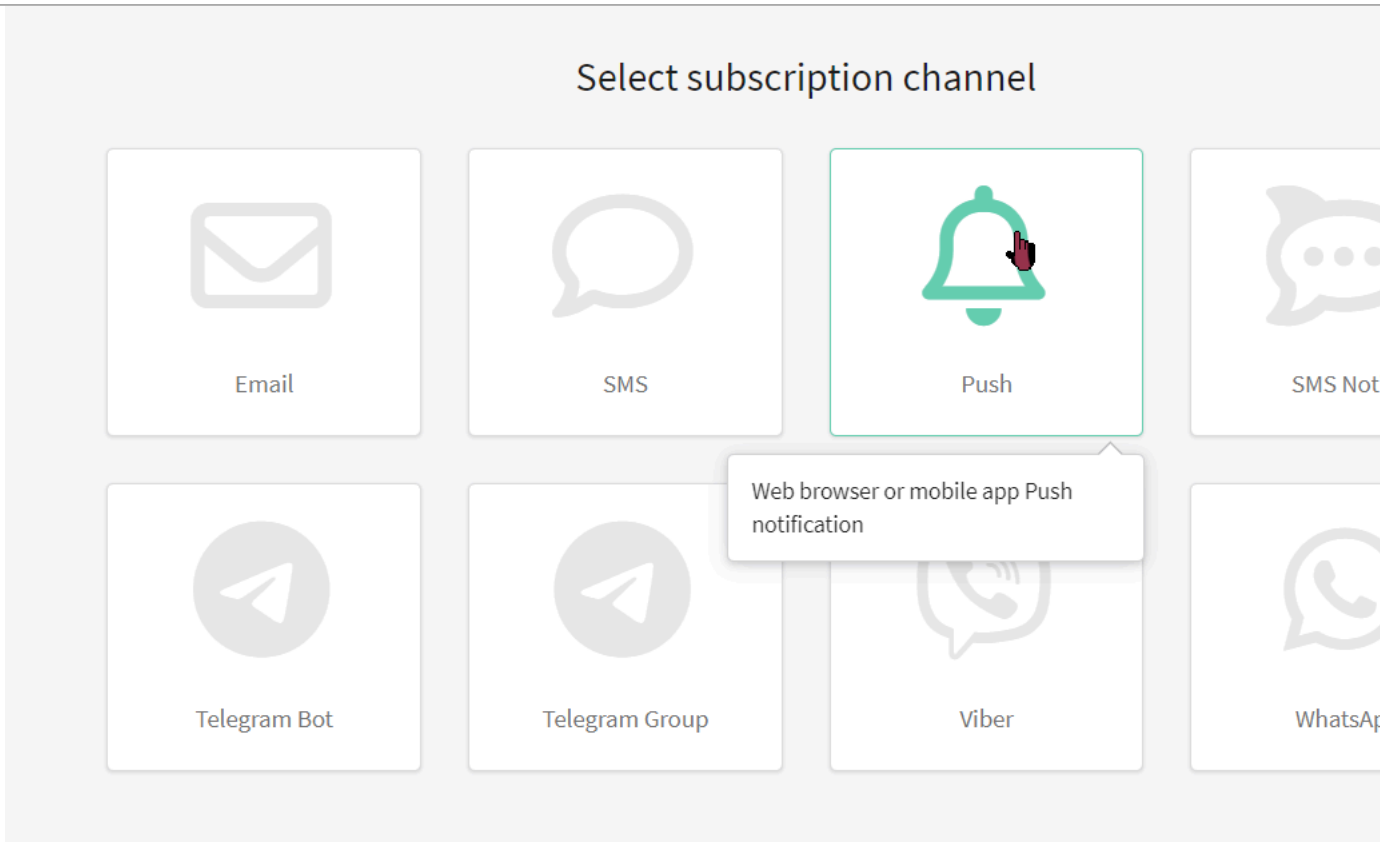
Profile databases and subscription resources

Altcraft mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

When you start your Push mailing, you use profile subscriptions to resources. This way, you can send not different browsers. Each client profile can be subscribed to one or more resources.

What can be a resource? A specific marketing area, a site, an application, or the type of content.

Select the section “Resources” in the main menu, then create a new resource or select an existing one. resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and spec domain will process your customers' clicks on links in messages:

General settings

Resource name: *

Promo messaging

Resource description:

Set group:

subaccount × Sales and promotion ×

Tags:

Set resource sending policy:

Undefined

Databases: *

12 - 09 database × 1 - Database ×

Website, form or application URL: *

https://examplesource.com

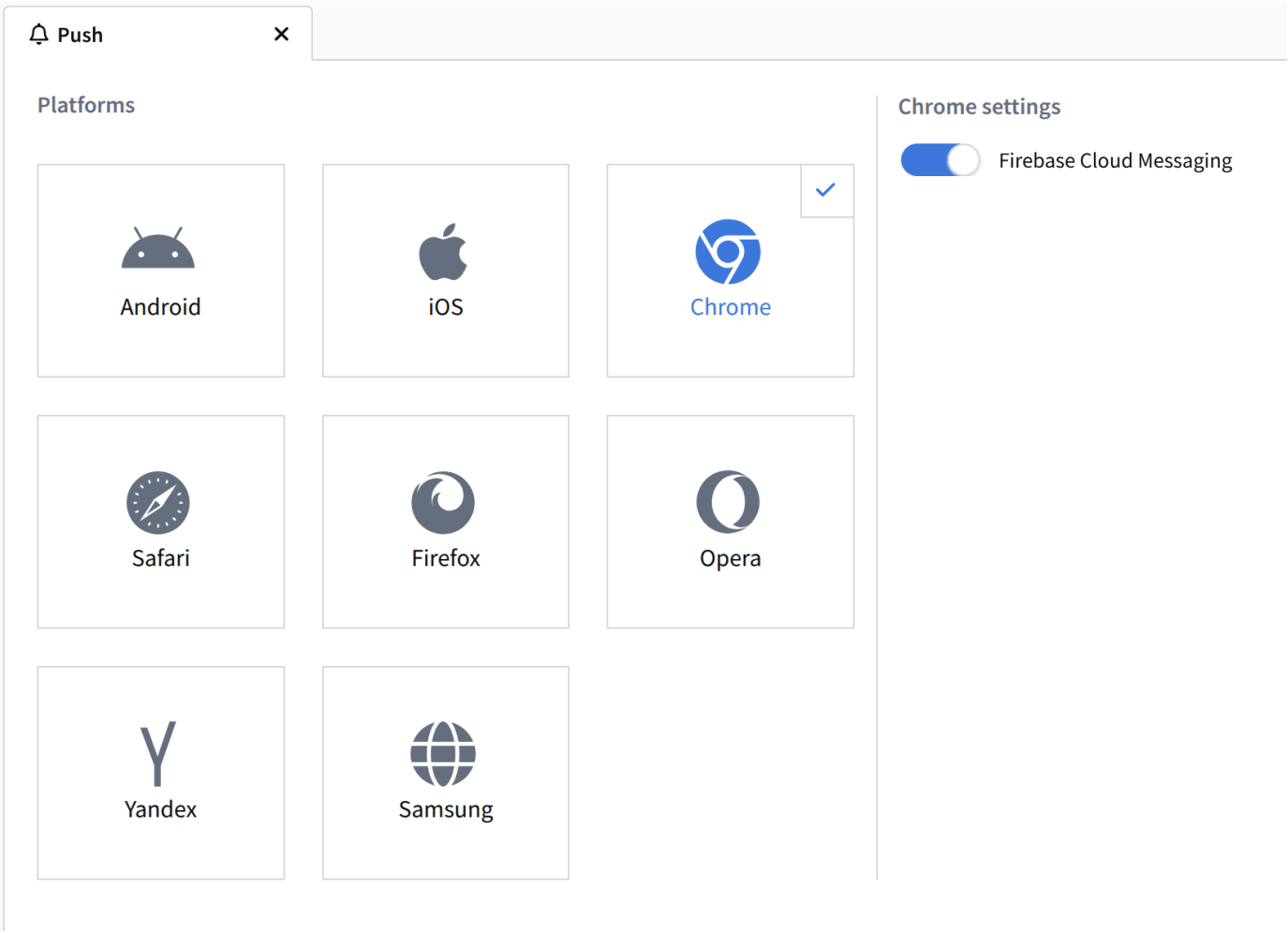
Default tracking domain: *

trk.example.com

Additional URLs

Increase priority for new subscriptions

In Push settings, select the browsers to which you are going to send notifications to clients and set then



INFO

Altcraft uses different push platforms for the supported browsers:

Service	Which browsers use this service to send Web push notifications?
Google Firebase	**Google Chrome, Mozilla Firefox, Opera, Yandex.Browser, Samsung Internet Browser**
Apple Push Notification Service	**Apple Safari**
Mozilla Services	**Mozilla Firefox**

Please note that sending notifications via Mozilla Firefox is possible both directly - Mozilla Services (SD through the Google Firebase service.

Don't know how to configure each platform? Take a look at this tutorial .

After selecting browsers, configure the corresponding providers — without this the Toolkit will not be ge not work. Detailed instructions are in the providers section. After configuring all needed providers, save

Customer profiles and subscription to the resource

So, we've just configured the profiles database and subscription resource. Now it's time to import custor client, Altcraft creates a single digital profile that includes personal data, history of actions and subscri

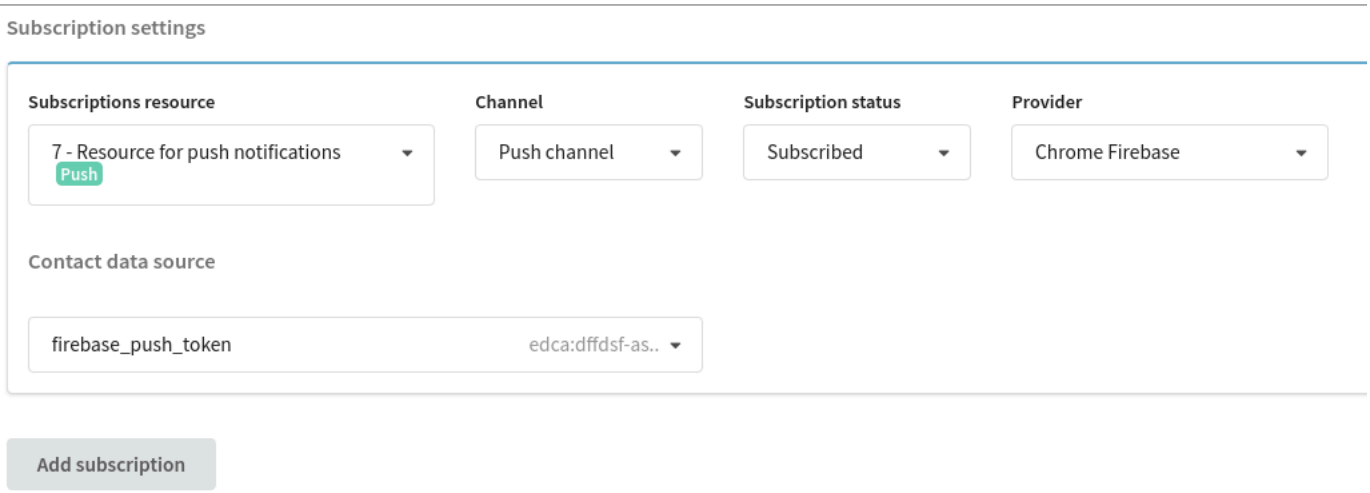
Existing subscriber database

If you already have the database full of tokens for your customers, you can upload customer data to the at a time, in a list, or from a table file. Different import methods are described in this article.

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. To do this, in the Subscription se communication channel, select a notification provider, in the Contact data source field, specify the colur



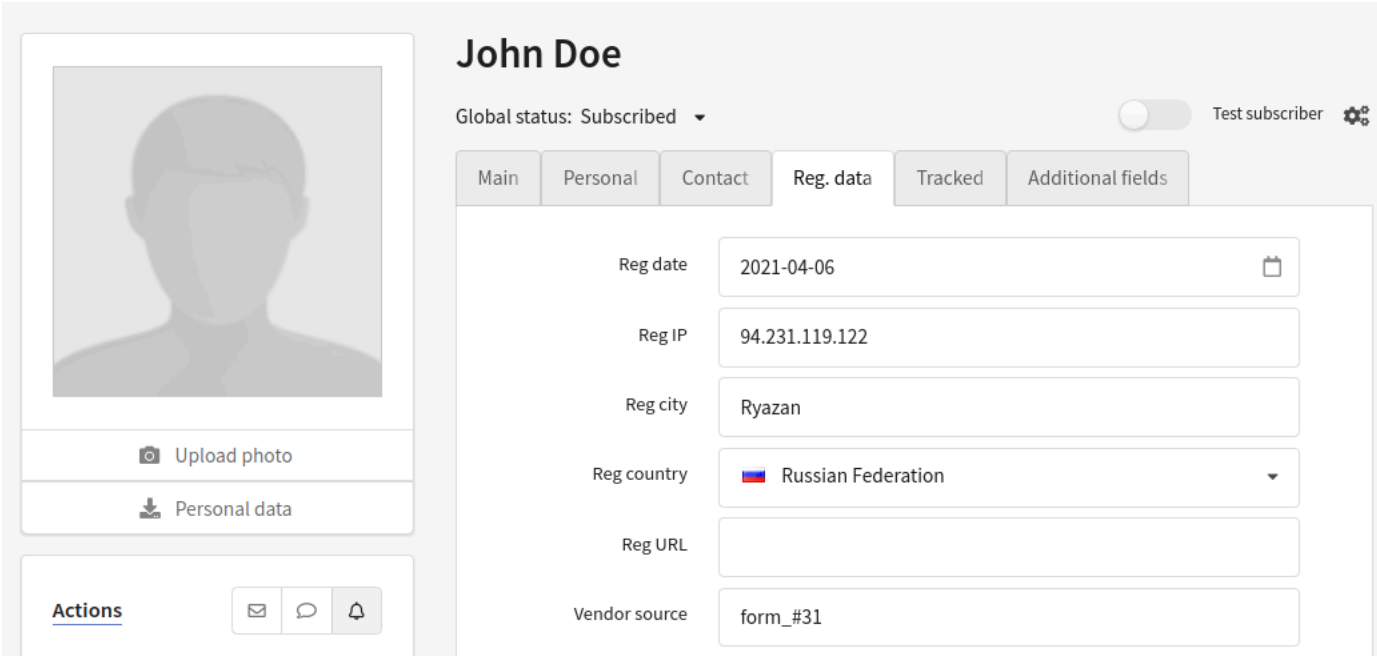
TIP

If you already have a profiles database, add a subscription to a new resource for them. There are two o

- **if customers have subscriptions to another resource with up-to-date email addresses — use bulk pr**
- **if you need to upload to the platform SMS phones for subscription, use the profile import in the "upd**

New push subscribers of your website

Altcraft imports the profiles of your website visitors when they allow push notifications. At the same tim device, IP address and geolocation determined by it are recorded in the profile:



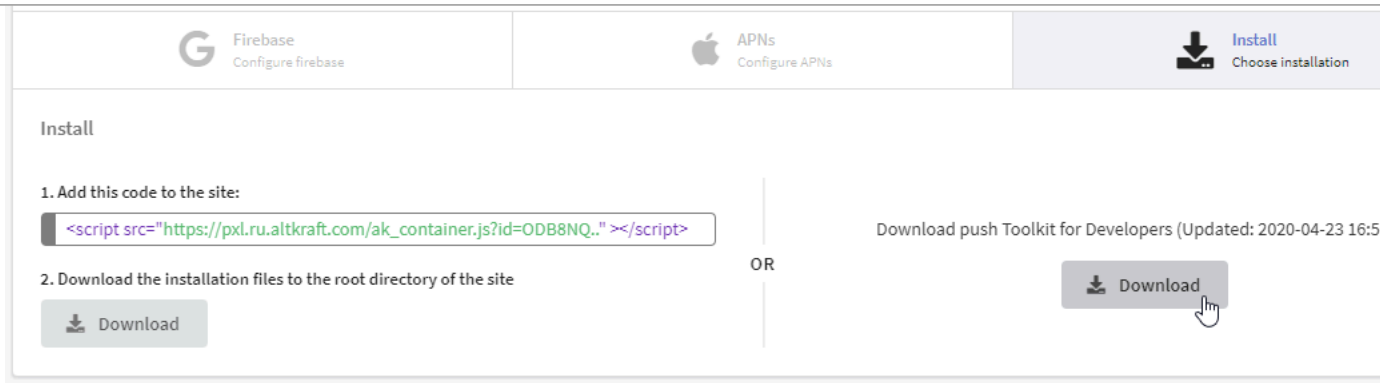
Go to the next step to configure the import of new push subscribers.

Step 3: Add Push Toolkit to your website

After you configure push notification channels, save the resource. This way, you will get access to push the toolkit on the tab "Install".

TIP

Download it and hand over the toolkit to the developers of your website. They'll have to add it to certain the link to this article or the article on Web Push configuration.



There are two ways to install the necessary files on your website:

1. Add the code to the site & Upload the installation files to the root directory

If you're going to use this method, read the instructions in the left side. Add the code to the site, save ch files to the root directory of the site:

- ****manifest.json** — project information for Google Chrome;**
- ****service-worker.js** — service worker, a users' browser extension to receive push notifications.**

Here `service-worker.js` file is dynamic. It is automatically updated every time you make changes to the res

```
importScripts(
  "https://pxl.polina-zakaryan.dev.altkraft.com/service-worker.js?id=MXw3"
);
```

2. Download push Toolkit for Developers

If you're going to use this method, read the instructions in the right side. Click the "Download" button. Tl .zip file. Unpack the archive and put Toolkit files to the root directory of your site.

This toolkit consists of four files:

- **README.txt — installation instructions for the Toolkit**
- **ak-push.js — web push opt in script for your website**
- **manifest.json — project information for Google Chrome**
- **service-worker.js — service worker, a user's browser extension to receive push notifications**

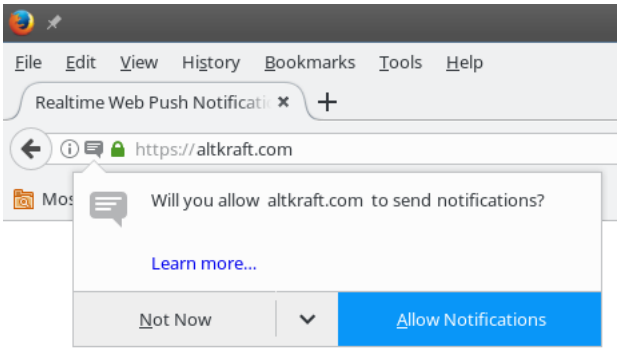
Here `service-worker.js` and `ak-push.js` files are static. It means that every time you change the resource s Toolkit and update the files on the site.

When the Toolkit is uploaded to the site, add the code, that will be responsible for showing a window for

Example for the Subscribe button:

```
<button id="init_sub">Subscribe to updates!</button> // The button

<script>
        document.getElementById('init_sub').addEventListener('click', function() { // On button click..
        try {
                const akPush = new AKPush();
                akPush.initSubscription() // ..show the subscription form
                }
        catch (e) {
                console.log(e); // ..or log the error to the browser console
                }
        });
</script>
```



Once everything is set up, you can go to the site and subscribe to push notifications. When the client cor worker is installed in their browser. The service worker can display messages even when the browser wi

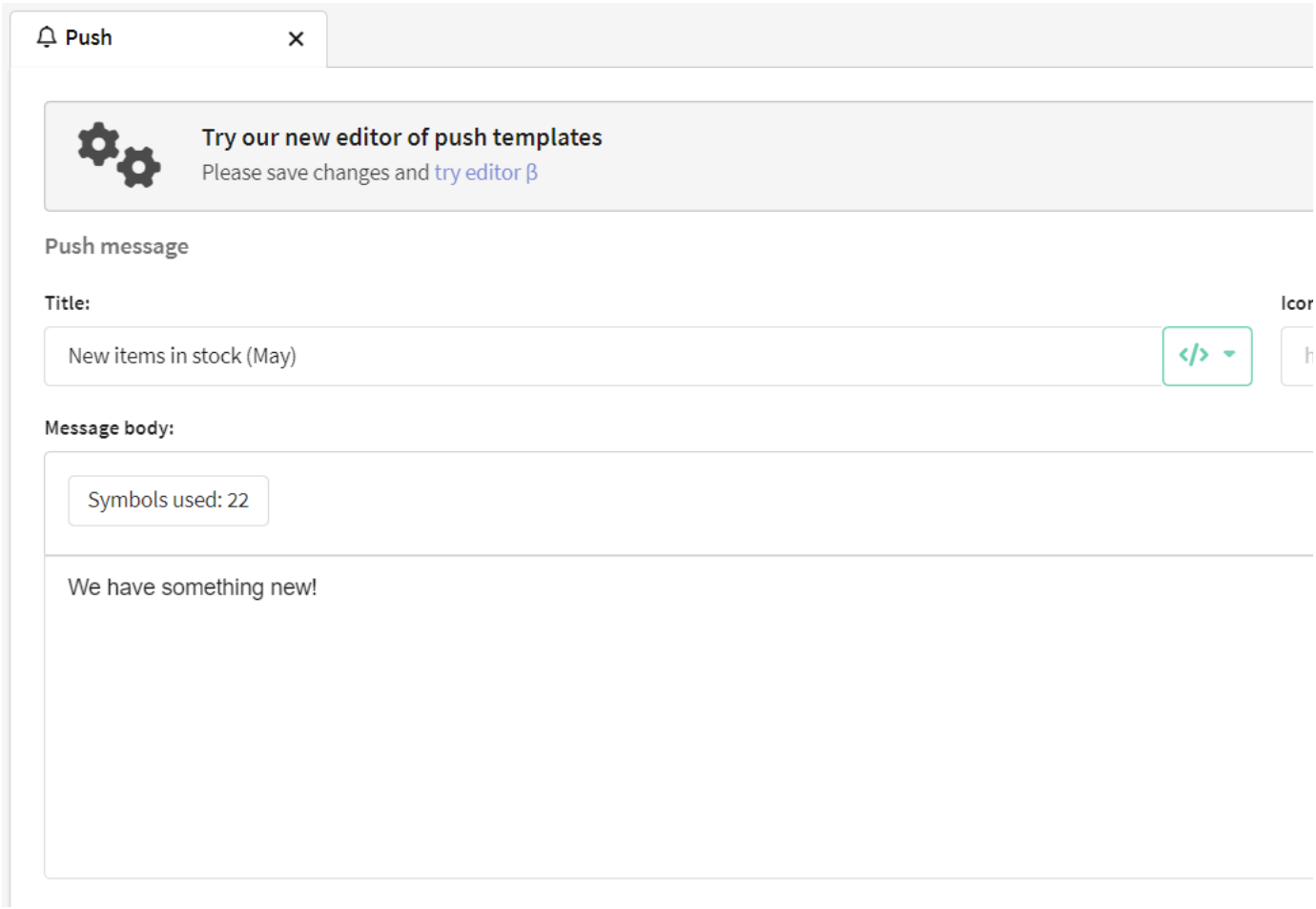
Step 4: Prepare content for sending

The content for your Push mailings is created in Message Templates. The templates contain content for create a new template, select the Push channel or add it for an existing one.

The template of the push notification can contain:

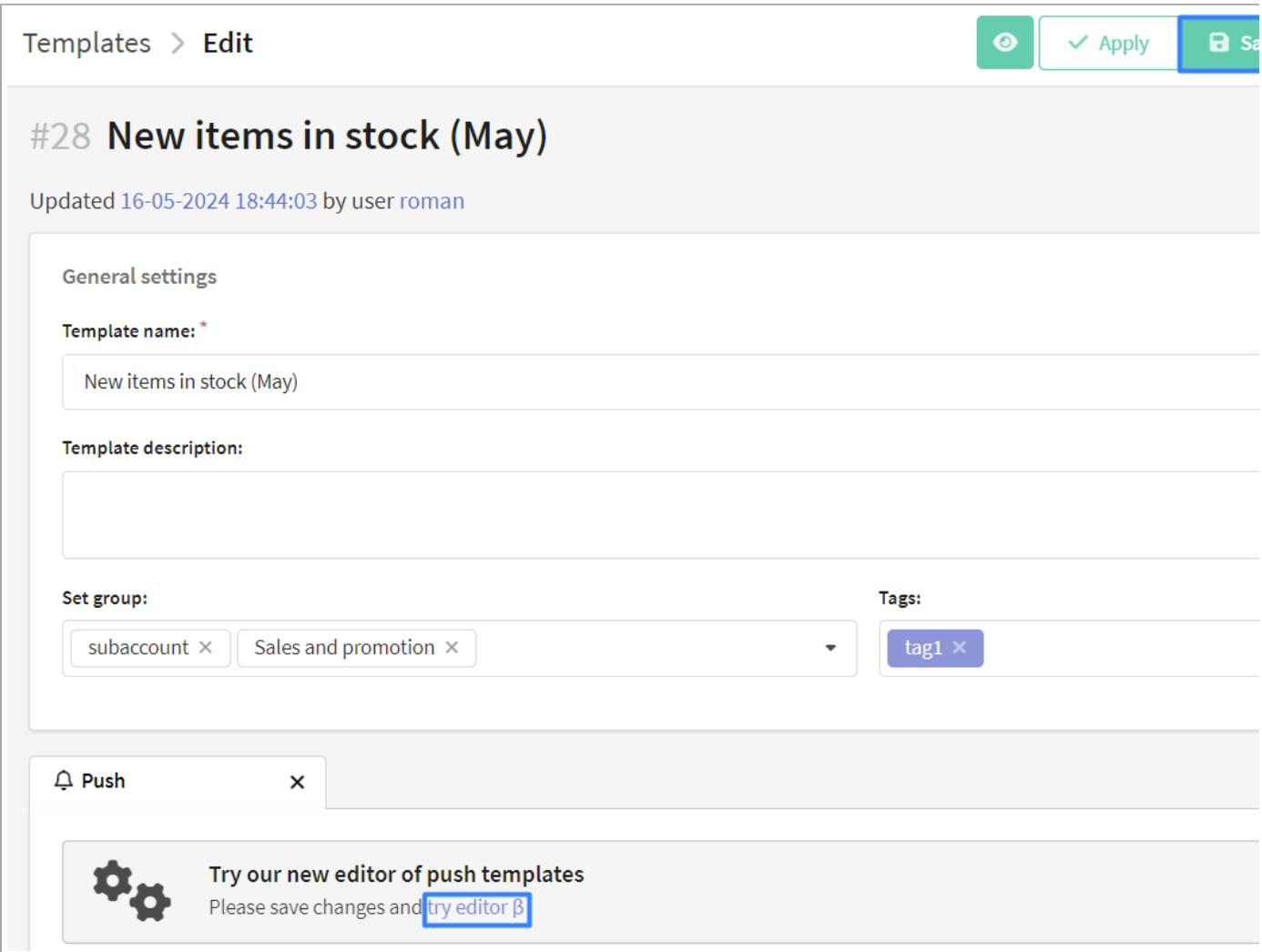
- **A **title** that is similar to email subject**
- **Notification **body** is the main text part of the message**
- **An **image** or an **icon** that will be displayed in the notification**
- **A clickable **link** is the link that the subscriber will follow when he clicks on the push notification.**

To the title, link and body, you can add the data of a specific client, as well as dynamic text content from



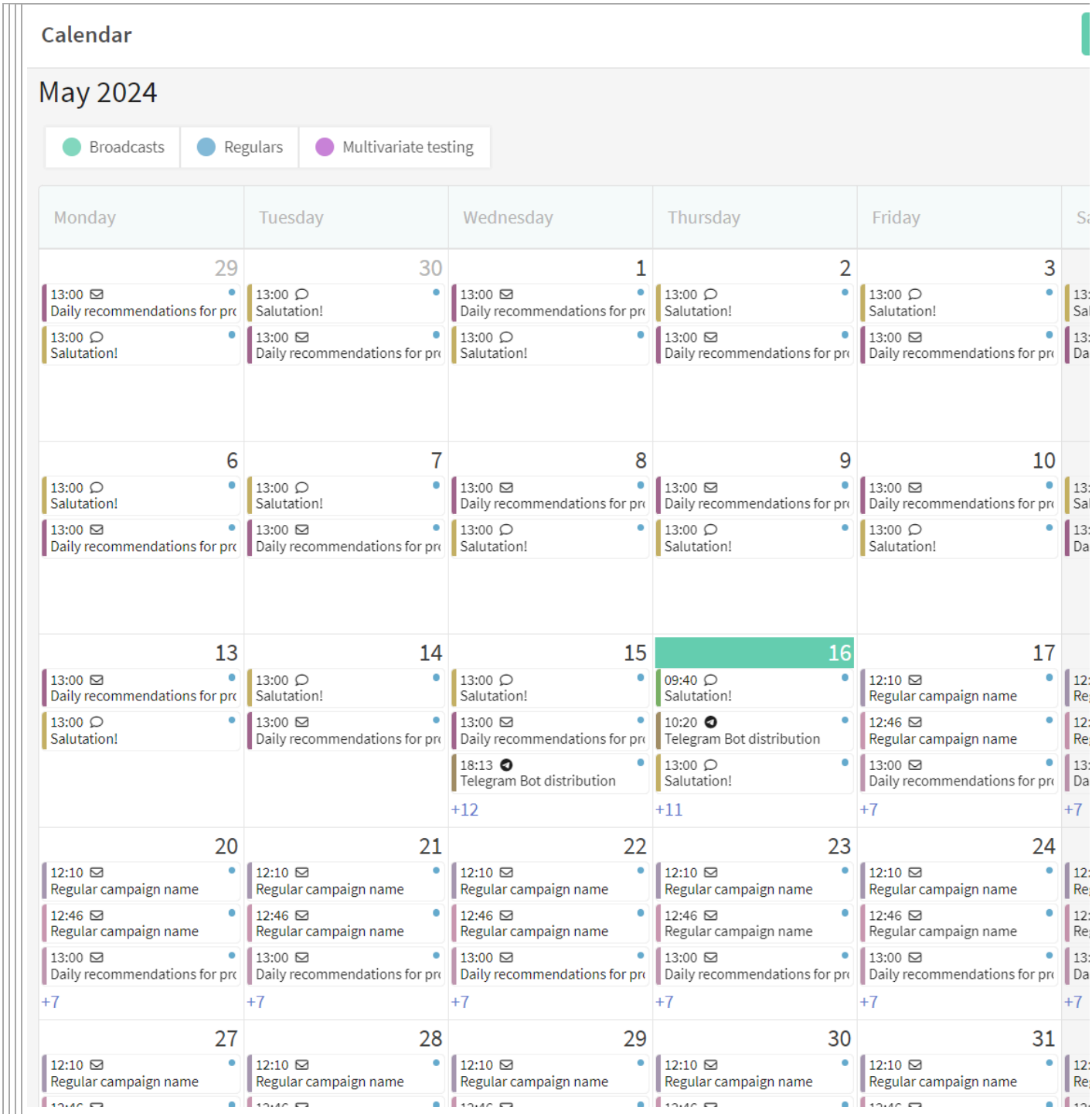
If you send not just Web push, but Rich push notifications, you'd better take a look at our special editor t notifications, you can add not only a text and an icon, but also buttons, images, banners, and even audio new editor will allow you to individually customize the notification for a specific browser, as well as add

To open the Rich push editor, you need to make an initial version of a push message in the usual templat how to do this above. Then click Save. Now go back to the created template and click "try editor β" to go

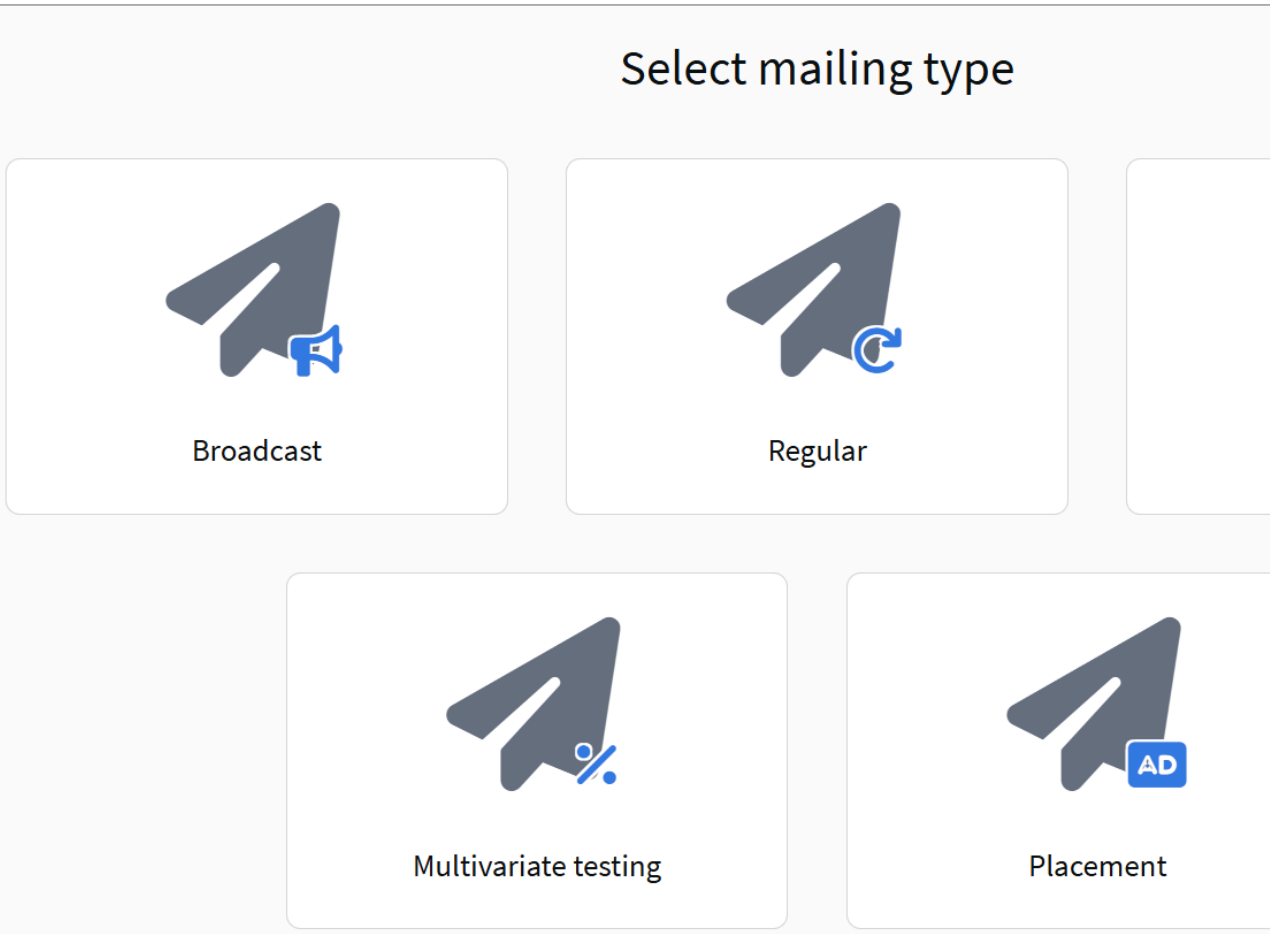


Step 5: Select the mailing type

![img] (/img/attachments/37849016/37849044.png)	**If you want to send a Push message, you need to use such a tool as a Mailing.** Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose ** Broadcasts** . If the mailing is periodic, your choice is a **Regular** mailing. In the main menu select **Mailing** and open **Calendar.** This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



TIP

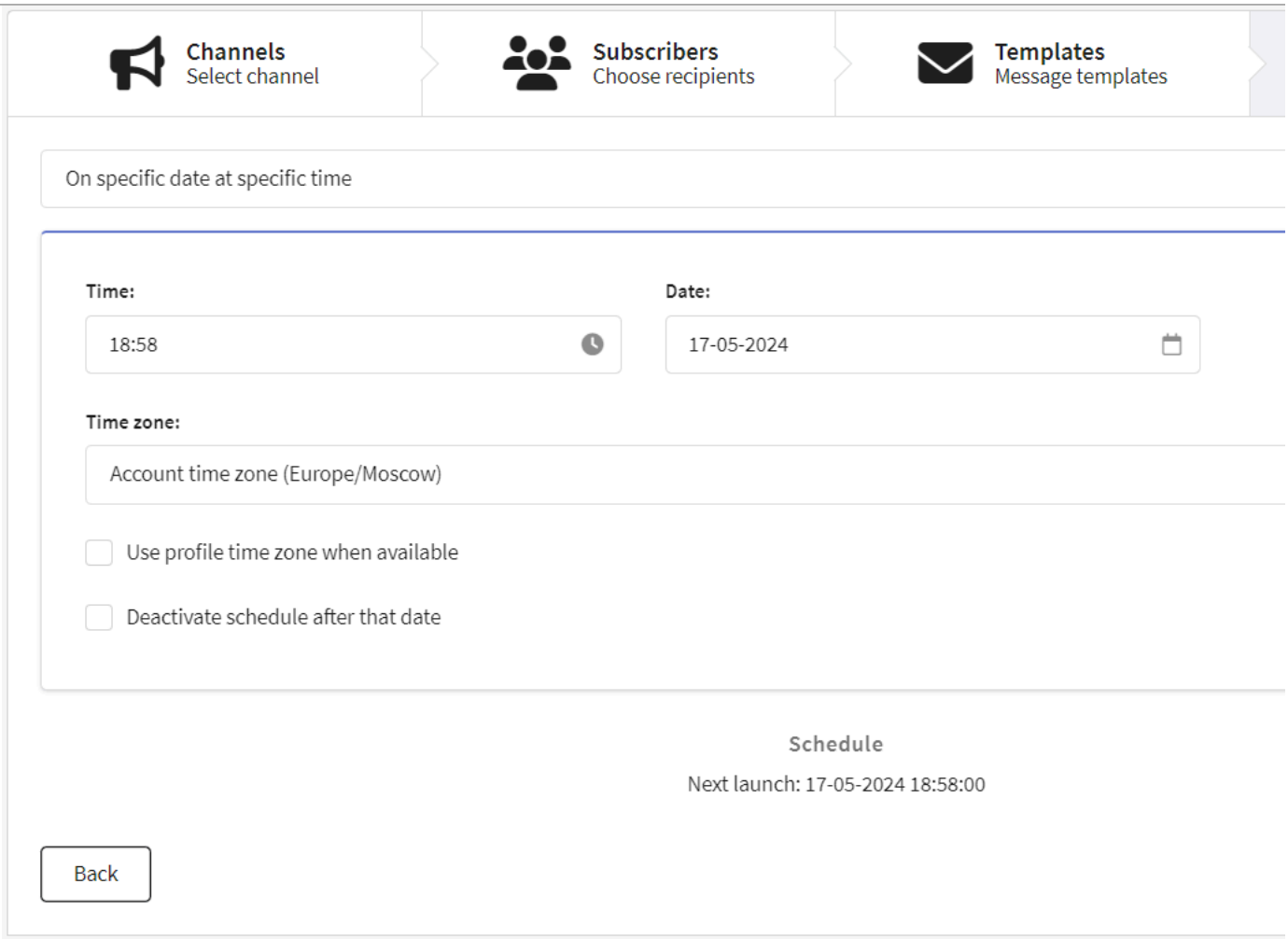
You can find details about how to set up Broadcast on this page.

TIP

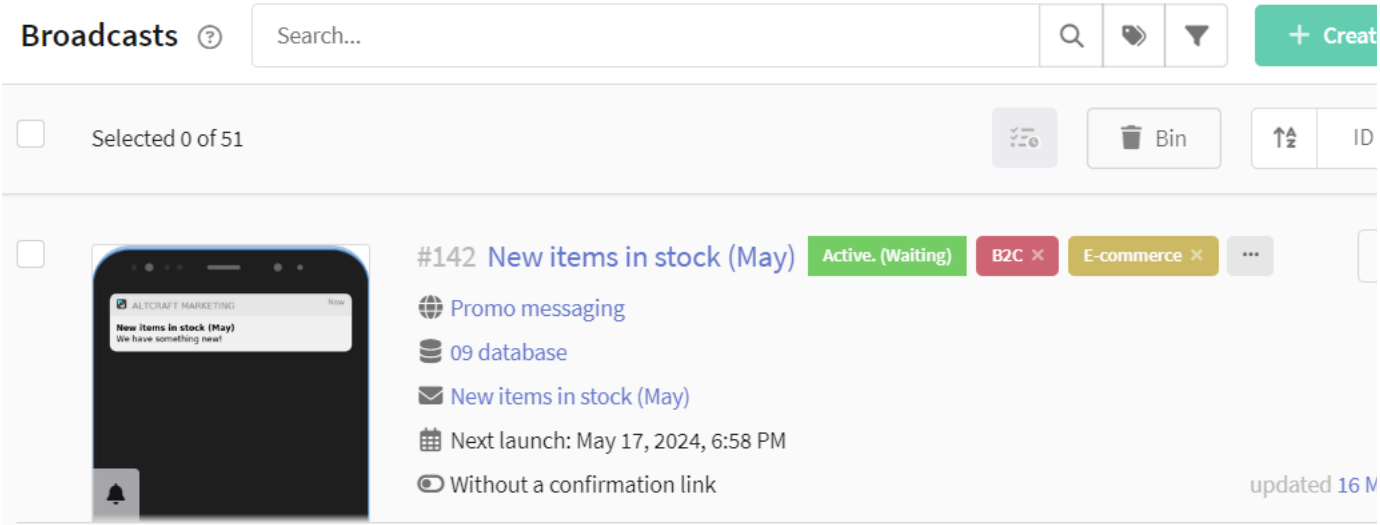
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select **Push** as the communication channel;**
2. **Select providers and sending mode. There are two sending modes - **fanout**, sending to all subscri subscription with the most recent creation date. If there are several of these (created at the same ti selected randomly. If mailings have filters by provider, custom fields or categories, then the filters a recent subscription is chosen;**
3. **Select the subscription resource you've just configured and the audience of recipients;**
4. **Add a message template to your message;**
5. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is availab**



Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



Step 6: Assess the mailing effectiveness

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your mailing. It logs key events, mai helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Descript
Mailing started	**Timestamp when the mailing execution began**
Limit	**Indicates whether a sending limit is configured**

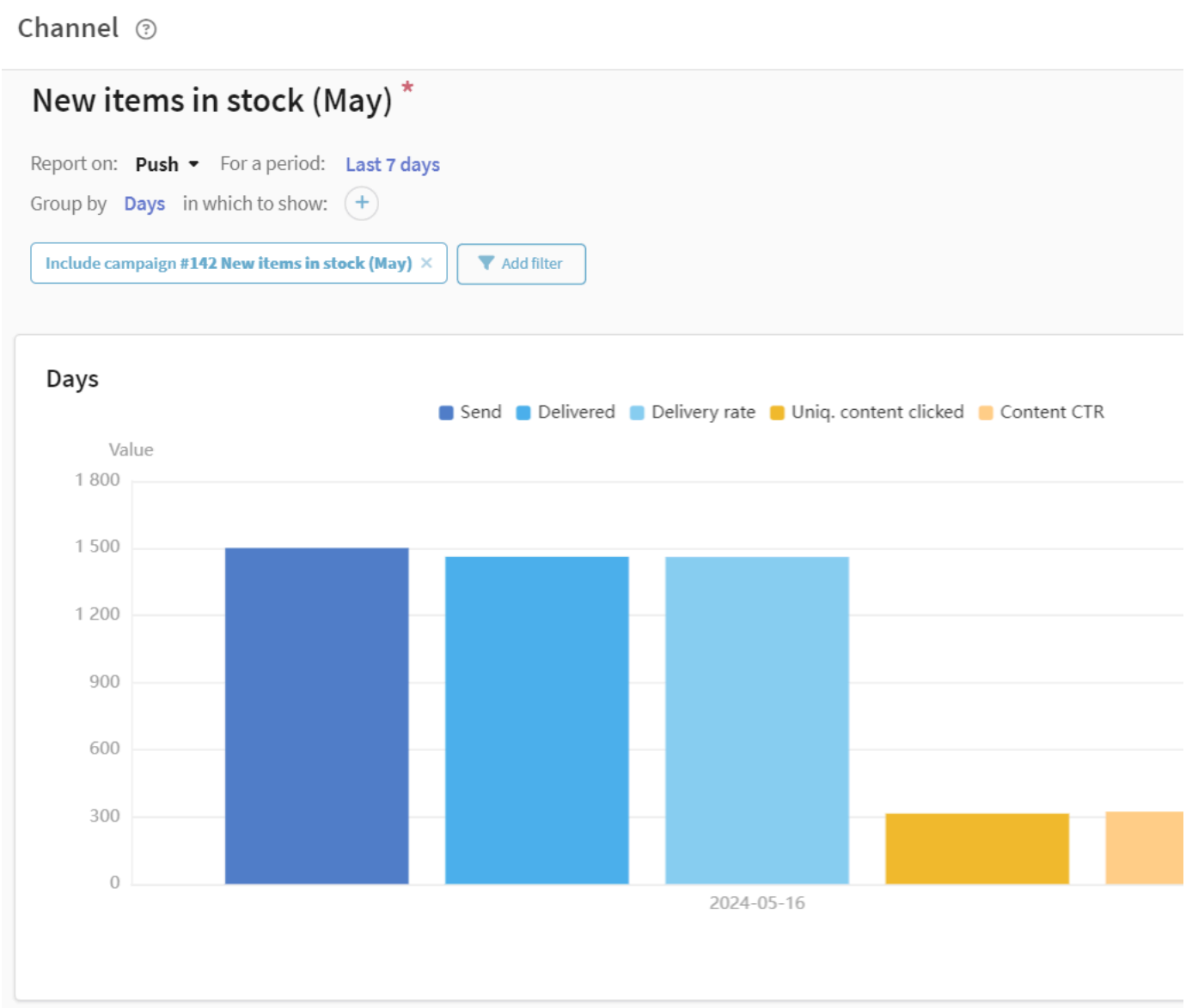
Type	Descript
Audience source	**Contact origin:** **List — Database** **Segment — Predefined contact group**
Database ID/Segment ID	**Unique identifier of the database being used**
Spent getting segment cursor	**Time required to load contact data from source (milliseconds)**
Number of subscribers received from the source	**Total contacts available in the specified source**
Number of subscribers found	**Contacts with active subscriptions matching mailing criteria**
Number of subscribers without push subscriptions	**Users who haven't opted in to receive push notifications**
Number of Push recipients	**Final count of devices that successfully received notifications**
Number of Push messages sent	**Total push notifications dispatched by the system**
Profiles ignored by resource settings	**Contacts excluded due to mailing configuration rules**
Mailing finished	**Timestamp when mailing processing completed**
Total time spent	**Complete duration from launch to completion (seconds)**

Reports and Analytics

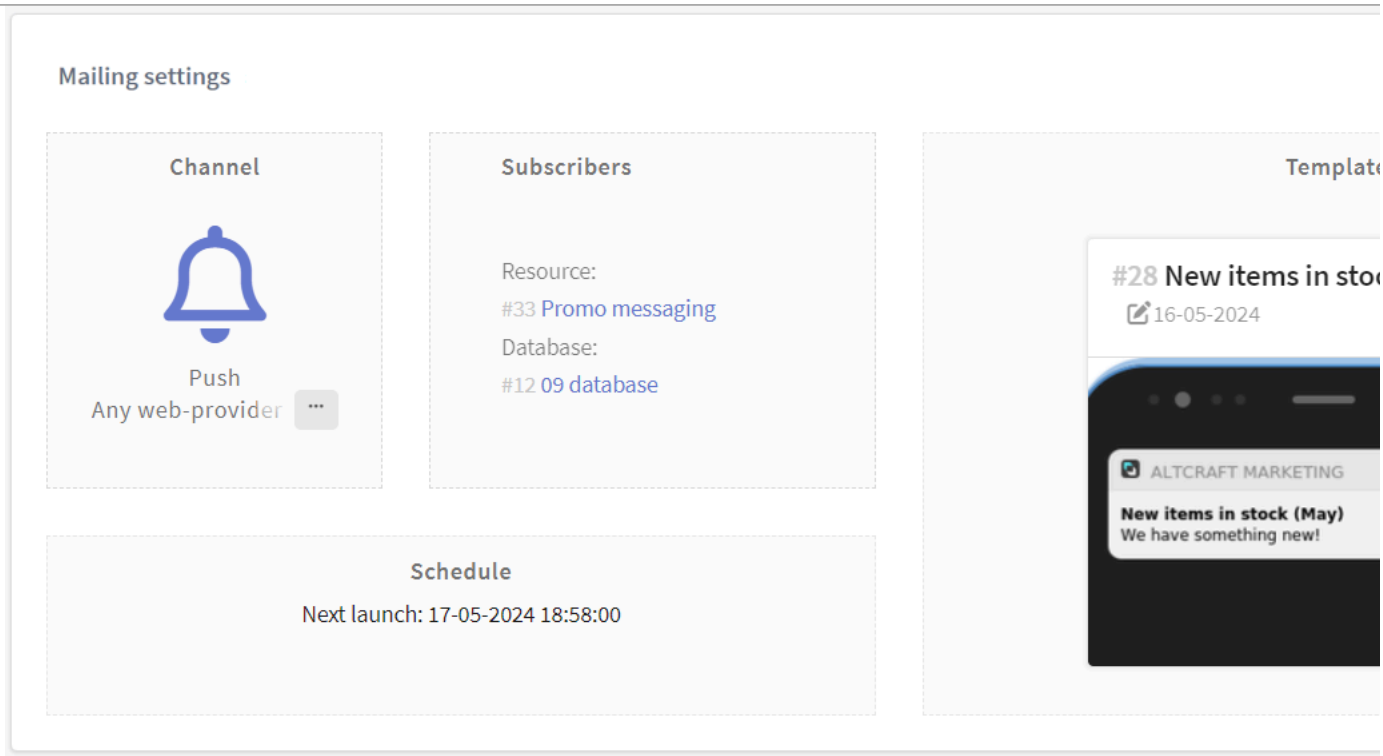
To evaluate the effectiveness of your push mailings go to the section Analytics in the main menu of the p

The channel report provides analytics for one or several mailings, for one or several communication char

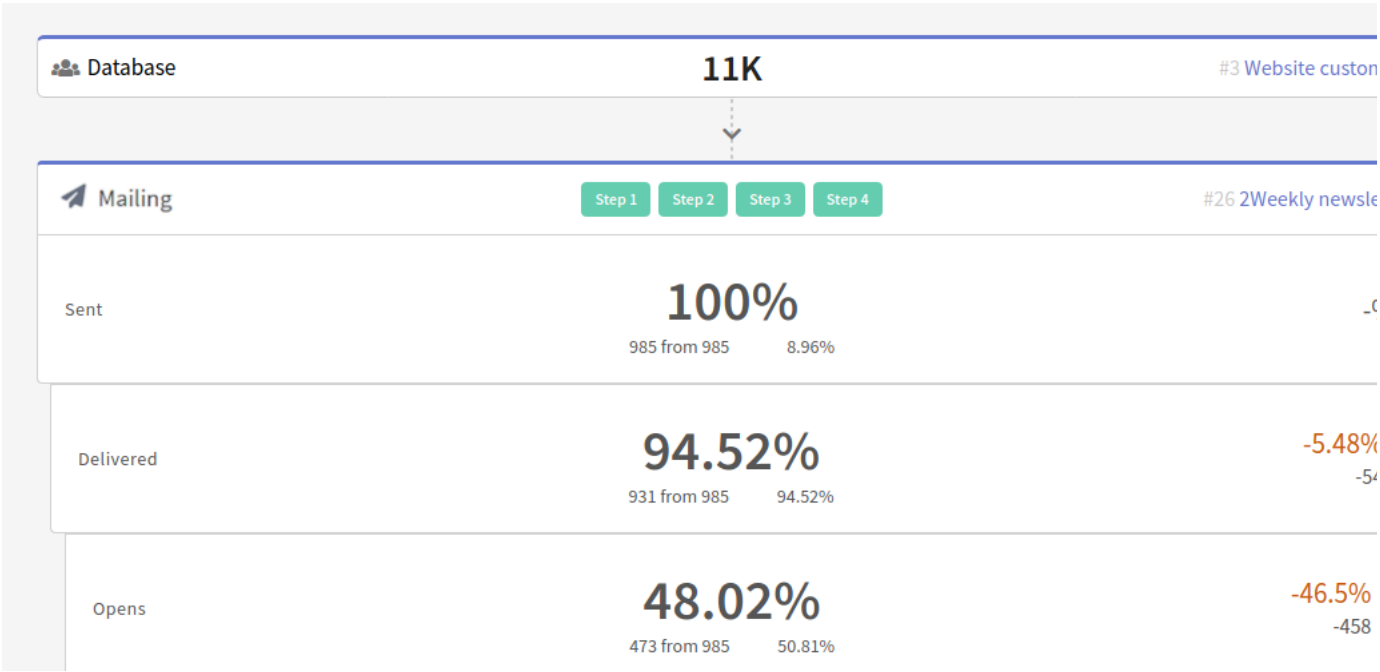
grouping, you can select a preset of indicators for Push:



You can also go to the channel report for a specific mailing by following the link in the mailing editor:



Actions in the Push mailing or in the template can turn into steps in the [interactive conversion funnel](#):



Resource and Website Setup

Preliminary Settings

- **Your website must use the HTTPS protocol. Web Push does not work on HTTP sites.**
- **The web pixels and push notification functionality must be activated in the Altcraft Platform.**
  - **Cloud version: If you are using the cloud version of Altcraft, contact support at support@altcraft. notifications and pixels for your account.**
  - **On-premises: Push notifications and pixels must be activated for all accounts from which sendin Account Management, click on the required account, and activate push notifications and pixels. corresponding administrator documentation article.**

![img](/img/attachments/37847712/en/37847714.png)

Creating a Resource

A resource is an object in the platform that links a subscriber database to a communication channel. Aft becomes available — a set of files for Web Push to work on your website.

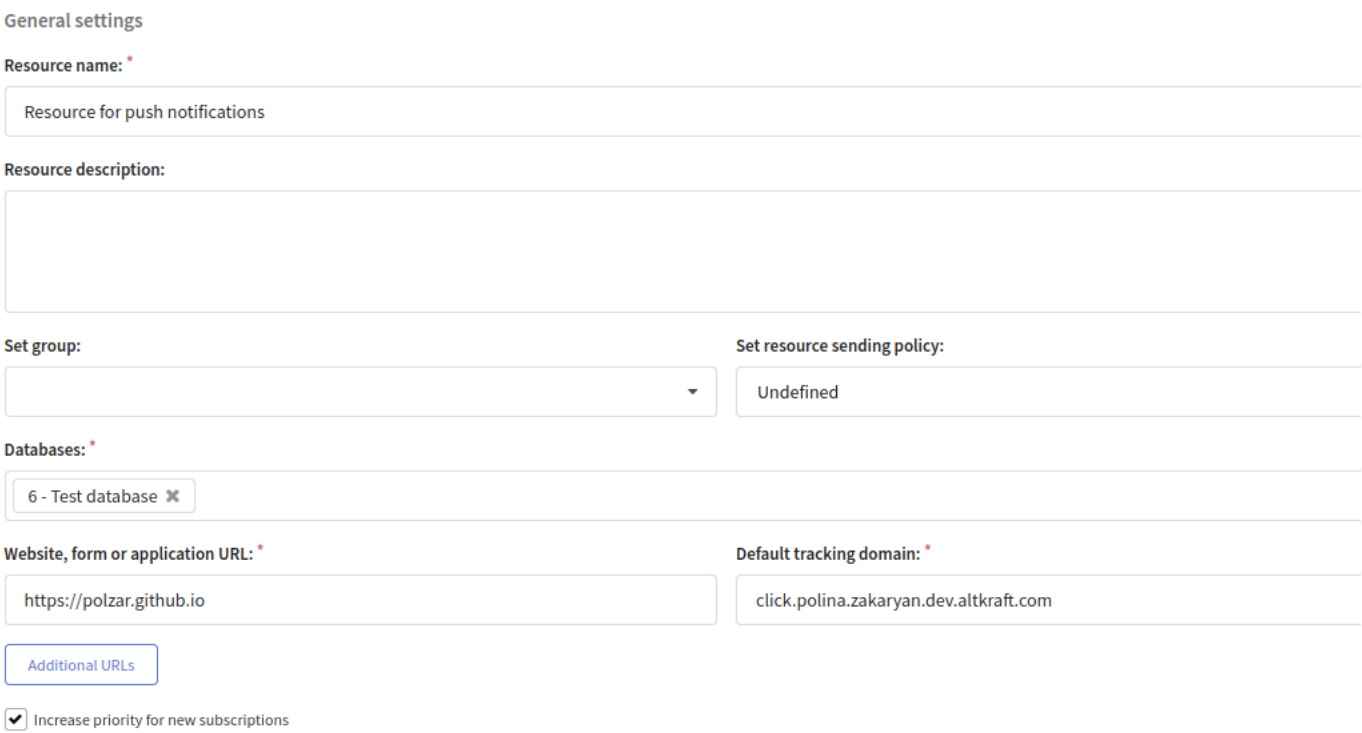
> **TIP**
>
> **A separate resource must be created for each website.**

In the platform, create or select a Database where push notification subscribers will be stored.

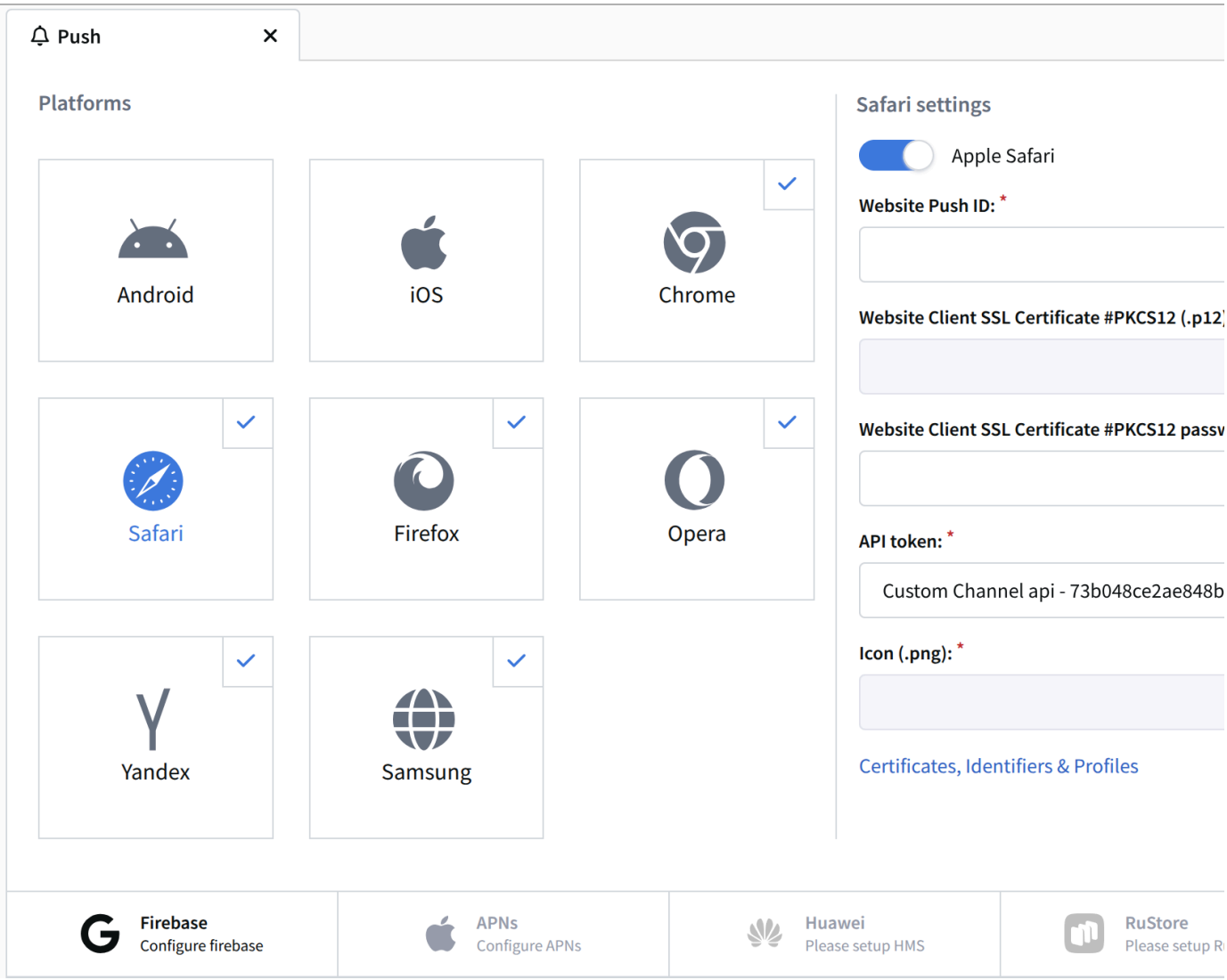
Add a push channel to an existing resource or create a new one. Enter the resource Name and Descripti

In the Website, Form, or Application URL field, specify the address of the website where push subscripti notifications will be sent. This address is added to the `manifest.json` file required for push setup, so push address is incorrect.

In the Default Tracking Domain field, specify the domain where Altcraft tracking is running. This domain



Next, select the desired browsers and providers for each platform. Settings for the selected providers wi



Detailed instructions for each platform are available in [this section of the documentation]. Without provid will not work.

Installing the SDK on Your Website

After all settings are configured, save the resource. The Toolkit — a set of files for Web Push to work on download. The Toolkit will appear in the Installation section.

> **INFO**
>
> **The Toolkit itself does not trigger the subscription window. It contains files and scripts for working witl subscription prompt, add the code that calls the `ak-push.js` function.**

Installing files on the website is possible in two ways:

Method 1: Dynamic `service worker`

Add the code that appears to your website and upload the installation files to the root directory:

- **manifest.json — project data for sending push via Google Chrome**
- **service-worker.js — service worker registered in user browsers**

The `service-worker.js` file is dynamic. It is automatically updated when resource settings change.

```
importScripts('https://pxl.test.com/service-worker.js?id=MXw3');
```

> **CAUTION**
>
> **When the Website, Form, or Application URL parameter is changed in the resource being used, you mus website.**

Method 2: Static Toolkit

On the right side of the interface, click Download, extract the archive, and place the files in the root dire consists of four files:

- `README.txt` — **installation instructions**
- `ak-push.js` — **script for displaying the subscription window**
- `manifest.json` — **project data**
- `service-worker.js` — **service worker**

When using the Toolkit, you can change the path to `service-worker.js`. Open `ak-push.js`, find the `swPath` pro

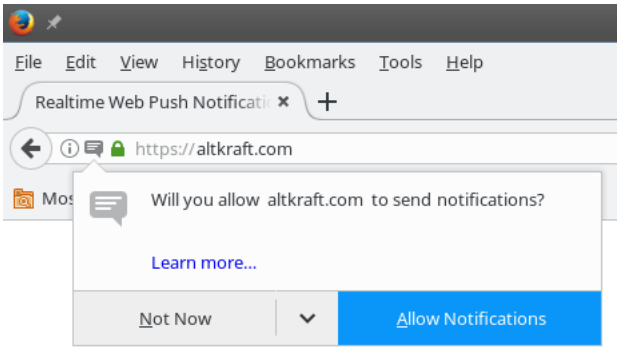
> CAUTION
>
> **When resource or provider settings are changed, you must re-upload the Toolkit to the website.**

Adding Subscription Code

After installing the Toolkit, add the code to display the subscription window to your website:

```
<button id="init_sub">Subscribe to Notifications</button>

<script>
    document.getElementById('init_sub').addEventListener('click', function() {
        try {
            const akPush = new AKPush();
            akPush.initSubscription()
        }
        catch (e) {
            console.log(e);
        }
    });
</script>
```



How it works:

1. **The user clicks the button**
2. **The `initSubscription` method is called**
3. **A system browser window pops up with a subscription prompt**

Subscription to one resource occurs separately for each browser. A single profile can have multiple activ subscribing from different browsers.

When a user grants permission to receive notifications, the platform automatically creates a profile for t other additional data can also be transferred. For more details on configuring data import, see here.

> TIP
>
> **Through the Altcraft Platform, you can create a custom subscription popup. For more details, see the a**

Transferring Data to the Platform

When a user grants permission to receive notifications, the platform automatically creates a profile for t other additional data can also be transferred. This article explains how to configure a script to collect in actions.

> **TIP**
>
> **Setting up Web Push on the Altcraft Platform requires using Web Push SDK methods. You can learn mor Instructions for installing the SDK on your website can be found here.**

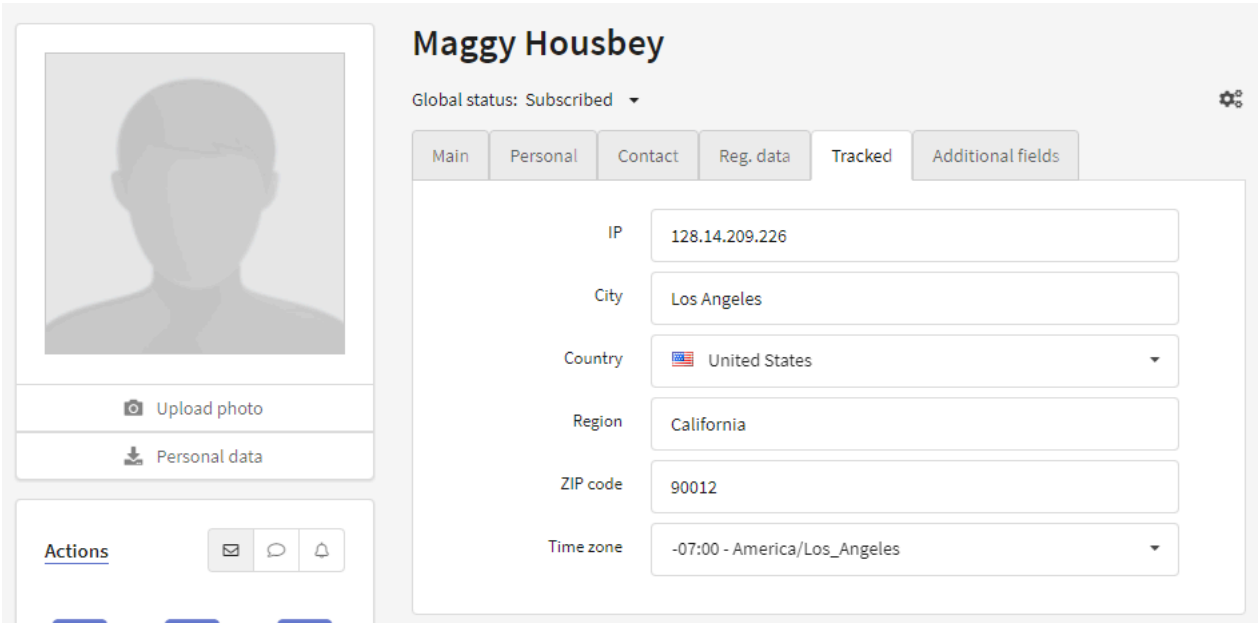
Importing Client Profiles When Subscribing to Web Push Notifications

When a user subscribes to Web Push notifications, a new profile will be added to the Databases and the embedded on the website was generated.

> **INFO**
>
> **If a database profile has previously opened your mailing from the same browser in which they are now s will be added to that profile. In this case, a new profile will not be created.**

When a subscriber clicks on a push notification (following a link), the Altcraft Platform collects profile da

- **Country, region, time zone, city, postal code, and IP address from which the push subscription occur**
- **Browser and operating system of the subscriber.**



If the token becomes outdated or the user resubscribes to push, the platform will automatically update t token. This prevents duplicate profiles from being created in the database.

> **NOTE**
>
> **Mobile browsers also support Service Worker and Web Push notifications.**

Transferring Client Data When Importing a Profile via Push Channel

At the time of push notification subscription registration, any known user data can be recorded into the client's first and last name, date of birth, pages viewed, and other business-related information. At the s existing subscriber already has the same data.

Example with email matching in the resource and adding the user's first and last name:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                email: 'example@email.com'
        }, // Searching for a profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Recording the subscriber's first and last name in the profile
)
```

If multiple databases are linked to a resource, you can specify the ID of the database where the profile s email matching in database `id=1`:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                db: 1
                email: 'example@email.com'
        }, // Searching for a profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Recording the subscriber's first and last name in the profile
)
```

Example with email matching in databases `id=15` and `id=20`:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                db: [15,20]
                email: 'example@email.com'
        }, // Searching for a profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Recording the subscriber's first and last name in the profile
)
```

> **TIP**
>
> **The list of profile fields is provided in this article.**

Updating Profile Data and Suspending Subscriptions

> **CAUTION**
>
> **akPush.updateSubscription will return an error if an up-to-date token has not been obtained via akPush removed from the cache using akPush.removeToken.**

A user subscribed to push notifications in a browser can change their status. For example, they may log receive a different matching profile. In this case, the current push subscription must be disabled on the new one. To achieve this, use the `akPush.updateSubscription` method:

```
const akPush = new AKPush();
akPush.updateSubscription(
        {
                db: [15,20]
                email: 'example@email.com'
        },

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        }, // Recording the subscriber's first and last name in the profile
        false // Flag for inexclusive, determines whether to pause all subscriptions to the resource with the current browser token for othe
)
```

Web Push Subscription Creation and Update Events

Platform users can subscribe to events for push subscription creation and updates (subscriber token cha subscription data can be sent to an external server using additional code.

To subscribe to events, add the following code to your website's pages:

```
AKPush.events.on('subscribe', (data) => { // Subscribe to the subscribe event
   console.log(data);
});
AKPush.events.on('update_subscription', (data) => { // Subscribe to the update_subscription event
   console.log(data);
});
```

This code will log the subscriber's token, as well as browser and push provider information.

Web Push SDK Methods

TIP

For more details on setting up Web Push mailings on the platform and installing the SDK on your websit

This article describes the Web Push SDK methods and their possible parameters. Each method is also ac

initSubscription

This method initializes a subscription. It is used to record client data in a profile and perform a search ar

Parameters

Parameter	Type	Required	Description
match	**object**	****Yes****	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will se**
update	**object**	**No**	**Contains data to be added to a new or existing profile.**

Usage Example

```
const akPush = new AKPush();
akPush.initSubscription(
	{
		db: 1
		email: 'example@email.com'
	}, // Searching for a profile by email. If no match is found, it will be recorded in the profile.

	{
		_fname: 'Anatoly',
		_lname: 'Wasserman'
	} // Recording the subscriber's first and last name in the profile
)
```

updateSubscription

This method updates profile data and modifies the status of push subscriptions.

Parameters

Parameter	Type	Required	Description
match	**object**	****Yes****	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will**
update	**object**	**No**	**Contains data to be added to a new or existing profile. If you want to pass the **inexclusive** parameter without updating the profile data, y**
inexclusive	**boolean**	**No**	**If the `inexclusive` flag is not provided or set to `false`:** **1. If no database filter is specified, each database of the resource will have only one profile with one subscription for the current token with** **2. If a filter is specified, each specified database will have only one profile with one subscription for the current token with the "Subscribed"** **resource for all profiles will be suspended.** **If the `inexclusive` flag is set to `true`, the call is equivalent to `AKPush.initSubscription`, meaning the profile data will be updated and the su**

Usage Example

```
const akPush = new AKPush();
akPush.updateSubscription(
	{
		db: [15,20]
		email: 'example@email.com'
	},

	{
		_fname: 'Anatoly',
		_lname: 'Wasserman'
	}, // Recording the subscriber's first and last name in the profile
	false // Whether to suspend all subscriptions to the resource with the current browser token for other profiles (default)
)
```

AKPush.events.on

This method subscribes to events related to push notifications.

Parameters

Parameter	Type	Required	Description
subscribe **update_subscription** **suspend_subscription**	**string**	****Yes****	**Name of the triggered event**

callback	**function**	**Yes**	**Function that will be called when the event occurs. For `subscribe`, the returned parameters will be:**
			{
			browser: "<subscriber's browser>"
			provider: "<push provider>"
			token: "<subscriber's token>"
			}
			For `update_subscription` and `suspend_subscription`, the returned parameters will be:
			{
			browser: "<subscriber's browser>"
			provider: "<push provider>"
			token: "<subscriber's token>"
			tokenCached: "<cached subscriber's token>"
			}
			The token parameter may return `null` as a value.

Usage Example

```
AKPush.events.on('subscribe', (data) => { // Subscribe to the subscribe event
   console.log(data);
});
AKPush.events.on('update_subscription', (data) => { // Subscribe to the update_subscription event
   console.log(data);
});
```

suspendSubscription

This method suspends a user's push notification subscription.

Parameters

Parameter	Type	Required	Description
match	**object**	**Yes**	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will s**
update	**object**	**No**	**Contains data to be added to a new or existing profile.**

Usage Example

```
const akPush = new AKPush();
akPush.suspendSubscription(
         {
                  db: 1
                  email: 'example@email.com'
         }, // Searching for a profile by email.

         {
                  _status: 'inactive'
         } // Updating the subscriber's profile status
)
```

AKPush.removeToken

This method is used to remove a device token from the cache. It is used, for example, in situations wher
No parameters are required for this method.

Usage Example

```
AKPush.removeToken()
```

PWA and Push Notifications

PWA (Progressive Web App) is a web application that a user can install on a device from a website. After separate application, can work with cache, use Service Worker, and receive push notifications.

> INFO
>
> **PWA is not a separate communication channel in Altcraft. It is a use case of Web Push for an installed v**

Setting Up Push Notifications for PWA

Setting up push notifications for PWA is no different from setting up regular Web Push. The order of actio

1. **Set up the resource and website — resource, Toolkit, SDK, subscription code**
2. **Set up providers — Firebase for Android, APNs for iOS**

The only difference is that the `manifest.json` placed on the website must contain parameters that allow th an application.

Manifest for PWA

Manifest is a JSON file that tells the browser how the application should be named, which icons to use, h installation.

For PWA to work, the manifest must specify:

- **`name` — full application name**
- **`short_name` — short name displayed next to the icon on the Home screen**
- **`start_url` — page that opens when the PWA is launched**
- **`display` — display mode; for PWA, `standalone` is used (opening without the browser address bar)**
- **`icons` — application icons; for installation on Android, icons 192×192 and 512×512 are required. Icon specified path.**

Minimal example:

```
{
  "name": "Example App",
  "short_name": "Example",
  "start_url": "/",
  "scope": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#ffffff",
  "icons": [
    {
      "src": "/icons/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

After creating the manifest file, it must be linked in the `<head>` of every page on the website. Without this manifest and will not be able to install the website as a PWA. Make sure the path to `manifest.json` is spec

```
<link rel="manifest" href="/manifest.json">
```

> TIP
>
> **The `manifest.json` file is provided in the download files and Toolkit when setting up a resource in Altcraf supplemented with the `name`, `short_name`, `display`, and `icons` parameters.**

Android Specifics

On Android, PWA Push works through the standard Web Push infrastructure of the browser. Supported in other Chromium browsers.

Requirements:

- **The PWA must be installed on the device;**
- **Firebase Cloud Messaging is used as the provider;**
- **The manifest must contain icons 192×192 and 512×512 pixels;**
- **Notifications are displayed in the Android system notification center.**

Detailed Firebase setup is described in the article Firebase Cloud Messaging.

iOS and Safari Specifics

Starting with iOS/iPadOS 16.4, installed web applications can receive push notifications. The user must a allow notifications.

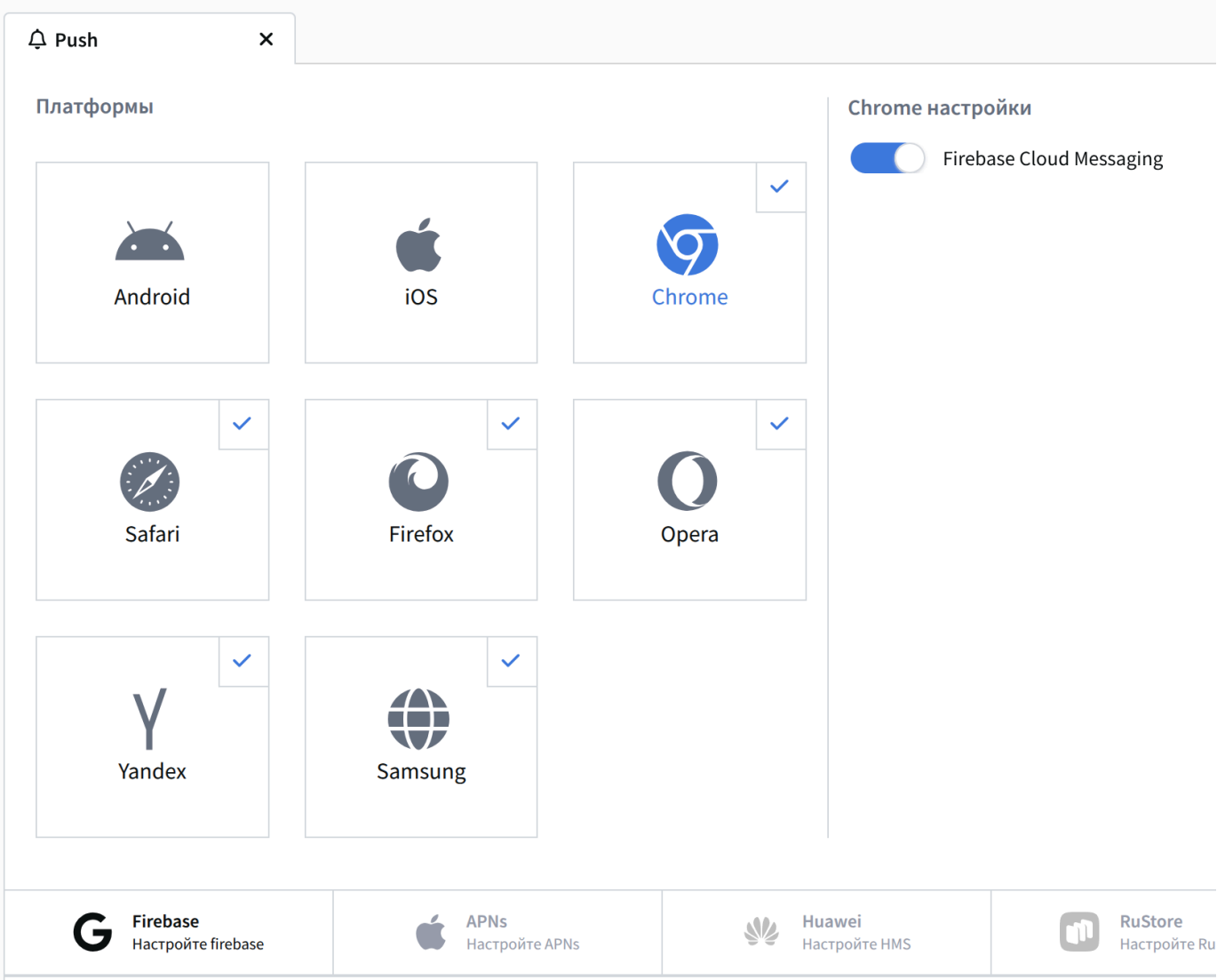
Limitations:

- **The PWA must be added to the Home screen;**
- **Notifications work through APNs;**
- **The user must explicitly allow notifications;**
- **Settings and certificates are tied to an Apple Developer Account;**
- **Behavior may differ from Android and desktop browsers.**

Detailed setup for Safari is described in the article Apple Safari.

Web Push Providers

A provider is a service that delivers push notifications to a browser. Each browser works with specific pr
audience, you may need to configure multiple providers.
Provider settings become available after activating the corresponding toggles when selecting browsers:



Detailed instructions for setting up each provider are contained in separate documentation articles.

Available Providers

Provider	Browsers
Firebase Cloud Messaging	Chrome, Firefox, Opera, Yandex Browser, Samsung Internet
Mozilla Services	Firefox
Safari / APNs	Safari

INFO

Sending to Firefox is also possible via Firebase Cloud Messaging.

Firebase Cloud Messaging

![img] (https://firebase.google.com/images/social.png)	To send Web Push to Mozilla Firefox, Opera, Google Chrome, Yandex Browser, Samsung Internet Browser, you must first set

Step 1. Creating a Project in Firebase Console

![img](/img/attachments/37847584/37847585.png)

![img](/img/attachments/37847584/project-name.png)

TIP

When creating a project, you can add the Gemini AI assistant and set up Google Analytics, but these st

Step 2. Obtaining Project Data

To configure Firebase Cloud Messaging in the Altcraft Platform, you will need your project data from Fire

![img](/img/attachments/37847584/general-settings.png)

![img](/img/attachments/37847584/web-app.png)

![img](/img/attachments/37847584/register-name.png)

![img](/img/attachments/37847584/register-script.png)

![img](/img/attachments/37847584/service-accounts.png)

![img](/img/attachments/37847584/confirm.png)

Step 3. Setting Up Firebase in the Altcraft Platform

![img](/img/attachments/37847584/en/browsers.png)

![img](/img/attachments/37847584/en/resource-fields.png)

![img](/img/attachments/37847584/en/installing.png)

INFO: Push in PWA

The resource and provider configuration for sending push notifications to PWA applications is done in th the article PWA Push.

Apple Safari

![img](https://uxwing.com/wp-content/themes/uxwing/download/brands-and-social-media/safari-icon.png)	To send Web Push to the **Safari** browser, you must first set up an Apple Push Notification Service project.

CAUTION

All actions are performed on a macOS device. An Apple Developer account is required for setup.

Step 1. Register Website Push ID

![img](/img/attachments/37847599/1-id.png)

![img](/img/attachments/37847599/2-register.png)

![img](/img/attachments/37847599/3-register-name.png)

![img](/img/attachments/37847599/4-register.png)

![img](/img/attachments/37847599/5-web-id.png)

Step 2. Creating a Certificate

![img](/img/attachments/37847599/6-register-cert.png)

![img](/img/attachments/37847599/8-cert.png)

![img](/img/attachments/37847599/9-cert.png)

![img](/img/attachments/37847599/10-cert.png)

Step 3. Saving the Push Certificate

![img](/img/attachments/37847599/11-cert.png)

![img](/img/attachments/37847599/12-keychain.png)

![img](/img/attachments/37847599/13-trust.png)

![img](/img/attachments/37847599/14-export.png)

Step 4. Setting Up in the Altcraft Platform

![img](/img/attachments/37847599/en/resource.png)

![img](/img/attachments/37847599/en/settings.png)

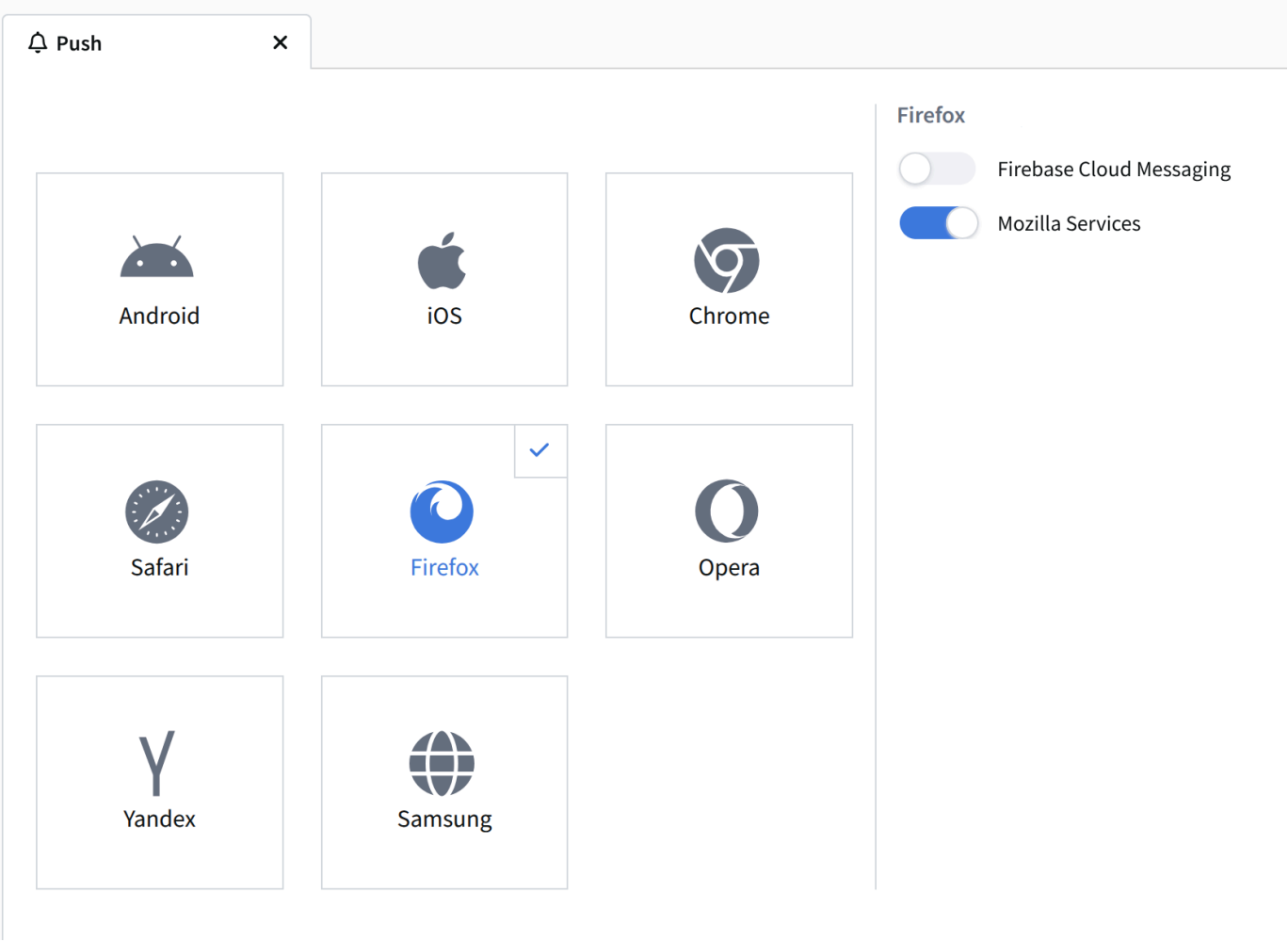
![img](/img/attachments/37847584/en/installing.png)

INFO: Push in PWA

The resource and provider configuration for sending push notifications to PWA applications is done in th the article PWA Push.

Mozilla Services

![img] (https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/Firefox_logo%2C_2019.svg/1280px-Firefox_logo%2C_2019.svg.png)	To send push notifications to the **Mozilla Firefox** browser, it is sufficient settings.



Further actions are described in detail in [this section] of the push channel configuration article.

INFO

Sending push notifications to Firefox is also possible via Firebase Cloud Messaging. For more details, se [Messaging].

Transferring Push Subscriptions from a Third-Party Service

This article describes transferring push subscriptions from a third-party service to the Altcraft Platform.

> TIP
>
> **VAPID keys and Application Server keys are the same thing.**

Step 1. Obtaining VAPID Keys

Request VAPID keys from the technical support of your current service. Both private and public keys are

---

Step 2. Importing VAPID Keys into Firebase

![img](/img/attachments/37848841/37848843.png)

![img](/img/attachments/37848841/37848844.png)

Step 3. Setting Up the Resource in the Altcraft Platform

Set up the resource for Web Push notifications. For more details, see the article [Resource and Website S]

> TIP
>
> **Resource setup differs for different browsers. Make sure the providers are configured correctly.**

Step 4. Preparing the CSV File with Subscriptions

When a user subscribes to push notifications, a `PushSubscription` object is created, which contains the dat

```
{
  "endpoint": "https://some.pushservice.com/something-unique",
  "keys": {
    "p256dh": "BIPUL12DLfytvTajnryr2PRdAgXS3HGKiLqndGcJGabyhHheJYlNGCeXl1dn18gSJ1WAkAPIxr4gK0_dQds4yiI=",
    "auth": "FPssNDTKnInHVndSTdbKFw=="
  }
}
```

Export this data from the third-party service to a CSV file. For each subscription, the fields `endpoint`, `p256`

Transfer the CSV file for processing:

- **On-premises — transfer the file to the platform administrator**
- **Cloud version — transfer the file to Altcraft support at [support@altcraft.com]**

The administrator (or support team) will convert the data to a format compatible with import. For more d documentation article.

After processing, you will receive a CSV file ready for import.

Step 5. Importing Profiles with Subscriptions

![img](/img/attachments/37848841/37848851.png)

![img](/img/attachments/37848841/37848852.png)

![img](/img/attachments/37848841/37848853.png)

> CAUTION
>
> **If the file contains data for matching existing users (e.g., email, phone), select the corresponding profil**

![img](/img/attachments/37848841/37848855.png)

> CAUTION
>
> **To create a subscription, you need the name of the field where the subscription ID is stored. In the proc `firebase_push_token`.**

![img](/img/attachments/37848841/37848856.png)

List of Web Provider Identifiers

In the Provider field of the CSV file, specify one of the following identifiers:

Provider	
Mozilla Firefox	`mozilla`
Safari	`safari`
Firefox Firebase	`firefox-firebase`
Chrome Firebase	`chrome-firebase`
Opera Firebase	`opera-firebase`
Yandex Firebase	`yandex-firebase`
Samsung Firebase	`samsung-firebase`
IOS APNs	`ios-apns`
IOS Huawei Mobile Services	`ios-huawei`

CAUTION

In the Skip Lines field, enter 1 if the file contains headers.

![img](/img/attachments/37848841/37848857.png)

TIP

If you are transferring subscriptions from OneSignal, see the article Migration from OneSignal.

Transferring Push Subscriptions for Safari

This article describes transferring push subscriptions configured for the Safari browser to the Altcraft Pl

CAUTION

Transferring Safari subscriptions is only possible if they were created with a certificate and key from y must have access to this account.

Step 1. Setting Up the Resource in the Altcraft Platform

Set up the resource for Web Push notifications and activate the Apple Safari provider. For more details, s Setup.

Step 2. Preparing the CSV File with Subscriptions

Export Safari subscription identifiers from the external service to a CSV file.

The file must contain at least two columns:

- **Column with push tokens — Safari subscription identifiers;**
- **Column with provider — each row must contain the value Safari (provider identifier from the table).**

Example CSV file structure:

```
push_token,provider
token_abc123...,Safari
token_def456...,Safari
token_ghi789...,Safari
```

Step 3. Importing Profiles with Subscriptions

![img](/img/attachments/37848859/37848867.png)

![img](/img/attachments/37848859/37848868.png)

![img](/img/attachments/37848859/37848870.png)

CAUTION

If the CSV file contains data for matching existing users (e.g., email, phone), select the corresponding p

![img](/img/attachments/37848859/37848871.png)

![img](/img/attachments/37848859/37848872.png)

CAUTION

In the Skip Lines field, enter 1 if the file contains headers.

![img](/img/attachments/37848859/37848874.png)

Migration from OneSignal

This article describes transferring push subscriptions from OneSignal to the Altcraft Platform.

> TIP
>
> **VAPID keys and Application Server keys are the same thing.**

Step 1. Obtaining VAPID Keys from OneSignal

> CAUTION
>
> **Skip this step if you are transferring subscriptions configured for the Safari browser.**

Request VAPID keys from OneSignal technical support. Both private and public keys are needed: private

---

Step 2. Importing VAPID Keys into Firebase

> CAUTION
>
> **Skip this step if you are transferring subscriptions configured for the Safari browser.**

![img](/img/attachments/37848878/37848879.png)

![img](/img/attachments/37848878/37848880.png)

Step 3. Setting Up the Resource in the Altcraft Platform

Set up the resource for Web Push notifications. For more details, see the article Resource and Website S

> TIP
>
> **Resource setup differs for different browsers. Make sure the providers are configured correctly.**

Step 4. Exporting Subscriptions from OneSignal

To export subscriptions, use the OneSignal API. Send a POST request via Curl — in response, you will rec compressed CSV file (`.csv.gz`).

> TIP
>
> **File creation may take several minutes depending on the number of users.**
>
> **You can check if the file is ready by sending a GET request with the `csv_file_url` parameter. If the file is returned.**

Example request:

```
curl -X POST \
  -H "Authorization: Basic YOUR_REST_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "extra_fields": ["web_auth", "web_p256"],
    "last_active_since": "1469392779",
    "segment_name": "Active Users"
  }' \
  "https://onesignal.com/api/v1/players/csv_export?app_id=YOUR_APP_ID"
```

In the request, you need to specify REST_API_KEY and APP_ID. This data is available in OneSignal: Settir

![img](/img/attachments/37848878/37848881.png)

Request body parameters:

Field	Value
`extra_fields`	**Additional fields. Be sure to specify web_auth and web_p256. Available:** `location`, `country`, `rooted`, `notification_types`, `ip`, `external_user_id`, `web_auth`, `we`
`last_active_since`	**Export devices with** `last_active` **greater than the specified timestamp**
`segment_name`	**Export devices from the specified segment. For more details about segments, see the OneSignal documentation**

> TIP
>
> **For more details on exporting subscriptions from OneSignal: CSV Export API.**

Step 5. Preparing the CSV File

> CAUTION
>
> **Skip this step if you are transferring subscriptions configured for the Safari browser.**

Transfer the CSV file with subscriptions for processing:

- **On-premises — transfer the file to the platform administrator**
- **Cloud version — transfer the file to Altcraft support at support@altcraft.com**

The administrator (or support team) will convert the data to a format compatible with import into the pla administrator documentation article.

After processing, you will receive a CSV file ready for import.

Step 6. Importing Profiles with Subscriptions

![img](/img/attachments/37848878/37848894.png)

![img](/img/attachments/37848878/37848895.png)

![img](/img/attachments/37848878/37848896.png)

> CAUTION
>
> **If the CSV file contains data for matching existing users (e.g., email, phone), select the corresponding p**

![img](/img/attachments/37848878/37848890.png)

> CAUTION
>
> **To create a subscription, you need the name of the field where the subscription ID is stored. In the proc** `firebase_push_token`**.**

![img](/img/attachments/37848878/37848891.png)

List of Web Provider Identifiers

In the Provider field of the CSV file, specify one of the following identifiers:

Provider	
Mozilla Firefox	`mozilla`
Safari	`safari`
Firefox Firebase	`firefox-firebase`
Chrome Firebase	`chrome-firebase`
Opera Firebase	`opera-firebase`
Yandex Firebase	`yandex-firebase`
Samsung Firebase	`samsung-firebase`
IOS APNs	`ios-apns`
IOS Huawei Mobile Services	`ios-huawei`

> CAUTION
>
> **In the Skip Lines field, enter 1 if the file contains headers.**

![img](/img/attachments/37848878/37848889.png)

SMS: ready-made integrations

Description

Altcraft Platform uses integrations with gateways of mobile operators to send SMS to your customers. If the list below - we can develop a custom integration during a pilot project (On-premises product trial). Co details.

Integrations

The following integrations are available out of the box:

- **SMSc.ru — SMS gateway by "СМС-Центр". Service documentation**
- **SMSimple.ru — SMS gateway by "Юлабс". Service documentation**
- **BulkSMS.com — SMS gateway by "Celerity Systems". Service documentation**
- **Devino.Online SMS — integration with API of Devino Telecom gateway. Service documentation**
- **SMSTraffic — SMS gateway by "СМС-Трафик". Service documentation**
- **Infobip — SMS gateway by "Infobip". Service documentation**
- **MTS — SMS outlet by "МТС". Service documentation**
- **MFMS EDNA — SMS gateway by "EDNA". Service documentation**
- **Sky Media — SMS gateway by Sky Media. Service documentation**
- **Play Mobile — SMS gateway by Play Mobile. Service documentation**
- **Rapporto — multipurpose platform Rapporto. Service documentation**
- **VK Notify — SMS gateway by Notify**
- **SberMobile — SMS gateway by SberMobile. Service documentation**
- **Stream telecom — SMS gateway by "Stream telecom". Service documentation**
- **Intellin — SMS gateway by Intellin**
- **Seven Tech — integration with the SEVENTECH SMS gateway. Service documentation**
- **i-Digital — SMS gateway by i-Digital. Service documentation**

> TIP
>
> **For proper operation of certain integrations, you need to provide specific information to the provider (fo sent SMS messages, you must provide a Rapporto URL to accept delivery status reports), or specify a p this, contact the platform administrators.**

Sender configuration

Cloud

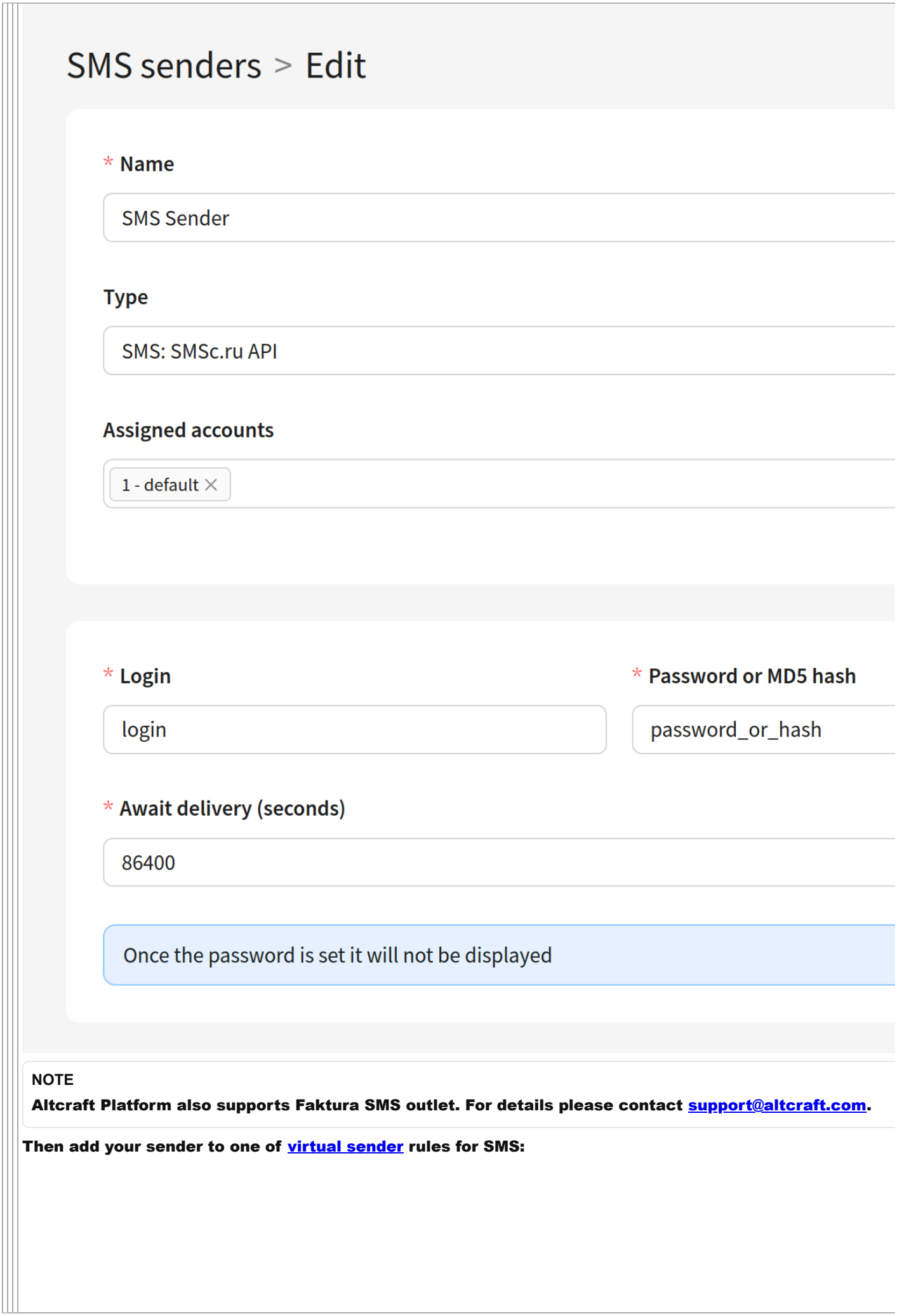
If your SMS gateway is listed among the ready-made integrations, contact support at support@altcraft.co

- **API token or key for your SMS gateway account;**
- **Caller ID (sender name) approved by your provider.**

The support team will configure the sender on your behalf.

On-premises

To start messaging using a ready-made integration, get the service account and credentials and add a n Administrative panel:



NOTE

Altcraft Platform also supports Faktura SMS outlet. For details please contact support@altcraft.com.

Then add your sender to one of virtual sender rules for SMS:

Rules for SMS senders

+ Add rule

Rule:

Default (match all)

Selected sender:

1 - SMS Sender

Percent:

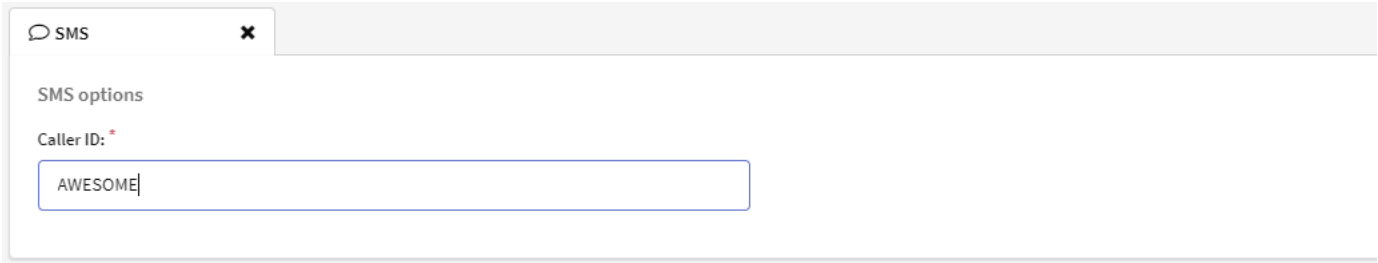
100

Override "From name":

✕ Remove sender

+ Add sender | ✕ Remove rule | ↑ Move up | ↓ Move down

After you get a CALLER_ID from your telecom service, enter it in your Resource SMS tab:



First SMS Mailing

![img] (/img/attachments/37848968/37848969.png)	**This tutorial will help you to create and launch your **SMS** mailing in the Altcraft Platform from scratch.** **SMS** as a channel of communication with clients is impossible to replace. It does not matter if a customer has Internet access, because today everyone has a mobile phone. The subscription process is also simpler than in other channels: the only thing you need to get started is a customer's phone number. Yes, there is no way to add media content to SMS, but you can add links to resources on the Internet. Altcraft Platform automatically **shortens** links in a message and also **tracks** recipients' clicks. </td> </tr>

Step 1: Connect SMS gateway

Altcraft Platform uses integrations with gateways of mobile operators to send SMS to your customers.

> TIP
>
> **You can read about our ready integrations in this article.**
>
> **In case you use different SMS gateway, write to us at support@altcraft.com and we will agree with you gateway.**

At first you'll have to connect the SMS sender. You need the authorization data of the telecommunicatio CALLER ID agreed with it.

> CAUTION
>
> **If you use an Altcraft Platform cloud account, please give the information to the support team at suppo to connect the integration.**

If the Altcraft Platform platform is installed on your equipment (On-premises), you need to configure SMS administrative panel. Contact your platform administrators to set up senders and virtual senders using t

- **Senders are configuration records that combine parameters for sender identification, sending speed behavior in case of sending errors.**
- **Virtual senders are the rules for sending from multiple senders. Can be configured both in the Admin Interface.**

> TIP
>
> **Or just contact our support team — and we will help you set up SMS sending.**

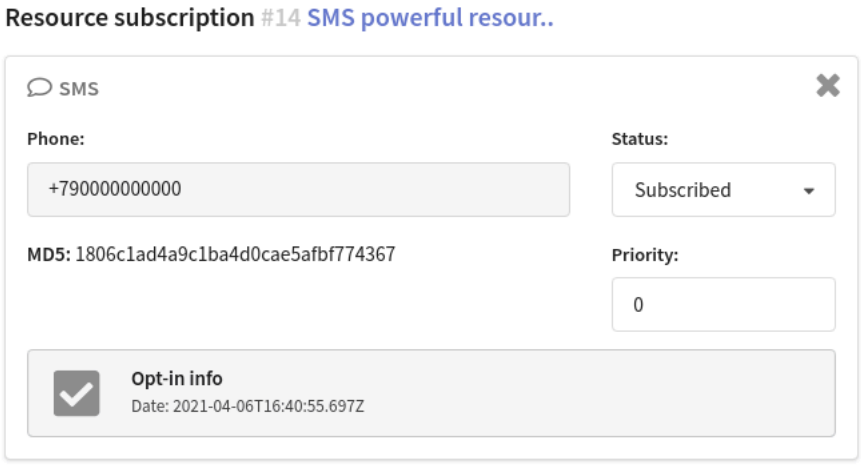
Step 2: Subscribe customers to the SMS channel

Profiles Databases

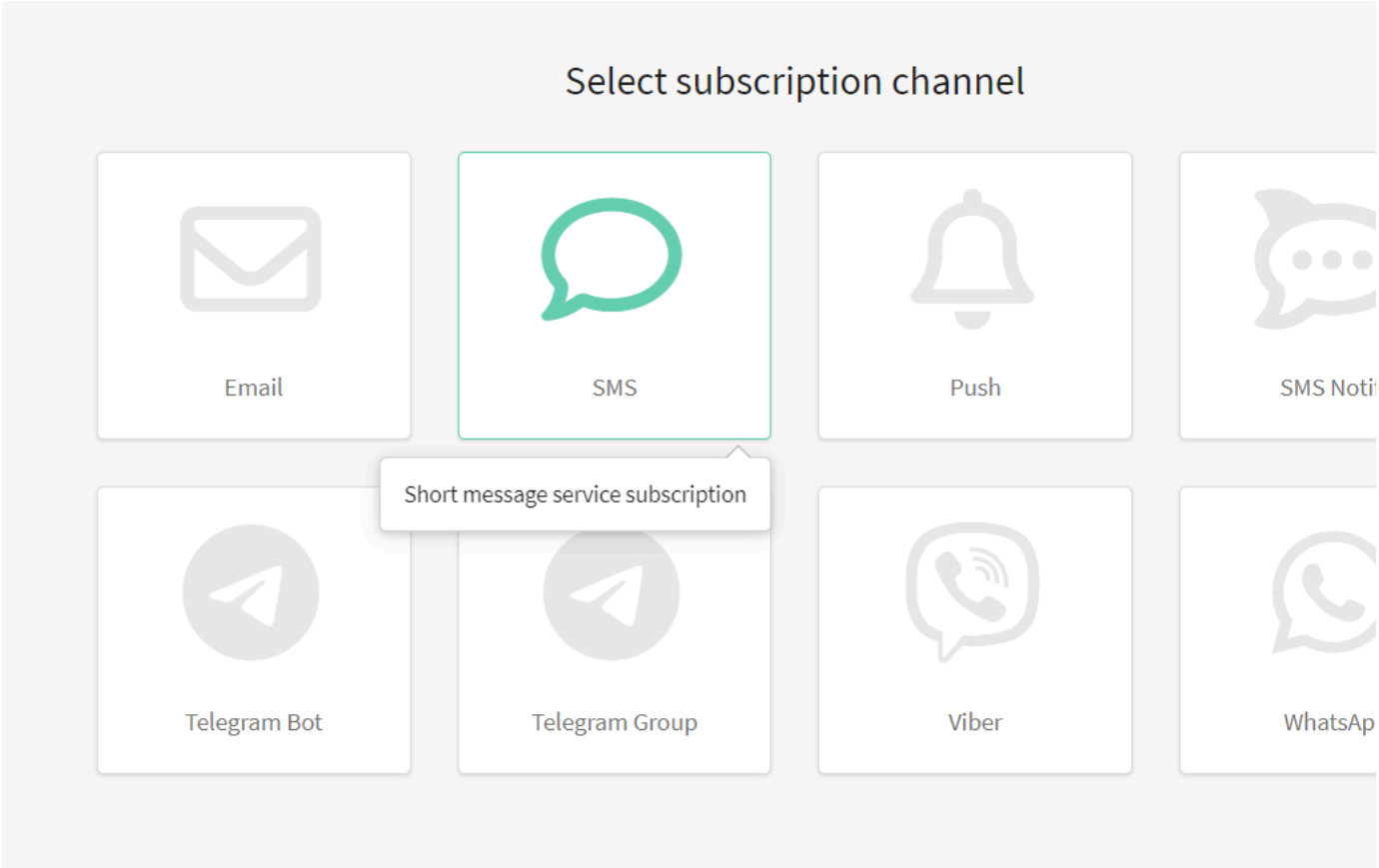
Altcraft Platform mailings are received by your subscribers. Subscriber profiles are stored in profiles dat Each customer profile has a system field "Phone". You can store your customers' phones there. However a customer in the platform.

Subscription resources

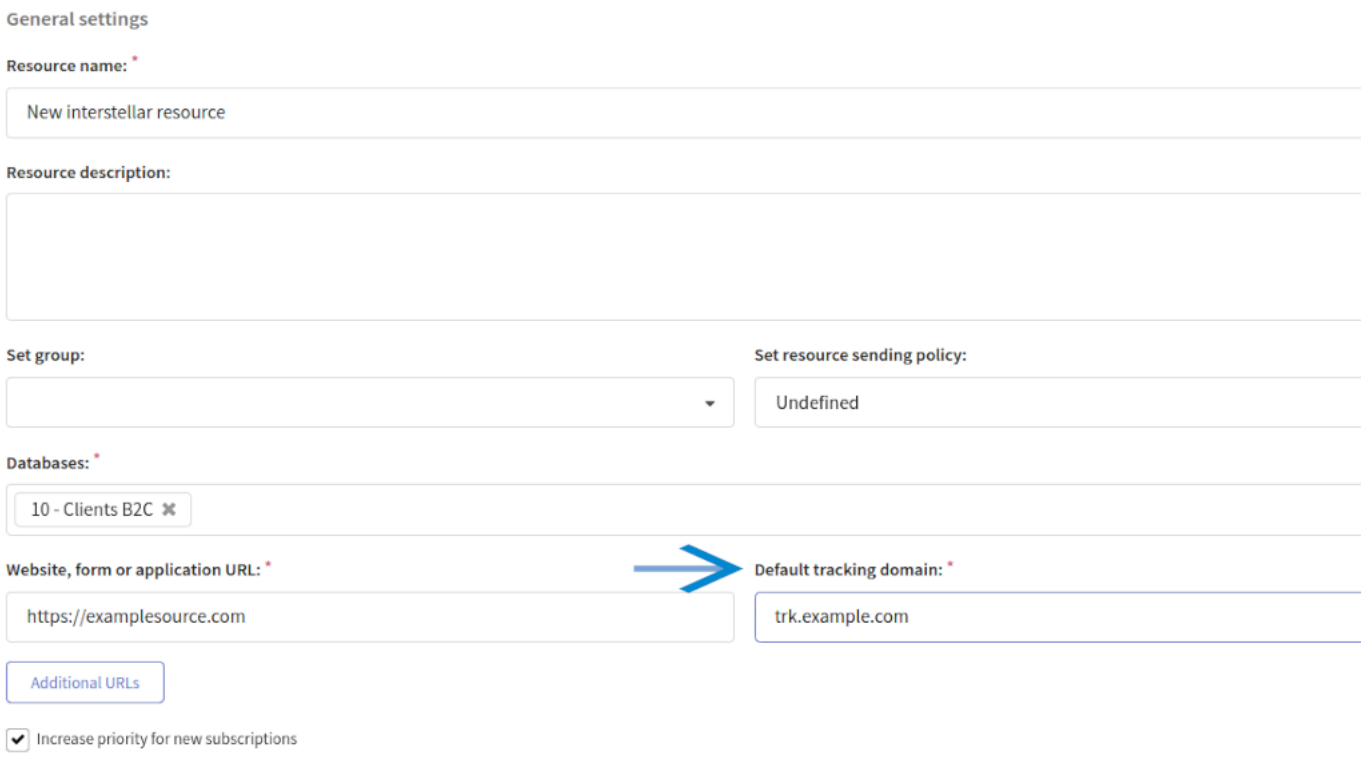
When you start your SMS mailing, you use profile subscriptions to resources. This way, you can notify th Each client profile can be subscribed to one or more resources . What can be a resource? A specific mar the type of content.



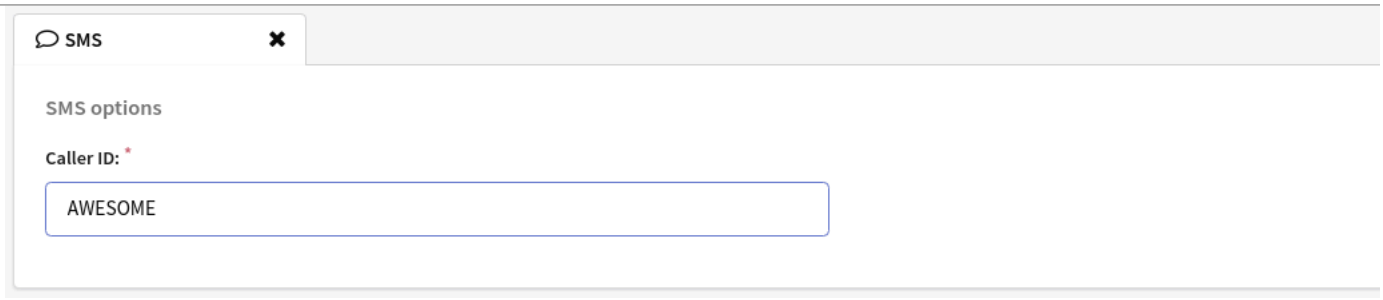
Select the corresponding item in the main menu, then create a new resource or select an existing one. S resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and spec domain will process your customers' clicks on links in SMS messages:



In the SMS options of the resource channel Caller ID agreed with your telecommunications service provi



Customer profiles and subscription to the resource

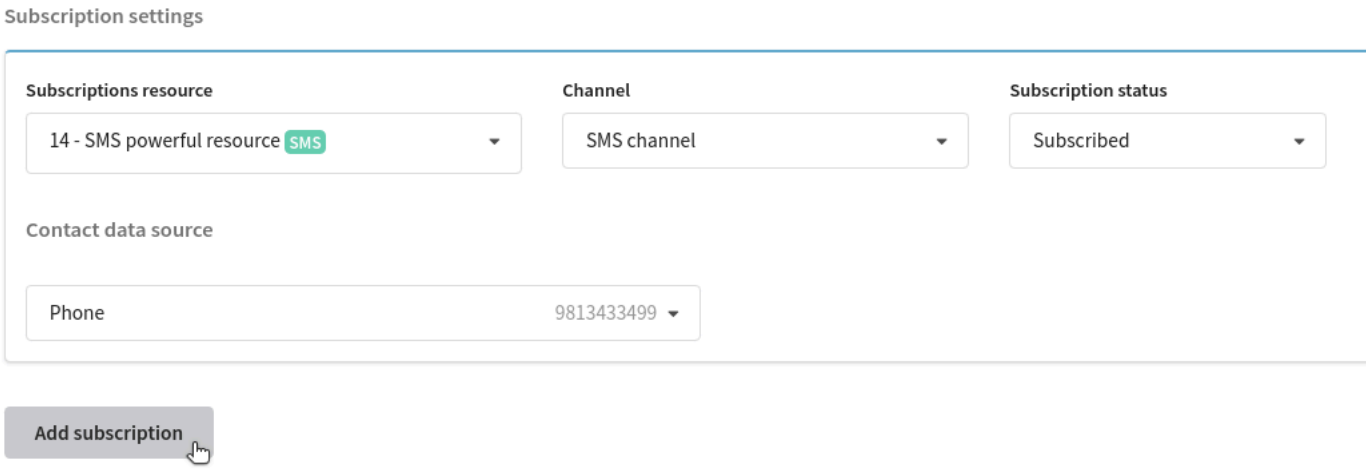
So, we've just configured the profiles database and subscription resource. Now it's time to import custor client, Altcraft Platform creates a single digital profile that includes personal data, history of actions and

You can upload customer data manually: one at a time, in a list, or from a table file. Different import met

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. To do this, select SMS as the co column with subscribers' phones:



TIP

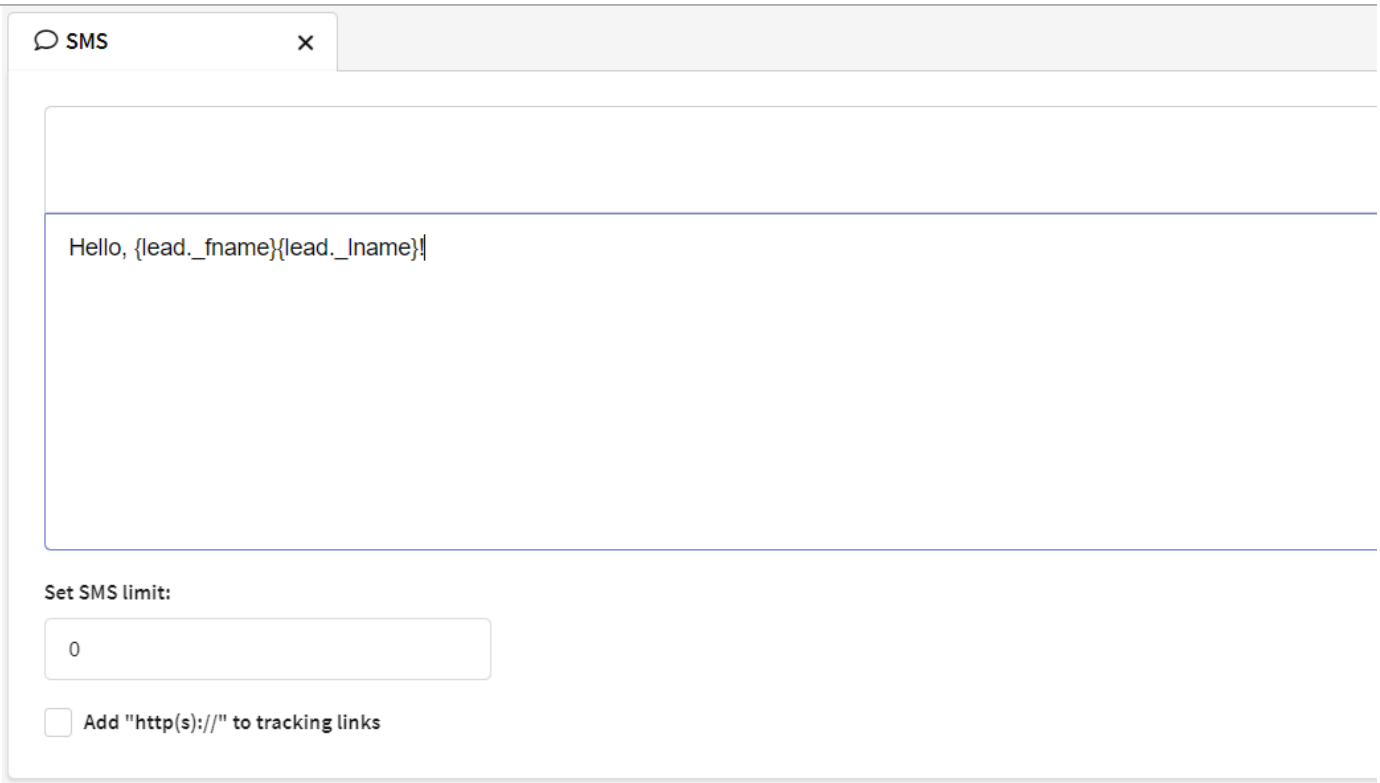
If you already have a profiles database, add them a subscription to a new resource. There are two optic

- **if customers have subscriptions to another resource with up-to-date SMS phone numbers — use bul**
- **if you need to upload to the platform SMS phone numbers for subscription, use the profile import in**

Step 3: Prepare content for submission

The content for your SMS mailings is created in Message Templates. The templates contain content for a create a new template, select the SMS channel - or add it for an existing one.

The SMS version of the template is a simple text editor where you can add data of a specific client, as w external sources:



INFO: Notice

When a template includes logical expressions or loops, the platform counts every formatting character of the final message. Each space or line break adds one character to the SMS and may affect the cost. message length, keep logical constructions on one line.

For example, instead of:

```
{if lead._city equal "Moscow"}
We invite you to a conference in Moscow on June 30!
{else}
We invite you to a conference in your city!
{end}
```

![img](/img/attachments/37848968/sms-template.png)

Use:

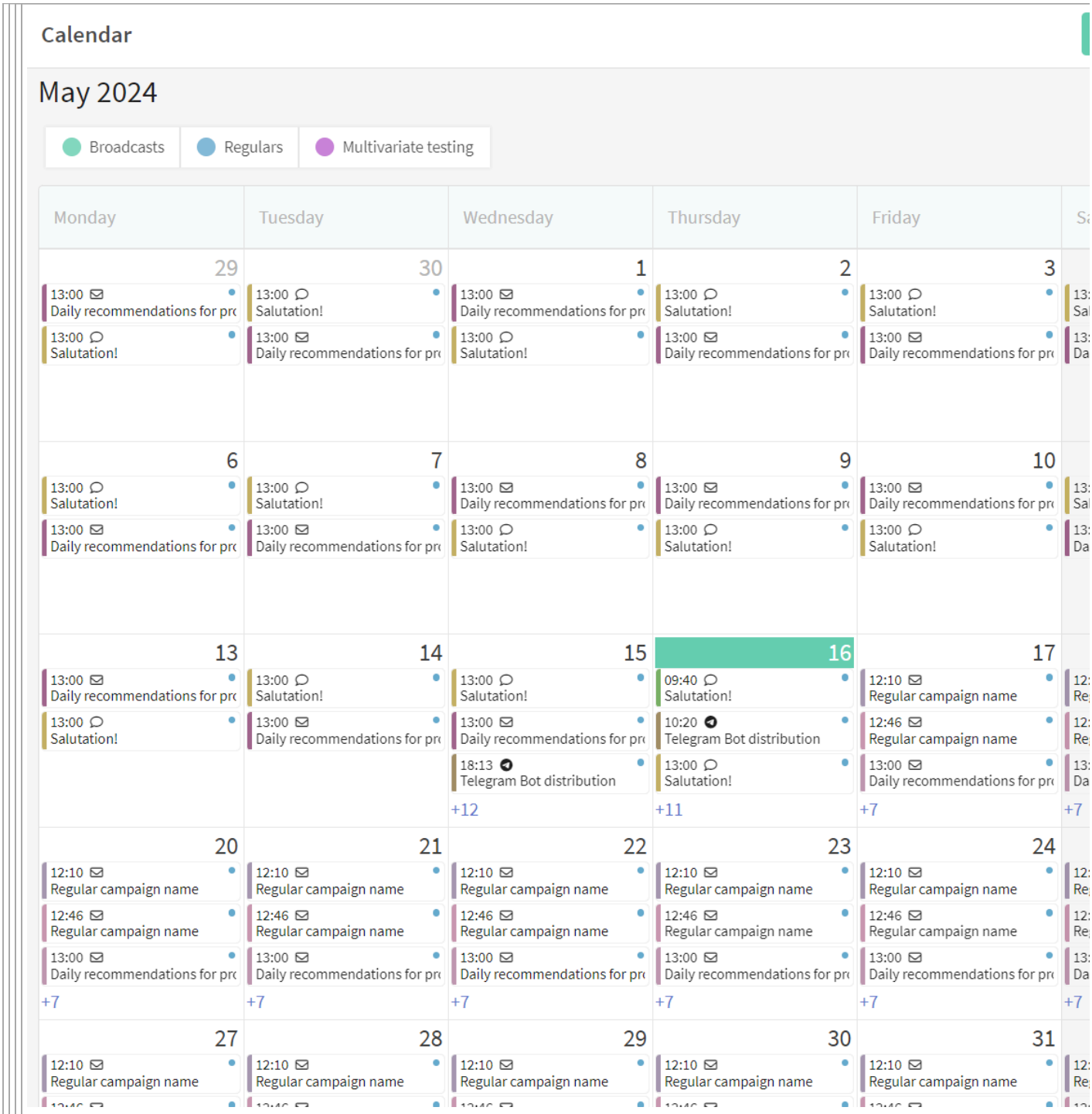
```
{if lead._city equal "Moscow"} We invite you in Moscow on June 30!{else}We invite you in your city!{end}
```

![img](/img/attachments/37848968/sms-template_2.png)

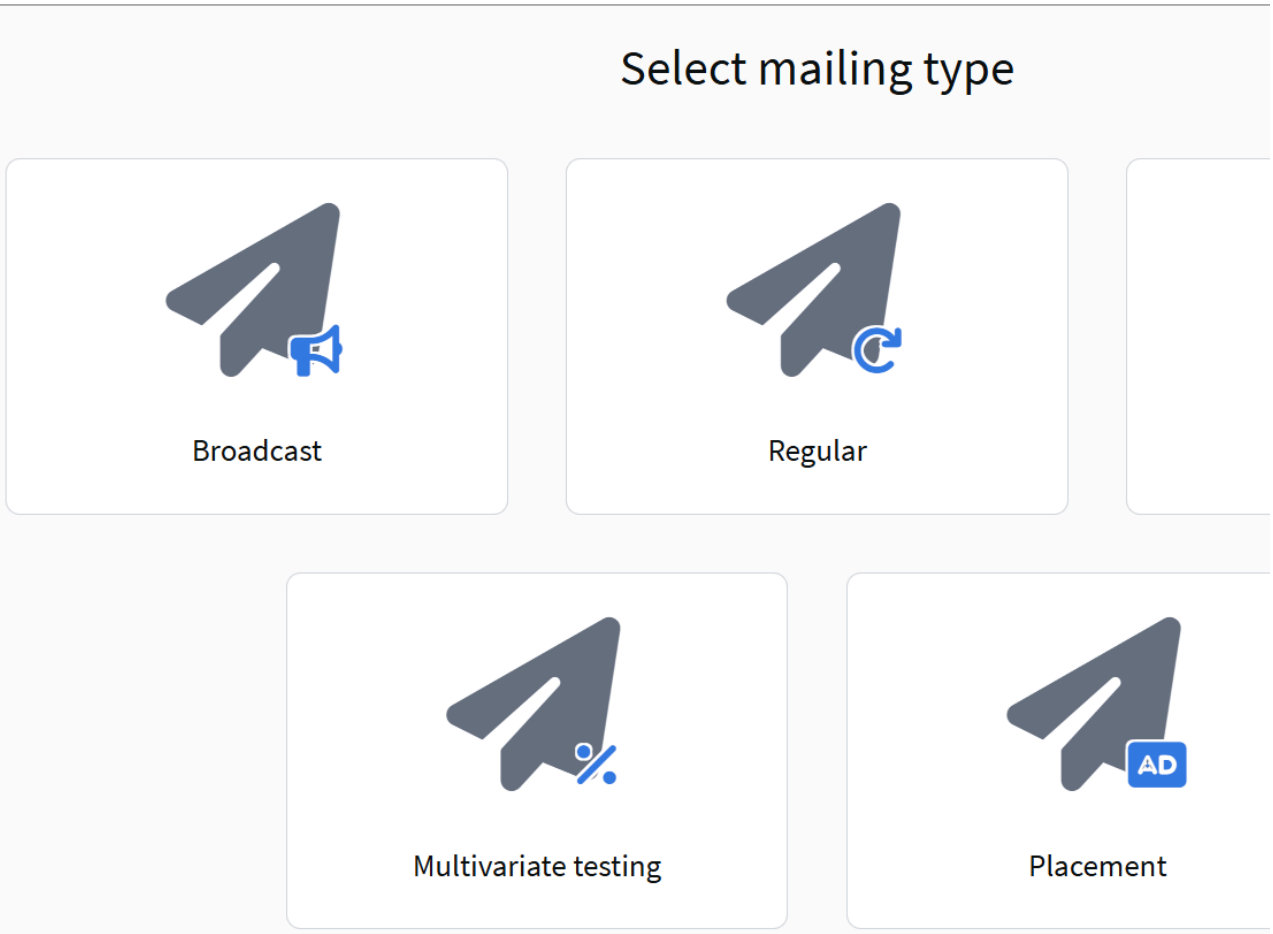
This approach keeps the logic intact and avoids unexpected SMS length growth.

Step 4: Select the type of email mailing

![img] (/img/attachments/37848968/37848997.png)	**If you want to send an SMS message, you need to use such a tool as a [**Mailing**]** Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose **Broadcasts** If the mailing is periodic, your choice is a **Regular** mailing. In the main menu select **Mailing** and open **Calendar.** This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



TIP

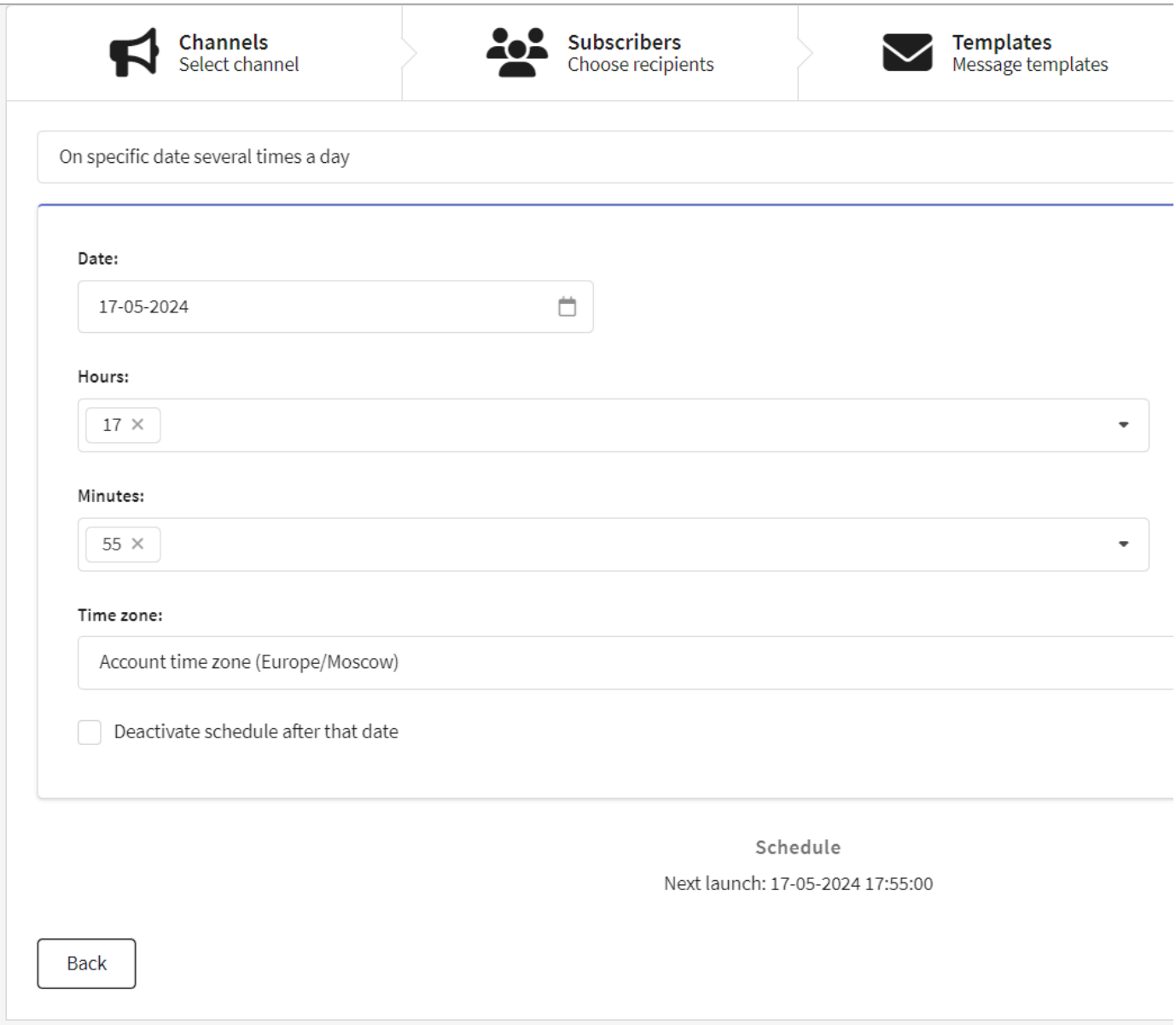
You can find details about how to set up Broadcast on this page.

TIP

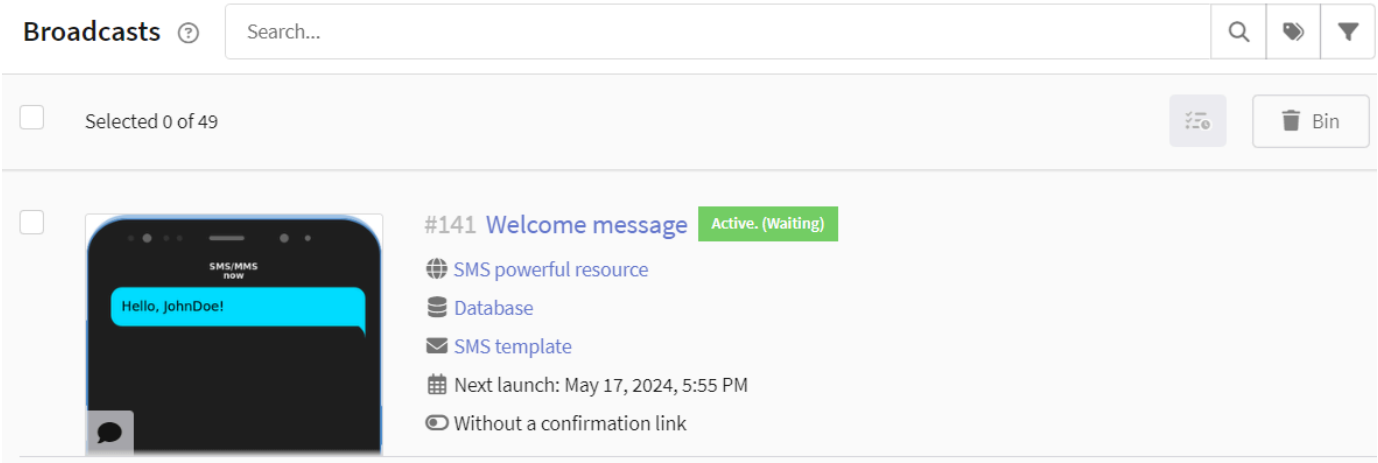
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select **SMS** as the communication channel;**
2. **Select the subscription resource you've just configured and the audience of recipients;**
3. **Add a message template to your message;**
4. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is availab**



Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



Step 5: Assess the effectiveness of the email mailing

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key event helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

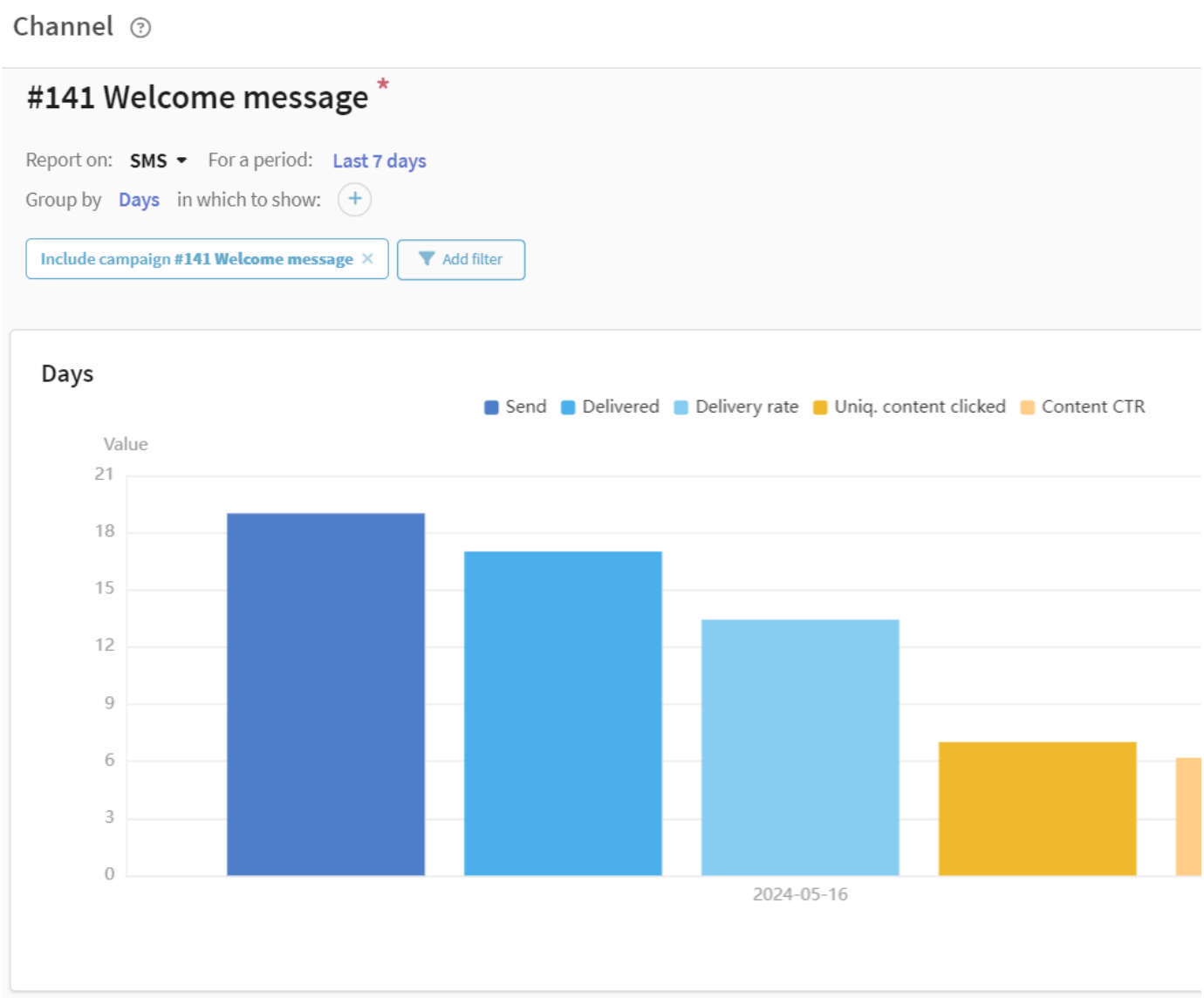
Type	Description
Mailing started	**The timestamp when the mailing execution began**
Limit	**Indicates whether a sending limit is set**
Audience source	**Contact source: List — Database Segment — Segment**
Database ID/Segment ID	**Identifier of the database being used**
Spent getting segment cursor	**Time spent loading contact data from the source (in milliseconds)**
Number of subscribers received from the source	**Total number of contacts in the specified source**
Number of subscribers found	**Number of contacts with active subscription to the selected resource**
Duplicate phones	**Count of duplicate phone numbers (excluded from sending)**
Number of SMS recipients	**Final count of successful SMS deliveries**
Mailing finished	**Completion timestamp of the mailing execution**
Total time spent	**Total duration from mailing start to completion (in seconds)**

Reports and Analytics

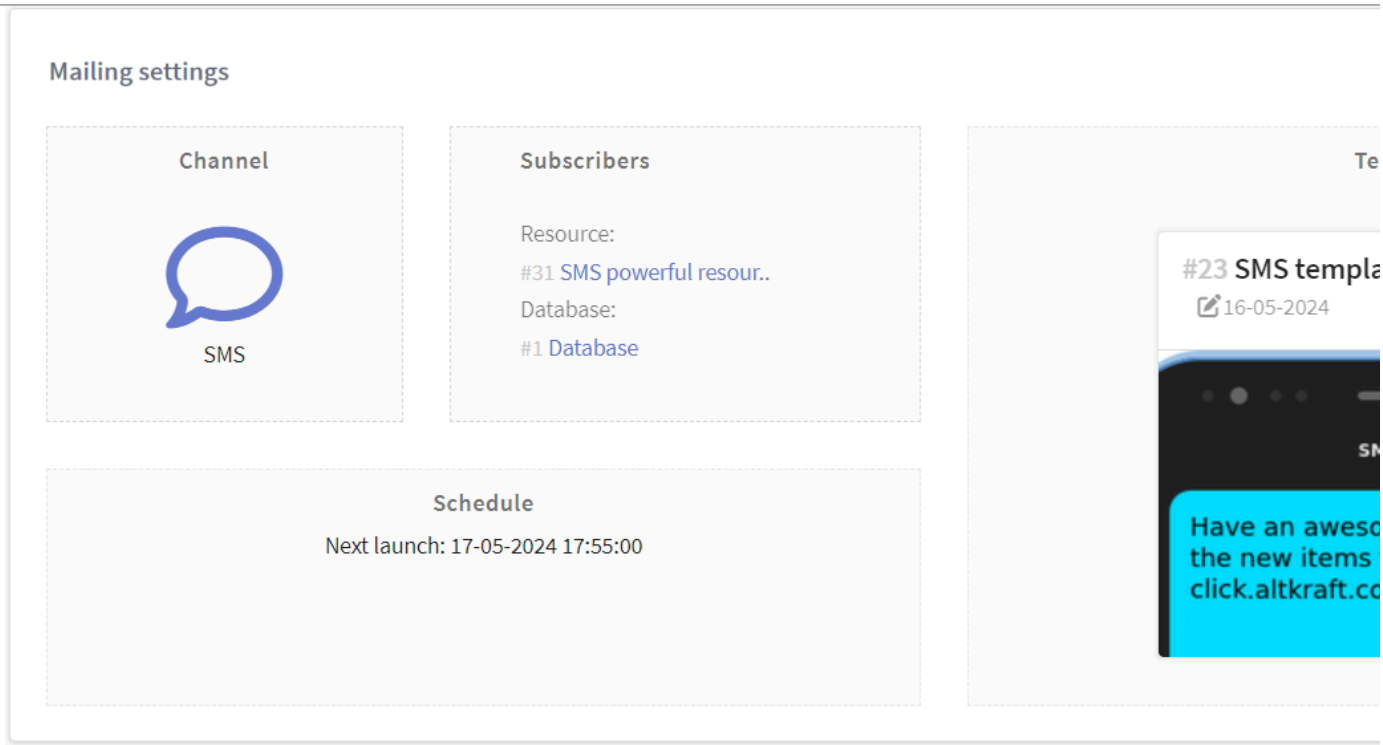
To evaluate the effectiveness of your email mailings go to the section Analytics in the main menu of the

The [channel report](#) provides analytics for one or several mailings, for one or several communication char

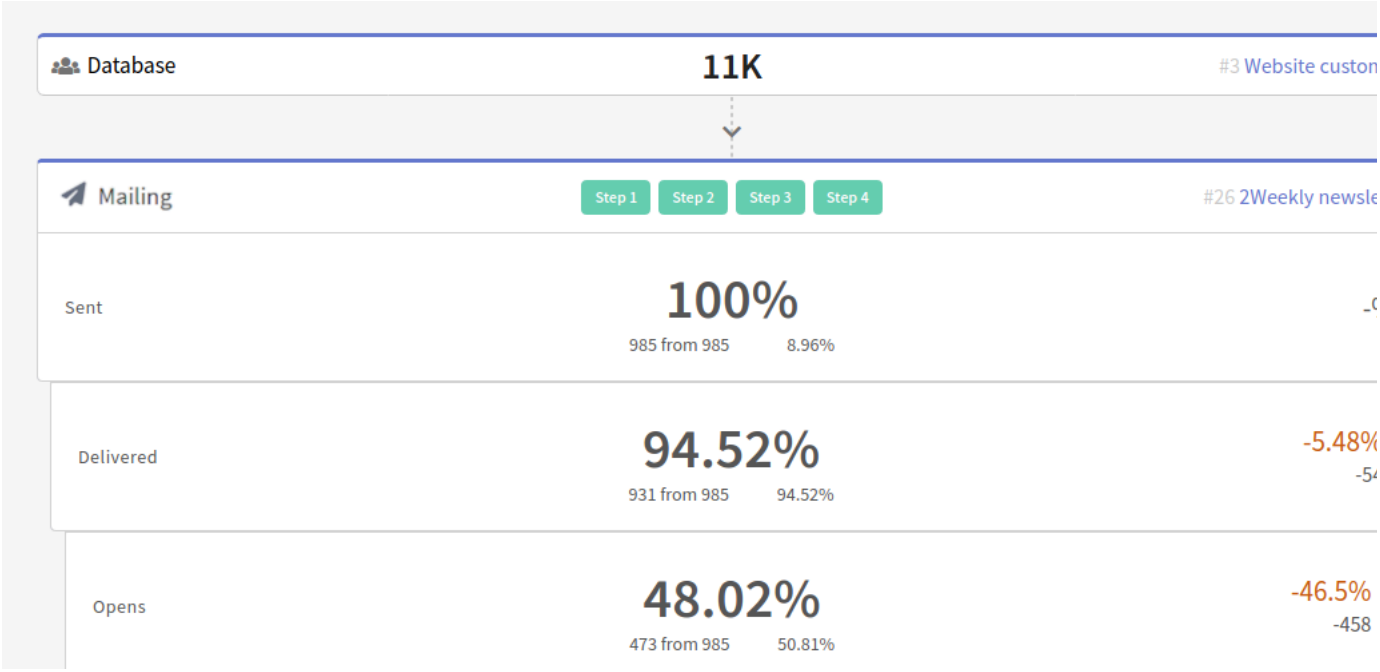
grouping, you can select a preset of indicators for SMS:



You can also go to the channel report for a specific mailing by following the link in the mailing editor:



**Actions in the SMS mailing or in the template can turn into steps in the [interactive conversion funnel](): **



Telegram

This section contains guides for working with Telegram in Altcraft Platform.

Telegram Bot — a channel for personalized messages. Subscribers interact with the bot individually, rec mailings. Suitable for customer communication, sending receipts, codes, order status notifications. Requ chat_id to the platform.

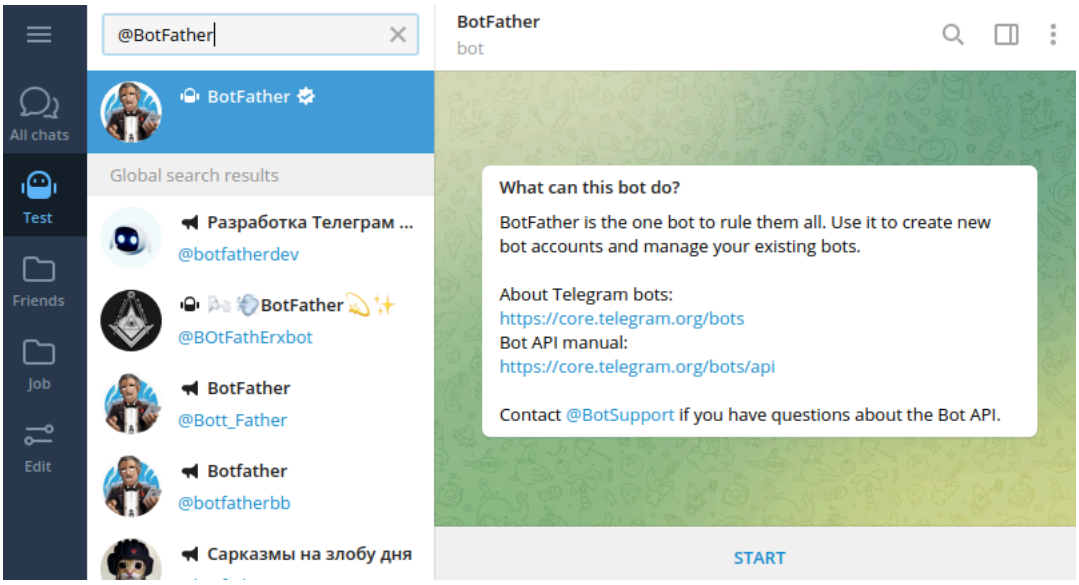
Telegram Group — a channel for publishing posts in groups and channels via placements. Suitable for ma public chats. Does not require individual profile subscriptions — just add the bot to the group with admin the channel or group.

Telegram Bot: a Guide for the First Mailing

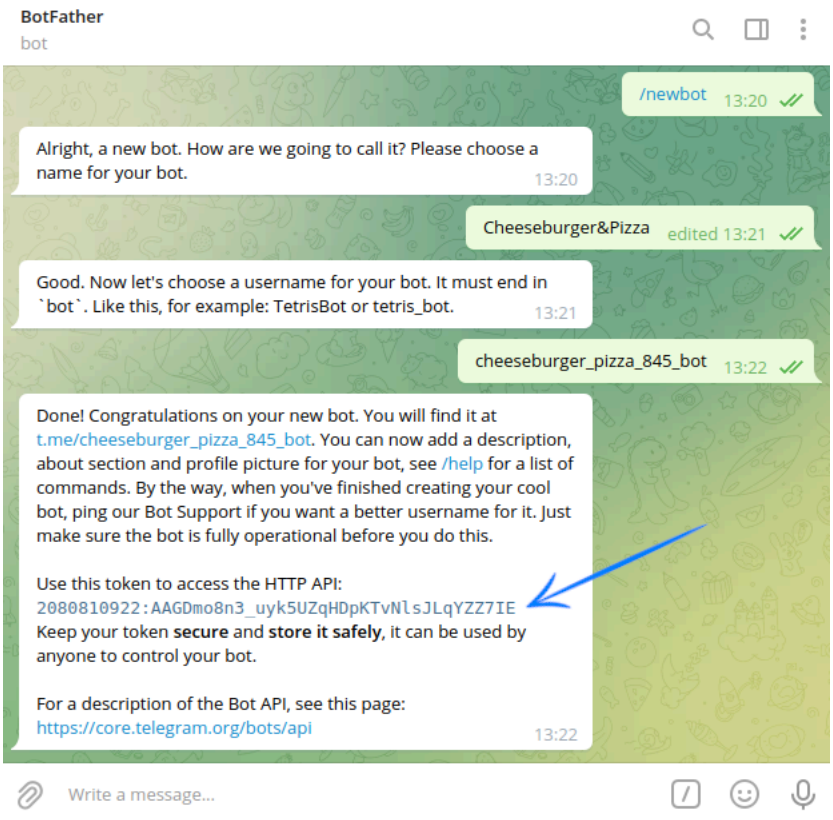
![img] (/img/attachments/56249058/56249069.jpg)	A bot is a direct channel that allows you to personally send messages to subscribers who have entered into a chat with the bot. bot.

Step 1. Create a bot and get a token

1. Log in to Telegram, enter `@BotFather` in the search box and select the desired bot.



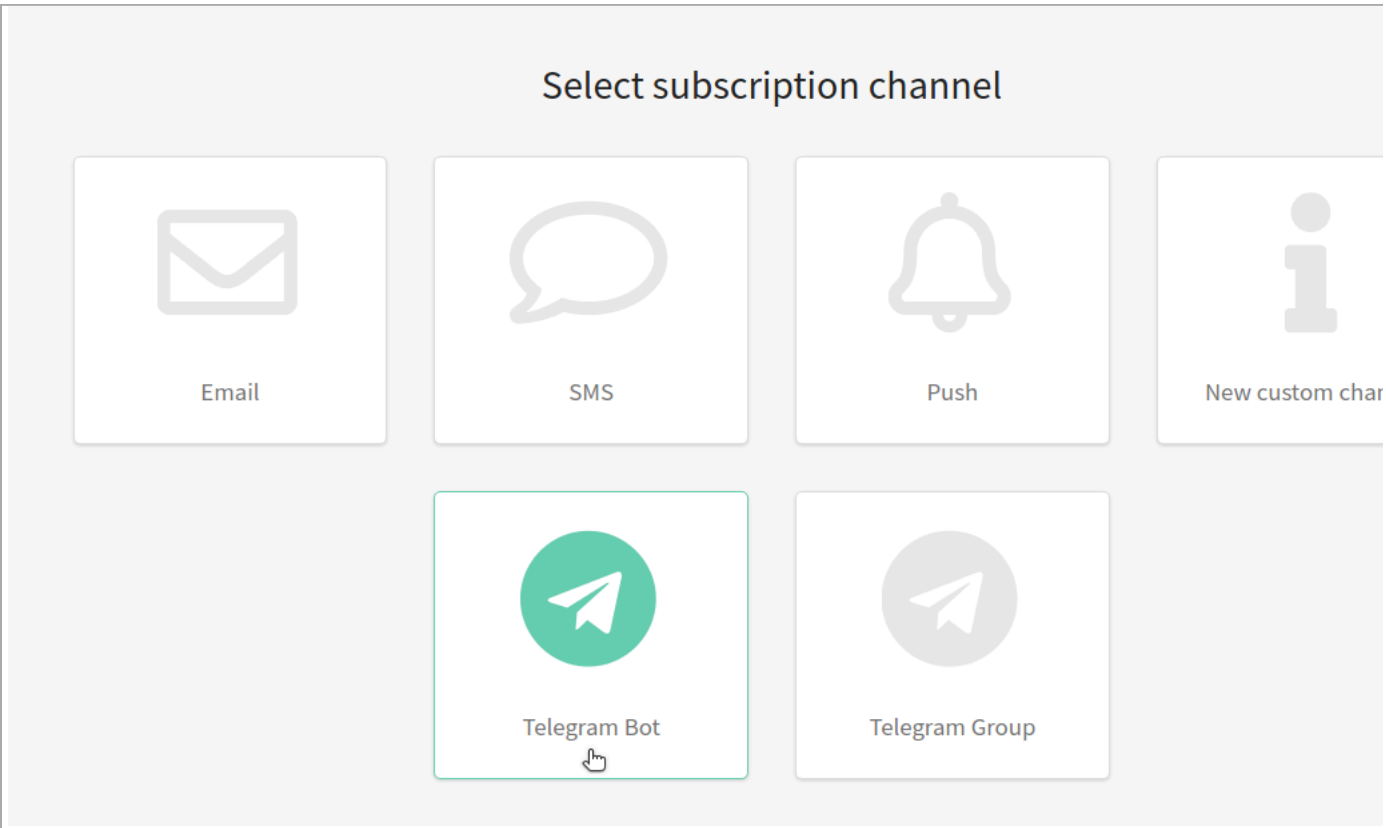
2. Start a dialogue with the bot by clicking the START button in the chat. Then send the `/newbot` comman username for your new bot. Please note that the username must end with `"bot"` or `"Bot"`. Copy the recei



Step 2. Subscribe your customers to this Telegram Bot

Go to the “Data” → “Profile database” and create a new database. The data of users subscribed to the bo Telegram IDs.

Then in the “Resources” section create a new resource or select a pre-existing one. Use Telegram Bot a



Add the necessary databases in the basic settings. Specify the default tracking domain that will process your Telegram Bot messages.

General settings

Resource name: *

Telegram bot resource

Resource description:

Set group:

Sales and promotion ✖

Set resource sending policy:

Undefined

Databases: *

26 - B2C clients ✖ 17 - Potential customers ✖

Website, form or application URL: *

https://examplesource.com

Default tracking domain: *

trk.example.com

Additional URLs

Increase priority for new subscriptions

Add the bot token copied from Telegram and save the resource:

Now you need to import user data into the database by subscribing them to the resource.

INFO

Please note that the platform only sends Telegram mailings. Subscriber data is collected and transmitt
Read more about how to do this in the Telegram documentation.

Example of getting a user's chat_id using the getUpdates method

Method: POST

URL:

`https://api.telegram.org/bot<bot_token>/getUpdates`

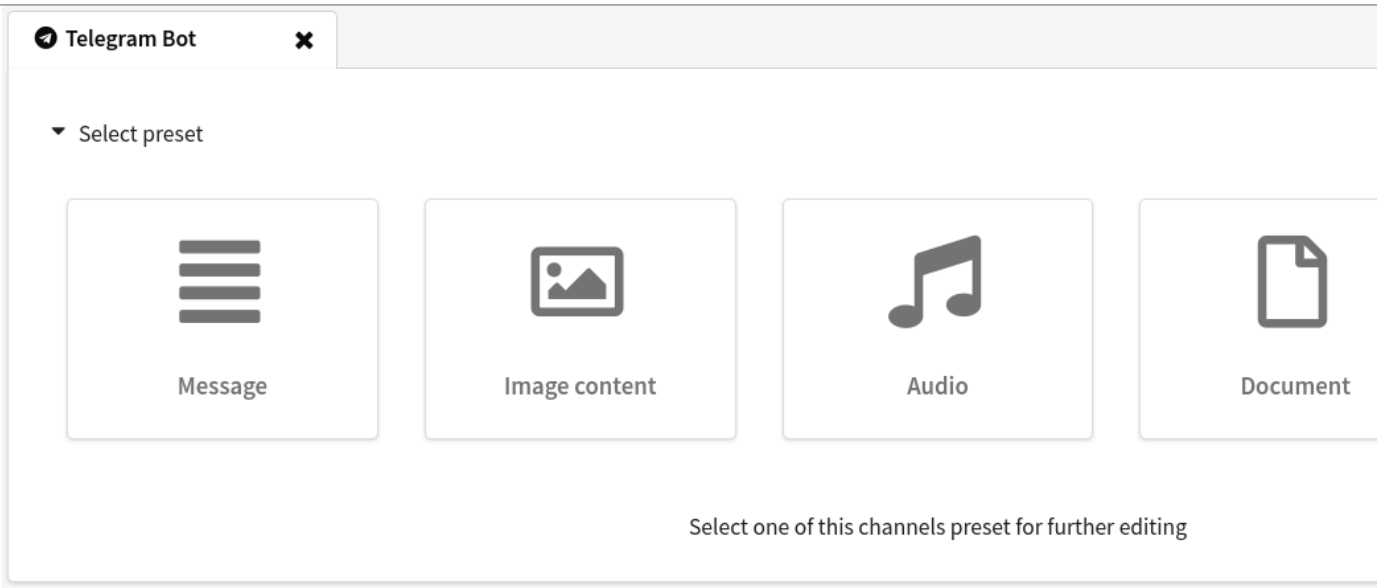
Response:

```
{
    "ok": true,
    "result": [
        {
            "update_id": 262756455,
            "message": {
                "message_id": 47,
                "from": {
                    "id": <chat_id>,
                    "is_bot": false,
                    "first_name": "John",
                    "last_name": "Doe",
                    "username": "john_doe",
                    "language_code": "en"
                },
                "chat": {
                    "id": <chat_id>,
                    "first_name": "John",
                    "last_name": "Doe",
                    "username": "john_doe",
                    "type": "private"
                },
                "date": 1652284613,
                "text": "Hello, world"
            }
        }
    ]
}
```

`<chat_id>` **is a unique identifier for a chat with a specific user. You need to pass it to the platform.**

Step 3. Create a template

Go to the "Templates" section. When creating a new template, select the Telegram Bot channel and add template. Select the content type in the Telegram version of the template:

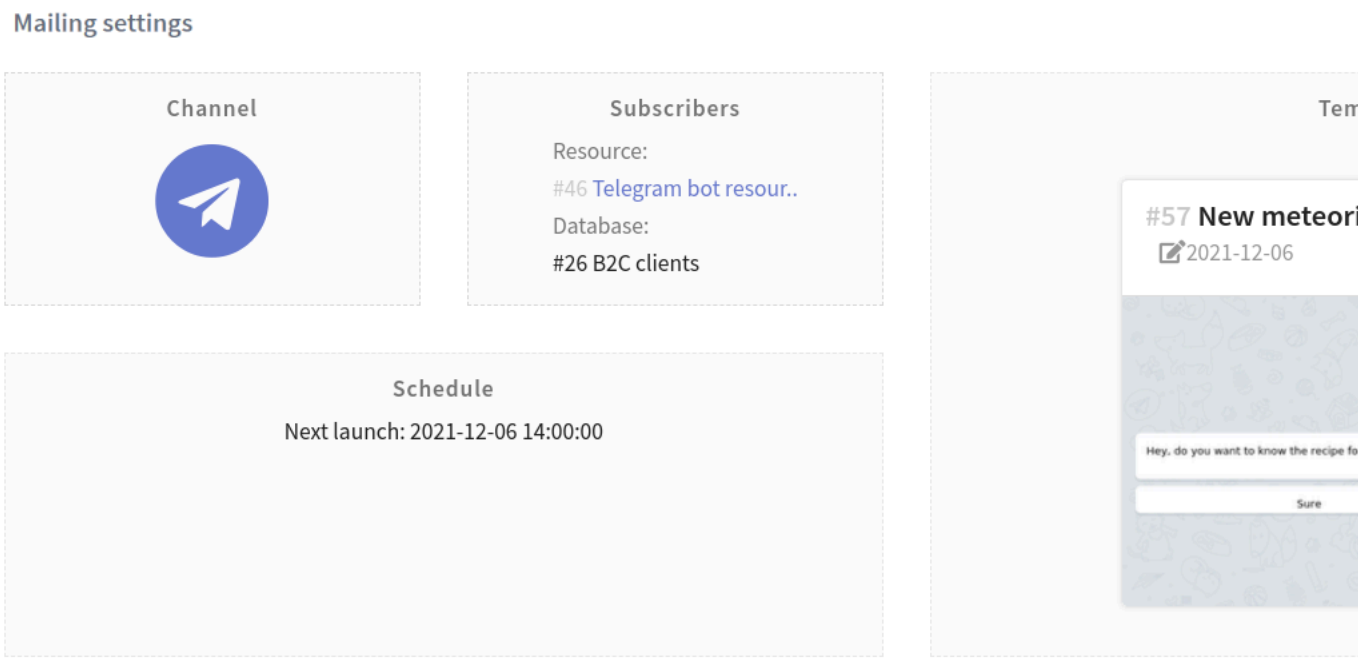


You can add data of a particular customer, as well as dynamic text content from external sources.

Step 4. Configure and launch a mailing

Open the section “Mailings” and select the distribution method: broadcast, regular or trigger mailing. Create a new mailing. After filling in the basic settings, you need to:

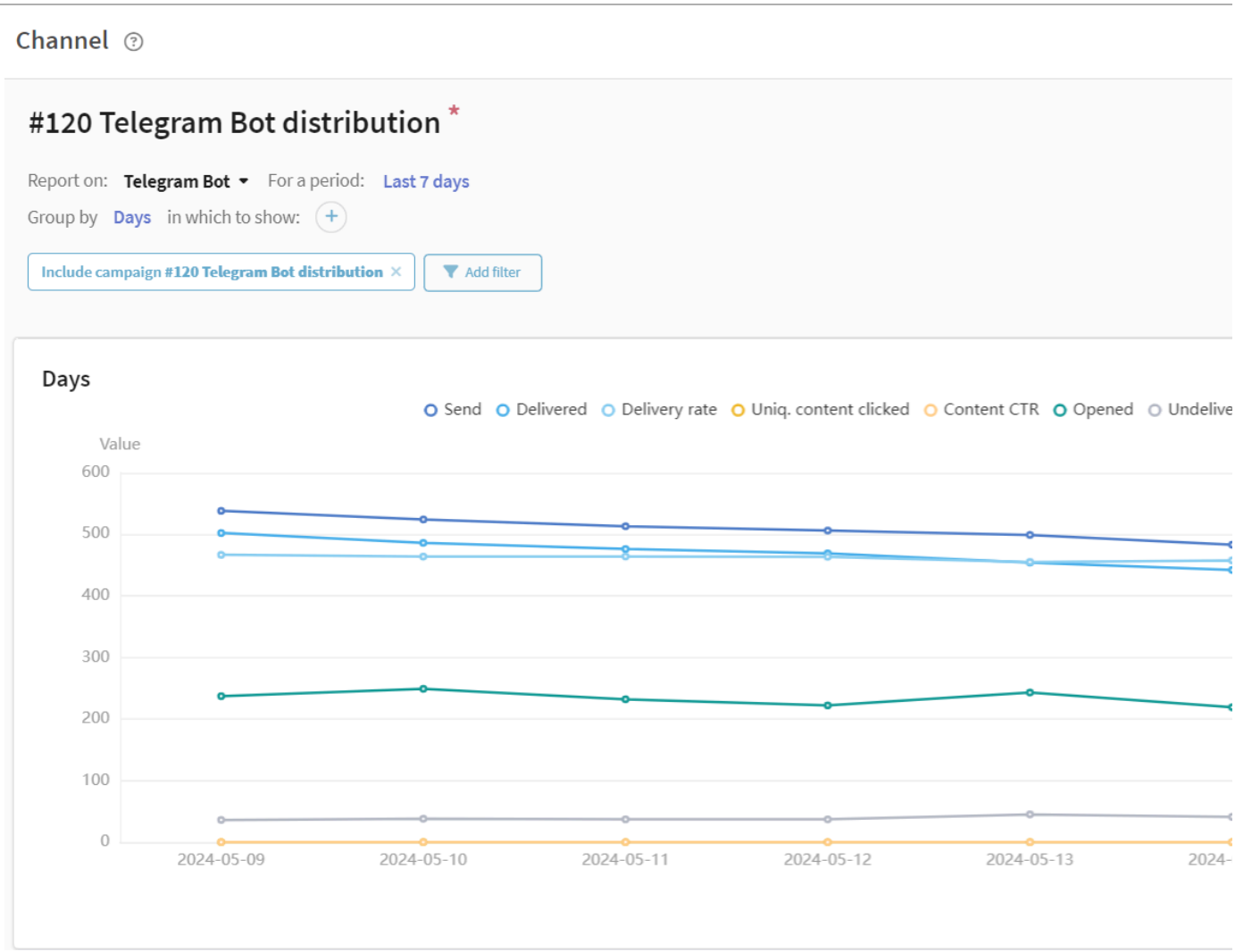
1. **Choose Telegram Bot as a communication channel**
2. **Select a previously configured subscription resource and target audience**
3. **Add a message template to your mailing**
4. **Set a schedule**



When the mailing is fully configured, save it. It will be displayed in the list of mailings of the correspondi on the right side of each entry. By its means, you can send test message

and, if everything is good, activate your actual mailing

:



Step 5. Evaluate your mailing effectiveness

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key event helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Descriptio
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resou**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push campaign use the Analytics section in the main menu of the p the analysis of one or more mailings.

Name	Send	Delivered	Delivery rate	Opened	Read
Total	3 533	3 257	92,19 %	1 603	1 295
2024-05-15	470	428	91,06 %	201	163
2024-05-14	483	442	91,51 %	219	176
2024-05-13	499	454	90,98 %	243	195
2024-05-12	506	469	92,69 %	222	180
2024-05-11	513	476	92,79 %	232	189
2024-05-10	524	486	92,75 %	249	200
2024-05-09	538	502	93,31 %	237	192

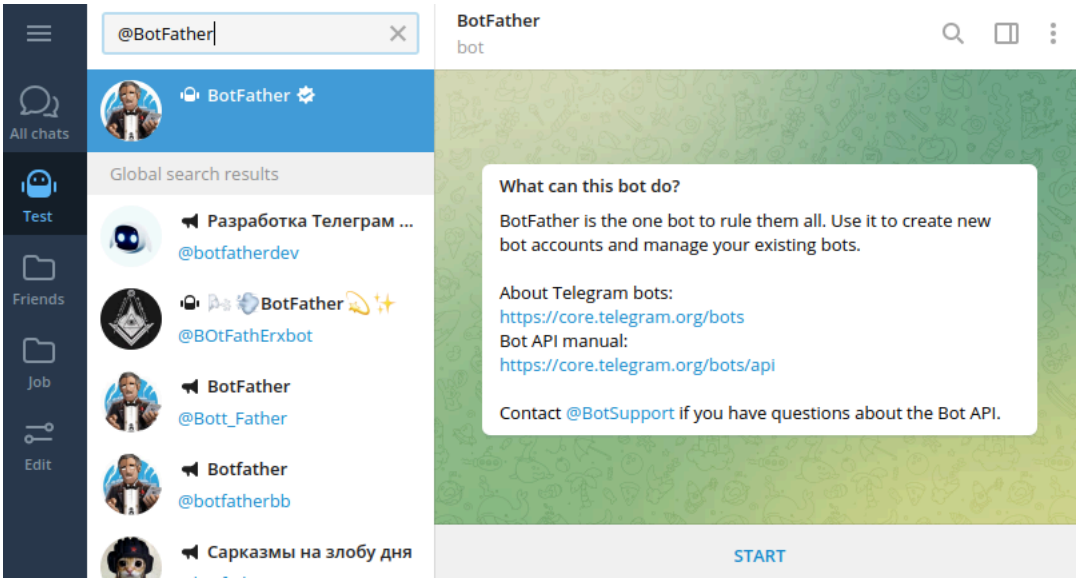
To make grouping easier, you can select a preset of indicators for Telegram Bot. You will see the data or undelivered messages in the table, as well as information on unique clicks and openings.

Telegram Group: a Guide for the First Mailing

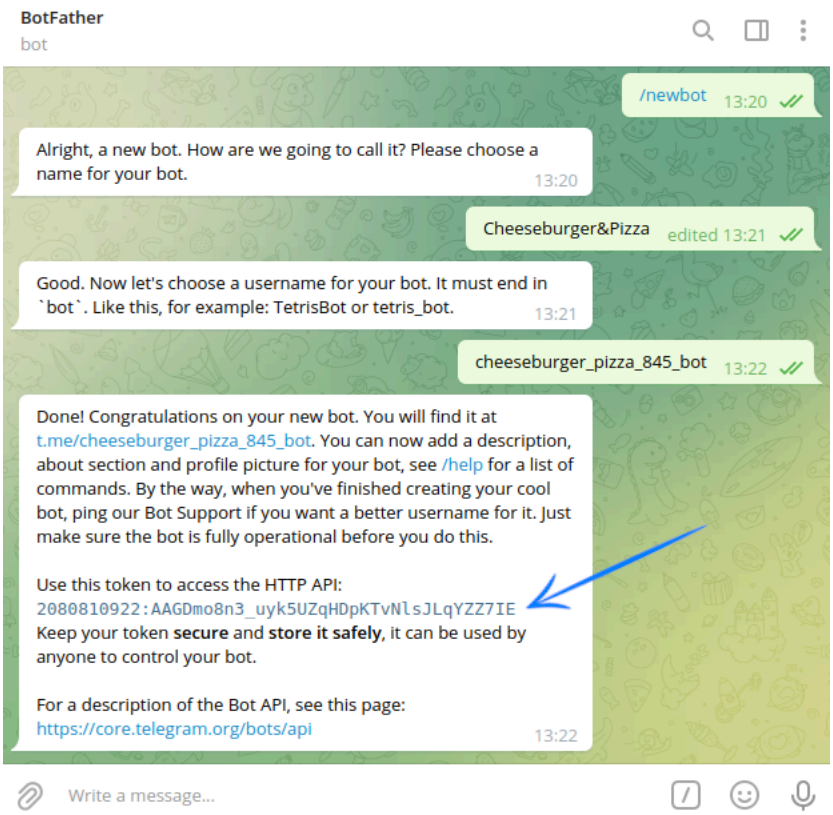
![img] (/img/attachments/56249074/56249076.jpg)	**Telegram channel is a feed of text messages, which are complemented by media files and emoji. Channel members can commen channels, two-way communication is available. This means that each group user can send a message to the chat.** The Telegram Group channel will allow you to set up automatic publication of posts in channels and groups of the messaging app using placement mailings. </td> </tr>

Step 1. Create a bot and get a token

1. Log in to Telegram, enter `@BotFather` in the search box and select the desired bot.



2. Start a dialogue with the bot by clicking the START button in the chat. Then send the `/newbot` comman username for your new bot. Please note that the username must end with `"bot"` or `"Bot"`. Copy the recei



Step 2. Add the bot to the group or channel and get a chat_id.

1. 1. Create a group or channel. You will find more detailed information on how to do this in the Telegram

2. Add the bot to a group or channel. Assign admin rights to the bot.

3. Send a test message to chat using an API request. This can be done through Postman or any other API

Method: POST

URL: `https://api.telegram.org/bot[BOT_API_KEY]/sendMessage?chat_id=[CHANNEL_NAME]&text=[MESSAGE_TEXT]`

- `[BOT_API_KEY]` — **the bot token received in the conversation with** `@BotFather`.
- `[CHANNEL_NAME]` — **channel or group name using** `@`.
- `[MESSAGE_TEXT]` — **message text.**

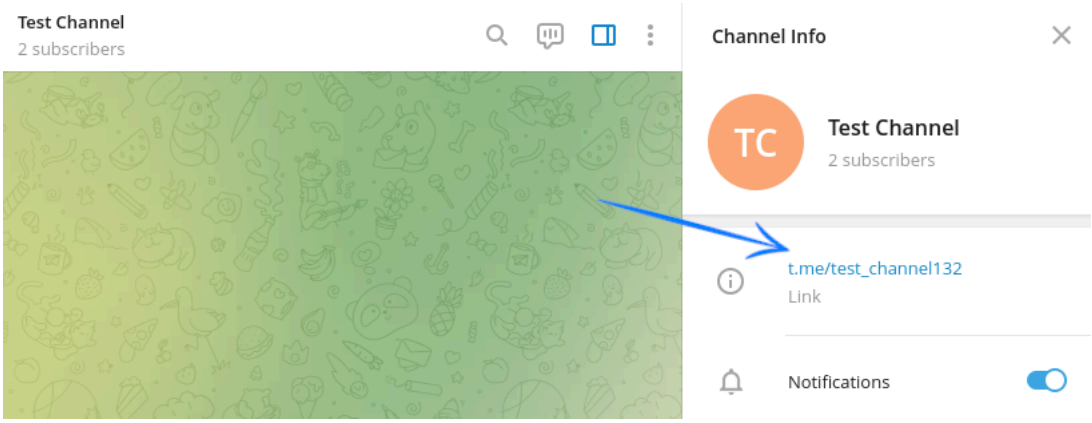
Example:

`https://api.telegram.org/bot2080910120:AAGfho8n3_uyk5trh4DpKTvNlsJLkadsZ7IE/sendMessage?chat_id=@new_test_group132&text=`

INFO

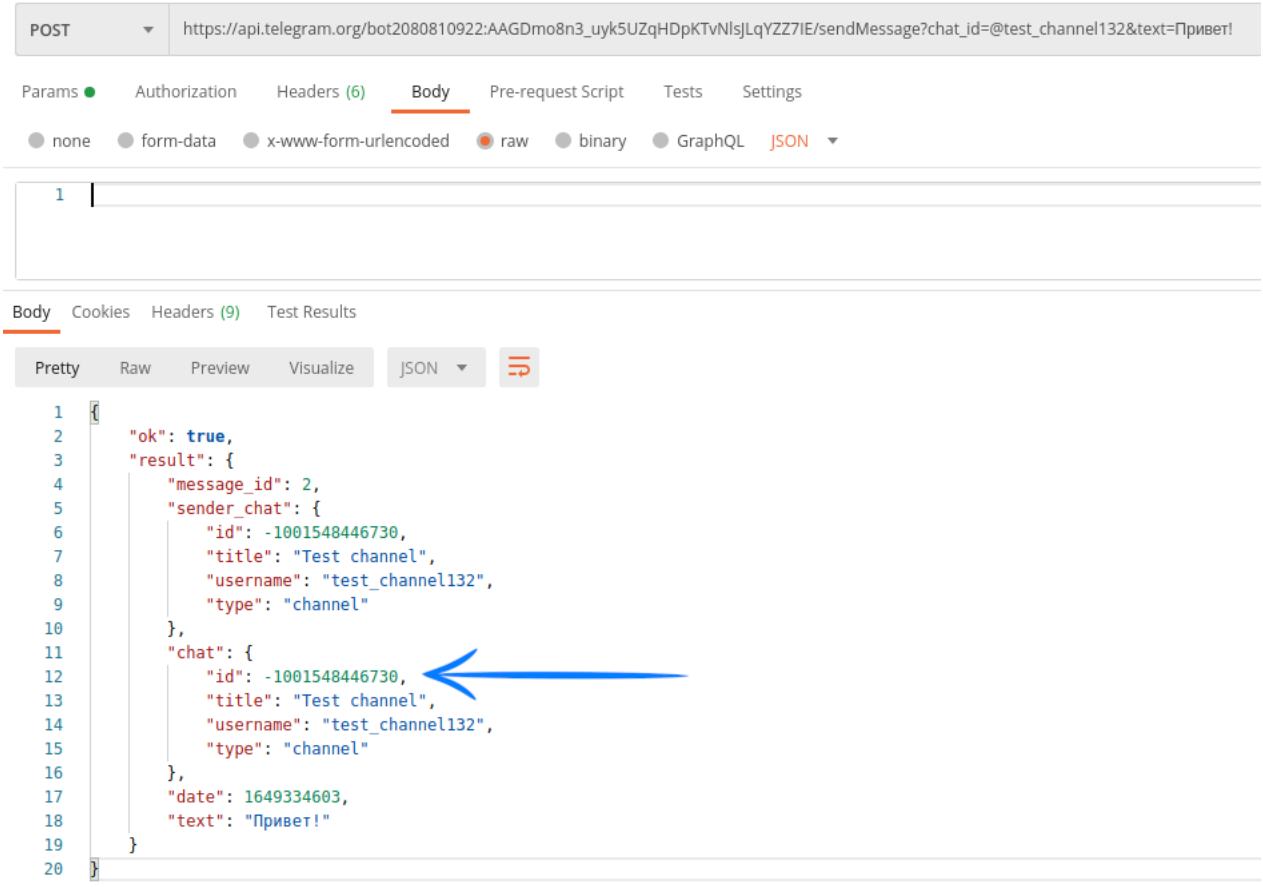
Where can I find the name of the channel or group?

The name is embedded in the link to the channel or group. For example, if the link is `t.me/test_channel132` **become the name of the channel —** `test_channel132`.



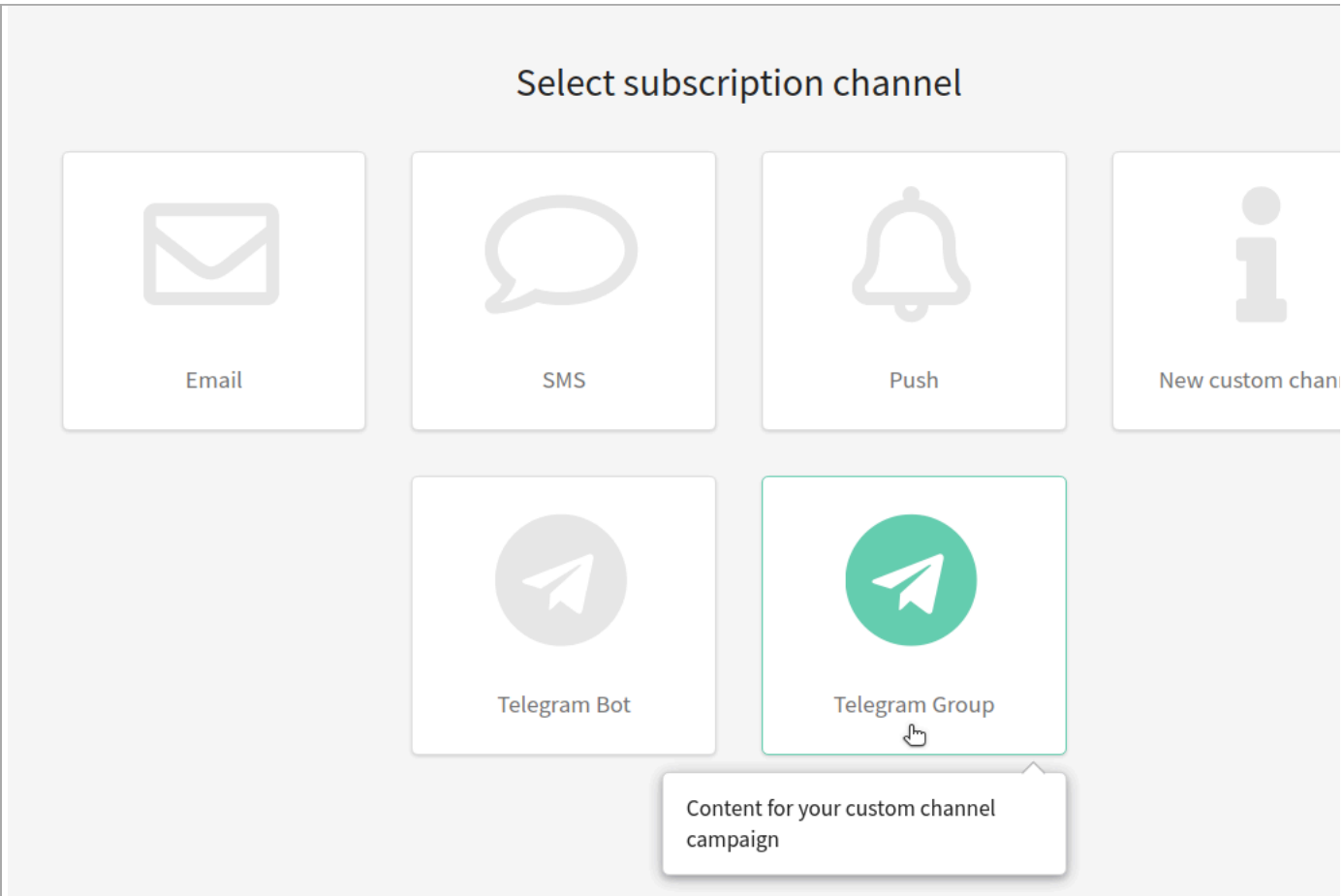
Response:

In the body of the response, you will receive the `chat_id.` **It must start with** `-100`.



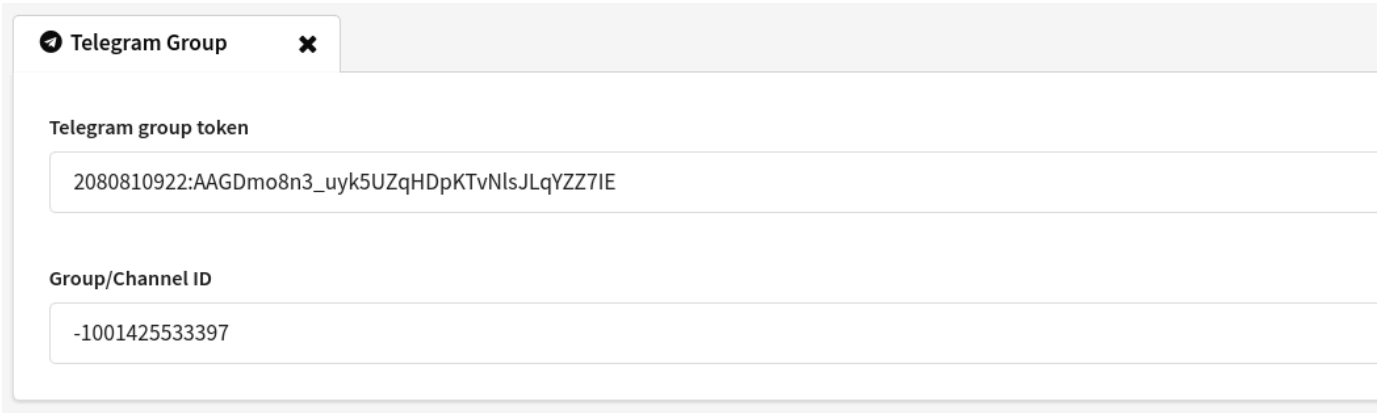
Step 3. Create a resource

Go to the “Resources” section create a new resource or select a pre-existing one. Use Telegram Group a



Add the necessary databases in the basic settings. Specify the default tracking domain that will process your Telegram Bot messages.

Add the bot token and group/channel ID copied from Telegram and save the resource:

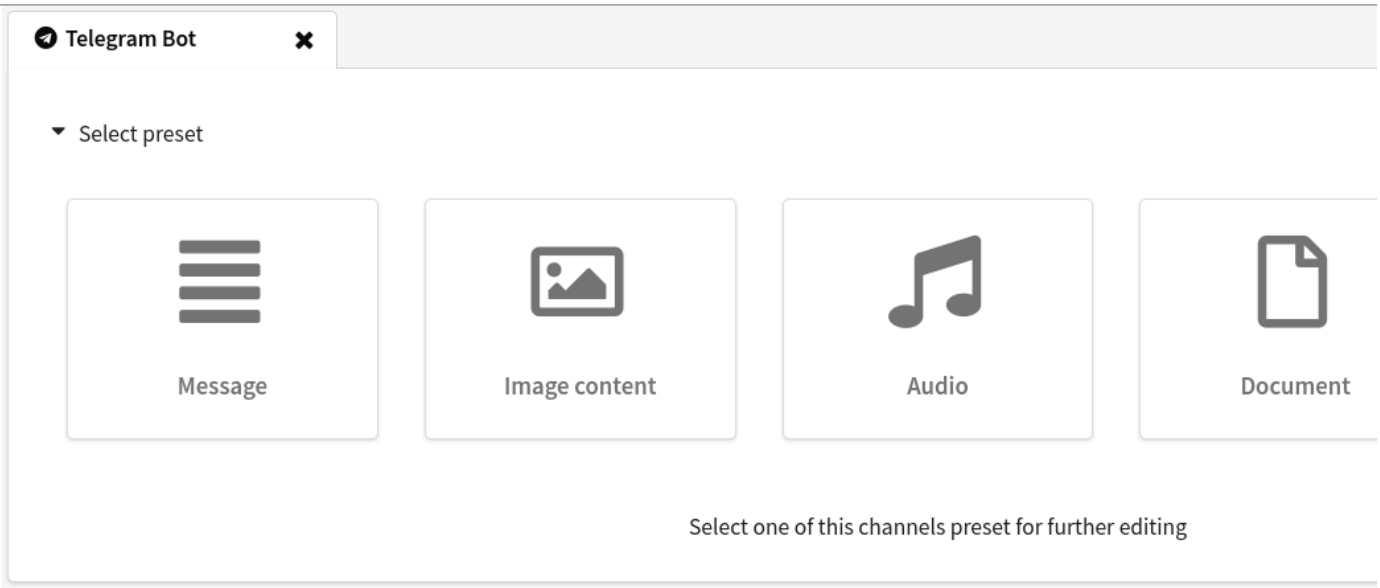


Now you need to import user data into the database by subscribing them to the resource.

Please note that the platform only sends messages. Subscriber data is collected and transmitted to the

Step 4. Create a template

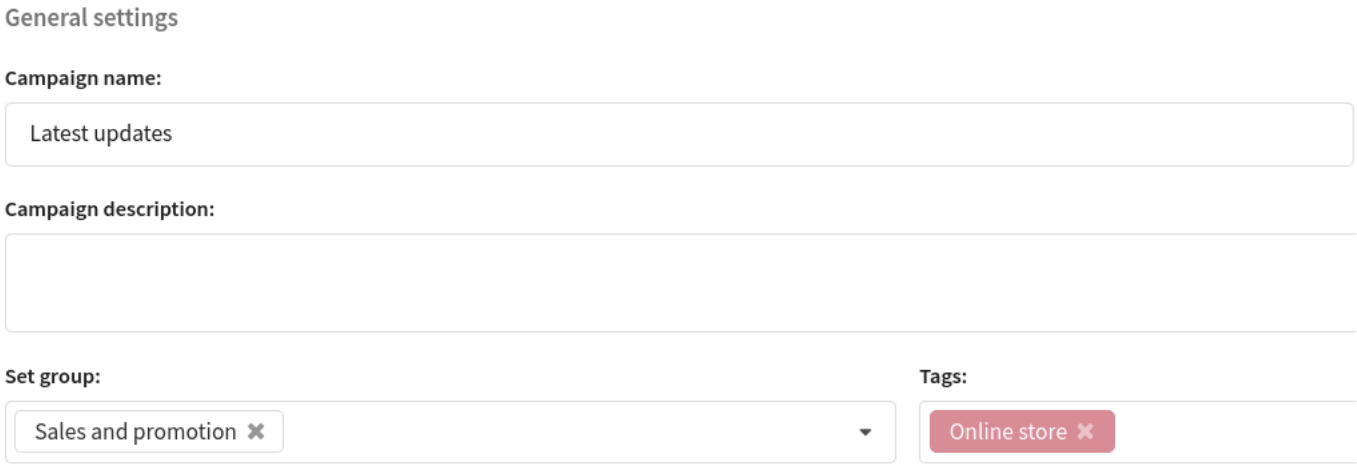
Go to the "Templates" section. When creating a new template, select the Telegram Bot channel and add template. Select the content type in the Telegram version of the template:



You can add data of a particular customer, as well as dynamic text content from external sources.

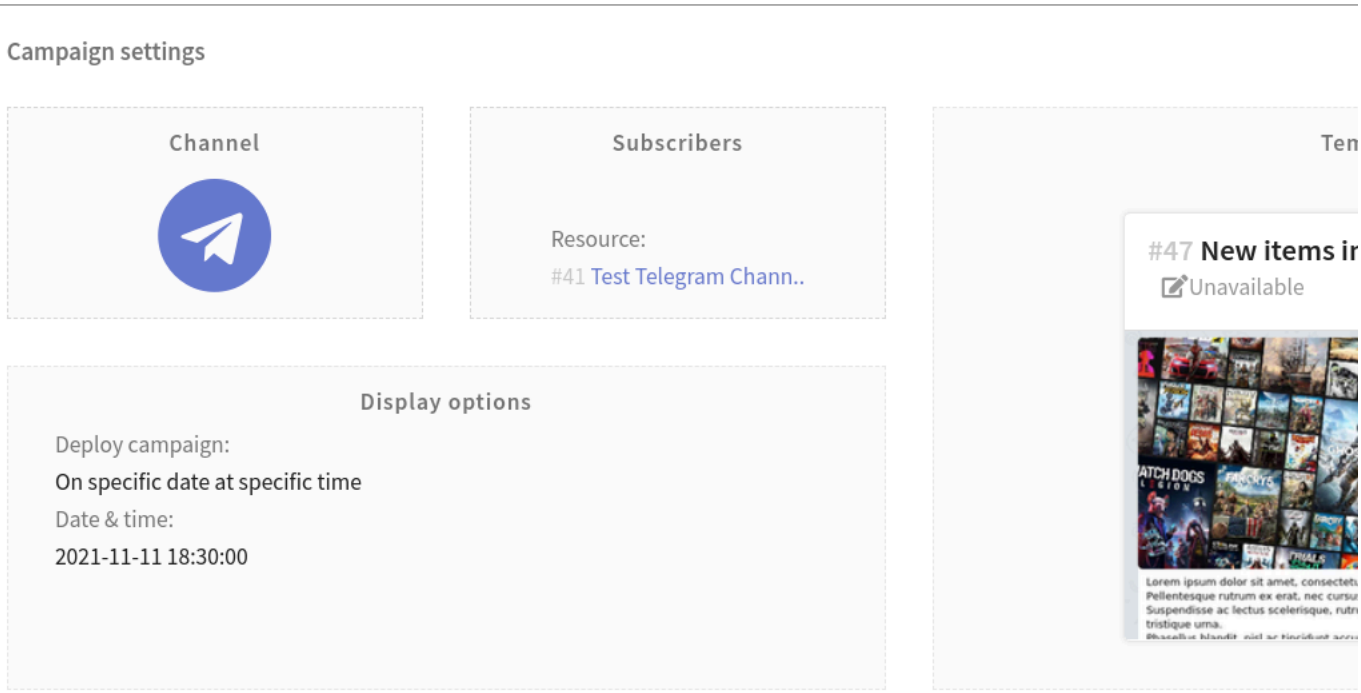
Step 4. Configure and launch a placement mailing

Open the section “Mailings and select Placements”. Create a new mailing.



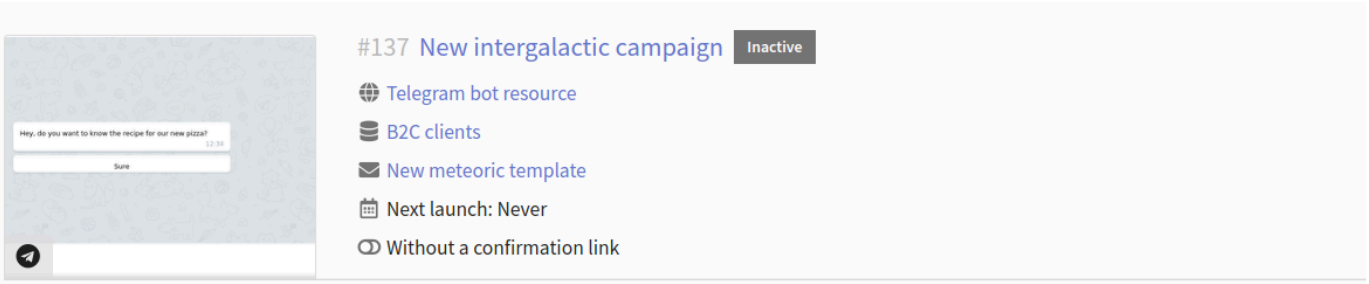
After filling in the basic settings, you need to:

1. **Choose Telegram Group as a communication channel.**
2. **Select a previously configured subscription resource.**
3. **Add a message template.**
4. **Add the date and time the post will be posted to the group.**



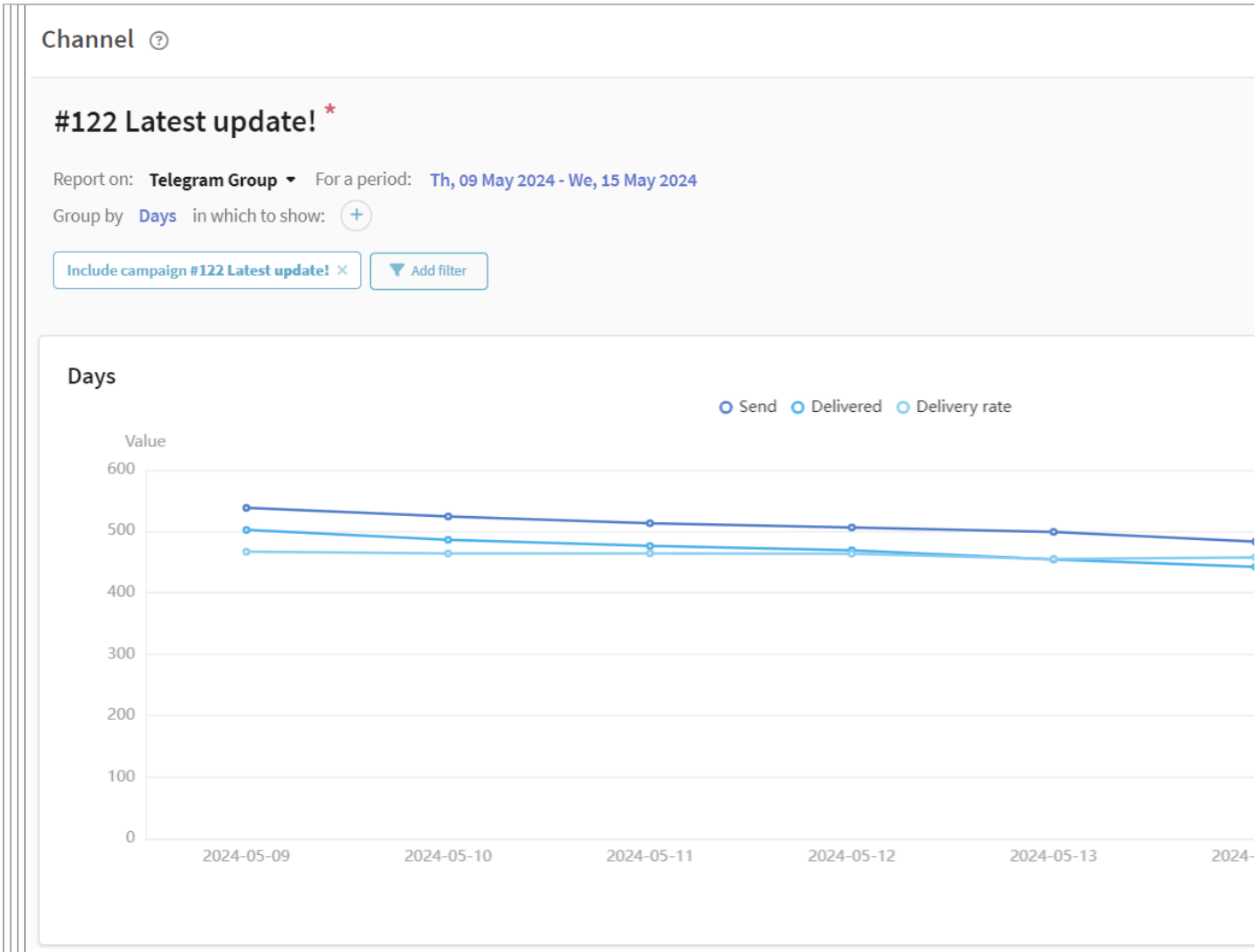
When the mailing is fully configured, save it. It will be displayed in the list of mailings of the correspondi on the right side of each entry. By its means, you can activate your actual newsletter

:



Step 5. Evaluate the effectiveness of your newsletter

To evaluate the result of your newsletters, use the “Analytics” section in the main menu of the platform. analysis of one or more mailings.



To make grouping easier, you can select a preset of indicators for Telegram Group. You will see the data undelivered messages in the table, as well as information on clicks and openings.

Max

This section contains guides for working with Max in Altcraft Platform.

Max Bot — a channel for personalized messages via bot. Suitable for customer communication, sending r personalized mailings. Requires passing the subscriber identifier to the platform.

Max Group — a channel for publishing messages in groups and chats. Suitable for mass content and anno individual profile subscriptions.

For more information about integration with the Max messenger, see this article.

First Campaign via MAX Bot

A MAX bot is an automated account in the MAX messenger that interacts with users via chat according t
To send Mailings via MAX Bot, users must be subscribed to the MAX Bot resource in the Altcraft platform
The subscription process works as follows:

1. **The user launches the bot.**
2. **The bot receives the user's `user_id`.**
3. **This identifier is sent to the Altcraft platform via API.**
4. **The profile is subscribed to the MAX Bot resource.**

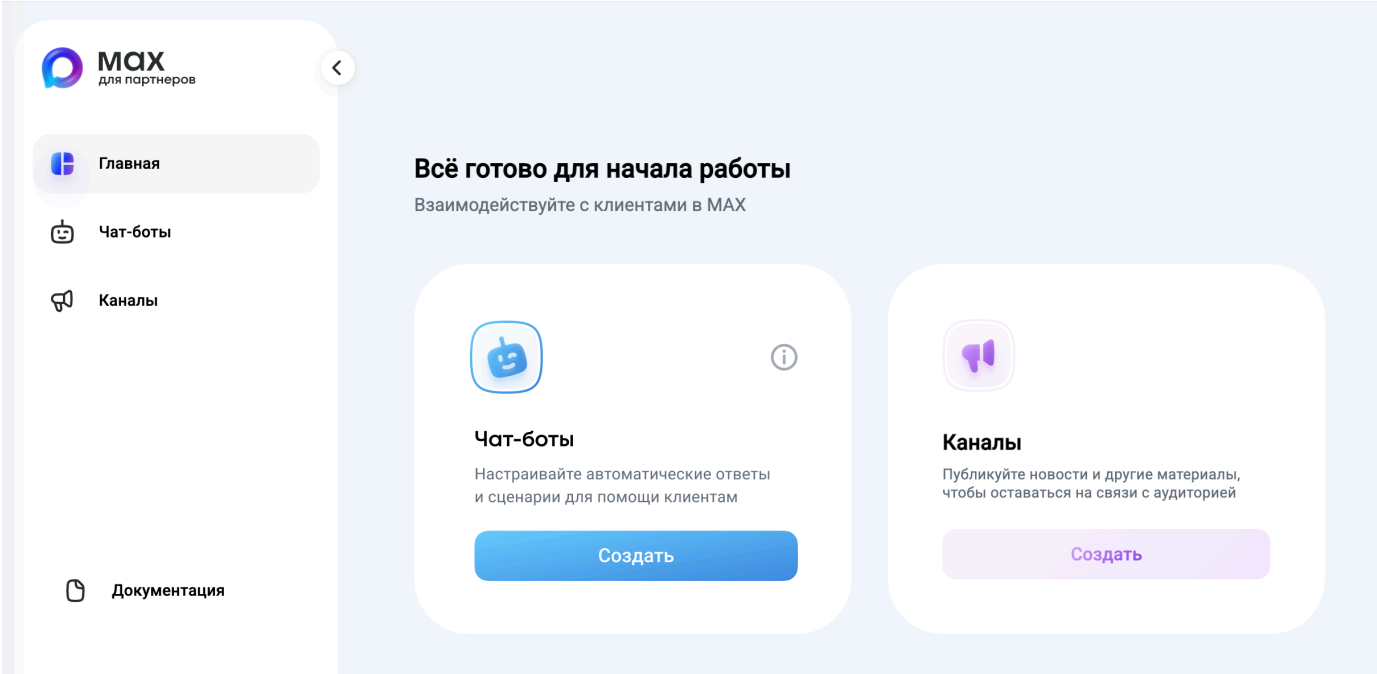
After that, the user can receive Mailings. Below is an example of how to configure such a Mailing via a M

Step 1. Create an organization and complete verification

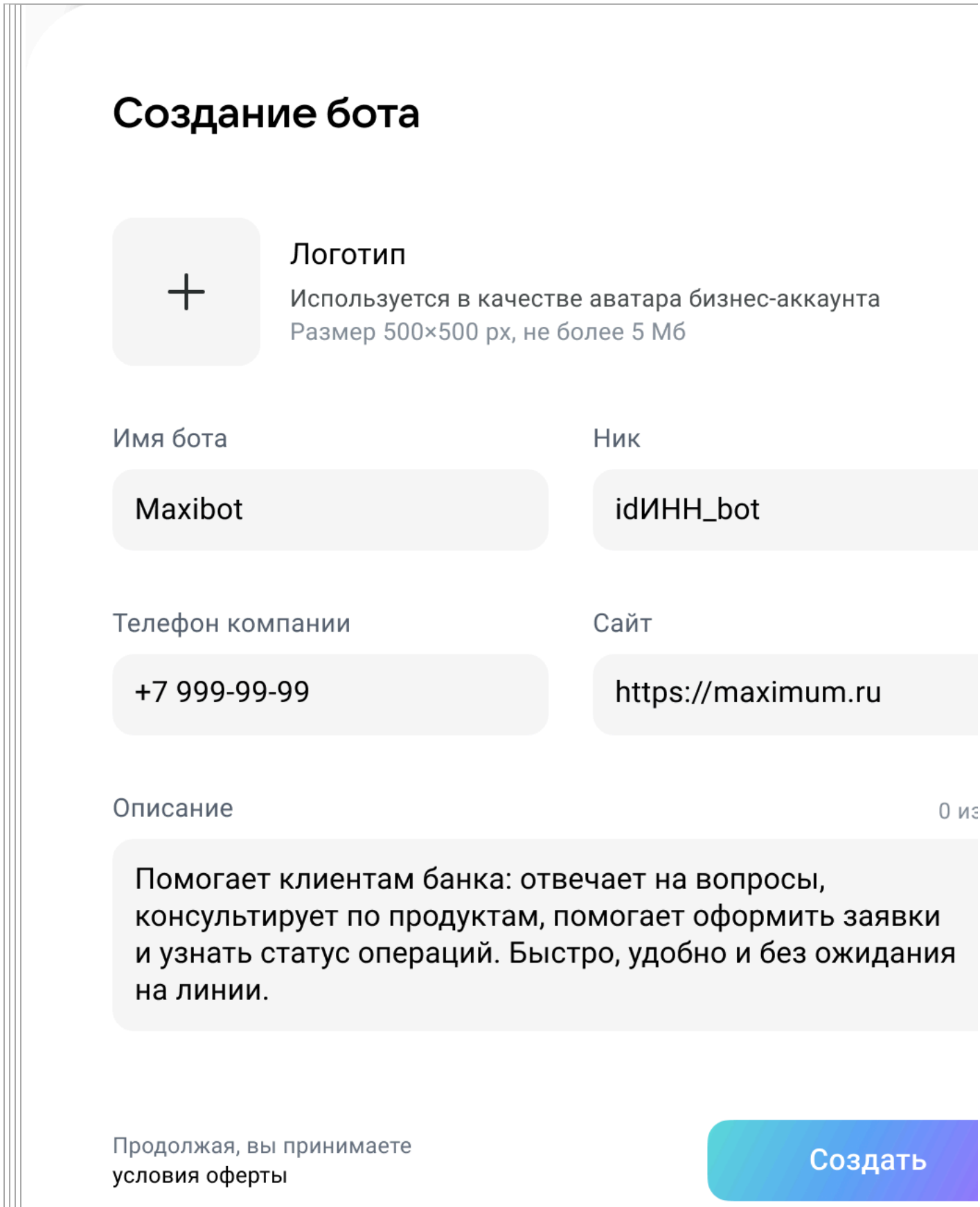
Creating bots and channels in MAX is available only for verified corporate accounts and individual entrep
create an organization in MAX and complete verification. More details about creating and verifying an or
documentation.

Step 2. Create a bot

On the MAX for Partners platform, go to your profile and open the Chatbots section:



Click Create and fill in the required fields:



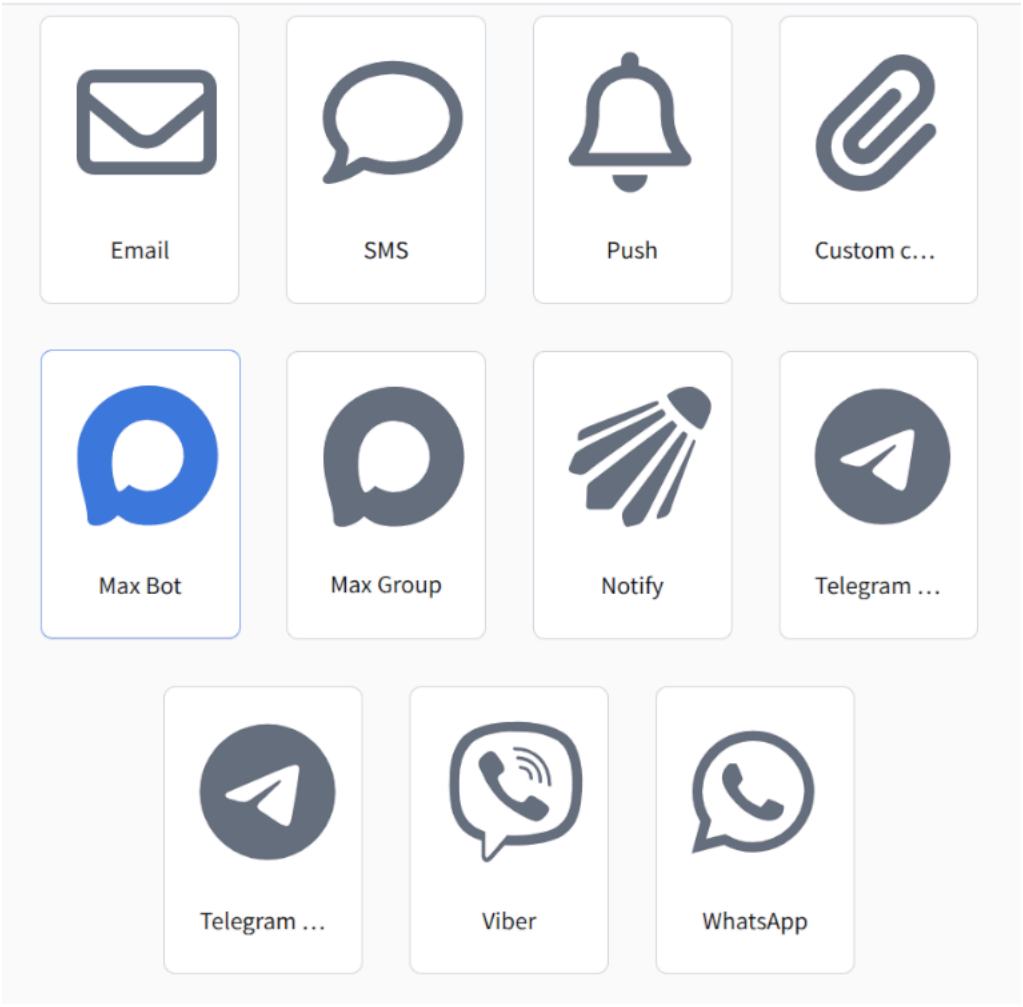
Click Create. The chatbot will be sent for moderation.

TIP: MAX chatbots

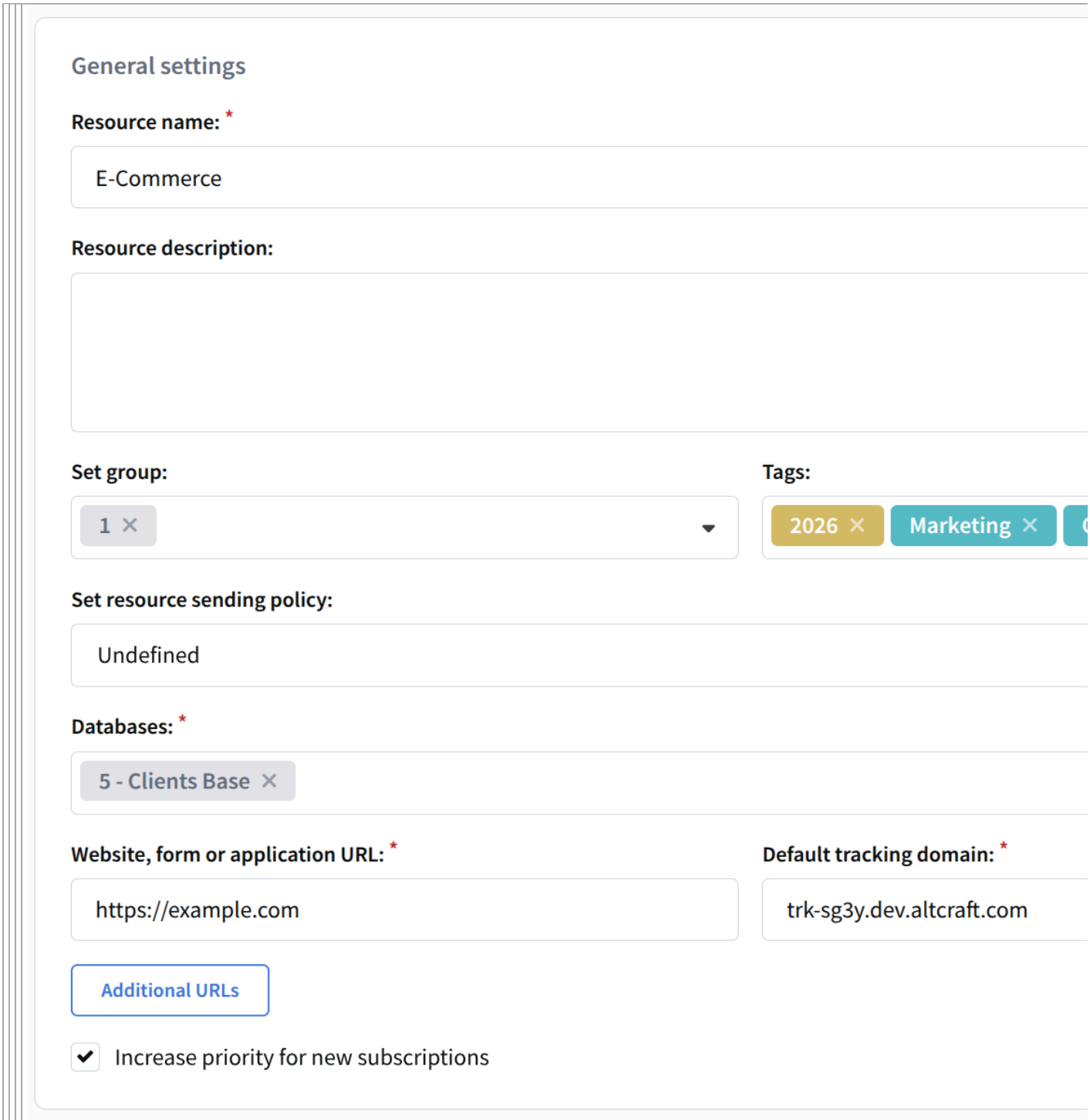
More information about creating a bot is available in the [MAX documentation](#).

Step 3. Subscribe clients to MAX Bot in the platform

Go to Data — Profile database and select a database where the data of users subscribed to the MAX bot Then open Resources and create a new resource or select an existing one. Use MAX Bot as the channel:



In the main settings, add the required databases. Specify the default tracking domain that will process s messages:



Save the resource. Next, import user data into the database and subscribe them to the corresponding re include the user's `user_id`.

NOTE

Note: the platform only performs message sending. Collecting and transferring user data to the platforn users who subscribed to the MAX bot, must be implemented on the client side. More details are availab

Example of retrieving a client `user_id` using the GET method `updates`

Method: GET

URL:

`https://platform-api2.max.ru/updates`

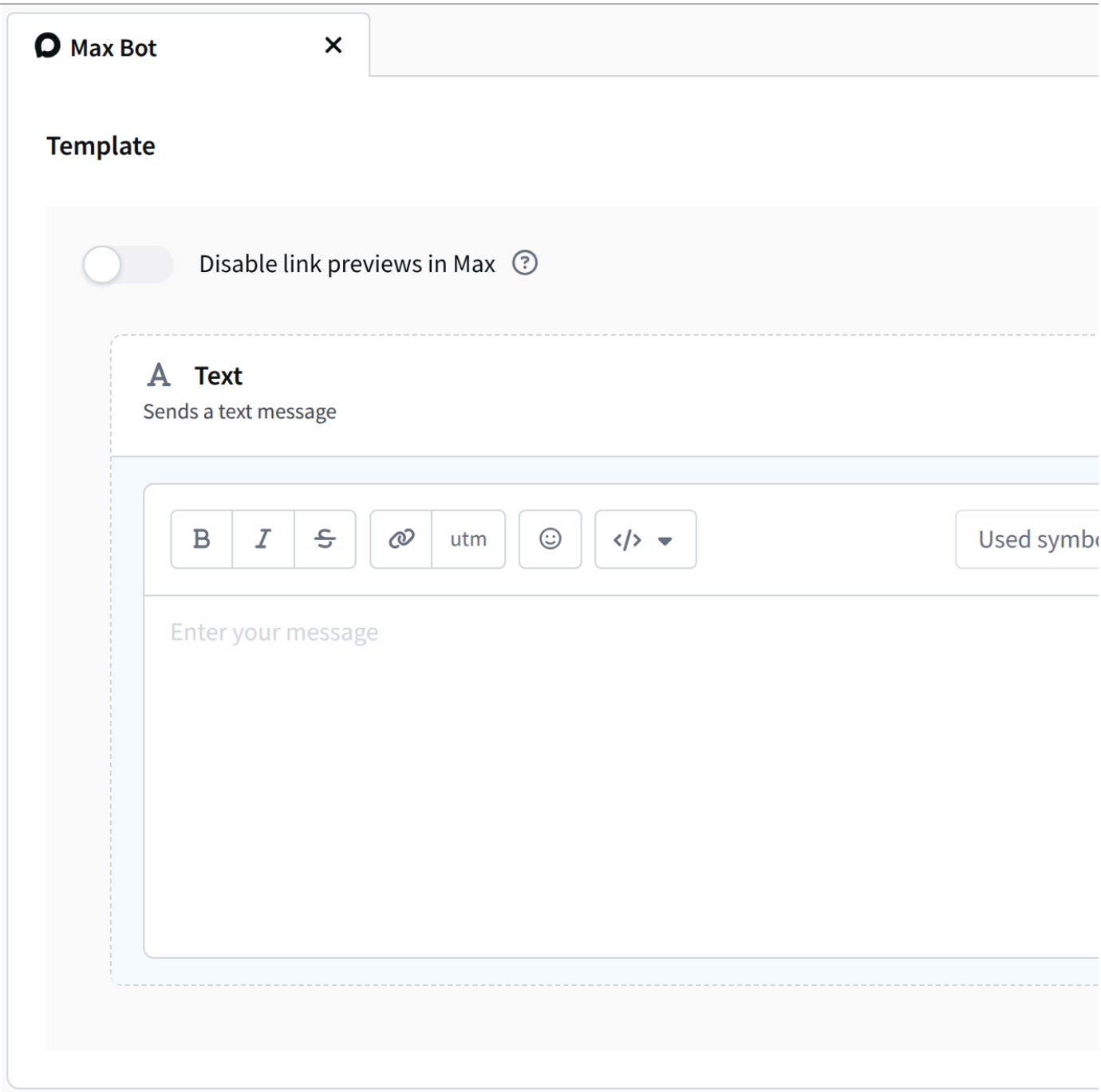
Response:

```
{
  "updates": [
    {
      "update_id": "9f3c2a71-8c4b-4c92-9e57-1d7f2c0b8a11",
      "message": {
        "message_id": "msg_847362915",
        "chat": {
          "chat_id": "7349281745621",
          "type": "private"
        },
        "from": {
          "user_id": "48291057",
          "first_name": "Ivan"
        },
        "date": 1719832145,
        "text": "/start"
      }
    },
    {
      "update_id": "b1a0c5e2-3c91-4a6e-8fbb-5e92c0a47d19",
      "callback": {
        "callback_id": "cb_12984756123",
        "data": "order_create"
      }
    }
  ],
  "marker": "1719832145:7349281745621:msg_847362915"
}
```


`<user_id>` is the identifier of a specific user that must be sent to the platform.

Step 4. Create a template

Go to the [Templates] section. When creating a new template, select the MAX Bot channel or add this cha

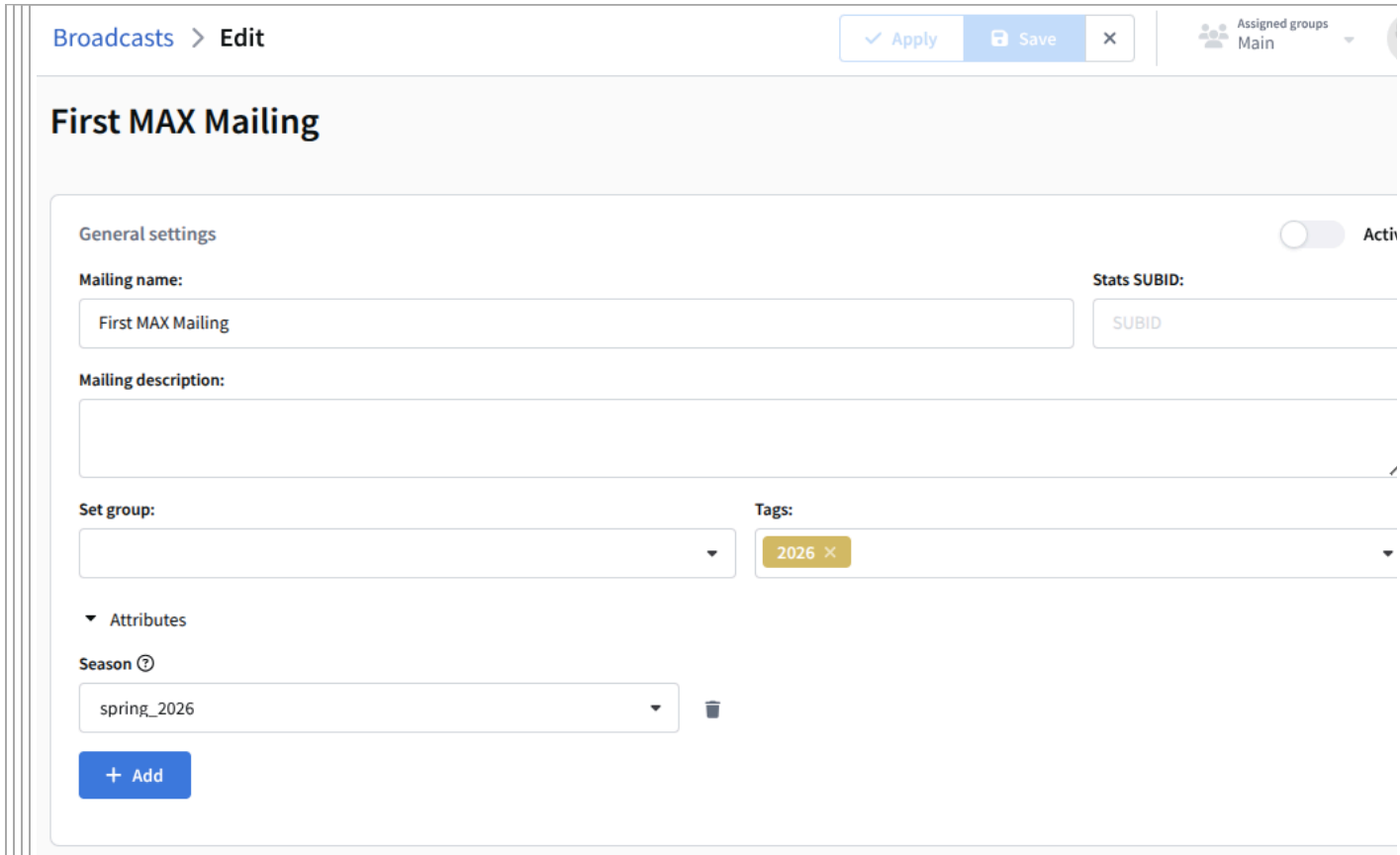


You can include specific client data as well as dynamic text content from external sources.
More details about creating a template for MAX are available in this article.

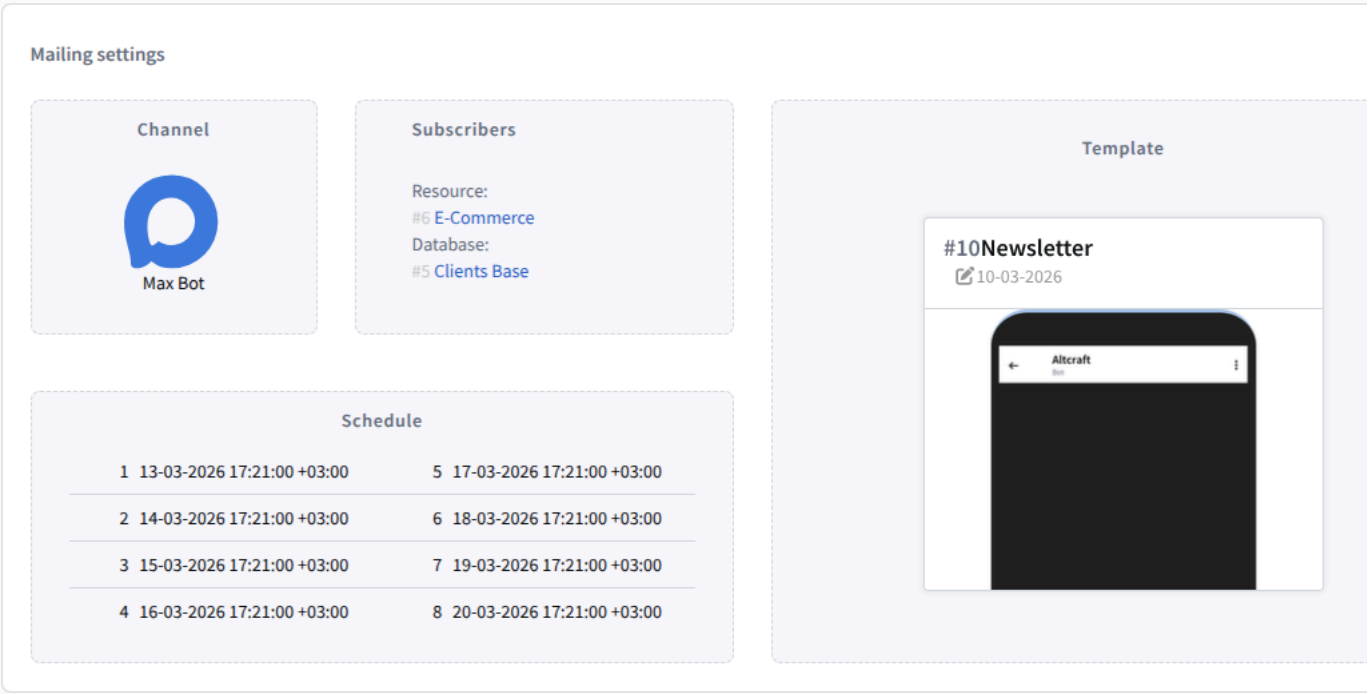
Step 5. Configure and launch the Mailing

Open the Mailings section and choose the appropriate type of Mailing: broadcast, regular, or triggered.
Create a new Mailing. After filling in the main settings, you need to:

1. **Select MAX Bot as the communication channel**
2. **Select the previously configured subscription resource and the recipient audience**
3. **Add a message template to the Mailing**
4. **Set the schedule**



After completing the Mailing configuration, save it. It will appear in the list of Mailings of the correspond record there is a quick actions menu. From there you can perform a **test send ![img](/img/attachment everything is successful, **activate** the Mailing ![img](/img/attachments/49710416/49710428.png):**



Step 6. Evaluate mailing performance

Mailing journal

The Mailing journal is a tool that provides a brief overview of Mailing execution. It records key events, M which helps identify potential issues.

To open the Mailing journal, click the Journal button in the quick actions menu:

The Mailing journal contains the following information:

Type	Description
Mailing start	**Time when the Mailing execution started**

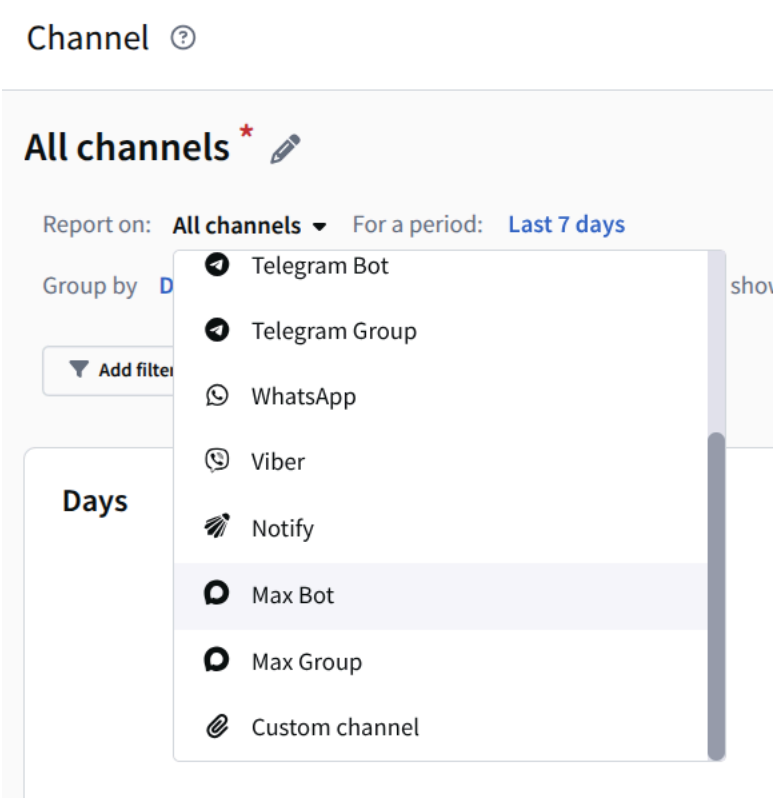
Type	Description
Limit	**Indicates whether a message sending limit is set**
Subscriber source	**list — Database segment — Segment (if applicable)**
Database ID / Segment ID	**Identifier of the database or segment used**
Data retrieval time	**Time spent loading contacts from the source (ms)**
Number of profiles in source	**Total number of contacts in the specified source**
Number of valid subscribers	**Number of users with an active subscription**
Subscribers without subscriptions	**Users without a subscription to the target channel (0 in the example)**
Records with incorrect fields	**Errors in subscription data (0 in the example)**
Number of duplicates	**Duplicate contacts (40 in the example)**
Number of messages sent	**Actual number of successful sends**
Campaign completed	**Time when the Campaign finished**
Total time spent	**Total Campaign duration**

Reports and analytics

To evaluate the results of the current Mailing, you can view the channel report. The Channel report provi Mailings.

To access statistics, open the Analytics section in the main platform menu. By default, the report for All displayed.

To group data by channel, select Report by: MAX Bot



Next, add a filter for specific Mailings. Select the required Mailing from the list. As a result, you will rece

Channel

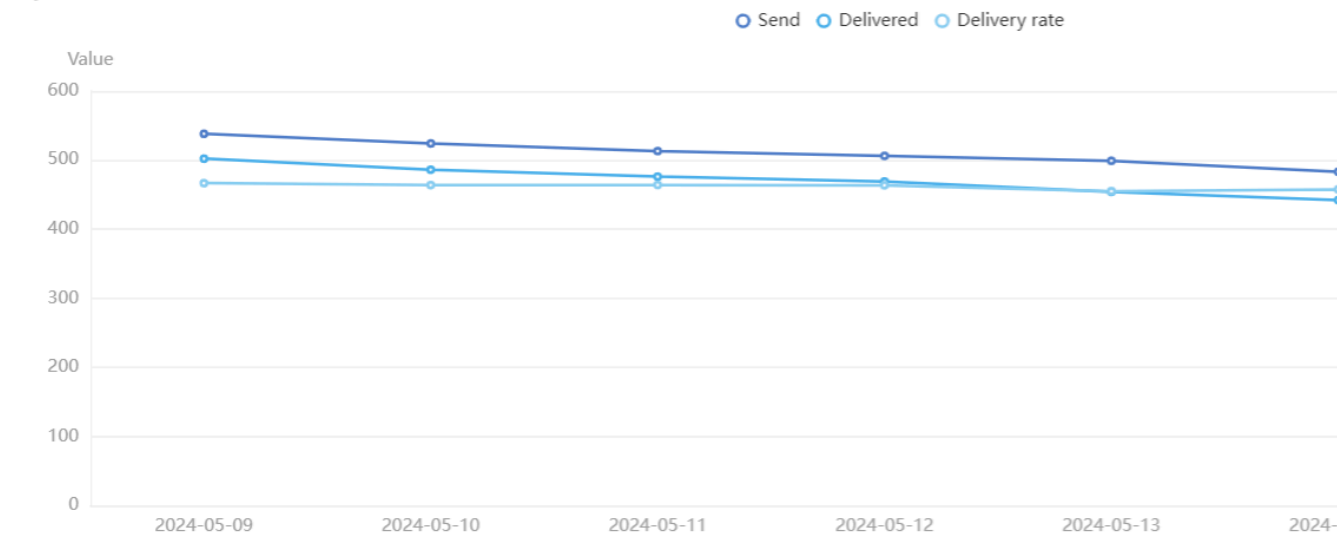
#122 Latest update! *

Report on: Max Bot For a period: Th, 09 May 2024 - We, 15 May 2024

Group by Days in which to show:

Include campaign #122 Latest update! × Add filter

Days



First Mailing in MAX Group

A channel in MAX is a feed of text messages that can include media files and emojis. Channels also supp
The MAX Group channel enables automatic publishing of posts in messenger channels and groups using

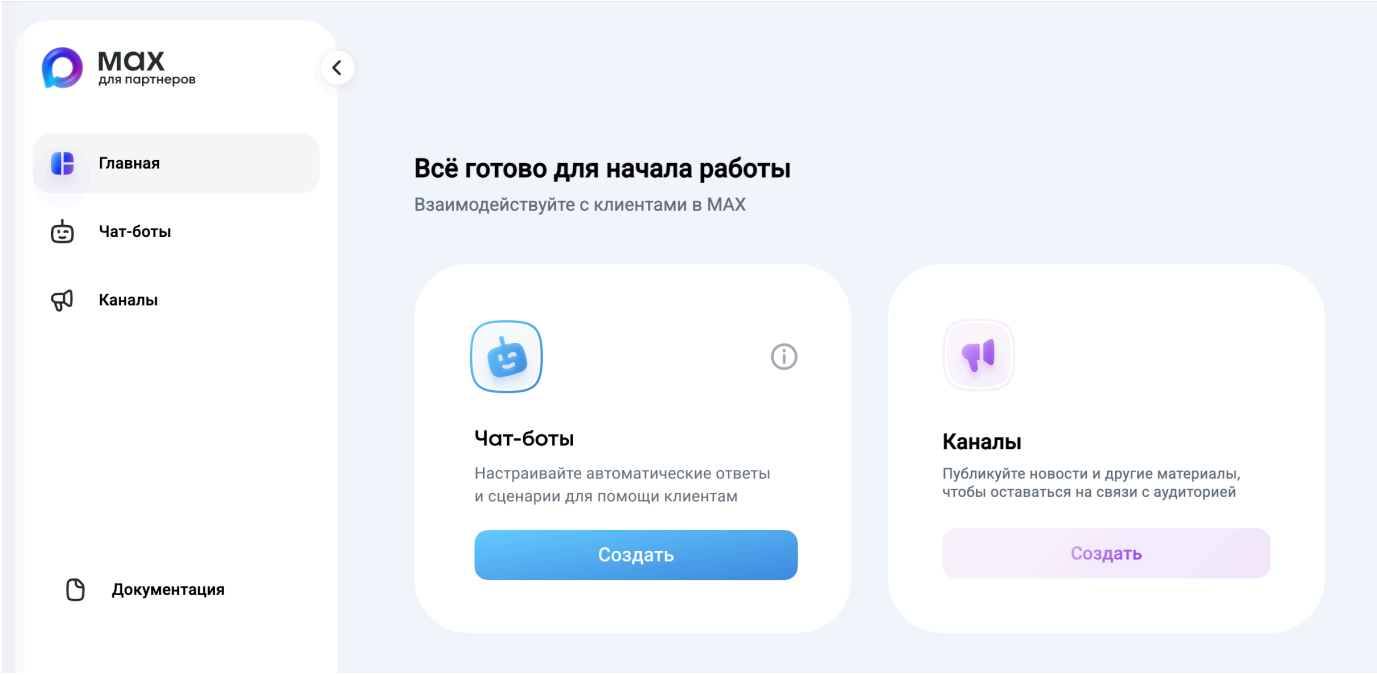
> TIP: Indirect channel
>
> **MAX Group is an indirect communication channel, which means Mailings in this channel cannot be pers variables.**

Step 1. Create an organization and complete verification

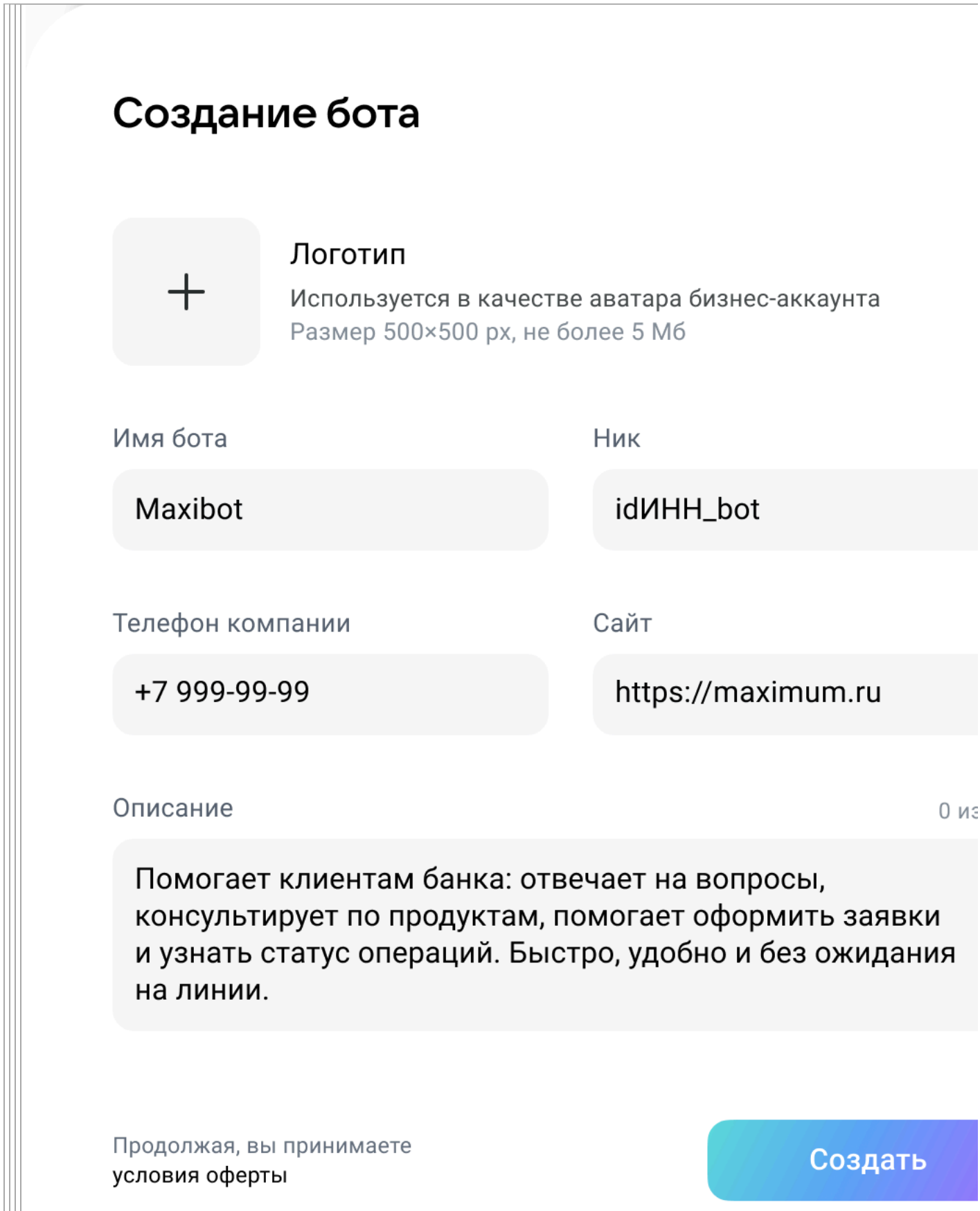
Creating bots and channels in MAX is available only for verified corporate accounts and individual entrep create an organization in MAX and complete verification. More details about creating and verifying an or documentation.

Step 2. Create a bot

On the MAX for Partners platform, go to your profile and open the Chatbots section:



Click Create and fill in the required fields:



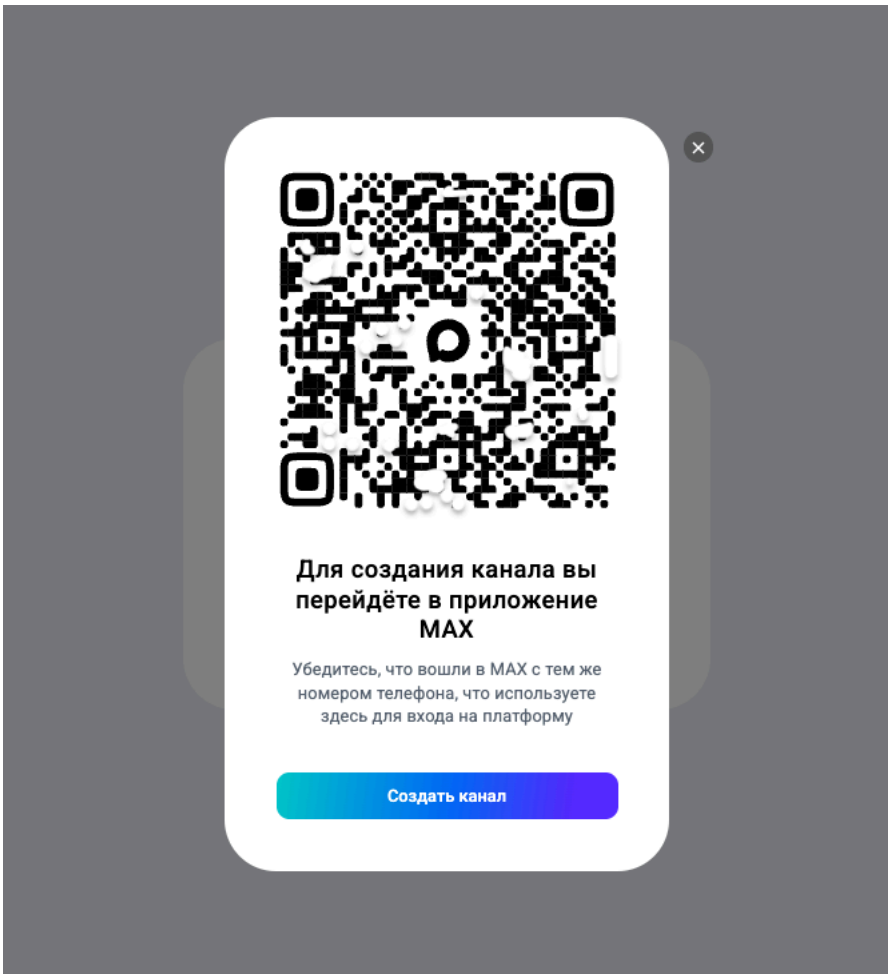
Click Create. The chatbot will be sent for moderation.

TIP: MAX chatbots

More details about creating a bot are available in the [MAX documentation](#).

Step 3. Create a channel or group in MAX

On the [MAX for Partners] platform, open Channels — Create — Create a new business channel. Scan the C button:



In the channel creation bot, confirm your phone number and wait for the request to be processed. After t created channel.

Creating a group does not require special permission.

TIP: Private channels

MAX supports private channels with access available only by invitation. More information about creatin available in the [MAX documentation.]

Step 4. Add the bot to the channel or group

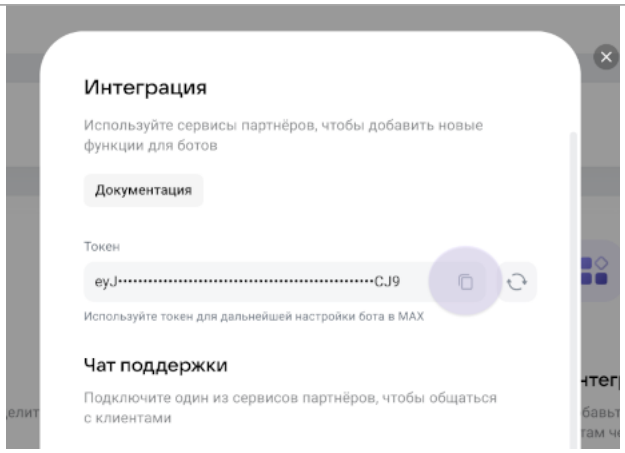
Add the created bot to the group or channel. Then grant the bot administrator rights. Messages will be p

Step 5. Create a connection and integration

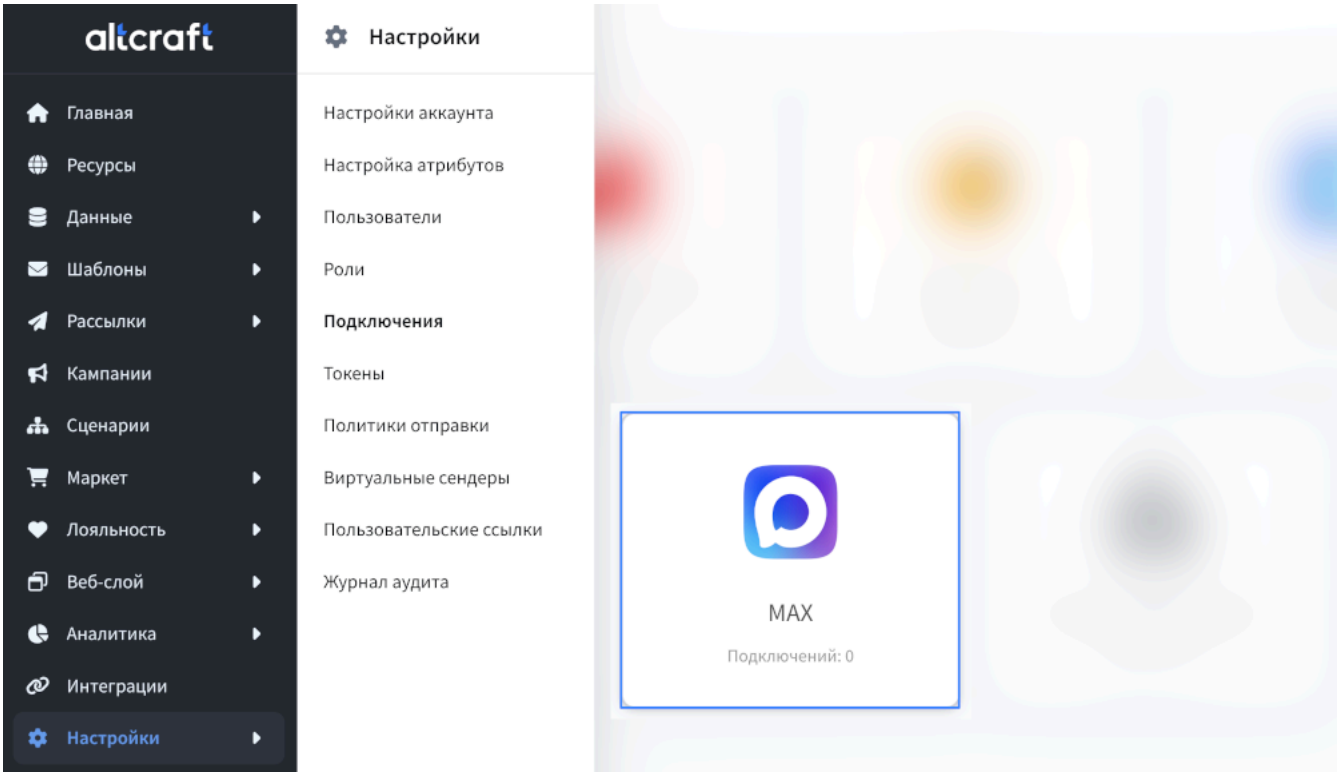
To enable interaction between the platform and MAX, you first need to create a bot connection and then communication channels.

To create the connection, you need the MAX bot token. It can be obtained on the MAX for Partners platfo

Open the Chatbots section on the [MAX for Partners] platform, then go to Integration — Get token. The AP

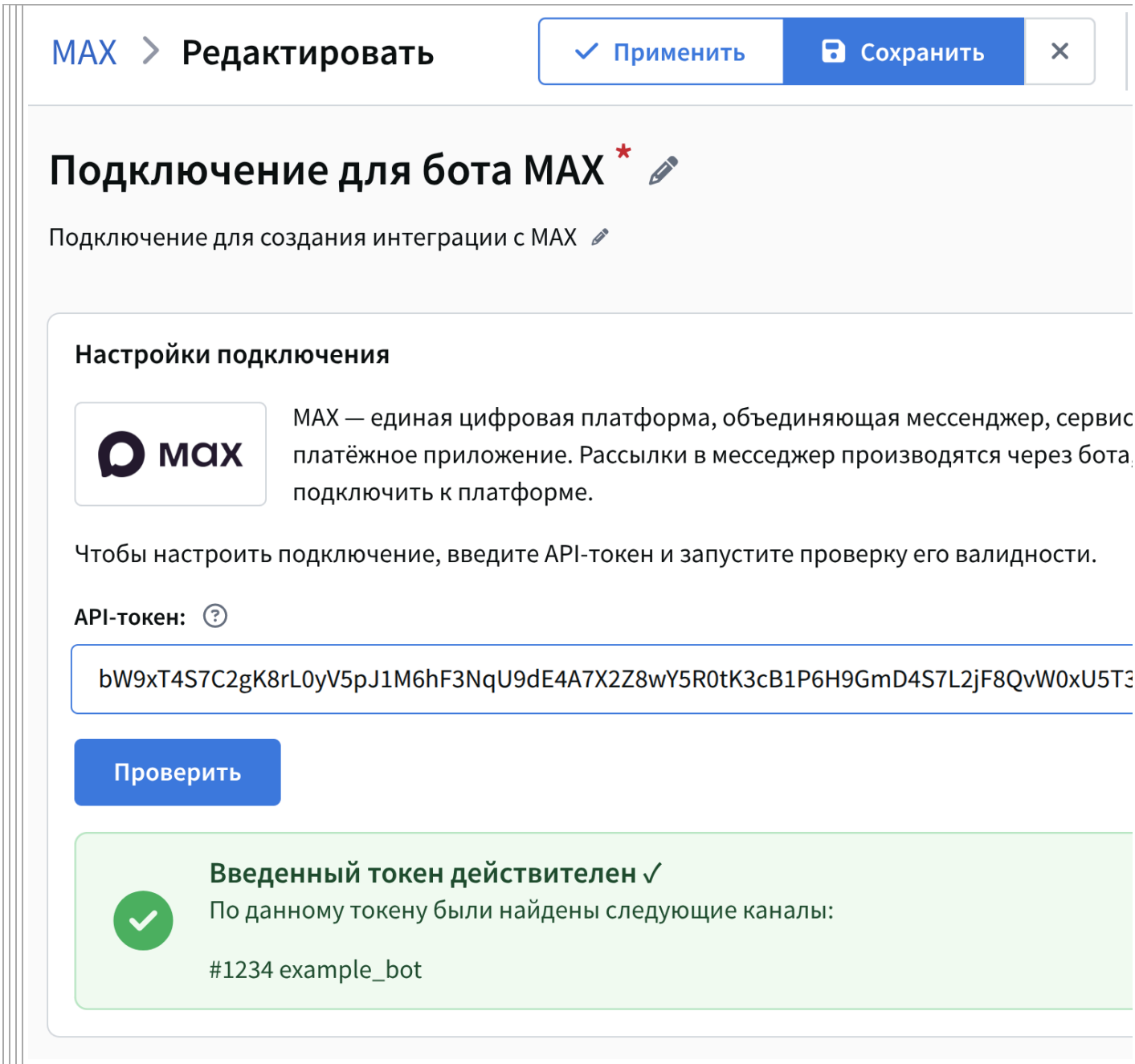


Then go to the platform and open Settings — Connections. In the list of connection types, select MAX:

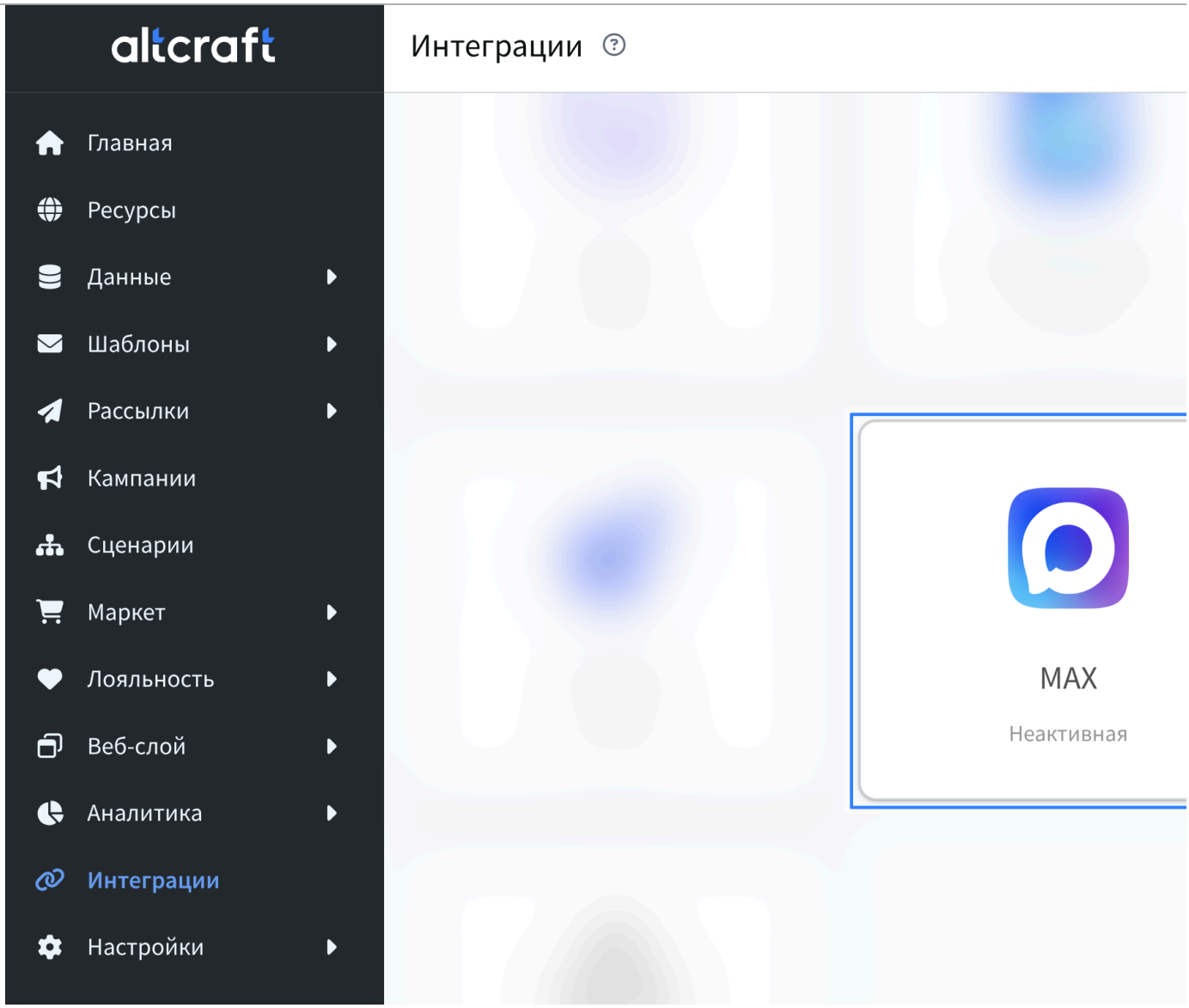


Click Create on the top panel and enter the connection name and description.

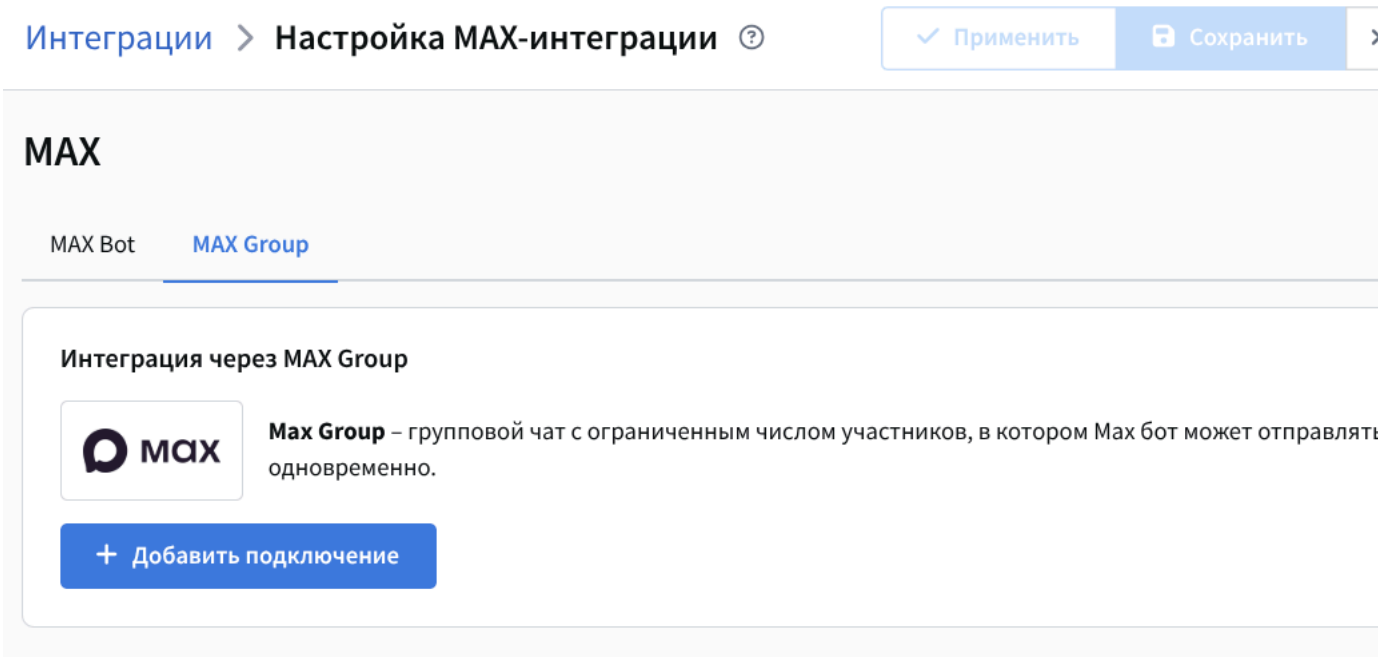
In the API token field, paste the token obtained from the MAX platform and click Check. If the data is cor successful verification message.



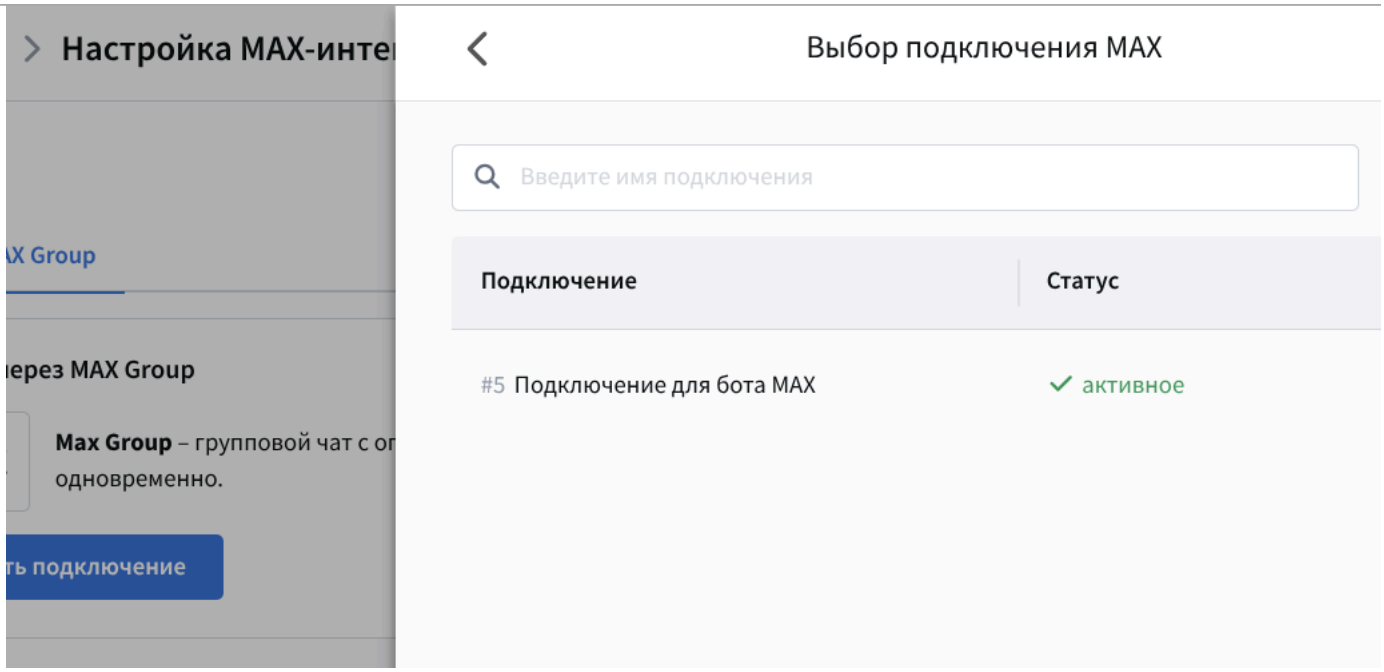
Click Save to finish creating the connection.
After creating the connection, configure the integration. Open the Integrations section and select MAX:



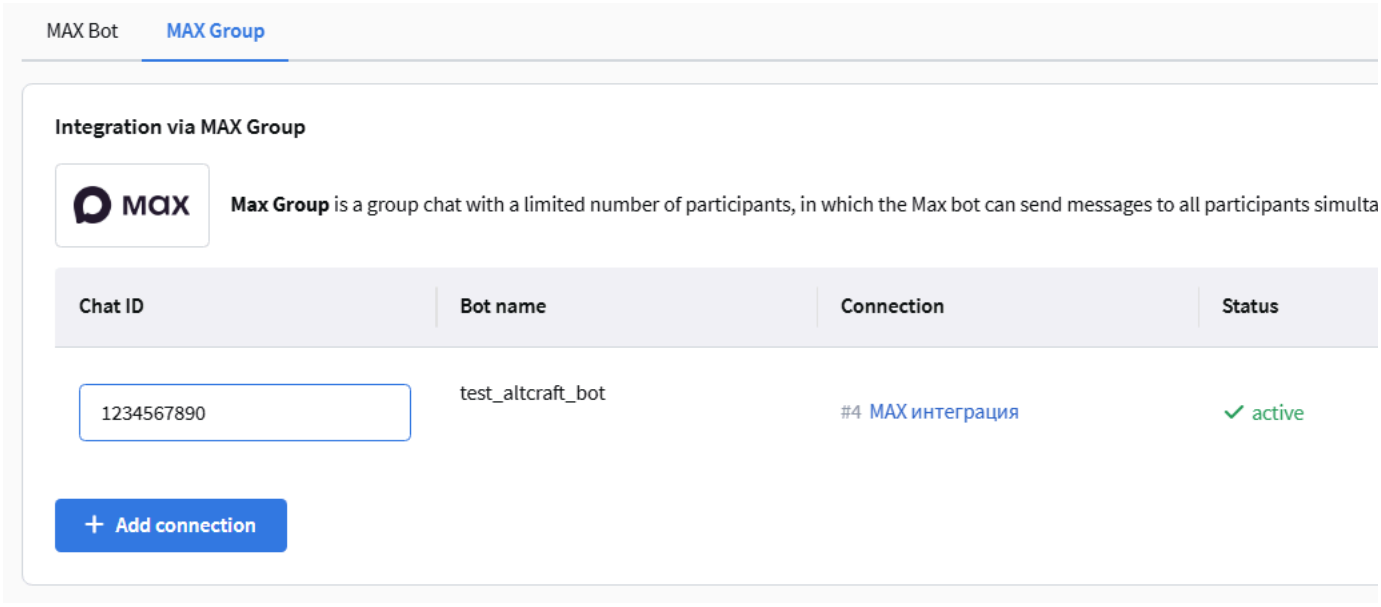
Open the MAX Group tab and click Add connection:



Select the created bot connection:



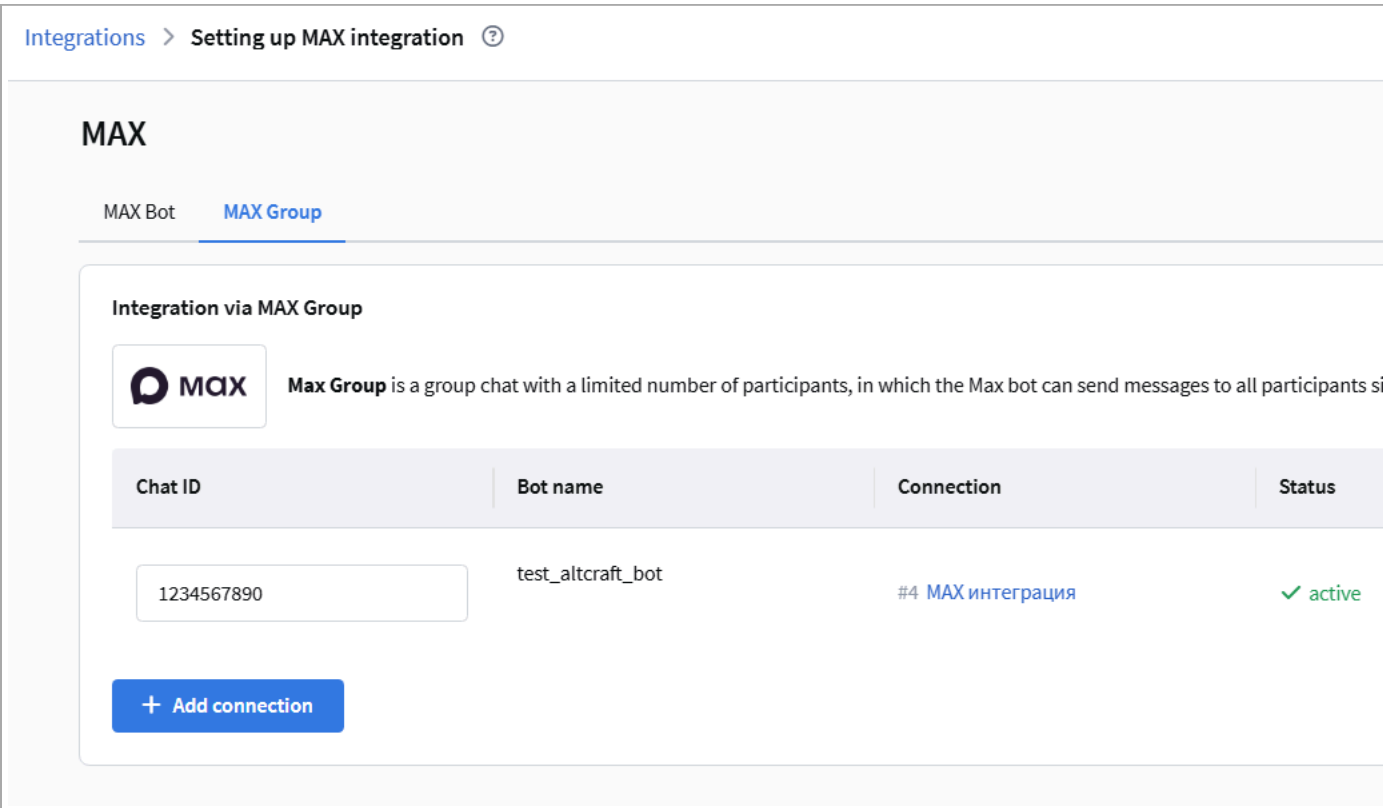
Then enter the chat_id — the identifier of the chat where the bot will send messages:



TIP: How to find the chat_id

Open [web.max.ru](web.max.ru), go to the desired chat, and look at the URL in the address bar. The number after the

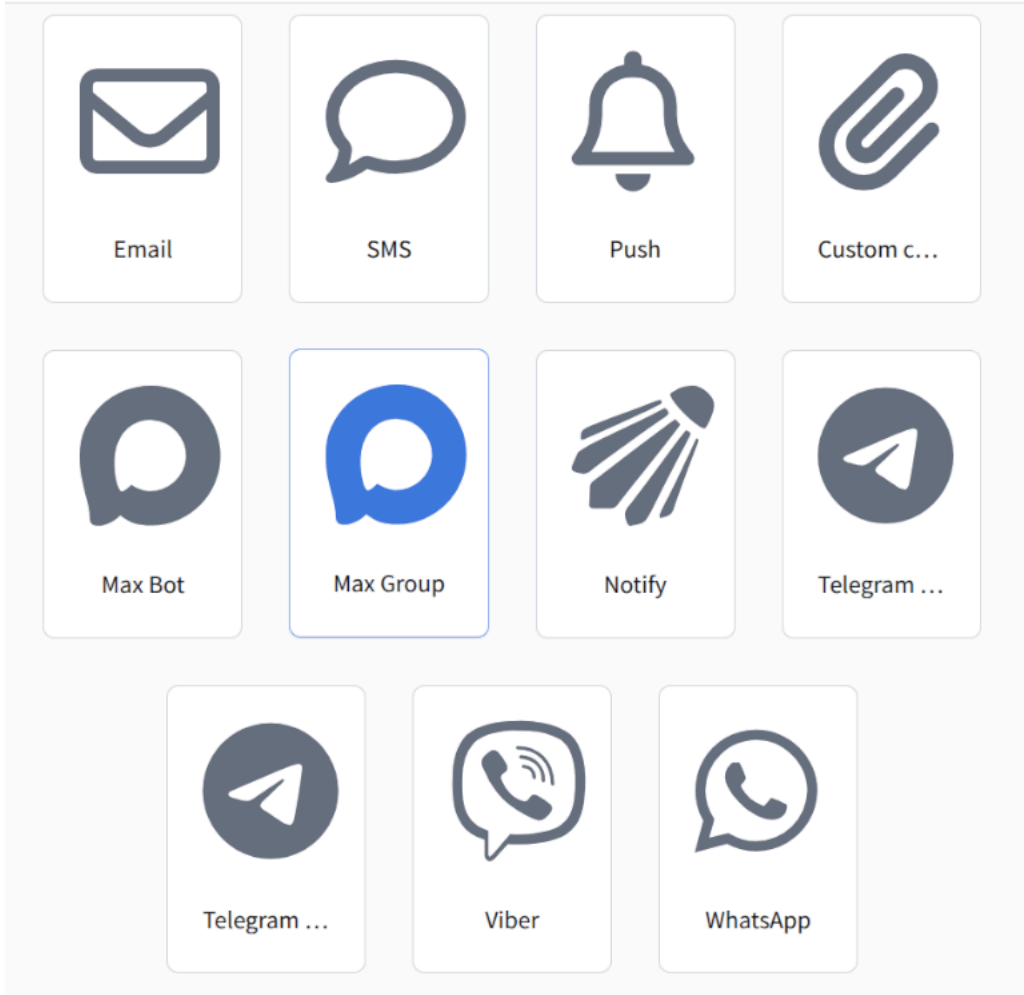
Enable the integration using the switch at the top of the page and click Save:



The integration is ready to use. Next, configure a resource that will be used to send messages to MAX m

Step 6. Create a resource

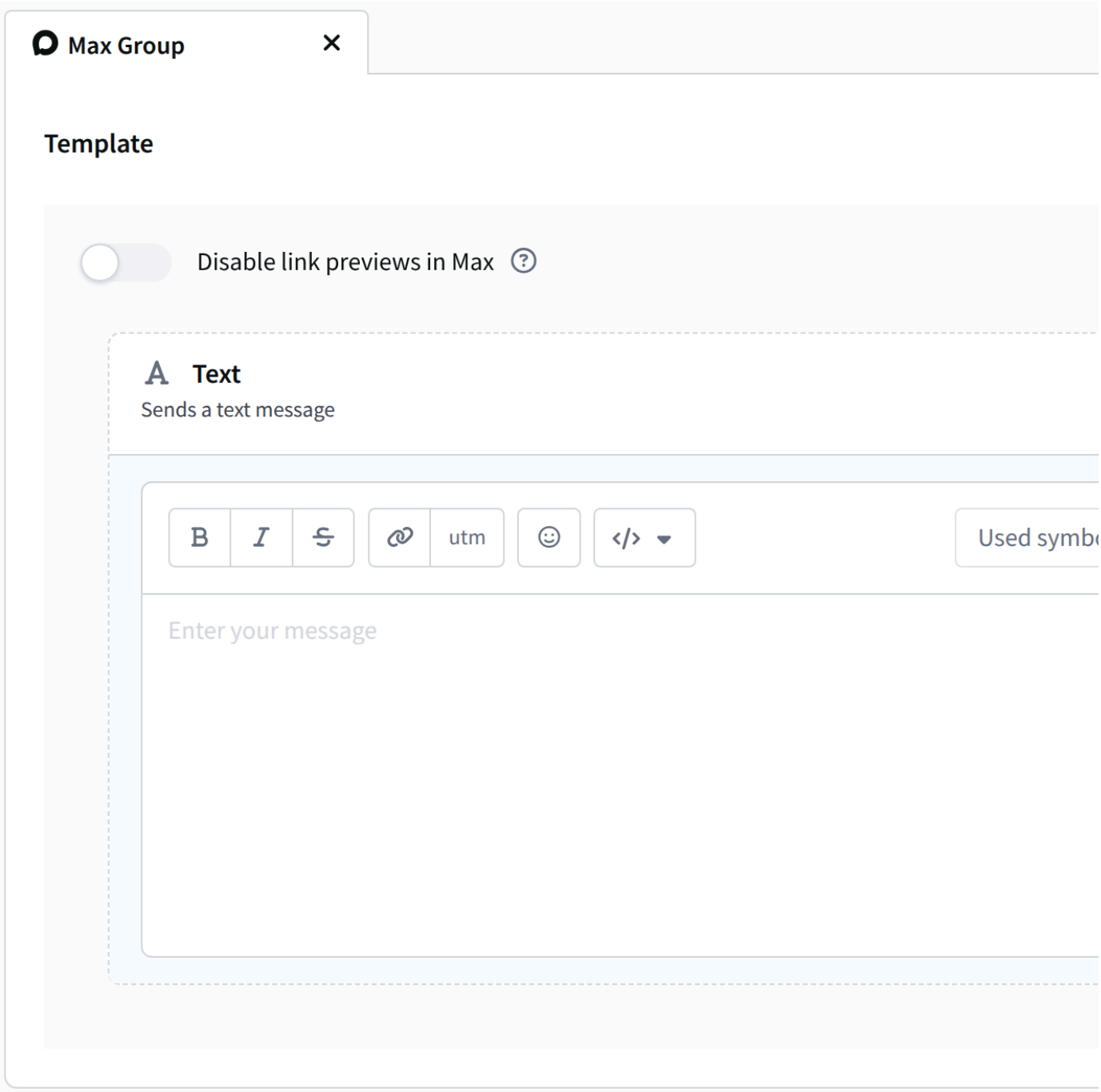
Go to the Resources section and create a new resource or select an existing one. Use MAX Group as the



In the main settings, add the required databases. Specify the default tracking domain that will process s group posts.

Step 7. Create a template

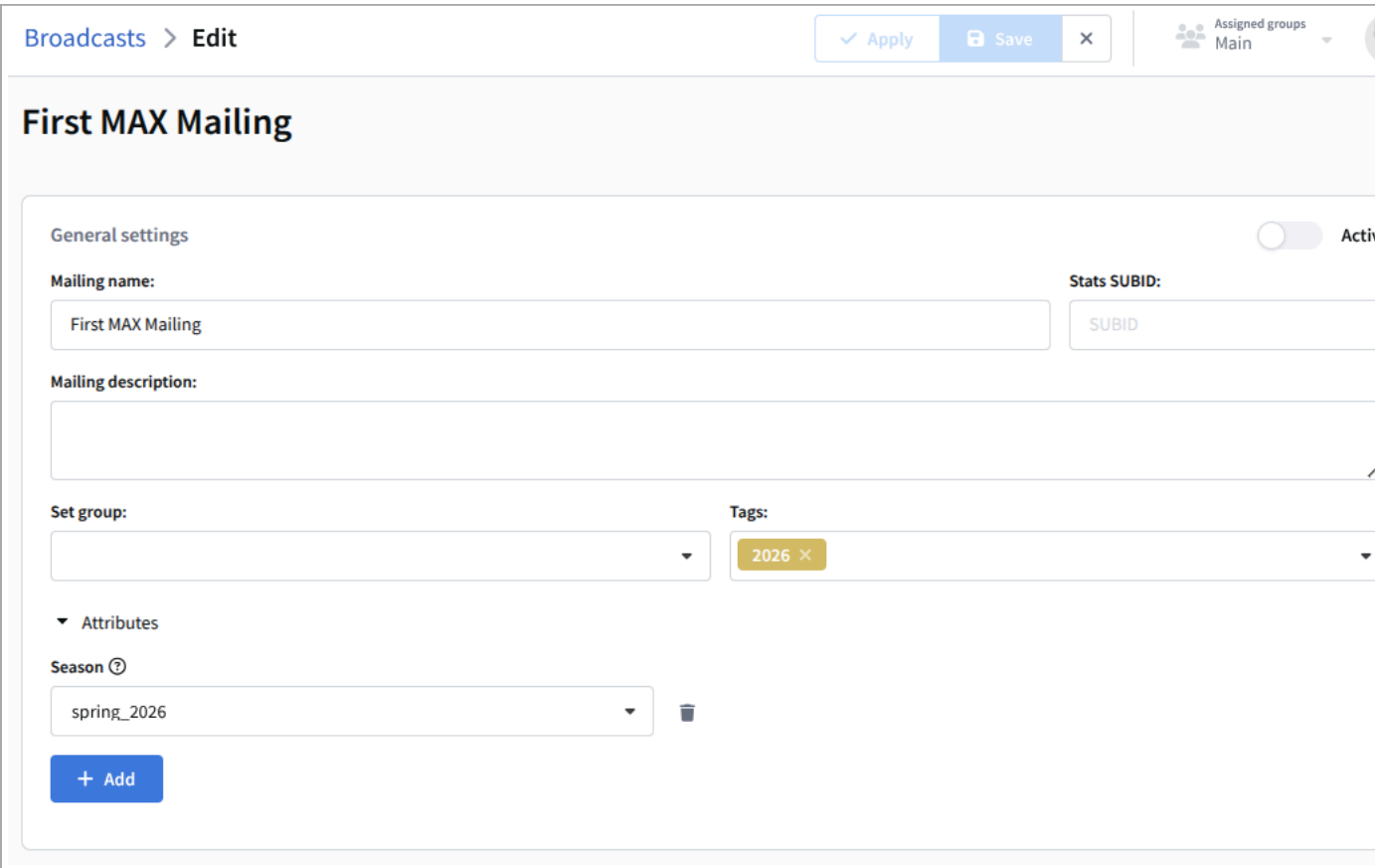
Go to the Templates section. When creating a new template, select the MAX Group channel or add this c



You can add dynamic text content from external sources.

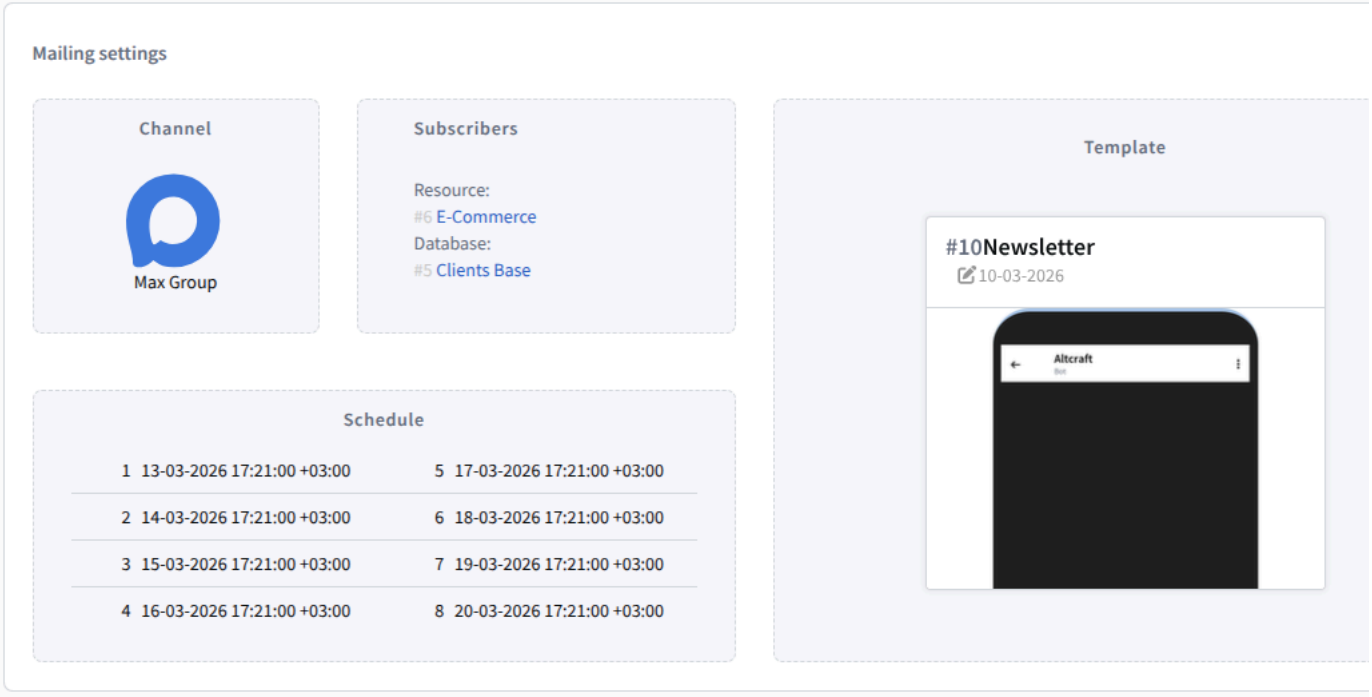
Step 8. Configure and launch a placement

Open the Mailings section and select Placements. Create a new Mailing.



After filling in the main settings, select:

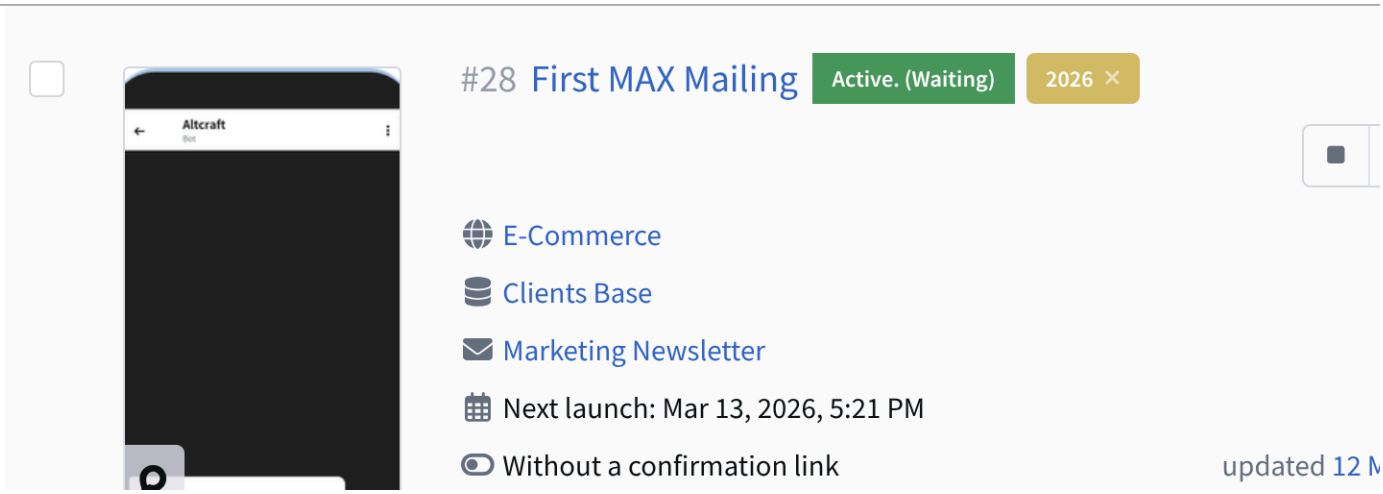
1. **MAX Group as the communication channel;**
2. **The previously configured subscription resource;**
3. **The message template;**
4. **The date and time when the post will be published in the group.**



After completing the configuration, save the Mailing. It will appear in the list of Mailings. On the right sid actions menu. From there you can activate the placement

▶

:

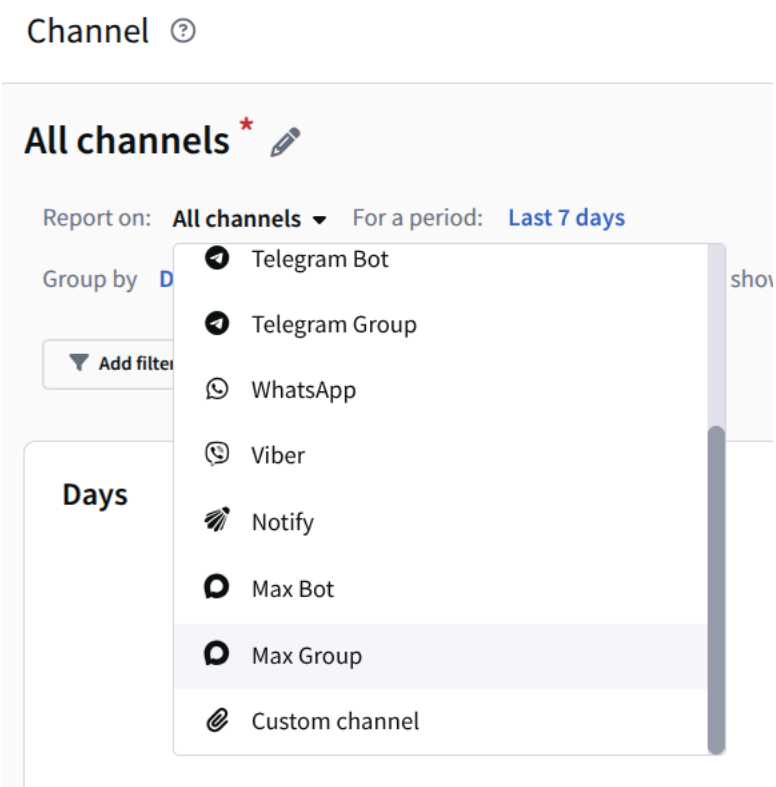


Step 9. Evaluate Mailing performance

To evaluate the results of the current Mailing, you can view the channel report. The Channel report provi Mailings.

To access statistics, open the Analytics section in the main platform menu. By default, the report for All displayed.

To group data by channel, select Report by: MAX Group



Next, add a filter for specific Mailings. Select the required Mailing from the list. As a result, you will rece

Channel

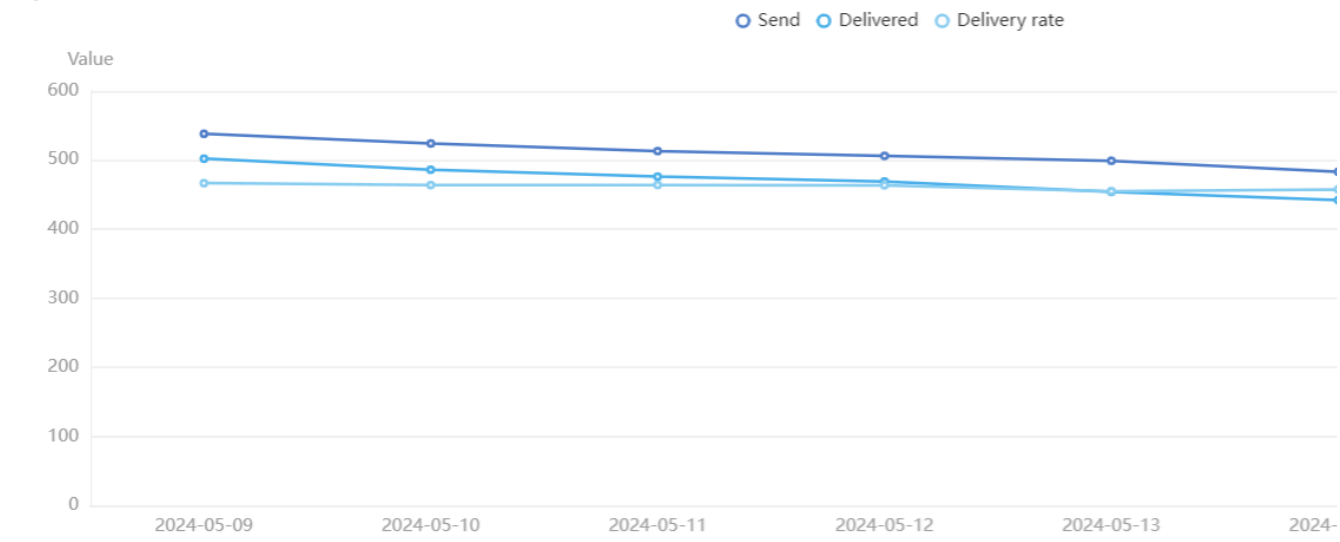
#122 Latest update! *

Report on: Max Group For a period: Th, 09 May 2024 - We, 15 May 2024

Group by Days in which to show:

Include campaign #122 Latest update! × Add filter

Days

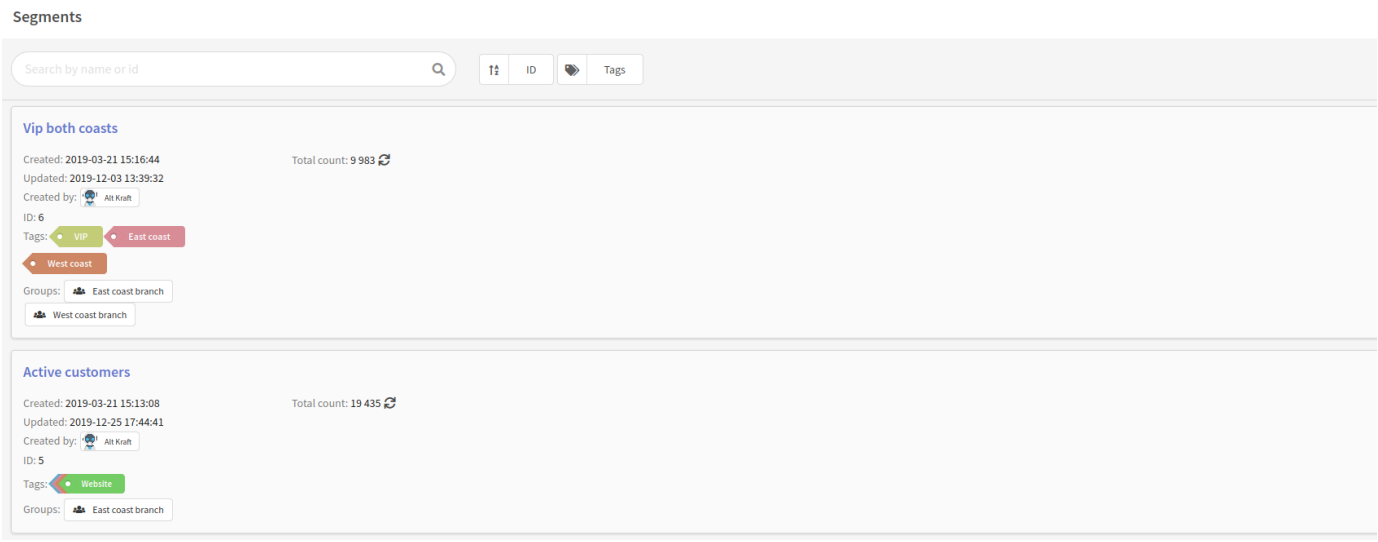


Segmentation

![img](/img/attachments/3114160/9408130.png)	Segment is a part of the database that includes profiles that meet certain conditions defined in the segment. Afte mailing as a source of subscribers.

Profiles segmentation allows you to target your marketing mailings and campaigns. Selection is based o history in messages, on the website or in the application, as well as based on queries to external source service APIs.

To segment the audience of subscribers, open Data → Segments in the main menu. In the list, for each s information will be displayed. Using the context menu, you can perform additional actions, for example,



Open an existing segment or add a new query by clicking + Create at the top of the page.

Segment types

Static segments

Static segment can be counted by selection from the database, but it will not be recounted automaticall into the static segment manually, and remove profiles in the same way. You can use these segments wh groups.

Dynamic segments

Dynamic segments are mainly used for targeting your marketing campaigns. Such segment does not sav segmentation conditions. Instead, the main thing which is saved is selection query structure. Counting i campaign launch - and actual information is used.

Updatable segments

The updatable segment is used for dynamic scheduled updates. It includes profiles that meet the specifi updated until this scheduled check. If necessary, the calculation can be performed manually.

Static Segments

A static segment is a fixed list of profiles from a selected database. After filling, its composition does no data in the database is updated.

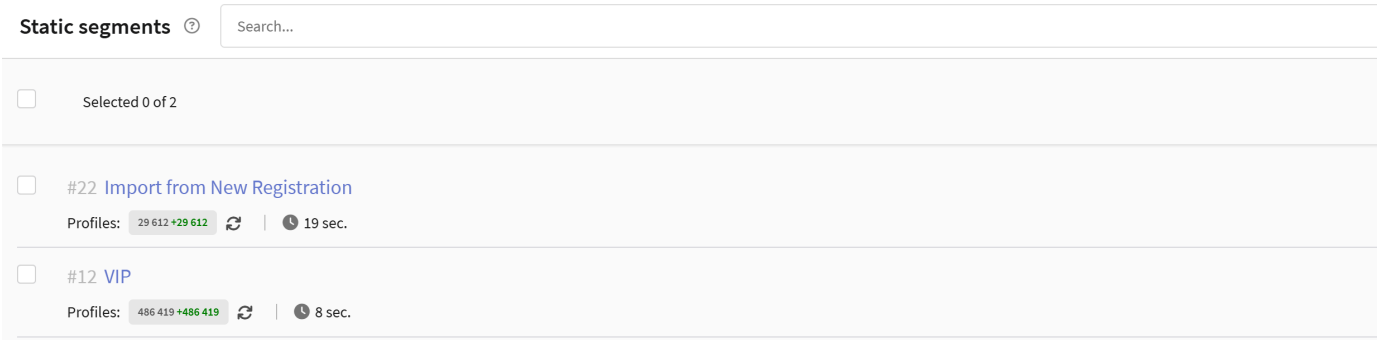
This type of segment is suitable for tasks where a fixed audience is needed, for example: mailings to a p time selections for reports or integrations. Also, the audience of a mailing is formed based on a static se

![img](/img/attachments/9407279/en/stat-segements-navigation-en.png)	To get started, go to **Data → Segments → Static Segments**.

Creating and General Segment Management

Creating a Segment

In the segments list, click + Create:



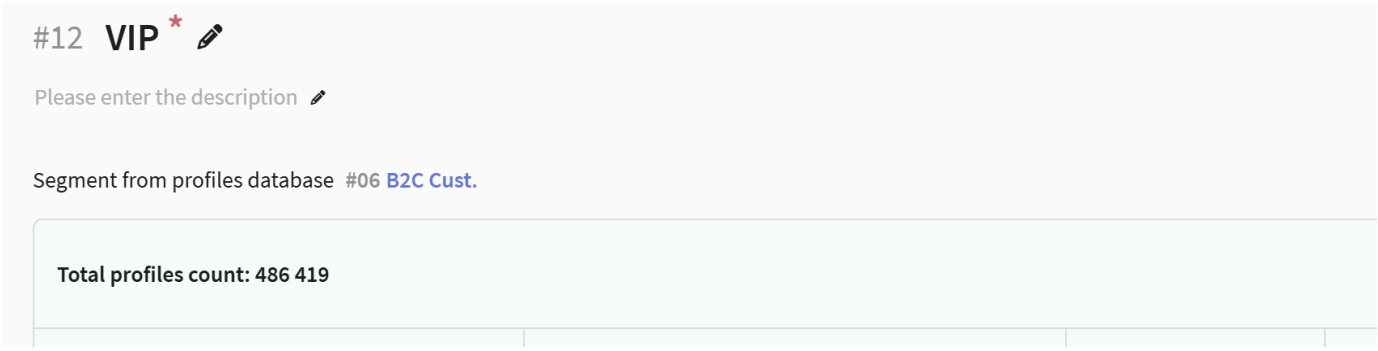
In the form that opens, specify the Name and Description of the segment, and select the Database from click Create:

![img](/img/attachments/9407279/en/stat-segemnt-name-en.png)

> INFO: Important
>
> **The database is set only during creation and cannot be changed later.**

Segment Configuration and Management

After creating or selecting a segment, its editor will open. In the upper right corner of the segment edito working with the entire current list of profiles:



- **Set Tags: Assign or change tags for categorizing the segment.**
- **Specify Access Groups: Configure which user groups can see and edit this segment.**
- **Copy Profiles to Profile Database: Copy all profiles from this segment to another selected profile dat**
- **Copy Profiles to Static Segment: Copy all profiles from this segment to another selected static segm**
- **Edit Profiles: Open the interface to modify data (for example, fields or subscriptions) for all profiles i**
- **Add to Scenario: Send segment participants to a selected scenario. By default, only profiles that are scenario are sent. To send all segment participants, including those already participating, check the**

> INFO: Note
>
> **You can set tags and access groups right after creating the segment. Other actions will become availal**

Audience Management

The segment composition can be changed in different ways: manually, by configuring conditions, using t that the conditions work only once when they are applied and are not saved for constant segment updat

- **Add: Clicking opens a side panel for configuring conditions by which profiles will be added to the seg**

- **Remove (becomes available after adding profiles): Clicking opens a similar side panel for configuring removed from the segment.**

Adding Profiles by Conditions

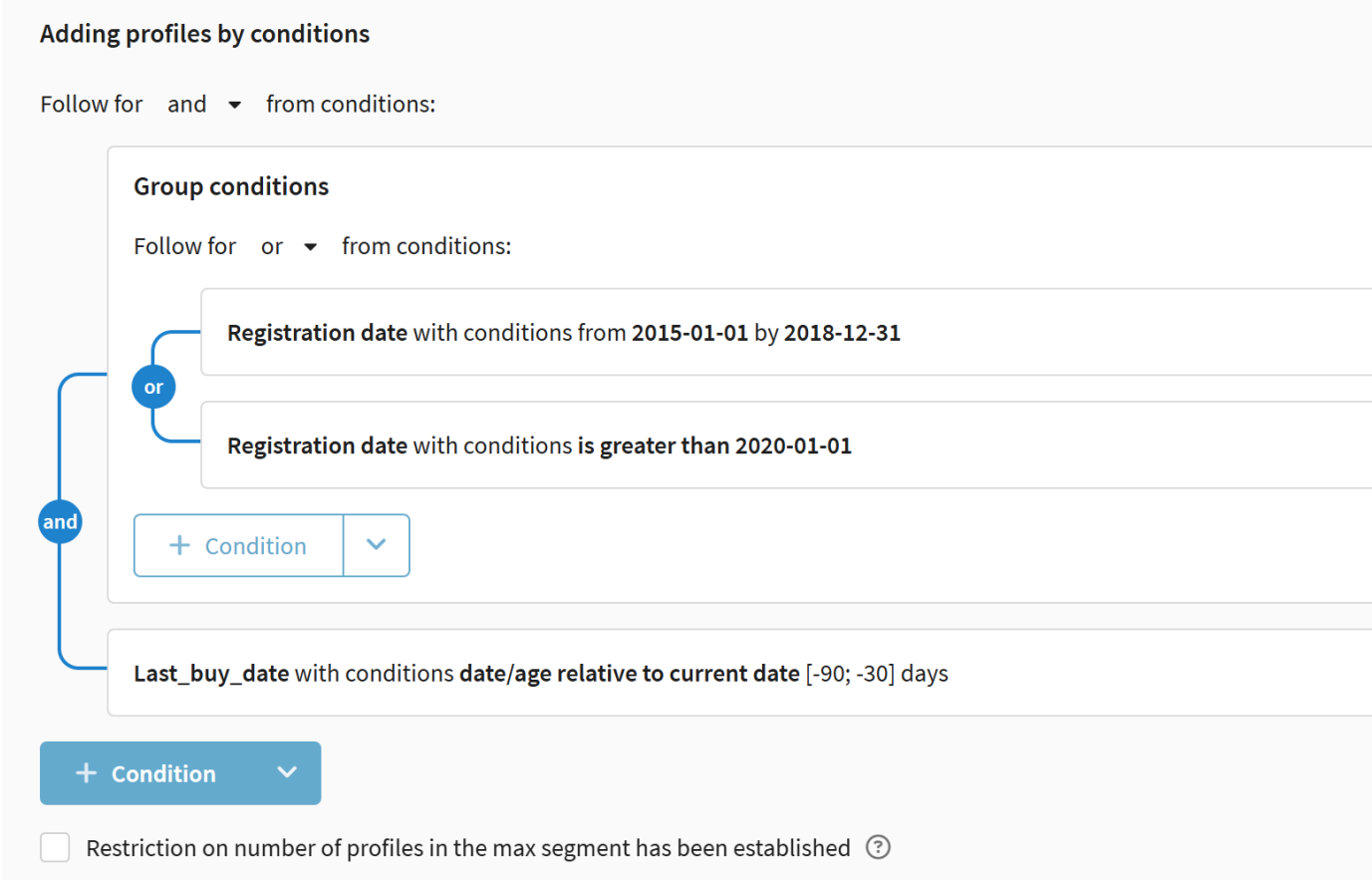
When you click the Add button, the conditions editor opens in the side panel. Here, categories are availa from the selected profile database:

![img](/img/attachments/9407279/en/stat-segment-cond-en.png)

Rules are built based on filters, which can be combined into groups. The following data categories are av

- **Profile data, including raw queries, profile structure, browser and operating system data, statuses, s identifier), main fields (email, phones, first name, last name, date of birth, gender, registration and lo custom fields.**
- **Interactions with entities: participation in scenarios and segments (dynamic or static), behavior in p working with promo codes, connections, forms, orders, goal achievements, participation in mailings test group).**
- **Based on Behavior in channels: interactions with Push notifications, SMS, Email, Telegram Bot, Tele WhatsApp, Viber, and VK Notify.**

Within a group, conditions can work using "AND" logic (must comply with each condition) or "OR" logic (r conditions):



You can quickly load pre-configured conditions by clicking Import Conditions and selecting an existing s rules will be copied.

INFO: Note

A static segment does not save segmentation conditions, so importing conditions is only available from

After configuring the rules, click Add. The system will start calculating the number of matching profiles. profile list in the segment will be updated.

Script

The segmentation query can also be created manually using the built-in JSON editor. To do this, click the

![img](/img/attachments/9407279/en/9407324-en.png)

Removing Profiles by Conditions

To remove profiles from the segment, click Remove. A similar conditions editor will open on the right. It a excluding profiles from the current segment. It also includes rules and rule groups, condition import, and

> NOTE
>
> **The Remove profiles that are in another segment function allows you to quickly exclude from the curren with another segment.**

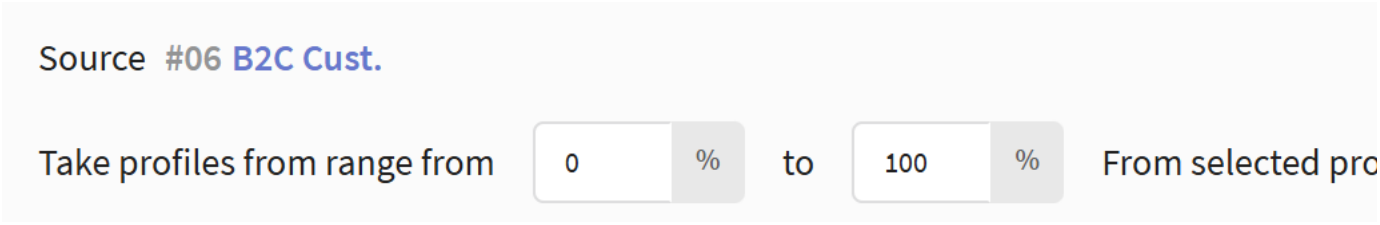
Audience Limitation

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can b segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 pro This restriction applies to the sample result of the segment. If a range is also used, the range restriction count limit is applied to the selected portion of the database.

Range



The range determines which portion of the entire database will participate in segmentation. By default, conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls wi you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% a intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% ra the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the da effect within that portion.

Importing Profiles from Another Segment

To import profiles from another segment, click the down arrow ▾ located next to the main Add button, a Import from Segment:

![img](/img/attachments/9407279/en/stat-segment-import.png)

This option allows you to quickly add to the current segment all profiles from another existing segment.

![img](/img/attachments/9407279/en/new-profile-import-en.png)

> INFO: Important
>
> **You can only select a segment created based on the same profile database. After selecting a segment, statistics, and history will be updated.**

Manual Addition and Removal of Profiles

You can manually add or remove a profile from a segment. To do this, in the profile card, go to the Intera select the desired segment, and edit:

![img](/img/attachments/9407279/en/vruchnuyu-en.png)

Viewing and Analyzing Segment Composition

Profiles Table

After calculation, a table with all profiles included in the segment is displayed in the static segment win table displays system fields, such as “Source Type” (for example, import, conditions) and “Addition Sour These fields help track the history of how each profile entered the segment.

![img](/img/attachments/9407279/en/stat-segment-table-en.png)

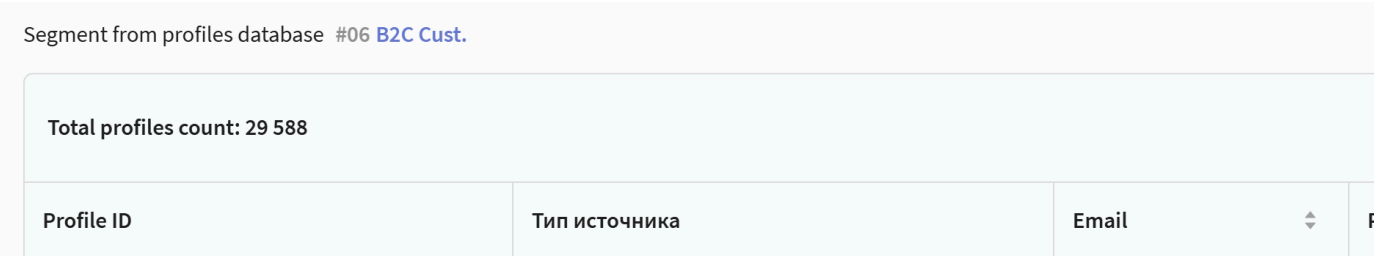
The following actions are available for working with this table:

- **Search**

  : find a specific profile by name, email, or other data.
- **Export Profiles**

  : export the displayed list to an external file.
- **Configure Displayed Fields**

  : to add or remove columns in the table, drag the desired Field from the side column Add Field to the



Update History

Below the profiles table is the Update History block. It visually presents information about changes in th segment over the selected period. You can specify the desired period (start and end dates) and group da

![img](/img/attachments/9407279/update-history.png)

When hovering the cursor over a chart column, a tooltip appears. It shows the total change in the numbe specific actions and in what quantity profiles were added to or removed from the segment:

![img](/img/attachments/9407279/en/update-history-2-en.png)

The tooltip groups changes by the following action types:

Added

- **Import from another segment: adding all profiles from another segment.**
- **Import from file to database: automatic addition of profiles after importing a file into the Profile Data was enabled.**
- **By segment conditions: addition based on filters (the “Add” operation).**
- **From profile card: manual addition via the profile card (the Interactions tab).**
- **From scenario node: addition of profiles by the “Update Static Segment” node in a scenario.**
- **Via API: addition via API.**

Removed

- **Import from another segment: removal of all profiles that are in another segment (the “Remove profil operation).**
- **By segment conditions: removal based on filters (the “Remove” operation).**
- **From profile card: manual removal via the profile card.**
- **From scenario node: removal of profiles by the “Update Static Segment” node in a scenario.**
- **From profile database: removal of a profile from the Profile Database itself (leads to its automatic re**
- **Via API: removal via API.**

Next to each action type, the exact number of affected profiles is indicated.

Manual Changes Log

Below the update history is the Manual Changes Log. This is a detailed log of all operations that affecte each operation, the following is displayed:

- **Start Date: the time when the segmentation process started (addition/removal by conditions).**
- **End Date: the time when the process ended.**
- **Initiator: the name of the user who initiated the operation.**

- **Added: the number of profiles added as a result of the operation.**
- **Removed: the number of profiles removed as a result of the operation.**
- **Status: the current state of the operation (for example, “Completed”, “In Progress”, “Error”).**

Manual updates journal

Start date	End date	Initiator	Added
29-12-2025 14:11	29-12-2025 14:11	New User	17265
29-12-2025 14:07	29-12-2025 14:08	New User	5
16-12-2025 16:41	16-12-2025 16:41	New User	12317

NOTE

The log does not display entries about adding profiles via API or from a scenario.

Dependent Objects

Below the manual changes log, the Dependent Objects block may appear. It appears automatically if the other platform entities.

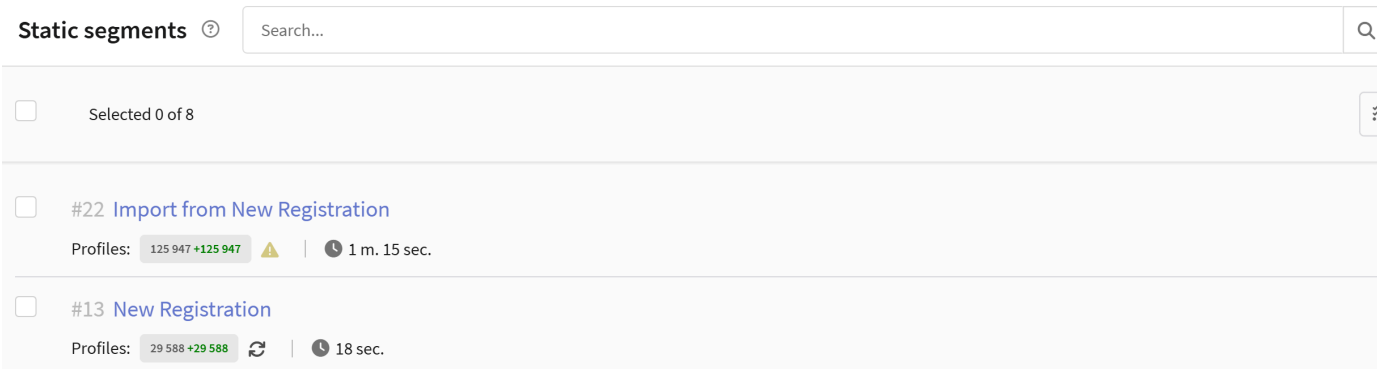
This block shows exactly where your segment is used (for example, in mailings or scenarios).

![img](/img/attachments/9407279/en/stat-segment-dependent-objects-en.png)

Segment Management

In the row of the desired static segment in the general list, the following actions are available for workin

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected s**
- **Update Profiles: Mass-modify profile data in the segment (for example, update fields, add a subscript**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Create an exact copy of the segment (with profiles, or without them).**
- **Delete Segment: Move the segment to the bin.**



Dynamic Segments

A dynamic segment is a segment that selects users directly at the moment of each use.

You can work with such a segment without saving it if you need a one-time quick selection. If you decide in mailings, scenarios, etc. Each time it runs, the segment will recalculate the profile selection based on

![img](/img/attachments/9407282/en/dynamic-segments-navigation-en.png)	To get started, go to **Data → Segments → Dynamic Segments**.

Creating a Dynamic Segment

In the dynamic segments list, click + Create. In the form that opens, specify the Name of the segment ar the side panel, select one Profile Database — this is a mandatory step; segment creation is impossible w

![img](/img/attachments/9407282/en/dynamic-segment-creating-en.png)

> **INFO**
>
> **Please note that after selecting a source, it cannot be changed.**

Configuring Range and Conditions

Here you can also set the Range, which defines the portion of the database to which the segmentation c

![img](/img/attachments/9407282/en/dynamic-segments-range-en.png)

By default, the range is set from 0% to 100% of the entire selected profile database. Profiles are selecte always selects the same users.

The main block is Condition Settings. To add a rule, click the + Condition button. In the dropdown menu, Import Conditions from another segment are available. After selection, the condition settings window wi

![img](/img/attachments/updatable-segments/en/upd-segments-range-1-en.png)

The mechanics are completely identical to static segments: all the same filter categories by profile data channels are available. Conditions within a group can be combined using must comply with each of the c one of the conditions (OR) logic.

In the “Condition Settings” block, the Delete button (clears the set rules) and the Script button (opens ar conditions in JSON format) are also available.

After configuring the conditions, in the same window, you can activate the checkbox Set a limit on the m the desired number. This limitation is applied to the selection result after selection by range and conditi

![img](/img/attachments/9407282/en/dynamic-segments-conditions-limits-en.png)

More about how audience limitations work

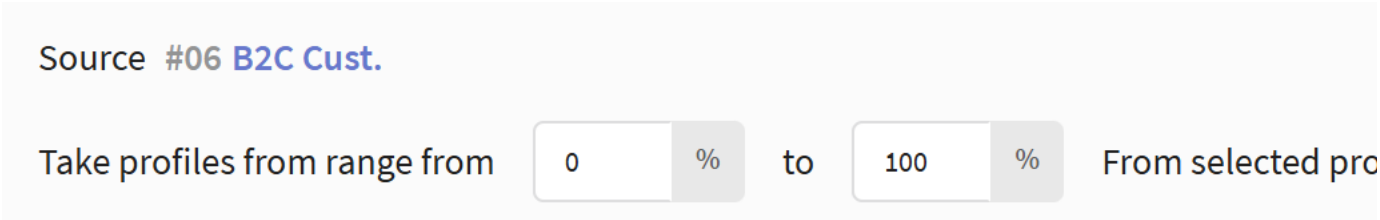
Limit on the maximum

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can b segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 pro This restriction applies to the sample result of the segment. If a range is also used, the range restriction count limit is applied to the selected portion of the database.

Range



The range determines which portion of the entire database will participate in segmentation. By default, conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls wi you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% a intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% ra the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the da effect within that portion.

Working with the Segment

Quick Segment

Before clicking the “Save” button, a dynamic segment can be used as a quick segment — for example, to static segment without the need to save the segment itself.

After adding at least one condition, additional actions for working with the current profile selection becc corner, next to the fields for tags and access groups:

- **Copy Profiles to Profile Database;**
- **Copy Profiles to Static Segment;**
- **Export Profiles;**
- **Bulk Edit Profiles;**
- **Add Profiles to a Scenario.**

![img](/img/attachments/9407282/en/dynamic-segments-actions-en.png)

Preview and Save

Also, after configuring the conditions, you can immediately estimate the size of the future audience. Clic calculation block — the system will process the database and show how many profiles match the conditi

![img](/img/attachments/9407282/en/dynamic-segmenets-number-of-profiles-en.png)

To see the profiles themselves, click Show — a table with the first 100 profiles from the calculated selec

![img](/img/attachments/9407282/en/dynamic-segments-subscribers-en.png)

After calculation, the profile count can be manually updated using the corresponding button next to the

Fill in the remaining fields in the form: if necessary, specify Description, Tags, and Access Groups. Then will automatically calculate its audience for the first time.

> **INFO**
>
> **If no conditions are set when saving the segment, all profiles from the selected database within the set included in the segment.**

Dependent Objects

If a dynamic segment is used in other platform entities (for example, in mailings or scenarios), the Deper automatically appear below the update log. This block shows exactly where your segment is used:

![img](/img/attachments/9407282/en/dynamic-segment-dependent-objects-en.png)

Managing Segments from the General List

In the row of the desired dynamic segment in the general list, the following actions are available for wor

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected s**
- **Update Profiles: Mass-modify profile data in the segment (e.g., update fields, add a subscription).**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Clone the segment conditions, tags, access groups, and Profile Database.**
- **Delete Segment: Move the segment to the bin.**

![img](/img/attachments/9407282/en/dynamic-segments-actions-in-list-2-en.png)

Updatable Segments

A scheduled segment is a segment that automatically recalculates its composition based on specified c (for example, once a day). Its list of profiles is current at the moment of the last recalculation.

INFO

- **Filling: The segment cannot be filled manually or via API. It is filled only based on specified conditic during manual recalculation.**
- **Usage: The segment can be used in the same places as a static segment (for example, in mailings,**

Creating a Segment

![img](/img/attachments/updatable-segments/en/upd-segements-navigation-en.png)	To get started, go to **Data → Segments → Scheduled Segments**.

In the scheduled segments list, click + Create. In the form that opens, specify the Name of the segment the segment source (Database). Segment creation is impossible without selecting a database:

![img](/img/attachments/updatable-segments/en/upd-segements-creation-en.png)

INFO

After selecting a source, it cannot be changed.

Configuring Conditions

After selecting the database, the Conditions and Schedule block will open. The mechanics of configuring to dynamic and static segments. You can set the Range, which defines the portion of the database for ap

![img](/img/attachments/updatable-segments/en/upd-segments-range1-en.png)

To add a rule, click the + Condition button. In the dropdown menu, the options Add Condition Group and I configuring conditions within a group, they can be combined using must comply with each of the conditi the conditions (OR) logic:

![img](/img/attachments/updatable-segments/en/upd-segments-cond-en.png)

In the same block, the Delete button, which completely clears all set rules, and the Script button, which conditions, are available.

After configuring the conditions, you can set limitations by activating the checkbox Set a limit on the ma segment to specify a limit. This limitation is applied to the selection result after selection by range and c

More about how audience limitations work

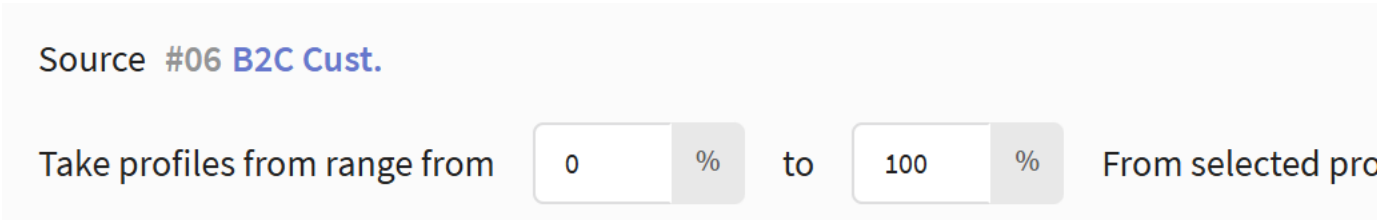
Limit on the maximum

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can b segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 pro This restriction applies to the sample result of the segment. If a range is also used, the range restriction count limit is applied to the selected portion of the database.

Range



The range determines which portion of the entire database will participate in segmentation. By default, conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls wi you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% a intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% ra the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the da effect within that portion.

Configuring the Schedule

In the Conditions and Schedule block, configure the parameters for automatic updates. Specify the exac the timezone. Then set the frequency of recalculation from the available options: daily, on a specific dat the month, etc.):

![img](/img/attachments/updatable-segments/en/upd-segments-timetable-en.png)

Optionally, you can activate the function Deactivate schedule after the specified date so that automatic

After configuring all parameters, click Save. The first calculation will start at the nearest scheduled time

Working with a Created Segment

After creating and saving the segment, its page will open. Now, in the upper right corner, next to the fiel additional actions for working with the current profile selection become available:

- **Copy Profiles to Profile Database;**
- **Copy Profiles to Static Segment;**
- **Export Profiles;**
- **Bulk Edit Profiles;**
- **Add Profiles to Scenario.**

![img](/img/attachments/updatable-segments/en/upd-segments-actions-en.png)

> INFO
>
> **Actions with profiles will become available after the first segment update.**

Managing Automatic Updates

The Automatic Segment Update block displays the current schedule. From here, you can Stop"/"Start (de Reconfigure Schedule, or Launch Manual Recalculation outside the scheduled time.

![img](/img/attachments/updatable-segments/en/upd-segments-upd-timetable-setting-en.png)

> CAUTION
>
> **Manual recalculation and editing of the update schedule are unavailable during segment update execut**

Below is the Show Update Conditions spoiler. When expanded, it displays all set conditions, range, and li can be viewed, or they can be deleted.

![img](/img/attachments/updatable-segments/en/upd-segmnets-cond-settings-en.png)

Analyzing Composition and History

After the first update, the Profiles Table with the current composition will become available on the segm

![img](/img/attachments/updatable-segments/en/upd-segments-user-table-en.png)

Update History

The Update History block is a graph visualizing changes in the number of profiles in the segment over th set using the calendar or ready-made options (today, last 7 days, this month, etc.), and data can be grou quarters, or years. When hovering over a point on the graph, a breakdown by profiles is displayed: added manually:

![img](/img/attachments/updatable-segments/en/upd-segments-upd-history-en.png)

Log

The Log block is a table with detailed information for each recalculation run. For each operation, the sta (Manual / Automatic), initiator (user or “System”), number of added and removed profiles, and status (`In` indicated.

![img](/img/attachments/updatable-segments/en/upd-segments-journal-en.png)

Dependent Objects

If the segment is used in other platform entities (mailings, scenarios, loyalty programs), the Dependent C update log:

![img](/img/attachments/9407282/en/dynamic-segment-dependent-objects-en.png)

Managing Segments from the General List

In the row of the desired scheduled segment in the general list, the following actions are available:

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected s**
- **Update Profiles: Mass-modify profile data in the segment (e.g., update fields, add a subscription).**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Clone the segment conditions, tags, access groups, profile database, and schedule.**
- **Delete Segment: Completely delete the segment.**

![img](/img/attachments/updatable-segments/en/upd-segment-actions-en.png)

Best Send Time (BST)

Best Send Time (BST) is a mechanism that determines the optimal time to send messages for each profil by a machine learning algorithm that identifies the time of highest profile activity based on data collecte

Sending messages at the right time directly impacts communication performance:

- **Open Rate increases as messages arrive when users are most active;**
- **engagement grows — users interact with content more frequently;**
- **the risk of messages being marked as spam decreases due to higher open rates;**
- **customer loyalty improves thanks to more relevant communication;**
- **resources are optimized through more efficient campaigns.**

How Best Send Time is calculated

The BST model uses machine learning algorithms to analyze user behavior. The ML service analyzes prof when the probability of message opening is highest. For each profile, one or several preferred sending ho

During analysis:

- **data from the last 6 months is used;**
- **message opens and other behavioral events are processed;**
- **the best hour or several hours for sending are determined.**

BST data is regularly recalculated (by default, once a day at 00:00 in the account time zone).

> CAUTION
>
> **By default, the ML model uses profile activity data from the last 6 months to determine the optimal sen of sends and opens within this period, the system cannot calculate BST for it.**

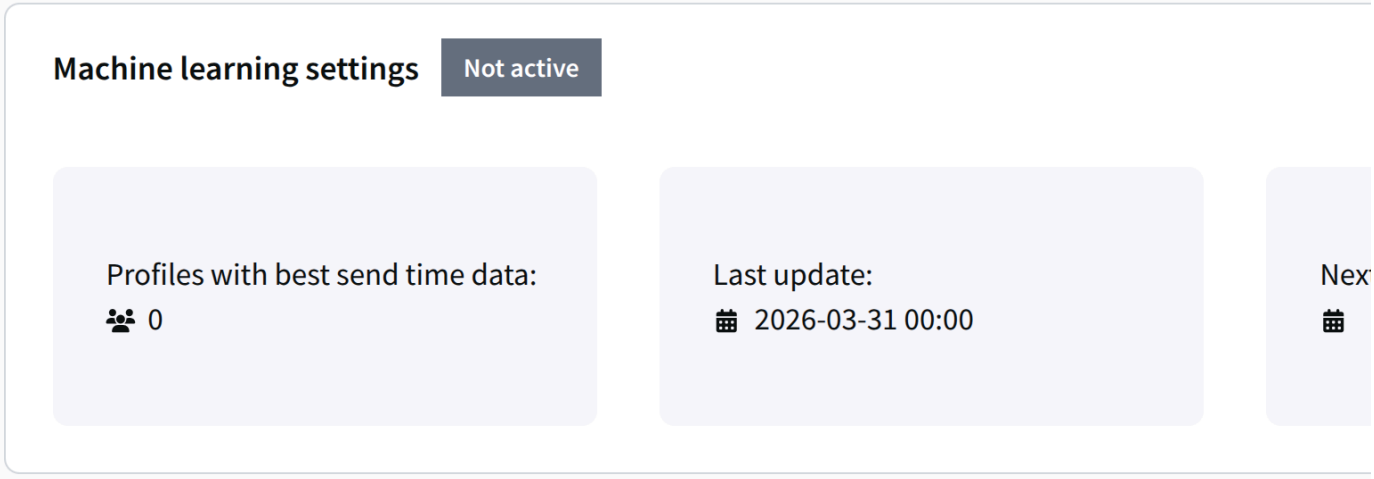
Using Best Send Time

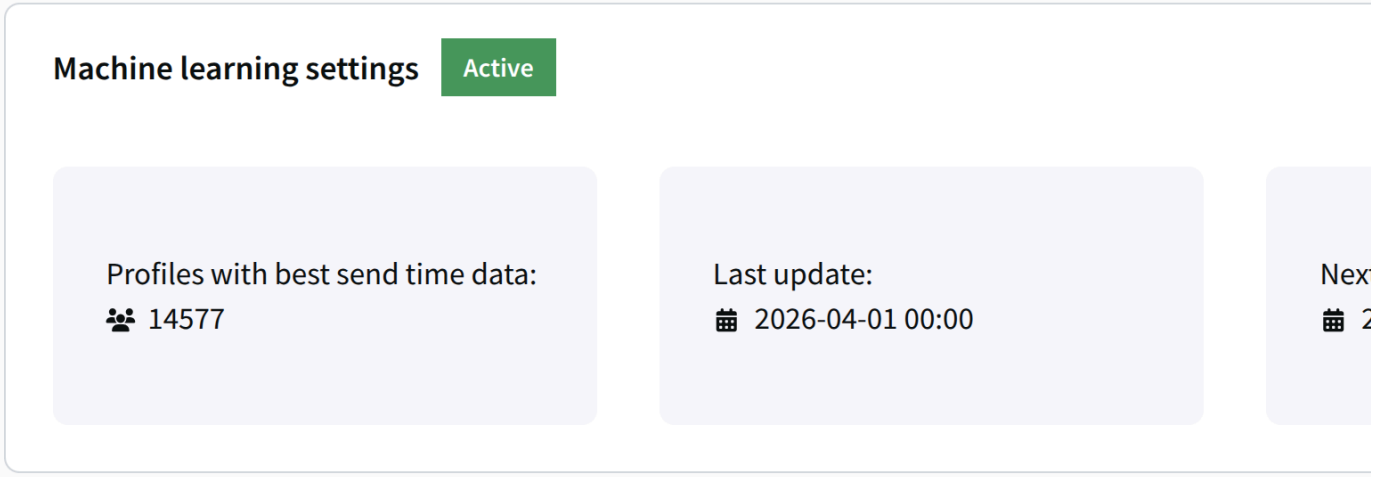
BST-based segmentation allows selecting profiles for which the current or specified time is optimal for s Mailing by selecting the segment audience as recipients — each user will receive the message at the mo

> INFO
>
> **To use the ML service, it must be activated at the platform level. If you are using the On-premises versi team.**

To calculate the best send time, you need to enable ML for the profile database. Go to the database setti and activate the module.

After activation, the model will start collecting and processing data. You can check the next update time enabled recently, wait for the first recalculation.





BST can be used as a condition for building a dynamic segment or in an automation Scenario node.

BST Segmentation

![img](/img/attachments/89325640/en/create-segment.png)

![img](/img/attachments/89325640/en/condition.png)

![img](/img/attachments/89325640/en/operator.png)

![img](/img/attachments/89325640/en/regular.png)

Scenario Node

This approach in Scenarios is suitable for complex communications with trigger chains and advanced log

![img](/img/attachments/89325640/en/scenario.png)

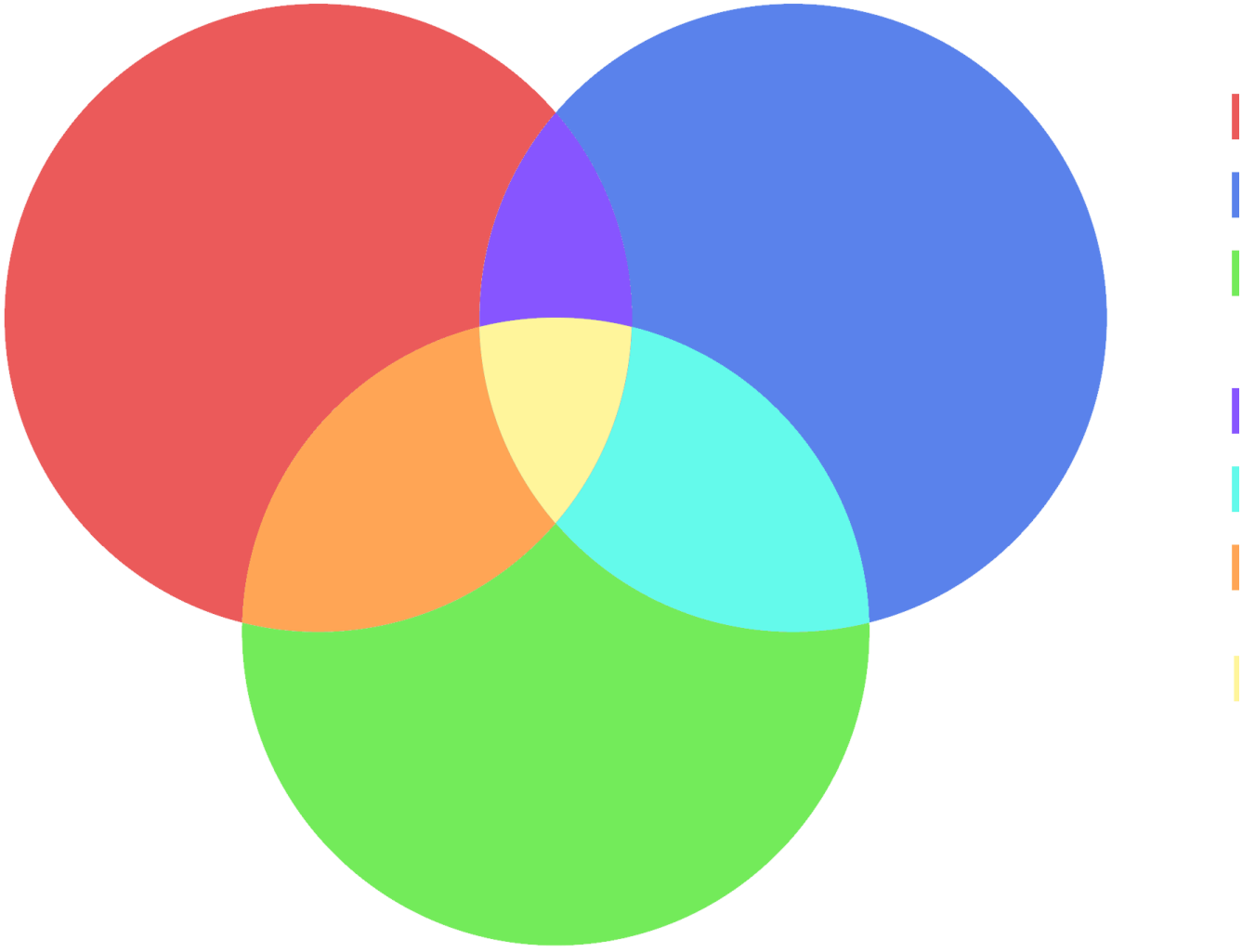
![img](/img/attachments/89325640/en/timer.png)

Logical operators “AND” and “OR”

**Segmentation on the platform is based on conditions. Often, a single condition is not enough to calculate
In such cases, it is necessary to specify two or more selection conditions or group of rules. The conditio
logical operators of formal logic - AND and OR. In this article, we will use examples to explain how condi**

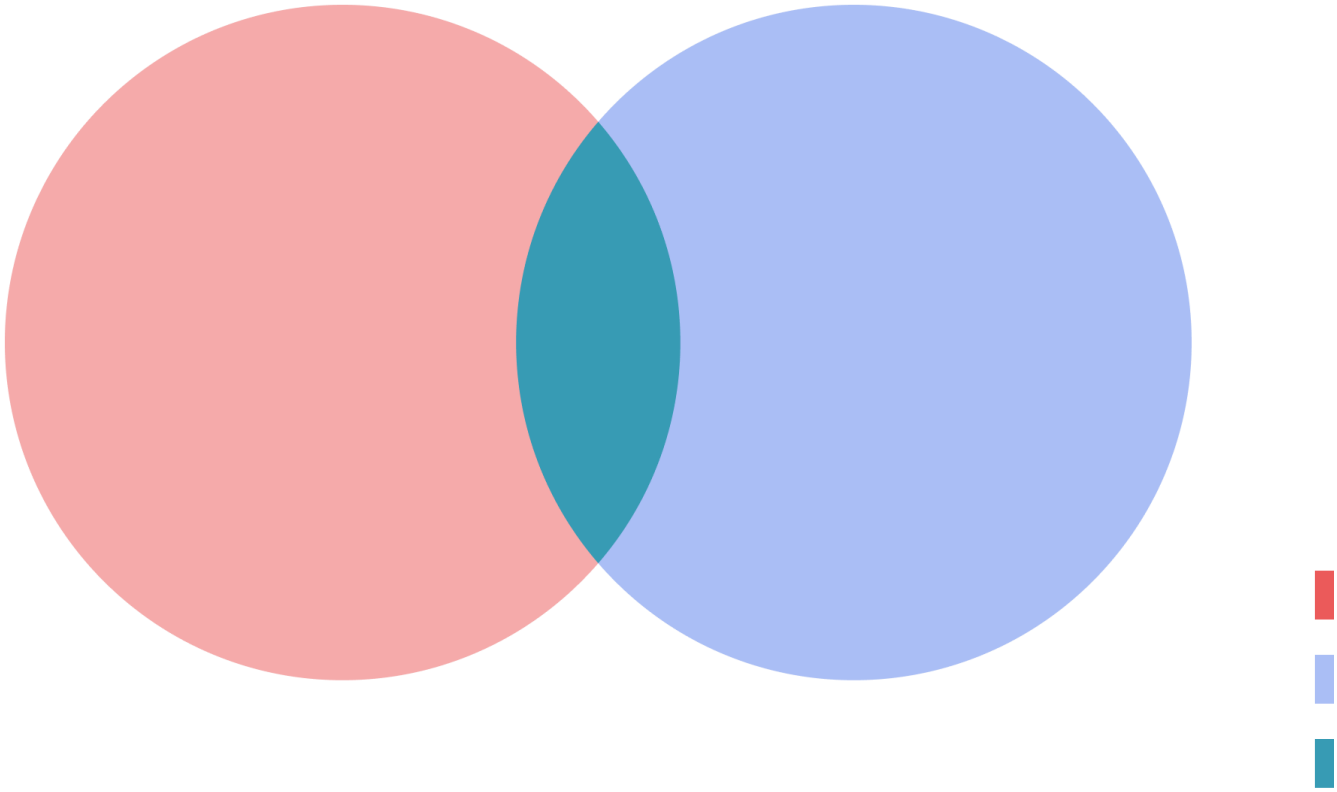
Operator "AND"

**The “AND” operator will select those subscriberss for which all rules and rule groups in the query are va
elements of the elements being joined must strictly match the condition. It can be represented schemati**

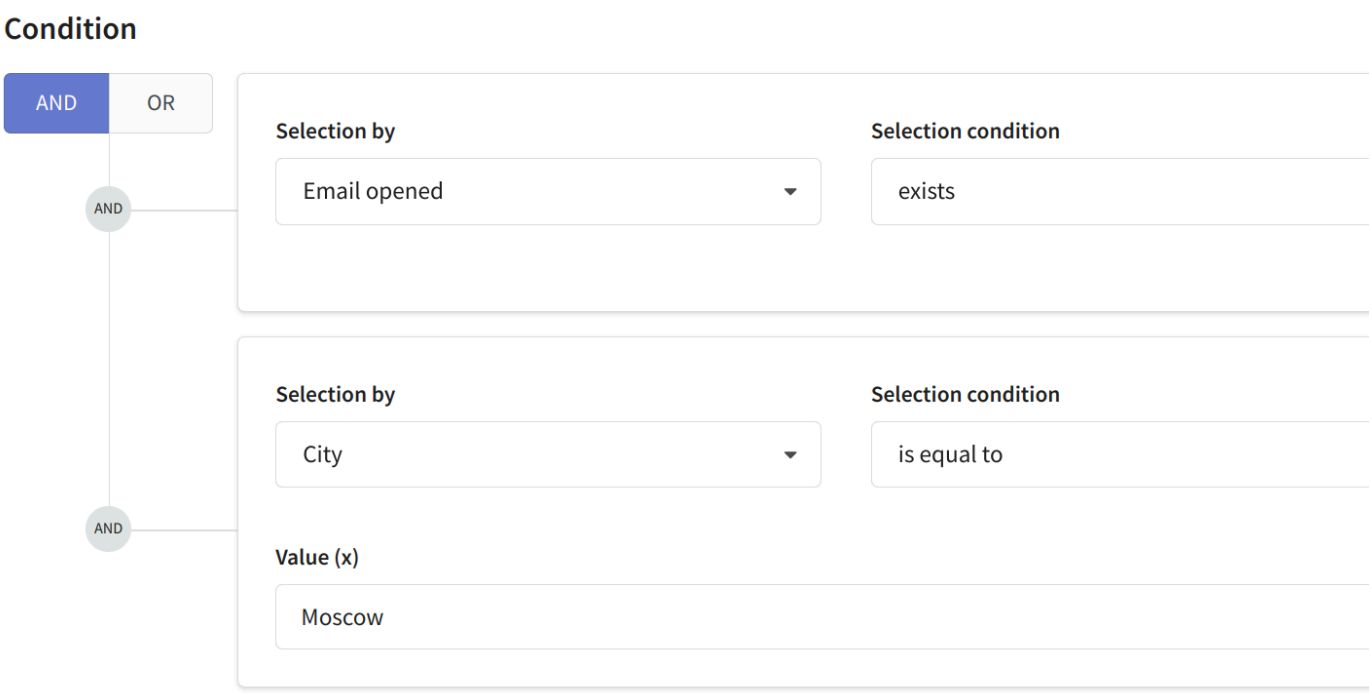


Example

**Let’s model the situation. To send out an invitation to a promotional event, we need to select subscribers
your email at least once:**

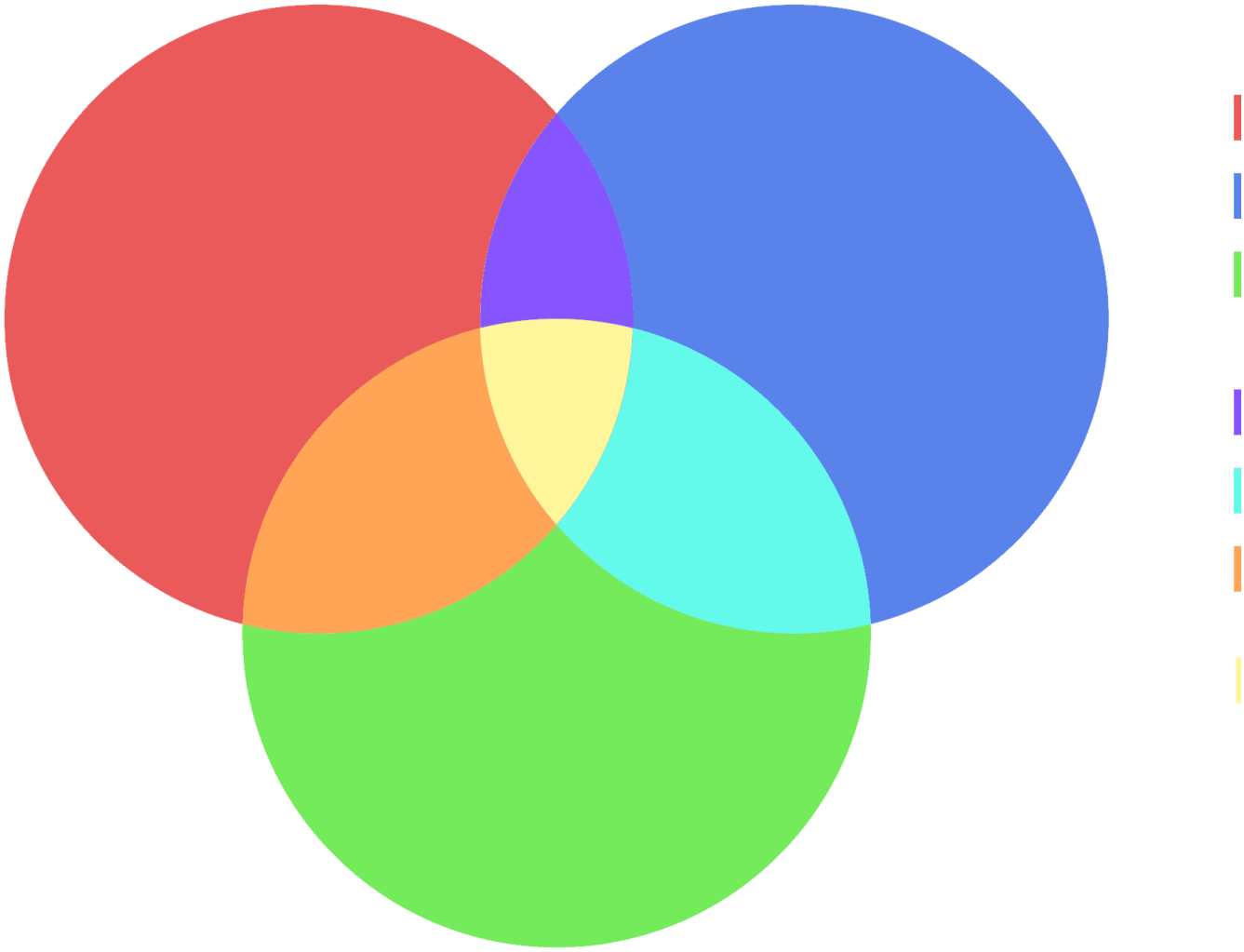


The conditions will look like this:

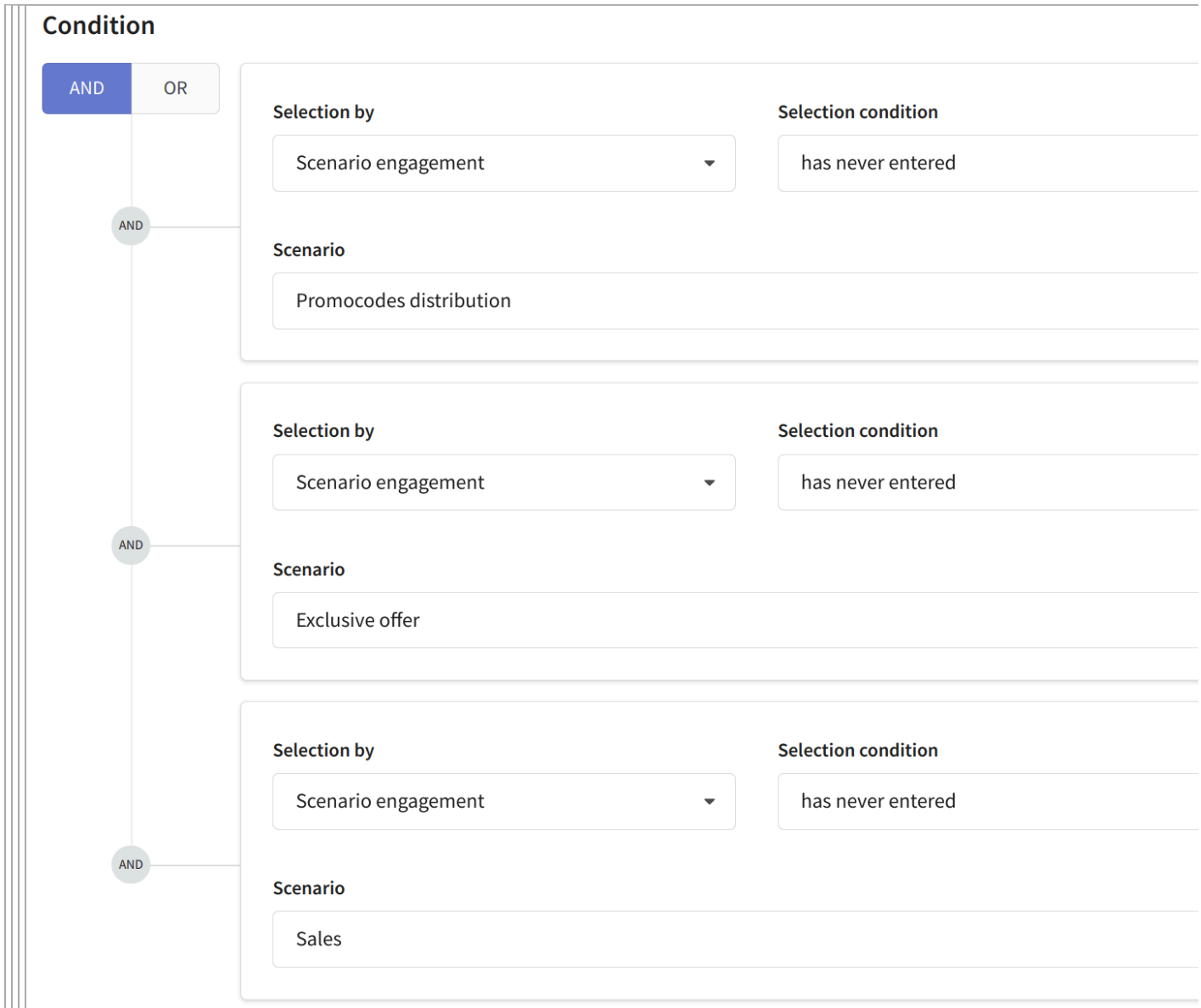


This way, we will select those customers who meet both conditions.

Let's say you've previously created several scenarios with different offers. Now you need to select those from you:



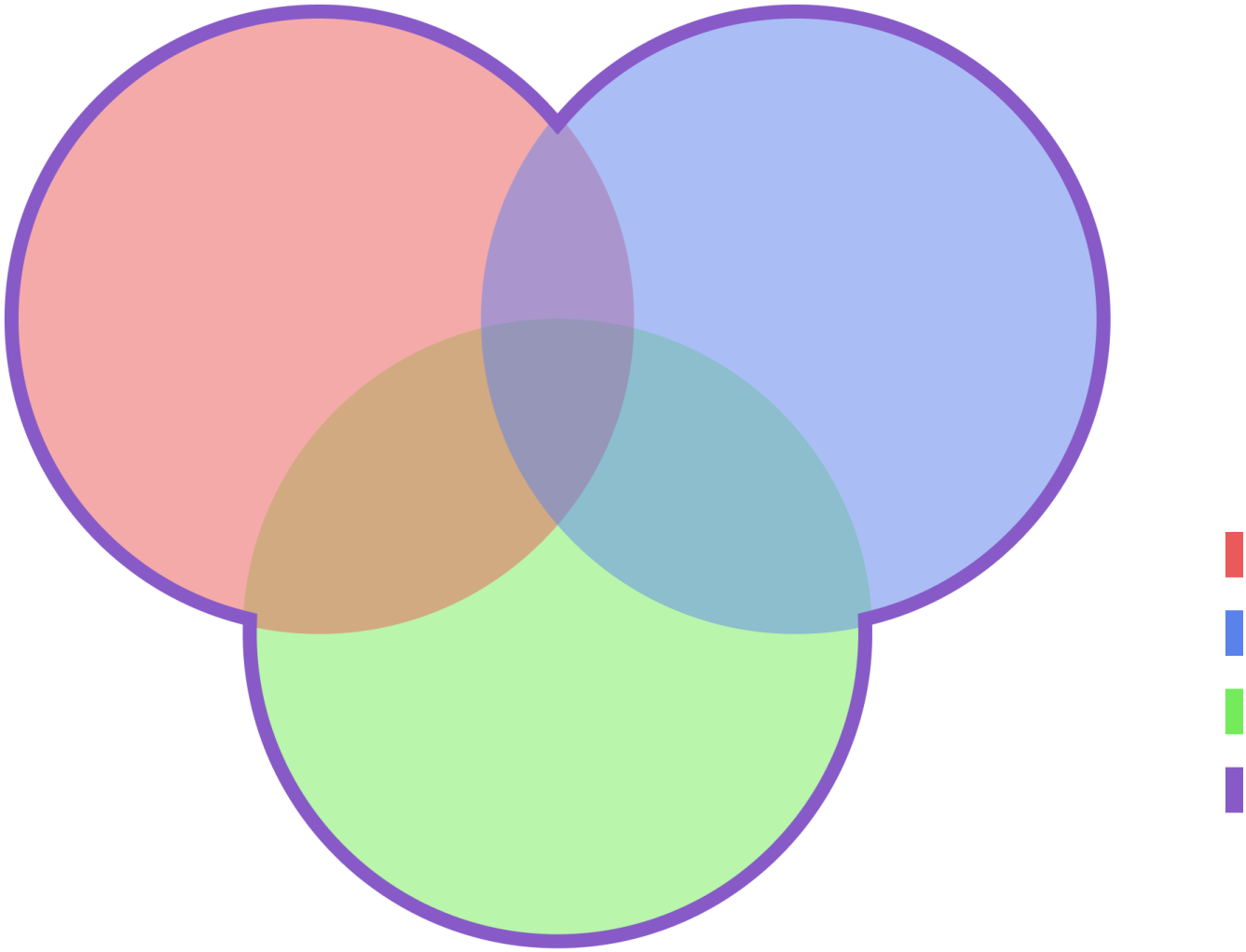
In this case, the conditions will also be connected by the “AND” operator:



This segment will include those profiles that have not received any discount messages, emails about exc promo codes.

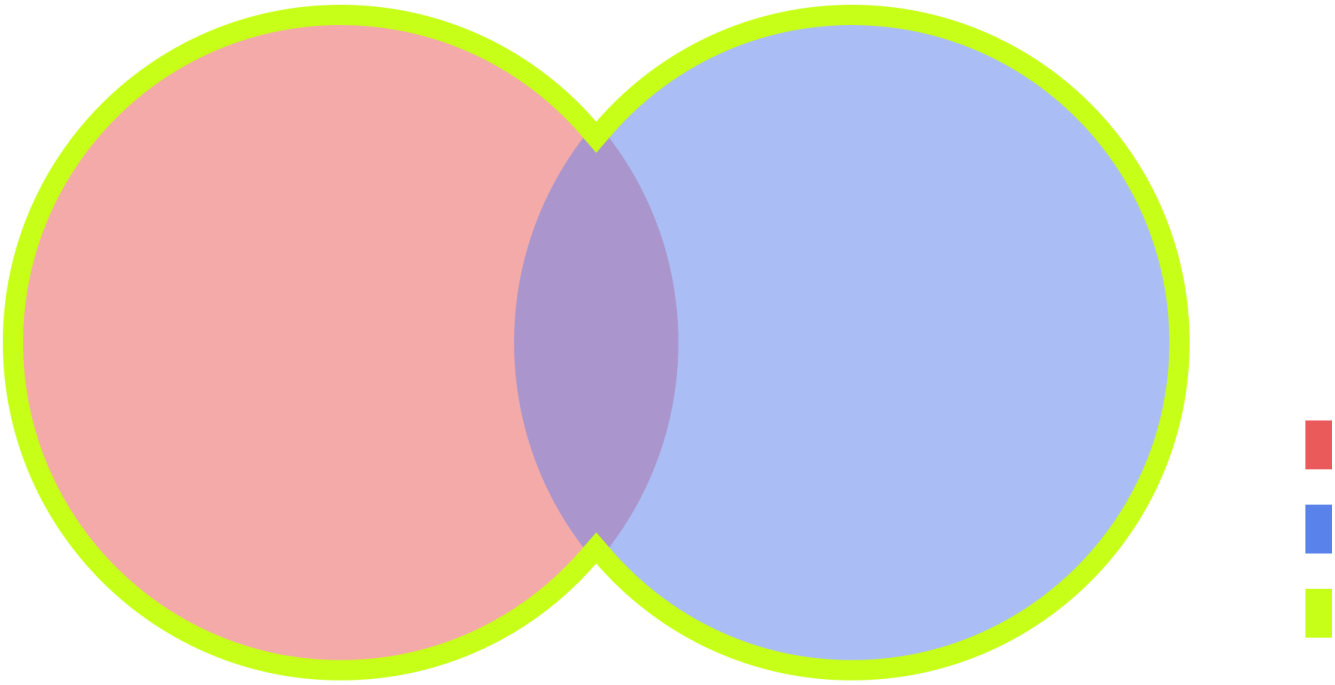
Operator "OR"

The “OR” operator selects those profiles for which at least one rule or group of rules is valid. If a profile included in such a segment. The scheme of its work is as follows:

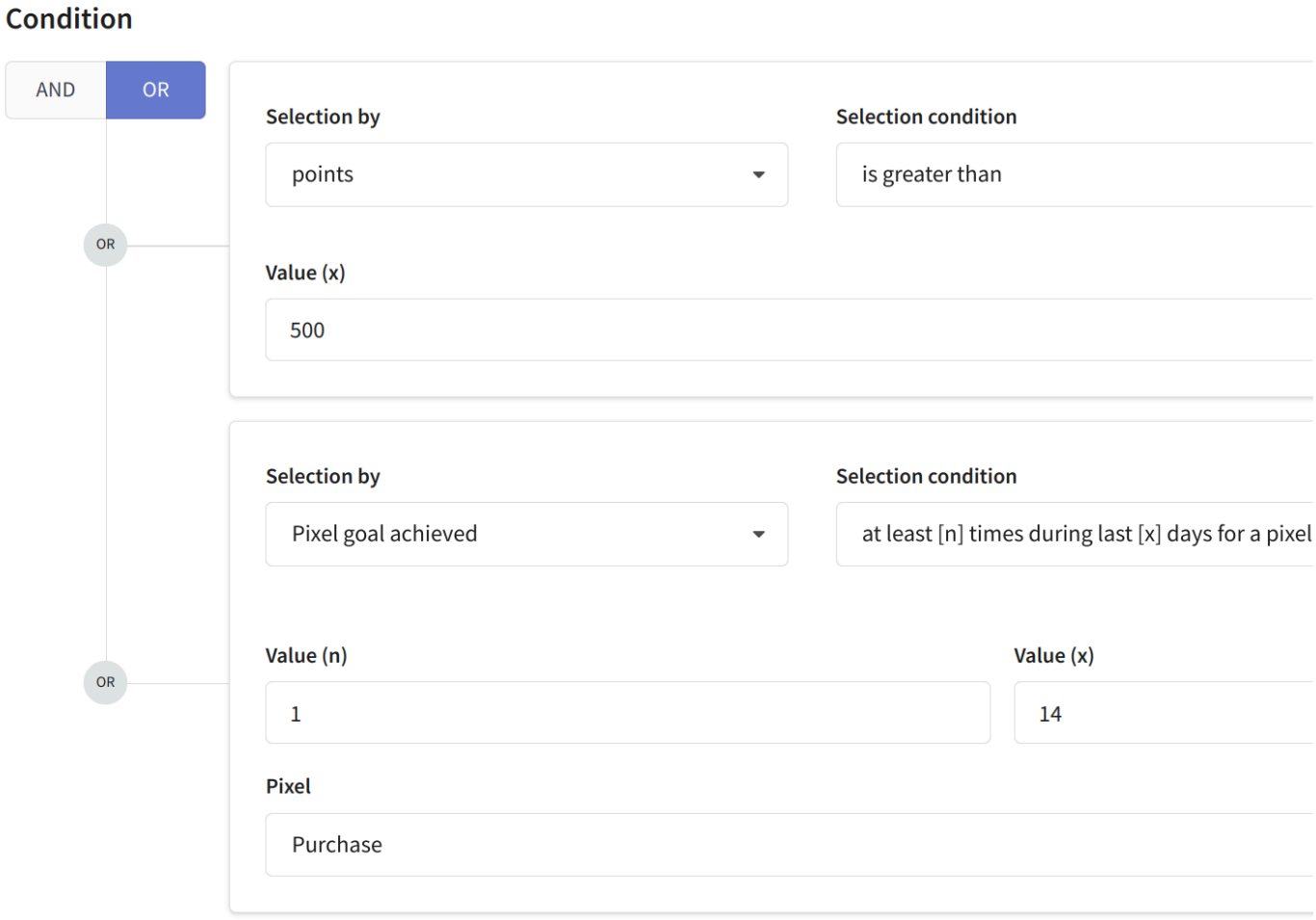


Example

Let’s assume that you came up with an idea to reward active customers with a special promo code. We v
either made a purchase in the last 2 weeks or have more than 500 bonus points on their account:

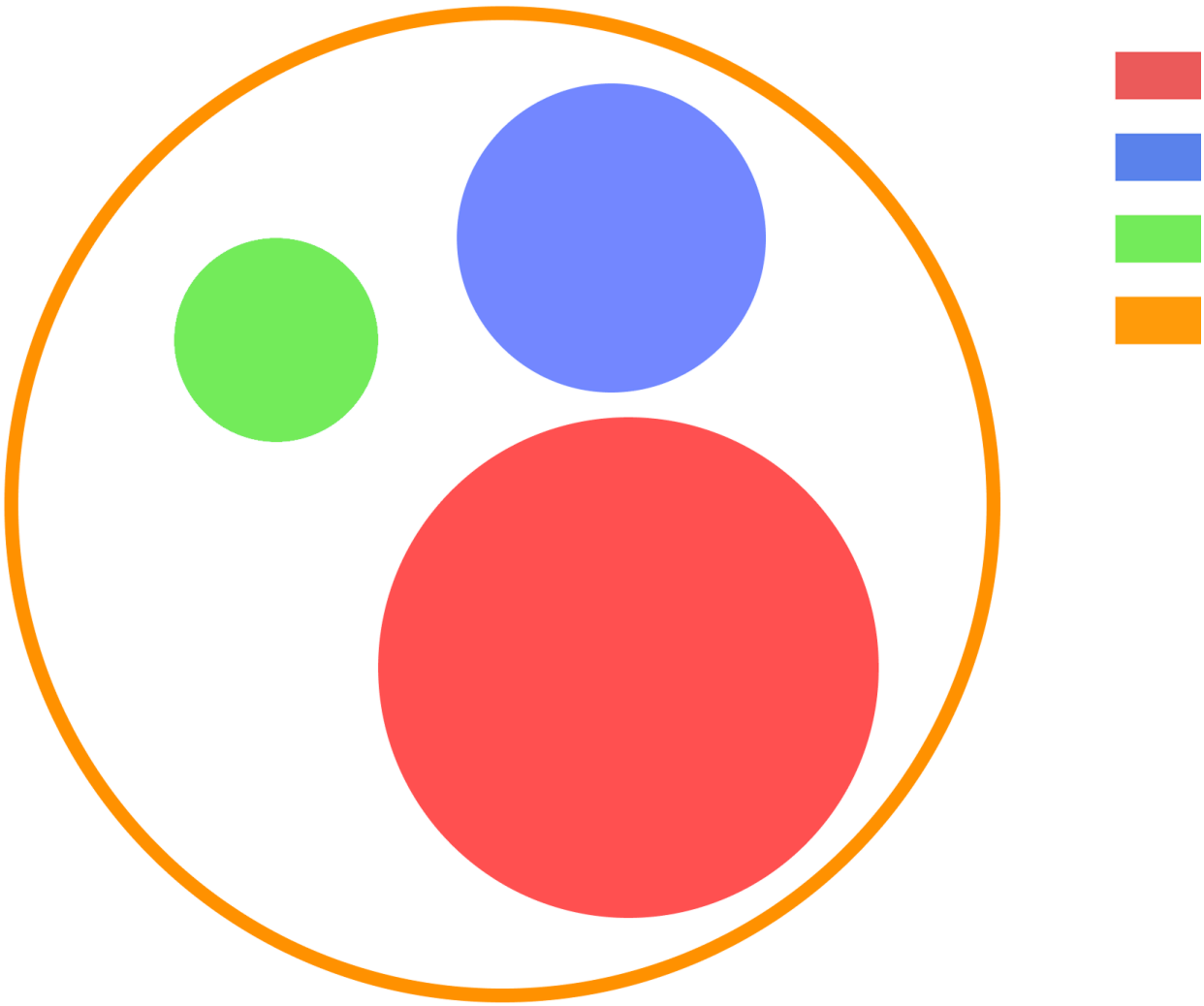


And a condition:

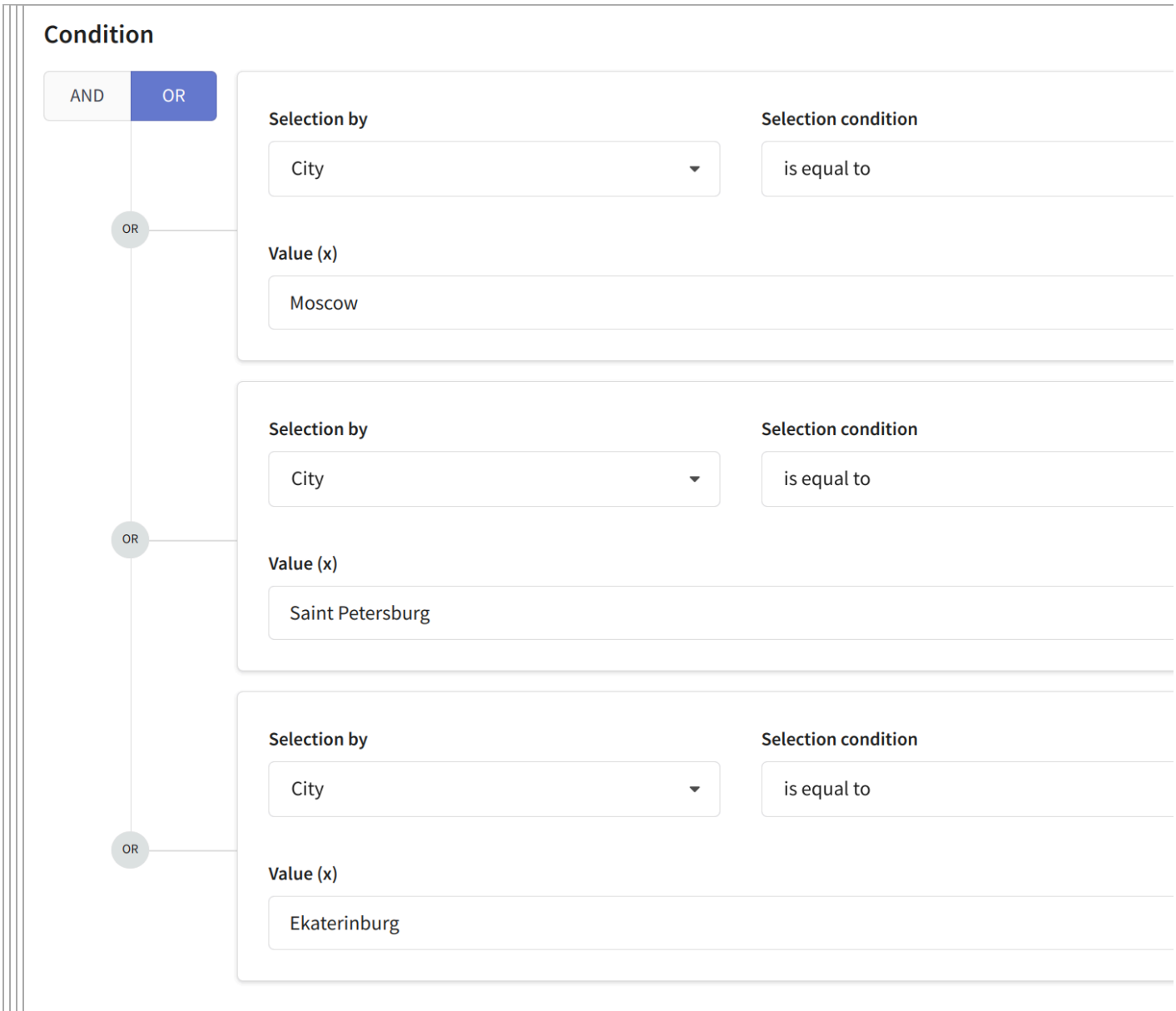


If any of these conditions are met, the customer will receive a promo code.

If you need to select people from few different cities as your audience, you would also use the OR operat



The conditions will look like this:



INFO

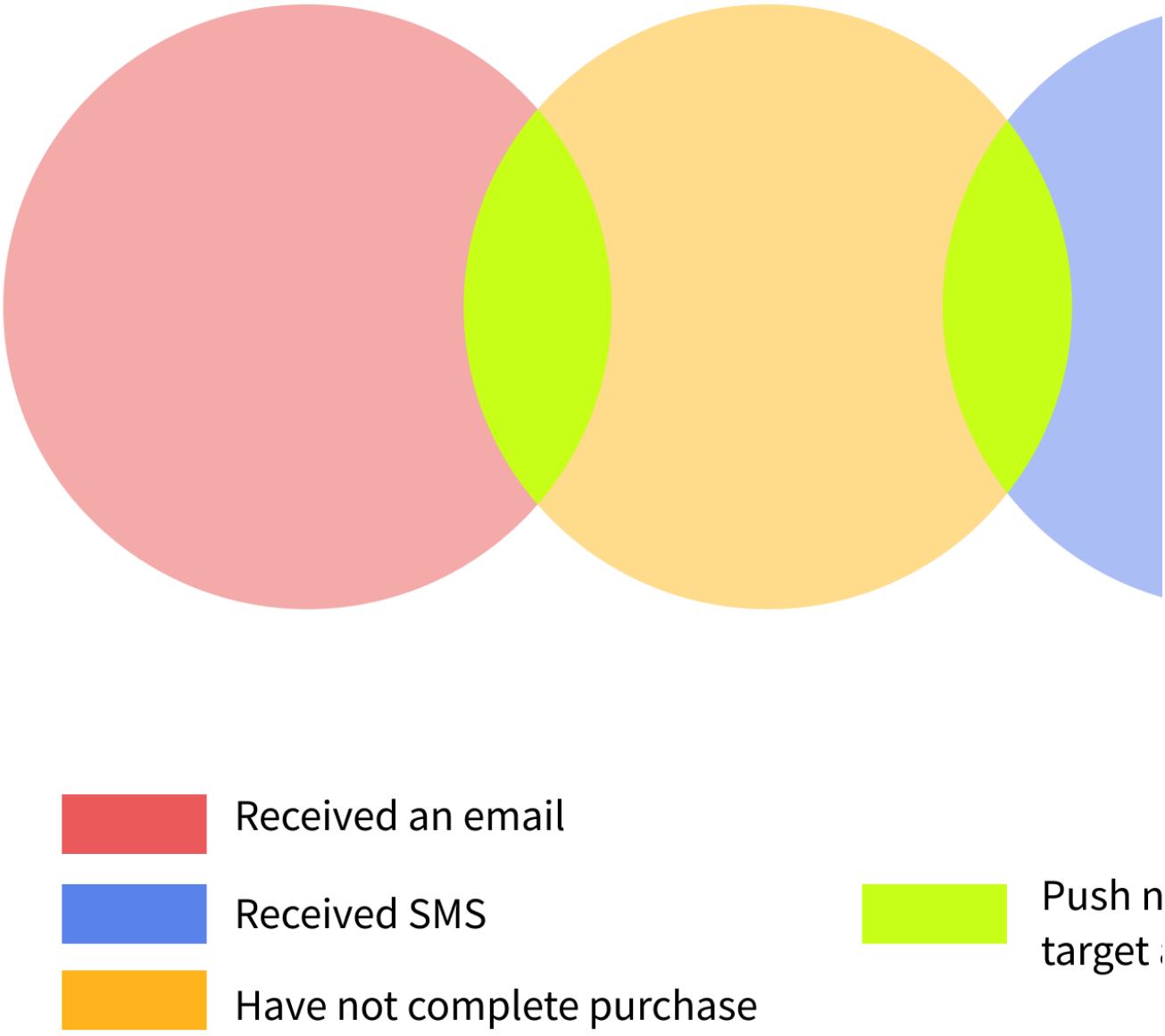
Using a condition with the “OR” operator and negations can lead to errors. Such conditions often conta
Avoid using these constructions — it’s better to replace them with equivalent conditions using “AND”. If
check with a visual logic diagram.

Group of rules

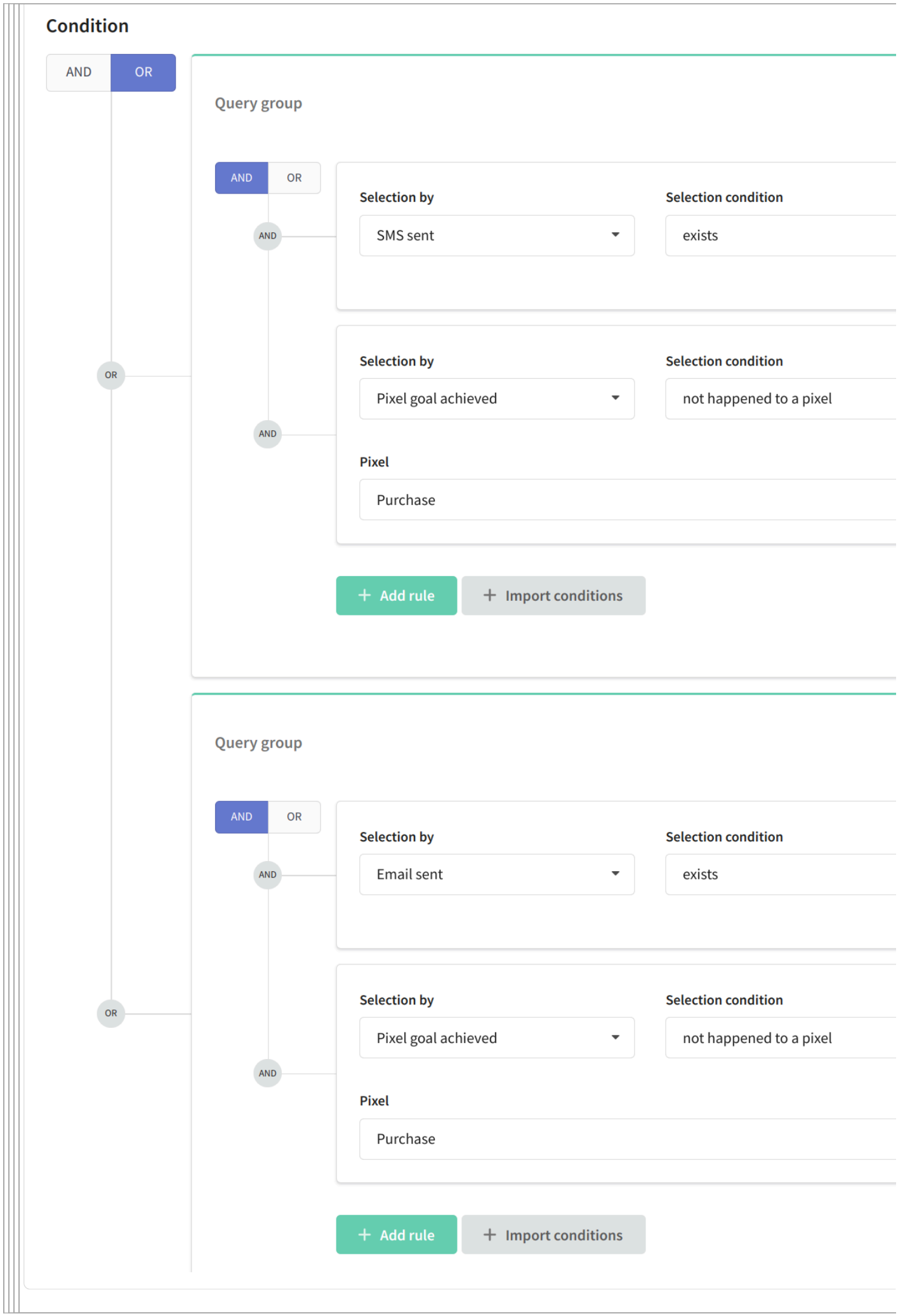
You can link multiple rules of the same level using either OR or AND, but you cannot use them together.
must use a group of rules. Within a rule group, you can use an operator different from the one used to li
allows you to build more complex audiences for your marketing mailings.

Example

Let’s explain it with an example. Let’s imagine a situation in which you need to create a Viber mailing fo
who have received both email and SMS messages, but never made a purchase. Schematically, the audie



For such a condition, the rule set will be composed as follows:


Condition
AND
OR
Query group
AND
OR
AND
Selection by
SMS sent
Selection condition
exists
OR
Selection by
Pixel goal achieved
Selection condition
not happened to a pixel
AND
Pixel
Purchase
+ Add rule
+ Import conditions
Query group
AND
OR
AND
Selection by
Email sent
Selection condition
exists
OR
Selection by
Pixel goal achieved
Selection condition
not happened to a pixel
AND
Pixel
Purchase
+ Add rule
+ Import conditions

Recommendations for working with segments

Segments are a key tool for working with data in a CDP platform. The quality of selections, analytics, an depends on how they are configured.

Segmentation is one of the most resource-intensive processes in the platform. Not all segments will wor how they are designed: what conditions are used, what fields are involved, and how external data source

This article provides recommendations for building efficient segments — from choosing a segment type t queries.

Segment types and calculation speed

Different segment types affect performance differently:

- **Static segments — calculated once at creation or manual update. After calculation, the profile list is segment does not require recalculation.**
- **Dynamic segments — recalculated every time they are used (when launching a mailing, campaign, et slow down mailing launches.**
- **Updatable segments — recalculated on a schedule. The load is distributed over time, but with a large frequent schedules, the overall load can be significant.**

Segmentation by events (profile history)

When segmenting by events — email opens, clicks, goals reached, link clicks — the platform scans the a

Always specify a date range. Without a date limit, the system will check the entire profile history, which

The larger the period, the longer the segment takes to calculate:

- **Segment "opened email in the last 7 days" — calculates quickly.**
- **Segment "opened email in the last year" — takes significantly more time.**
- **Segment "opened email" without a date limit — scans the entire history, which can take a very long t**

Recommendations:

- **Always limit the date range. Use relative periods whenever possible ("last N days").**
- **For recurring segments, a range of 30–90 days is usually sufficient.**
- **Instead of checking events for all time, use profile attributes that do not require scanning history. Fo of "had at least one order".**

Segmentation by profile data

Indexes and custom fields

Indexes are internal database structures that speed up searching for profiles by field values. Without an through all profiles sequentially — the segment builds slowly, especially on large databases.

Some indexes are created automatically when the platform starts. These are system indexes necessary cannot manage them.

For fast segmentation by custom fields, you must create an index on that field in the database. On large can increase segment calculation time several times.

> CAUTION
>
> **No more than 64 indexes can be created per account. For more details on database indexes, see the ad**

Recommendations:

- **Analyze which fields you most often build segments by.**
- **Pass the list of these fields to the administrator for index creation.**
- **Take into account the 64-index limit and prioritize the most important fields.**

Composite indexes for large databases

For databases containing one million profiles or more, segmentation performance depends even more on presence of indexes.

MongoDB supports composite indexes, which speed up segmentation by multiple fields simultaneously. I segments by the combination "city + customer status", a composite index on these two fields will signifi

> CAUTION
>
> **The number of indexes on a single profile database is limited by MongoDB limits. Before creating compo which field combinations are used most often and prioritize them. Each additional index increases the t the database.**

Storing lists of values

If you need to select profiles by a specific value from a list (e.g., interests, categories, tags), it is better t type rather than in a string.

For example, the `interests` field contains `tennis, football, chess`. If you need to select everyone interested will take longer when stored in a string than when stored in a "tags" field.

Segmentation by external data

Segmentation by external sources (SQL queries, APIs, files) has its own performance characteristics. De separate article:

Recommendations for segmentation by external data

Key points:

- **Design queries so that they return the minimum amount of data.**
- **Combine conditions from one external database into a single SQL query.**
- **Conditions with "NOT" on external data lead to scanning the entire profile database — minimize thei**
- **The platform can automatically combine queries to one external database into a single SQL query (v but only if the queries use the same connector and are bound to the same profile field.**

The "NOT" operator

Too many "NOT" conditions

Excessive use of negations, especially the NOT IN SEGMENT operator, complicates logic and reduces pe

The "NOT IN SEGMENT" operator.

This is the most performance-heavy operator. When used, the system dumps the entire profile database t segment. This is very slow.

Example of a problematic construct:

```
NOT IN (segment A)
AND NOT IN (segment B)
AND NOT IN (segment C)
...
```

Recommendations:

- **Avoid the "NOT IN SEGMENT" operator whenever possible. Instead, duplicate the conditions from the sign.**

For example, you need to select profiles that are not in the segment "Not in GCG". Instead of `NOT IN (segm in GCG`. The result will be the same, and performance will be higher.

- **Do not use more than 2–3 conditions with "NOT" (of any kind) in a row without a good reason.**
- **Replace a set of "NOT" conditions with a single positive rule. Instead of listing where an object shou be.**
- **Document segmentation rules: state why each negation is added. This helps avoid situations where the logic remains unnecessarily complex.**
- **Use "NOT IN SEGMENT" only when it is truly unavoidable (e.g., the excluded segment is dynamically**

Using "NOT" with "OR"

When building a segment, a condition like the following is often set:

```
NOT (has active contract) OR NOT (verified)
```

At first glance, it may seem that this condition excludes all customers without a contract and without ve differently: the segment will include all customers except those who simultaneously have both an active

This means that the condition excludes only those for whom both conditions are true at the same time, r them.

Recommendations:

- **Avoid long expressions that alternate NOT and OR.**
- **If you need to select only customers who have neither a contract nor verification, use:**

```
NOT (has active contract) AND NOT (verified)
```

Segmentation Conditions

This section contains conditions for segmenting profiles:

Segmentation by Profile data

Profile data is the main information about a subscriber that can be used for precise audience segmentati everything: from name and email to system labels and subscription statuses.

Main Fields

Main fields include standard profile data such as Email, Phone, First Name, Last Name, Date of Birth, and different types, and each type has its own comparison operators.

Operators

For most fields, the following operators are available:

- **Is equal to / Is not equal to — exact match of the subscriber's field to the segmentation rule. The val selected from a list.**
- **Exists / Does not exist — checks whether the field is active for a specific profile. An existing field ca**
- **Empty / Not empty — checks whether an existing field contains any value.**
- **Is in file in column / Is not in file in column — the rule looks for a match of the field in a column of a t attached to the rule.**
- **Is in data table / Is not in data table — the rule looks for a match of the field in a query to an externa**
- **Is in a field of post url response / Is not in a field of post url response — the rule looks for a match of method request to an external API.**
- **Is in list / Is not in list — the rule looks for a match of the field in a list that can be copied into the da separated by a line break (pipe).**

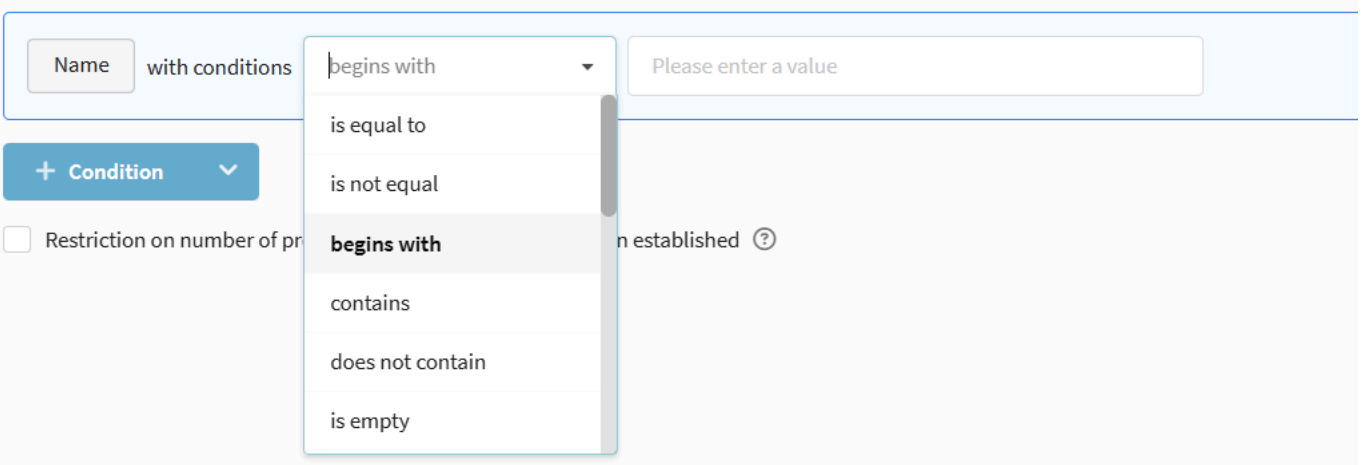
Field Types and Their Operators

String Fields

String fields store any values: letters, numbers, special characters. These include: `First Name`, `Last Name`, `Code`, and `Provider`.

For string fields, an additional set of operators is available:

- **Starts with — search based on the beginning of the field value.**
- **Contains / Does not contain — search based on a part of the field value (case-insensitive). The "Does lot of system resources.**

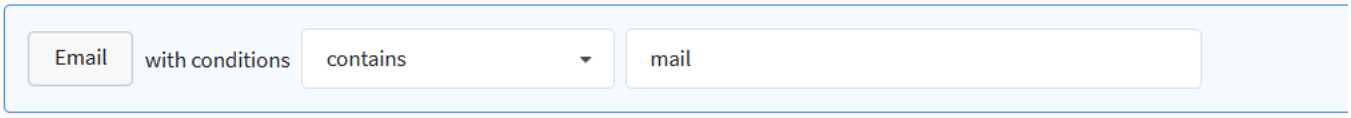


> **NOTE**
>
> **The standard Phones field has the same set of segmentation rules as string fields. However, this field c format +71234567890.**

Email Type Fields

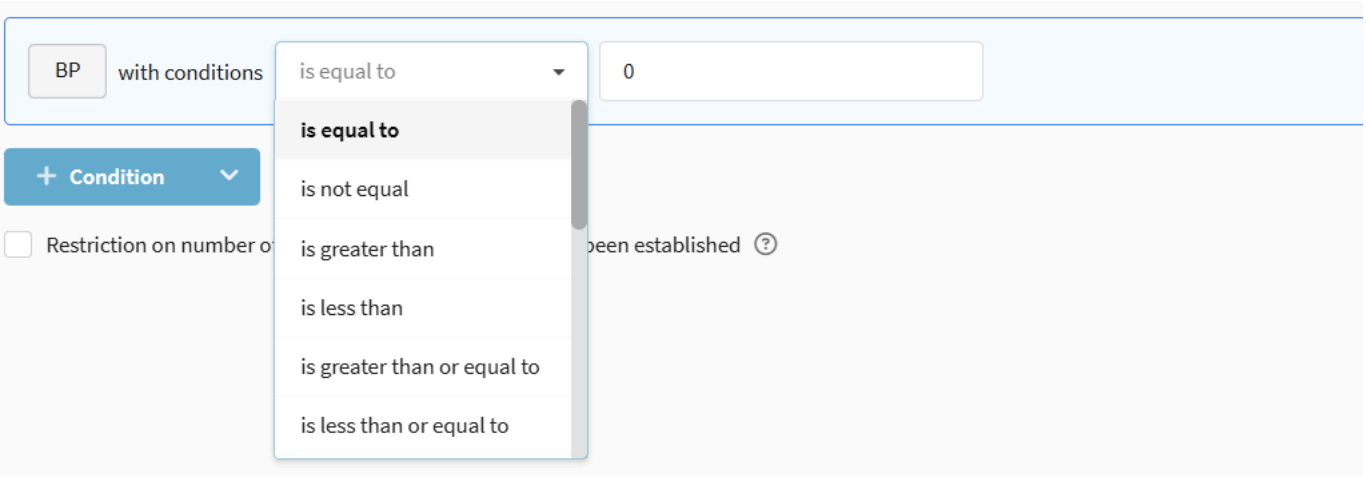
The Email field is used to identify a profile and has a special set of operators related to domains and don

- **Domain group is / Domain group is not — search for email addresses of a selected domain group (ma**
- **Domain is / Domain is not / Domain contains — search by the domain of subscribers' email addresses organizational domains).**



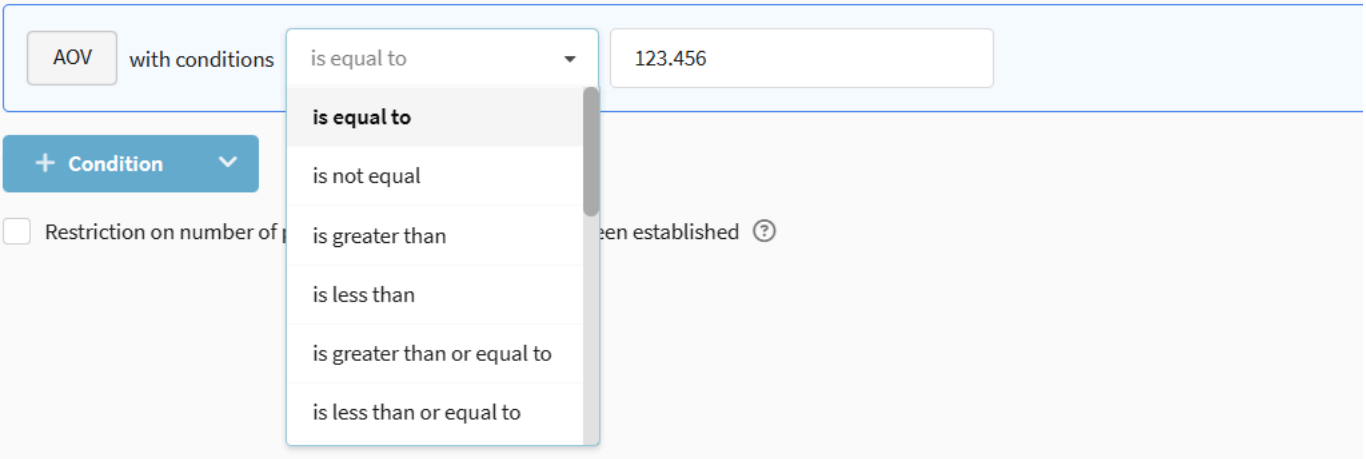
Integer Fields

Integer type fields can contain positive and negative whole numbers. Available operators: Is equal to / Is than.



Floating Point Fields

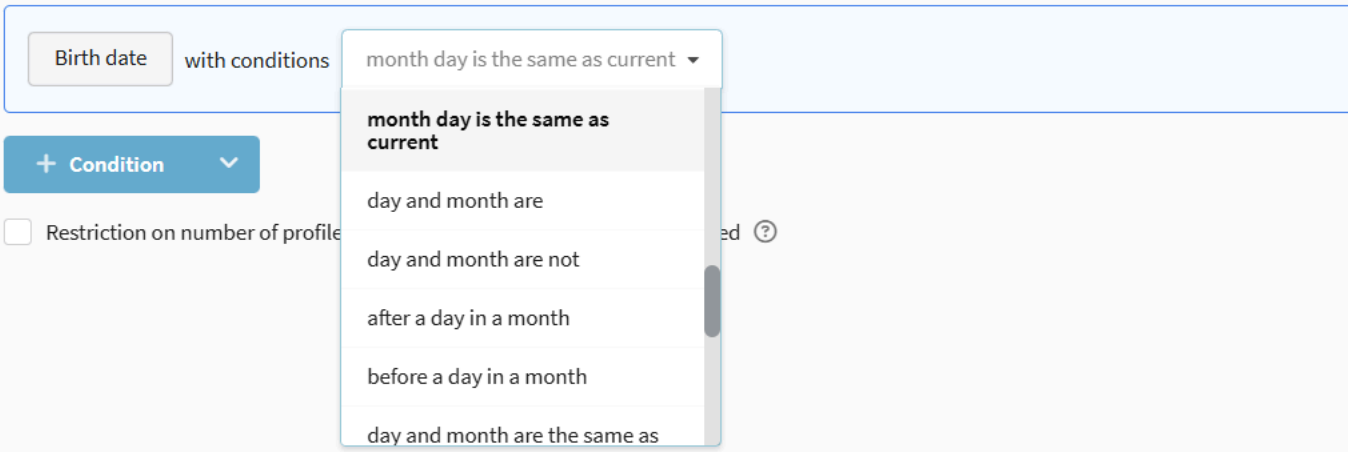
Floating-point fields can contain positive and negative decimal values. Available operators: Is equal to / I than.



Date

Date type fields store dates in RFC 3339 format (e.g., 2018-02-22T21:00:00Z). These include: `Registration`
Date fields have an extended set of segmentation operators:

- **Greater than / Less than — selection by date after or before the specified one.**
- **Is in a date range — selection by date falling within the specified period.**
- **Date/age relative to current date — selection by a moment in the past or future.**
- **Month is / Month is not / Month is the same as current — selection by month.**
- **Month day is / is not / after month day / before month day / month day is the same as current — selec**
- **Day and month are / are not / after / before a day in the month / day and month are the same as curre (without year).**



TIP

Dynamic segmentation by a date field is excellent for trigger reminders (e.g., about the end of a service [regular mailing] based on such a segment.

Date/age relative to current date

This operator allows selecting profiles by a date counted from the current moment, with an optional age

Parameters:

- **Specified period — a range in days relative to the current date. Negative values mean days in the pas future, 0 means today.**
- **Specify age — a range "from X to Y years", both boundaries are required. The left boundary is inclusi**

Profiles with an empty date field never enter the segment.

How it works

The system checks whether the profile's age fell within the specified range on at least one day of the giv years from the date in the field to the checked day.

NOTE

If the date field includes a time, it is taken into account when calculating age. Therefore, on a person's that they have not yet reached the specified age (N-1 instead of N).

Examples

Period [0; 0] (today only), age 20–30.

Profile	Age today
turned 20 yesterday	20
turned 20 today	20 (or 19)
turns 20 tomorrow	19
turned 30 yesterday	30
turned 30 today	30 (or 29)
turns 30 tomorrow	29

Explanation:

- **Turned 20 yesterday — age is 20 today, falls within the 20–30 range.**
- **Turns 20 tomorrow — still 19 today, does not match.**
- **Turned 30 today — depending on the time stored in the date field, age may be 29 or 30. If 29 — matcl**

Period [-30; 0], age 20–30. The profile matches if its age was in the range on at least one day during the l

Profile	Result	Reason
turned 20 yesterday	✓	age 20 throughout the period
turns 20 tomorrow	✗	age 19 throughout the period
turned 30 yesterday	✓	was still 29 at the beginning of the period
turns 30 tomorrow	✓	age 29 throughout the period

Period [0; +∞], age 20–30. All profiles who will ever be 20–29 years old in the future match.

Profile	Result	Reason
15 years old	✓	will turn 20 in 5 years
25 years old	✓	age already in range
35 years old	✗	age will never be in range

Inverted condition

The "Does not fall in period" operator gives the opposite result: a profile matches if it would not match u

Example

Condition: date of birth does not fall in period [0; 0], age from 20 to 30.

Profile	Result
15 years old	matches
25 years old	does not match
35 years old	matches

Explanation: age 25 falls within the 20–30 range, so the profile is selected under the direct condition and and 35 are outside the range, so the result is reversed.

Ignoring the year

When this option is activated, only the day and month are compared. Useful for selecting by recurring da

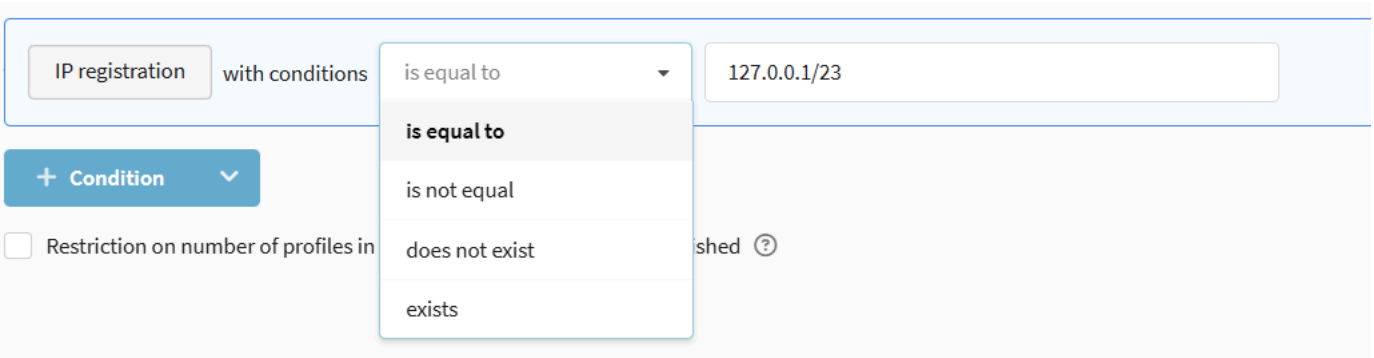
NOTE

Ignoring the year and specifying age are mutually exclusive options.

- **If the lower boundary is not set (-∞), all profiles match — any day.month has fallen into the period at s**
- **If the upper boundary is not set (+∞), all profiles match — any day.month will fall into the period at so**
- **Profiles with an empty date field do not match.**

IP Address

IP Address type fields store addresses in IPv4 format. These include `IP Address` and `Registration IP Address` operators are available: Is equal to / Is not equal to, Exists / Does not exist.



NOTE

For IP, network mask search is implemented:

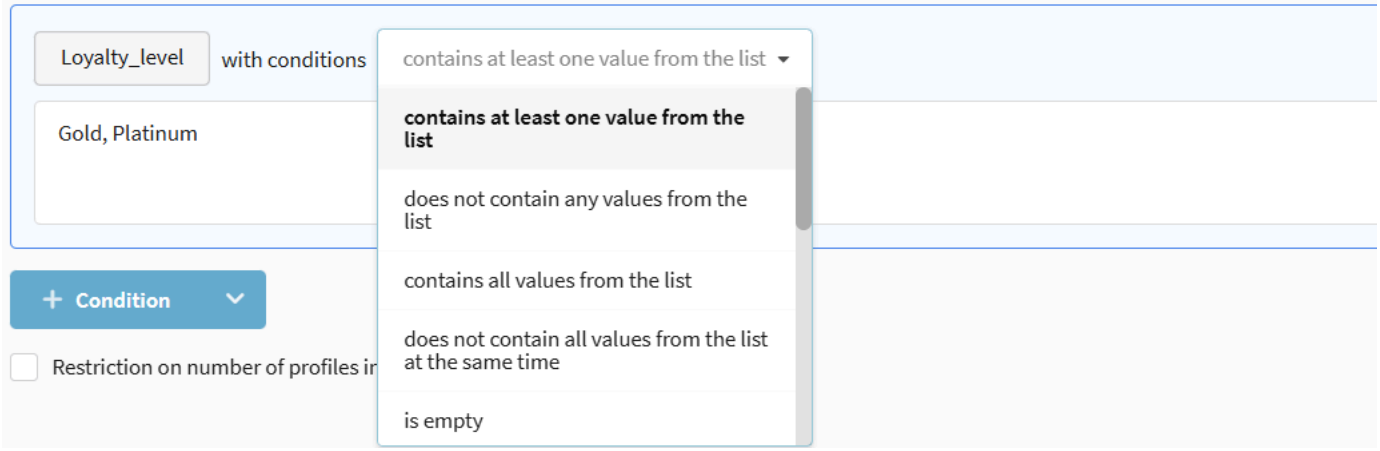
- **Is equal to — find subscribers whose IP addresses are within the specified subnet (e.g., `127.0.0.1/25`**
- **Is not equal to — find subscribers whose IP addresses are not within the specified subnet. To searc the AND condition.**

Tags

The Tags field is a list separated by commas. These can be, for example, purchased products.

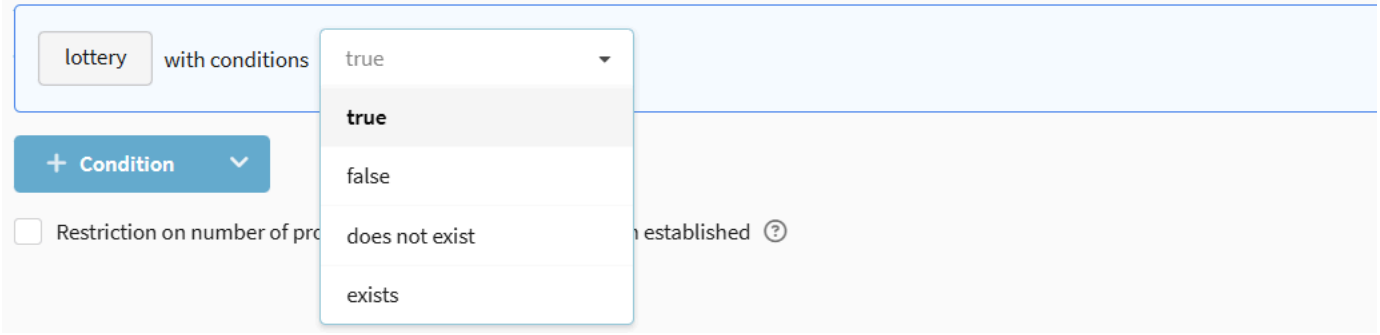
Tag segmentation operators:

- **Contains one of — the field contains at least one of the values from the rule.**
- **Contains none of — the field does not contain any of the values from the rule.**
- **Contains all of — the field contains all values from the rule.**
- **Contains not all of — the field does not contain some of the values from the rule.**



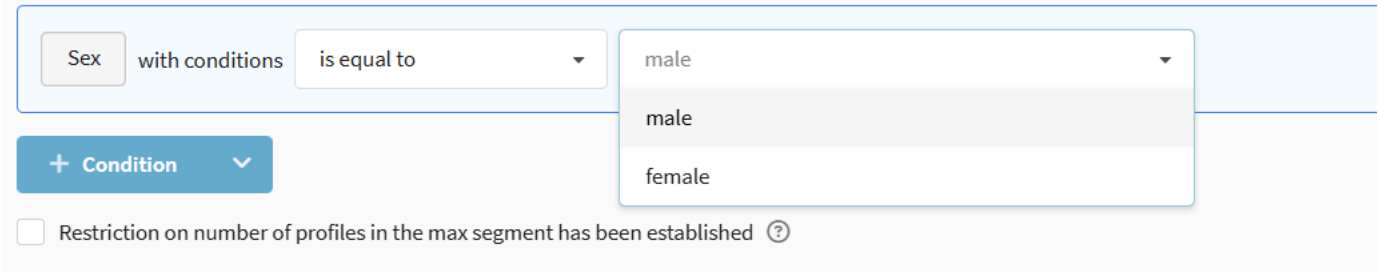
Boolean Field

A field containing logical YES/NO (true/false). Convenient for statuses or presence of characteristics. Op false.



List (Enumeration)

A field containing one of predefined values. Among standard fields, the Sex field has this type.
Operators can only compare the value with one selected from the list or check for the existence of the fi

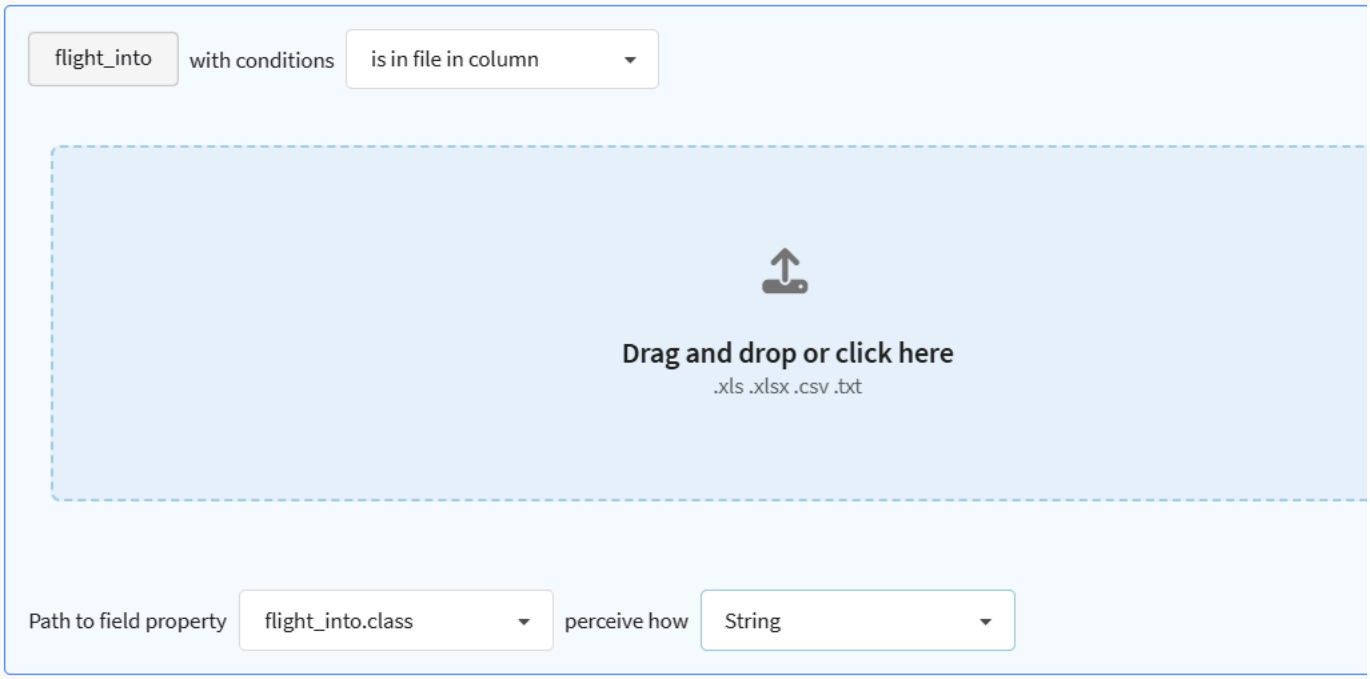


Custom (Additional) Fields

You can create your own fields in the database structure to store specific information. The same operato for main fields, depending on the field type.

Object

The field contains an arbitrary object for storing complex data structures.
Available conditions: Exists / Does not exist, as well as checks in lists and external sources.
If necessary, specify the path to a specific property of the field. The type of the property will be determi override it by changing the value in the “Perceive type as” field.

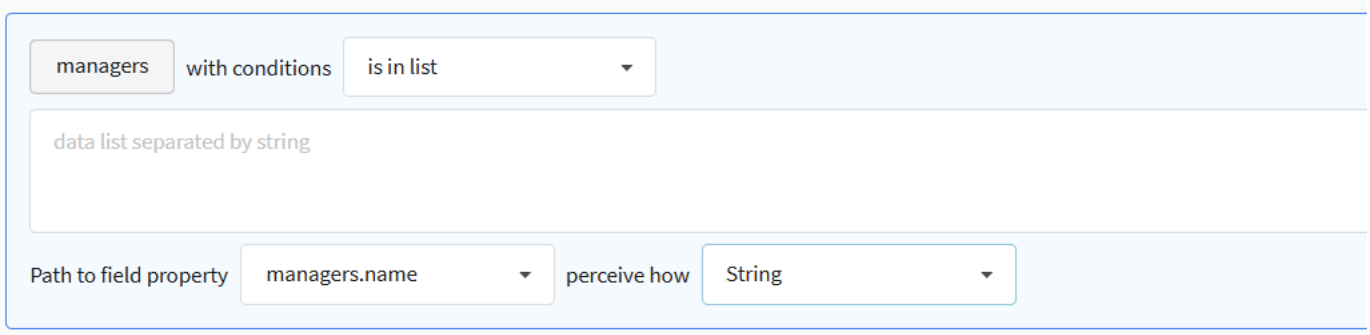


Array of Objects

The field contains an array of arbitrary objects for storing a sub-table (e.g., a list of contracts).

Available conditions: Exists / Does not exist, as well as checks in lists and external sources with the spe in", or "is not in".

You can also specify the path to the property and its type.



System Fields

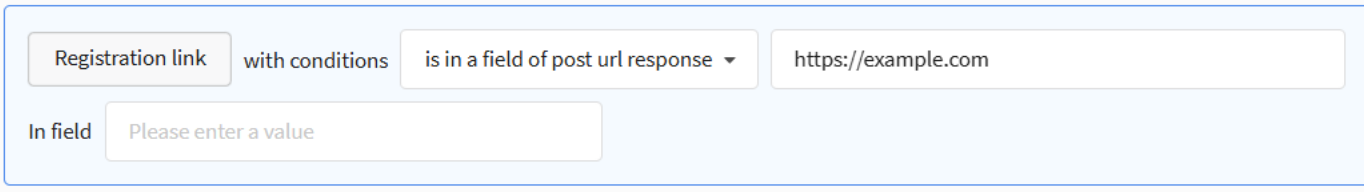
System data available for segmentation includes fields automatically filled by the platform.

Profile Status

Contains only the global profile status, which determines the possibility of communication with the clien dropdown menu.

Profile Identifier

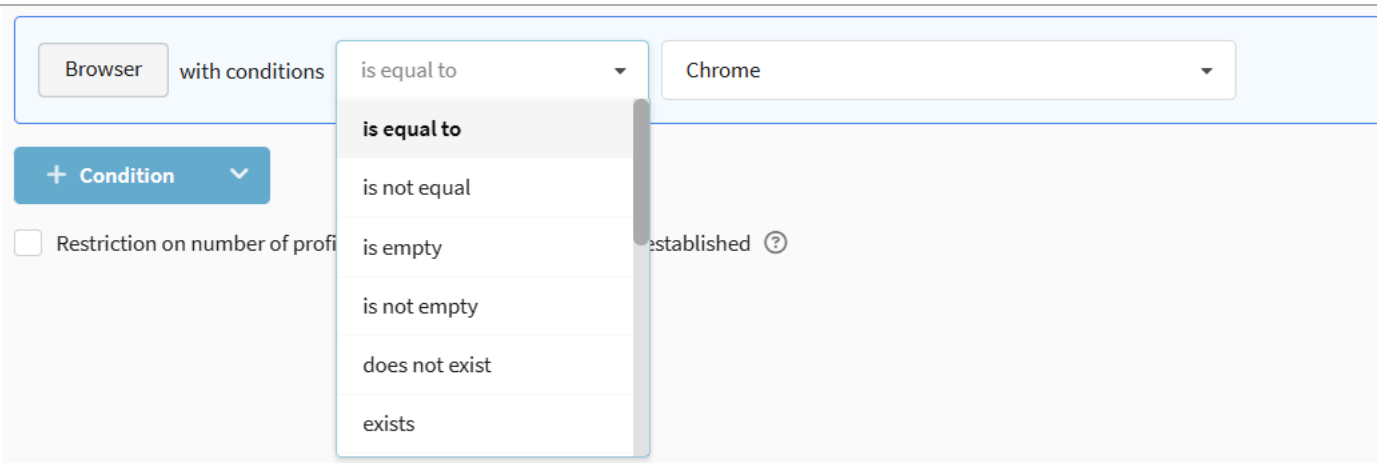
A unique identifier for each customer profile. Options available: Is equal to / Is not equal to, as well as ch identifier in lists and external sources (file, SQL, API).



Browser and Operating System

The Browser and Operating System fields are automatically filled when a client opens an email or perfor User agent header).

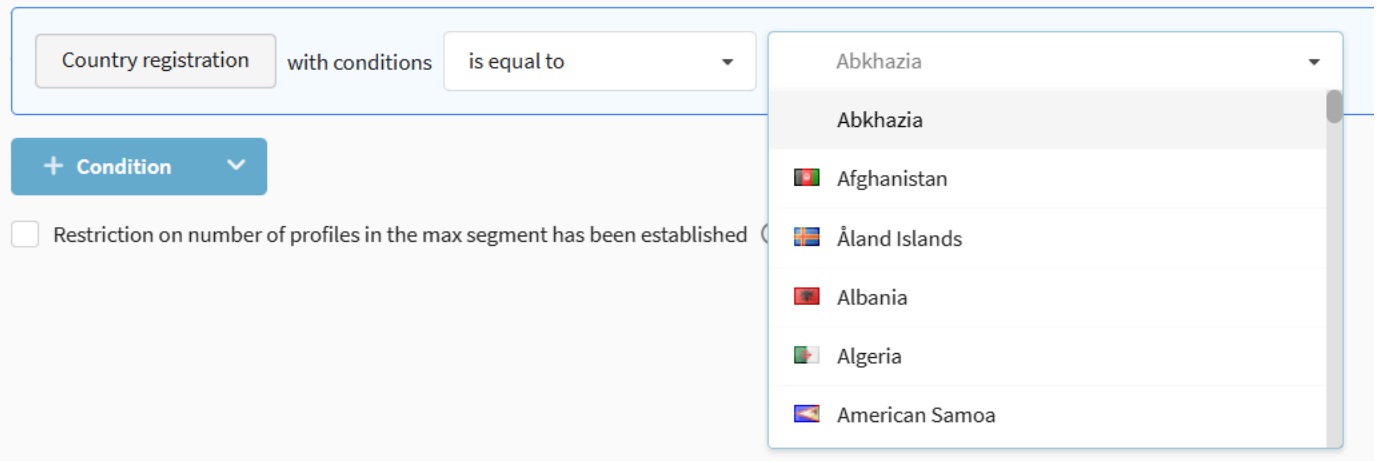
Options available: Is equal to / Is not equal to, Exists / Does not exist, Empty / Not empty, as well as chec



Geolocation Data

Location data (Country, Region, City, Zip Code, Time Zone) can be imported manually or determined auto moment of opening an email or clicking a link.

Use standard string field operators for segmentation by geolocation.



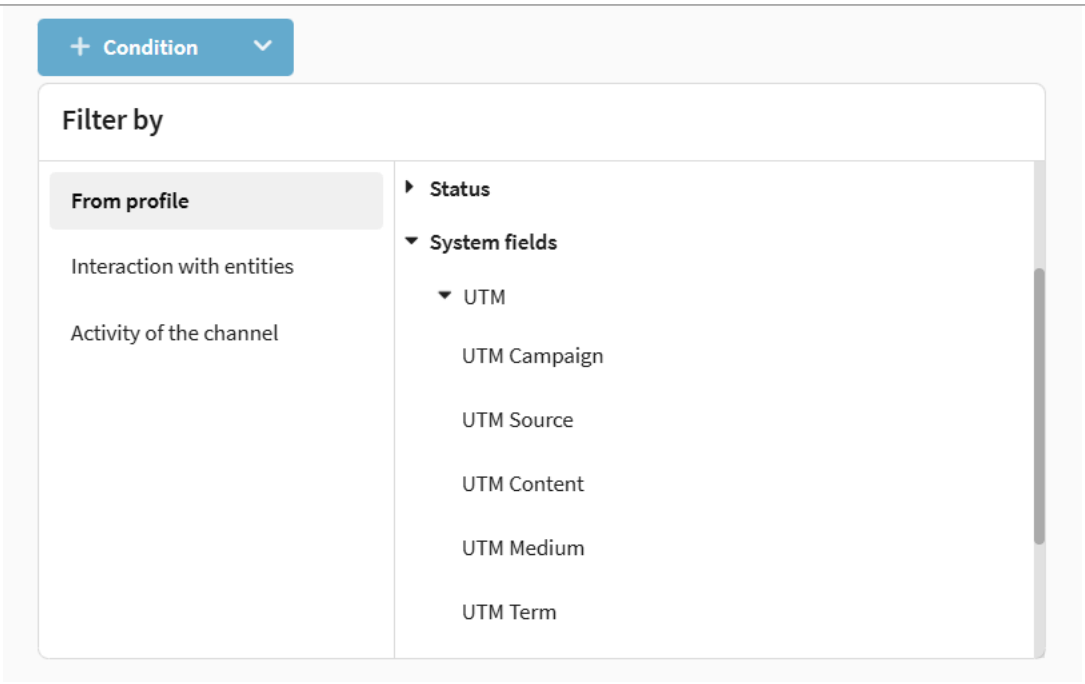
UTM Labels

UTM labels are used to track traffic sources. The following fields are available:

- **UTM mailing — the name of the mailing.**
- **UTM Source — the traffic source (advertising system).**
- **UTM Medium — the type of mailing (contextual, banner, etc.).**
- **UTM Content — additional information for differentiating ads.**
- **UTM Term — the keyword.**

For UTM fields, the following operators are available:

- **Is equal to / Is not equal to**
- **Starts with**
- **Contains / Does not contain**
- **Is empty / Is not empty**
- **Exists / Does not exist**
- **Is in file in column / Is not in file in column**
- **Is in data table / Is not in data table**
- **Is in a field of post url response / Is not in a field of post url response**
- **Is in list / Is not in list**



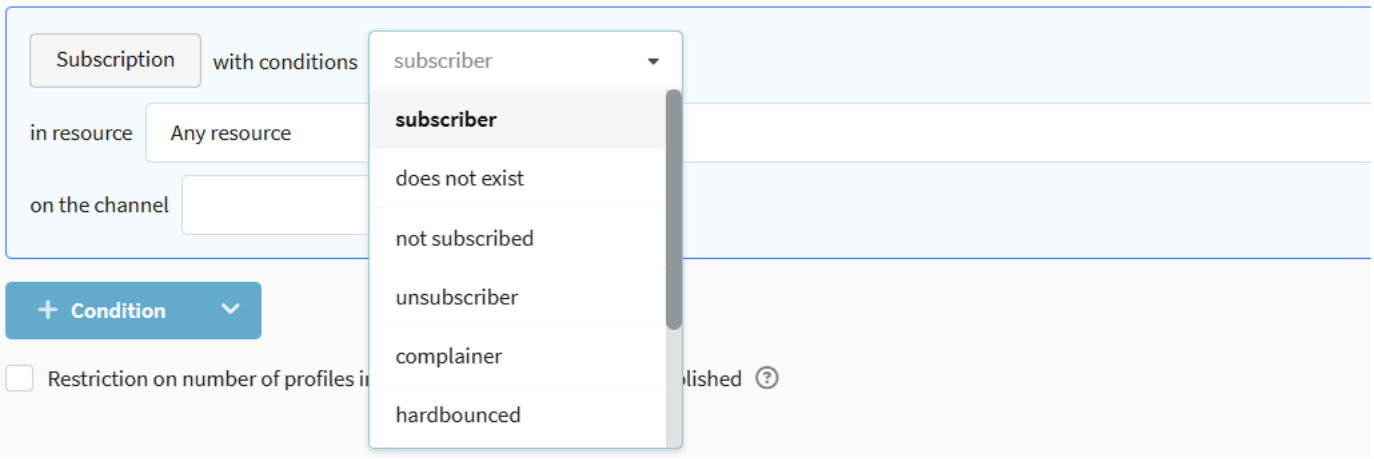
Subscriptions

Segmentation by subscription data allows you to determine the possibility of communication with clients channels and set up omnichannel mailings.

Subscription

Allows selecting profiles by the presence and status of a subscription to a specific resource and commu

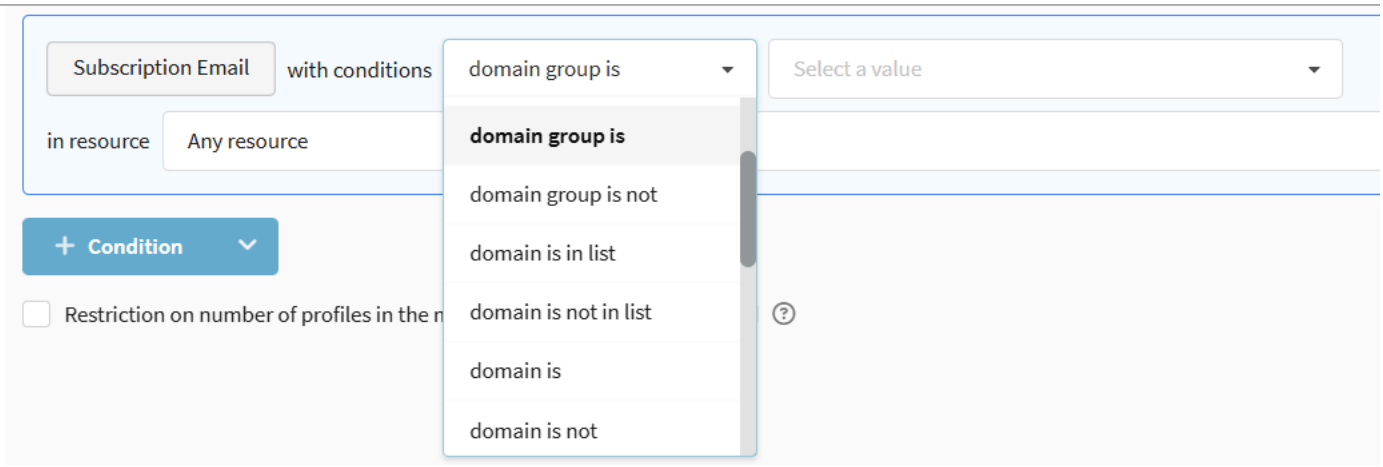
- **Does not exist — the profile does not have a subscription to the selected resource and channel.**
- **Subscribed — the profile has active subscriptions (“Subscribed” status).**
- **Not subscribed — the subscription exists, but its status does not allow sending messages.**
- **Unsubscribed, complainer, hard bounced, unconfirmed, suspended, not valid — selection by one of th**



Email Subscription

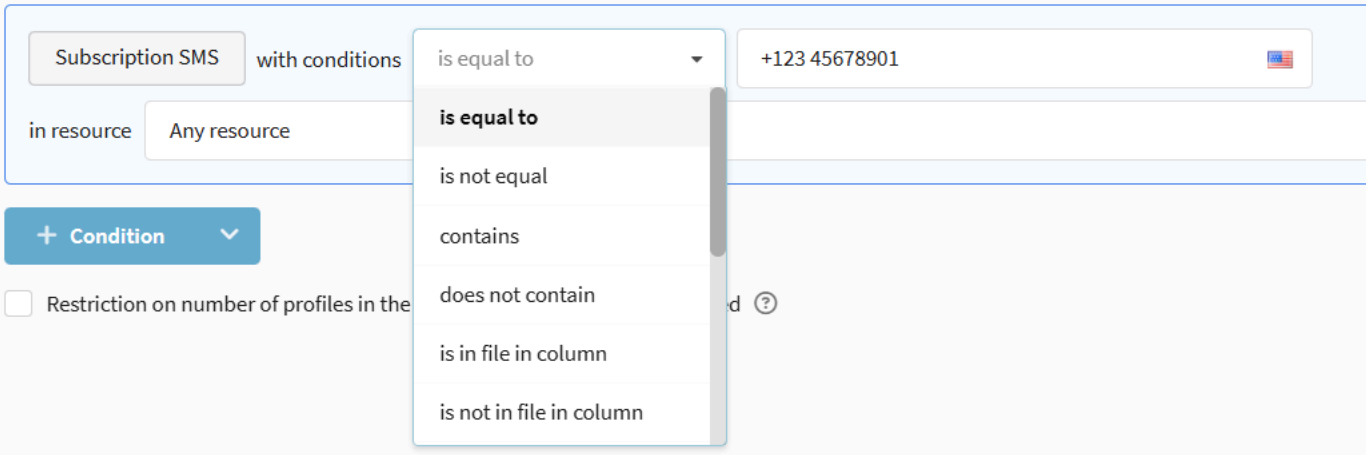
Allows isolating groups of subscribers by email provider or finding specific emails. The set of operators i

- **Is equal to / Is not equal to — exact match of the address.**
- **Contains — search by part of the email address.**
- **Domain group is / Domain group is not — search by domain group (mail.ru, yandex.ru).**
- **Domain is / Domain is not / Domain contains — search by domain or part of it.**
- **Standard operators for searching data in lists and external sources.**



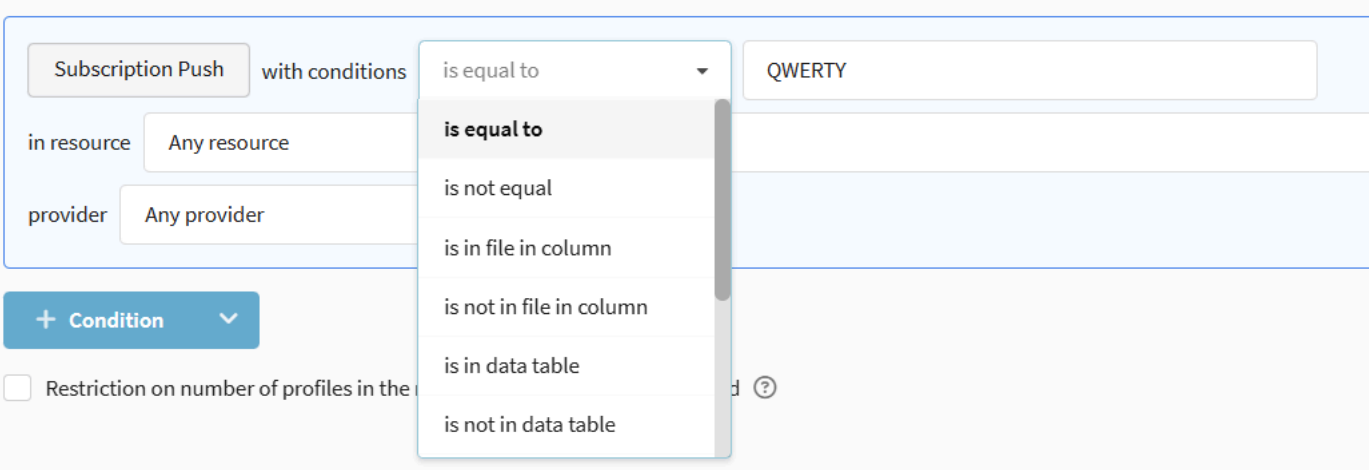
SMS Subscription

Segmentation by data contained in the SMS subscription. Selections by exact or partial match are availa Contains / Does not contain, as well as checks in lists and external sources.



Push Subscription

Segmentation by data contained in the Push subscription. Selections are available only by full match Is e operators for searching data in lists and external sources. Additionally, you can specify the push notifica



TIP

Detailed information about field types in the database and permissible values can be found in the separ

Segmentation by Interactions with Entities

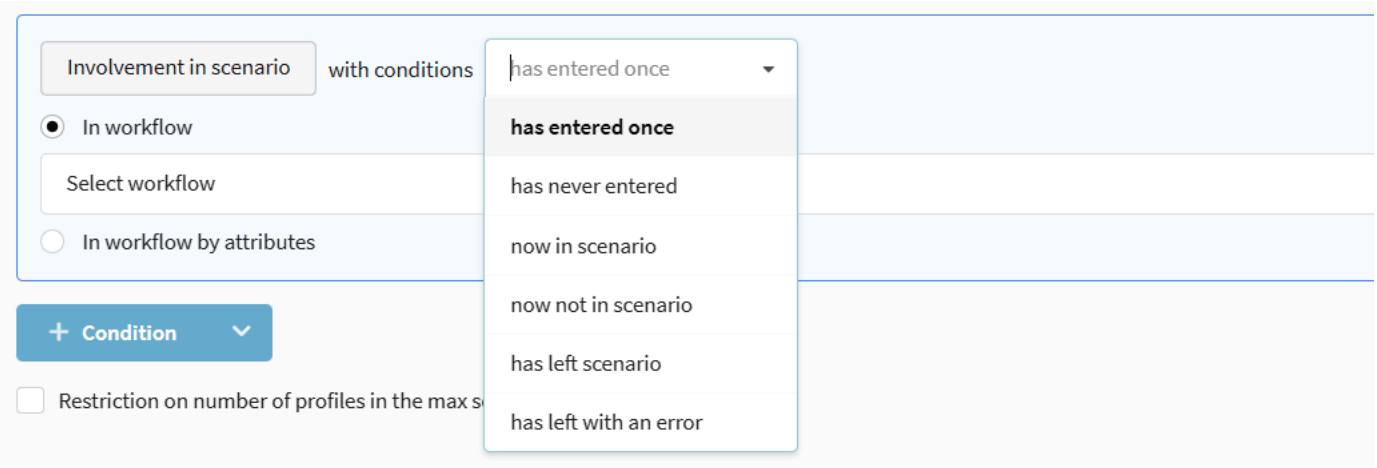
This section contains rules for segmenting profiles based on their interactions with various functional er as automation scenarios, forms, popups, segments, and loyalty programs.

Scenarios

Segmentation by participation in automation scenarios. Allows selecting profiles that are in a scenario, l error.

Available operators:

- **has entered once — the profile has ever participated in the selected scenario.**
- **has never entered — the profile has never participated in the selected scenario.**
- **now in scenario — the profile is currently active in the scenario.**
- **now not in scenario — the profile is currently not active in the scenario.**
- **has left scenario — the profile successfully completed the scenario.**
- **has left with an error — the profile exited the scenario due to an error.**



Segments

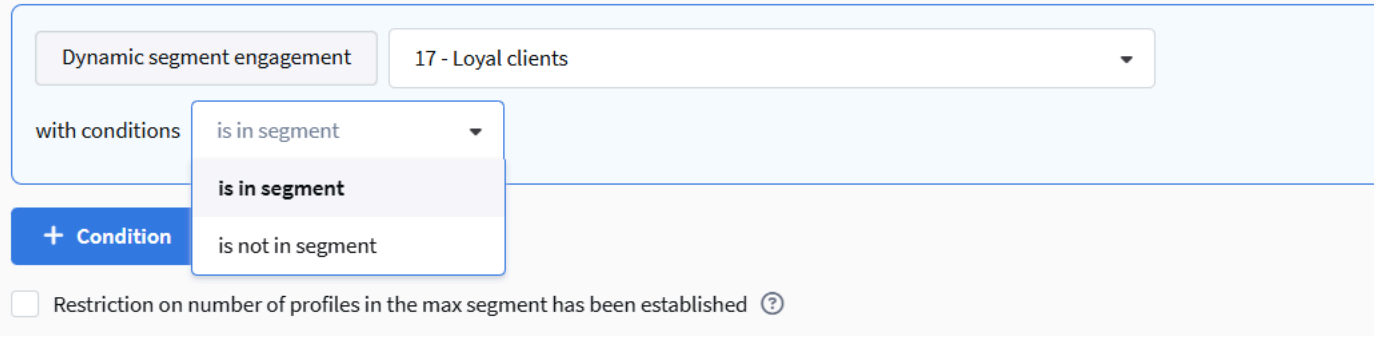
Selection of profiles based on their participation in other segments (dynamic or static). Allows creating

Participation in a Dynamic Segment

The rule checks whether a profile is included in the calculation result of another dynamic segment.

Available operators:

- **Is in segment — the profile is included in the selected dynamic segment.**
- **Is not in segment — the profile is not included in the selected dynamic segment.**



CAUTION

Try to avoid the Is not in segment operator for large selections, as such a rule consumes significant sys arrays.

NOTE

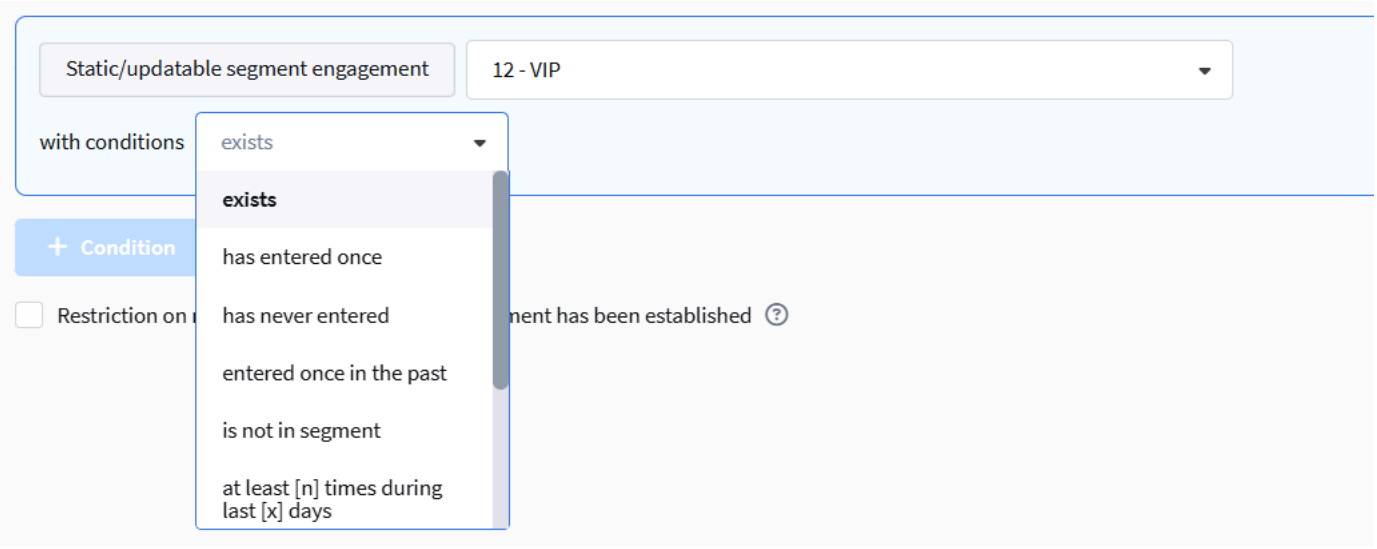
The dynamic segment for selection must be configured for the same database as the current segment.

Participation in a Static Segment

The rule checks whether a profile is in a static segment (control group) or has ever been in it.

Available operators:

- **exists — the profile is currently in the static segment.**
- **has entered once — the profile has ever been in this segment.**
- **has never entered — the profile has never been in this segment.**
- **entered once in the past — the profile is currently not in the segment but was in it previously.**
- **is not in segment — the profile is currently not included in the segment.**
- **entered at least [n] times during [x] days — the profile entered the segment the specified number of**
- **entered at least [n] times during period — the profile entered the segment the specified number of ti**
- **never entered during [x] days — the profile has not entered the segment in the last X days.**
- **never entered during period — the profile has not entered the segment in the selected period.**

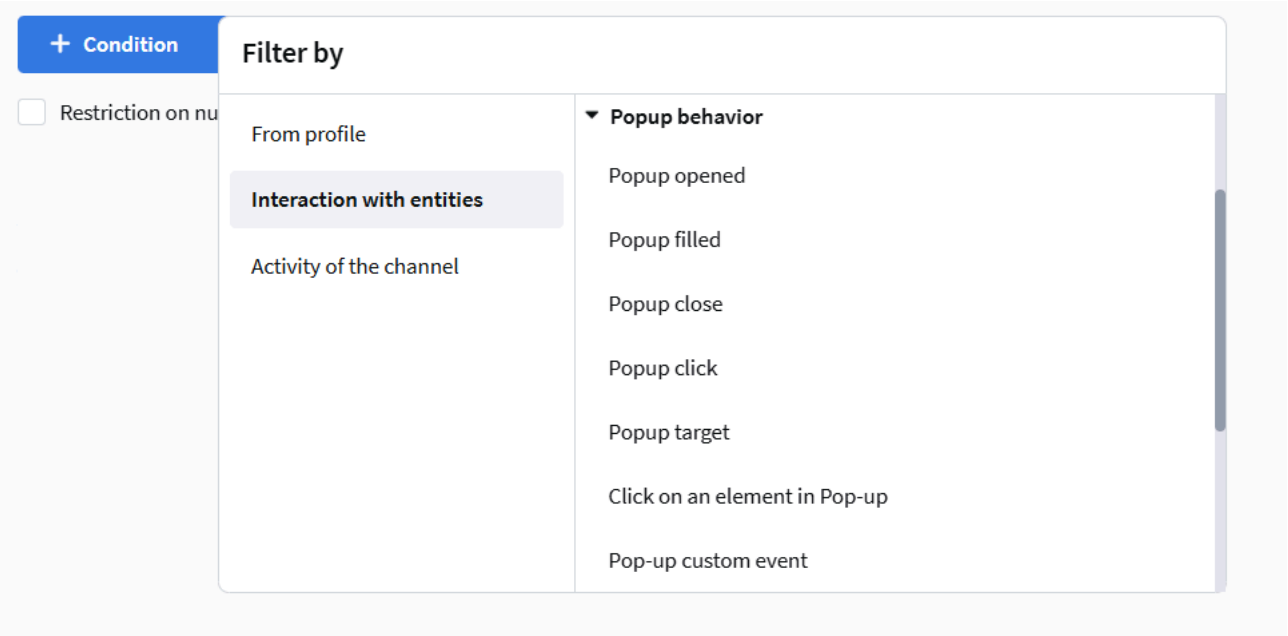


Popup Behavior

Segmentation based on user interaction with [popups] on your website.

Available events for selection:

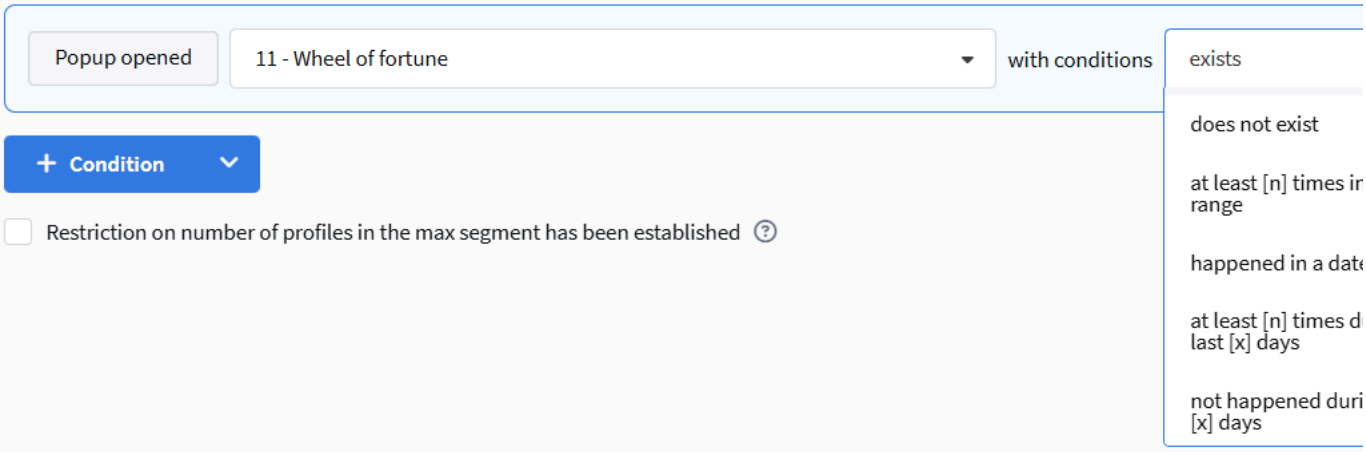
- **Pop-up opened — the popup was shown to the user.**
- **Pop-up filling — the form in the popup was successfully filled and submitted.**
- **Pop-up closed — the user closed the popup (click on the "Close" button).**
- **Click on pop-up — click on any element of the popup, except the close button.**
- **Pop-up target — the conditions for showing the popup were met (even if it wasn't shown due to limits**
- **Click on an element in Pop-up — click on a specific HTML element with a given `name` attribute.**
- **Pop-up custom event — an event configured via JavaScript code.**



Available selection conditions:

- **Exists — the event was registered at least once.**
- **Does not exist — the event was never registered.**

- **At least [n] times in a date range — the event occurred at least N times during the specified period.**
- **Not happened in a date range — the event did not occur during the specified period.**
- **At least [n] times during [x] days — the event occurred at least N times in the last X days.**
- **Not happened during last [x] days — the event did not occur in the last X days.**



Loyalty

Promo code

Segmentation by participation in loyalty programs using promo codes.

Available operators:

- **Assigned / not assigned — selection of subscribers to whom a promo code was assigned by a loyalty**
- **Activated / not activated — selection based on the fact of promo code activation (for example, when**

Loyalty Programs

Profile segmentation can use data based on actions in loyalty programs.

Available selection conditions:

- **Participation in a loyalty program — whether the profile is a participant in the selected loyalty progr**
- **Points balance on members' accounts — the number of points in the loyalty program.**
- **Points accrued in promotions.**
- **Points expiration in promotions.**

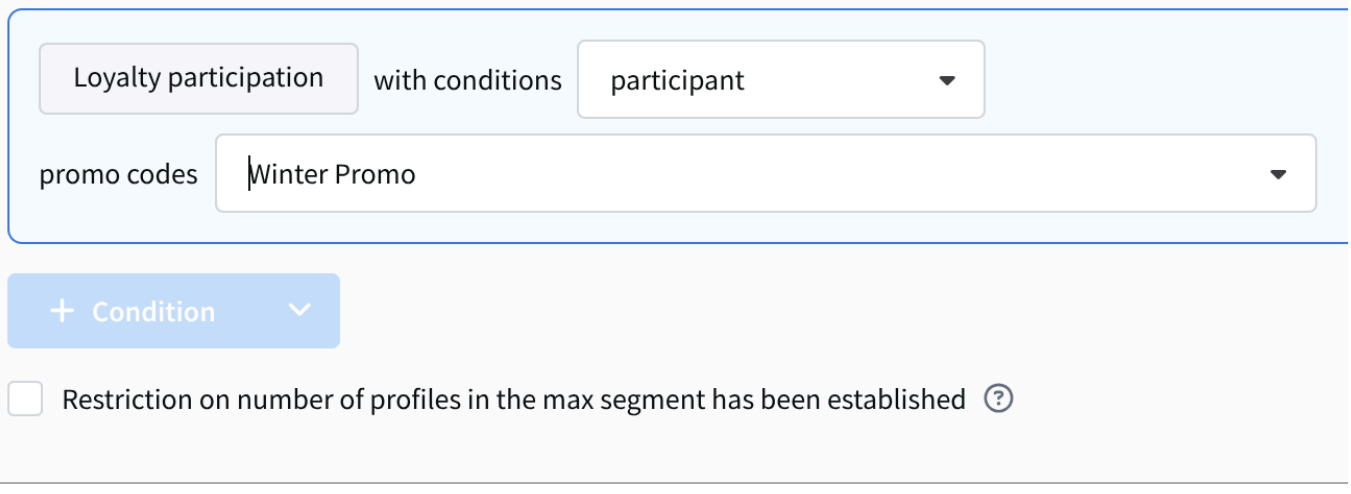
Available operators:

- **Member / Not a member.**
- **Points balance equal / not equal / is within points range.**
- **points amount from n to m within a period / points amount from n to m in the next x days.**
- **in the next x days / points amount from n to m in the next x days.**

To configure these conditions, you may need to select a loyalty program, a promotion, a balance type (ac

TIP: Final and active balance

Accrued points can have an activation time. If it is not specified, the points become active immediately points cannot be used until that moment. The active balance is the number of points that can be used r total points balance, including inactive points.



Relations

Segmentation by the presence of relations between subscriber profiles.

Available operators:

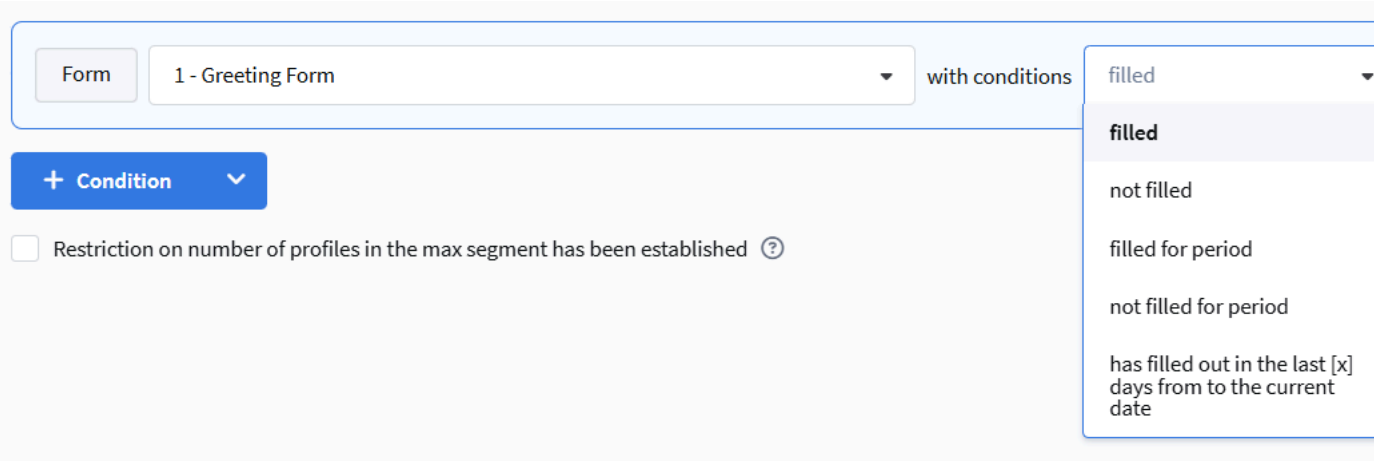
- **Is / Is not in direct relation — selection of subscribers who have or do not have a direct relation with**
- **Is / Is not in reverse relation — selection of subscribers who have or do not have a reverse relation fr**

Forms

Segmentation based on interaction with forms placed on the website.

Available selection conditions:

- **filled out — the subscriber has ever filled out the specified form.**
- **did not fill out — the subscriber has never filled out the specified form.**
- **filled out for the selected period — the subscriber filled out the form during the specified dates.**
- **did not fill out for the selected period — the subscriber did not fill out the form during the specified d**
- **filled out in the last [x] days from the current date — dynamic selection for the last X days.**



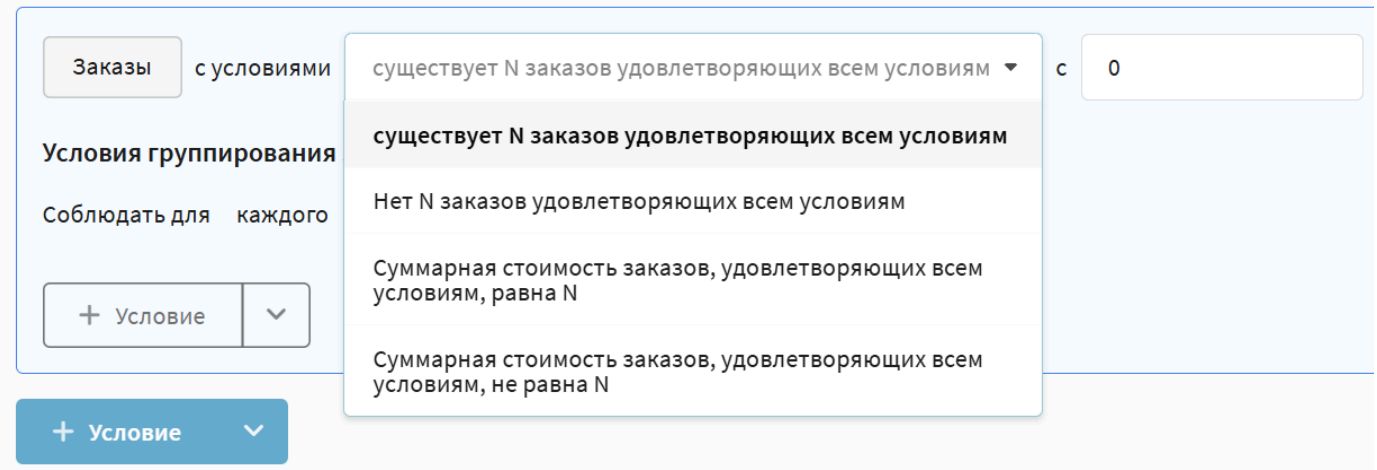
Orders

Segmentation based on orders registered in the Market module. Allows building selections by quantity, a composition, as well as by geography, sales channels, and statuses.

Basic Conditions

Rules of this type allow you to select profiles that have (or do not have) orders meeting the specified par available:

- **here are N orders that meet all the conditions — selection of profiles with the specified number of or conditions.**
- **No N orders that meet all the conditions — selection of profiles that do not have the specified numbe**
- **Total cost of orders that meet all the conditions equals N — selection by total order amount taking in**
- **Total cost of orders that meet all the conditions does not equal N — exclusion by total amount.**
- **Total cost of orders that meet all the conditions in range — selection of profiles whose total order ar numeric range.**
- **Total cost of orders that meet all the conditions is not in range — exclusion by total amount range.**



Purchase Analytics Conditions

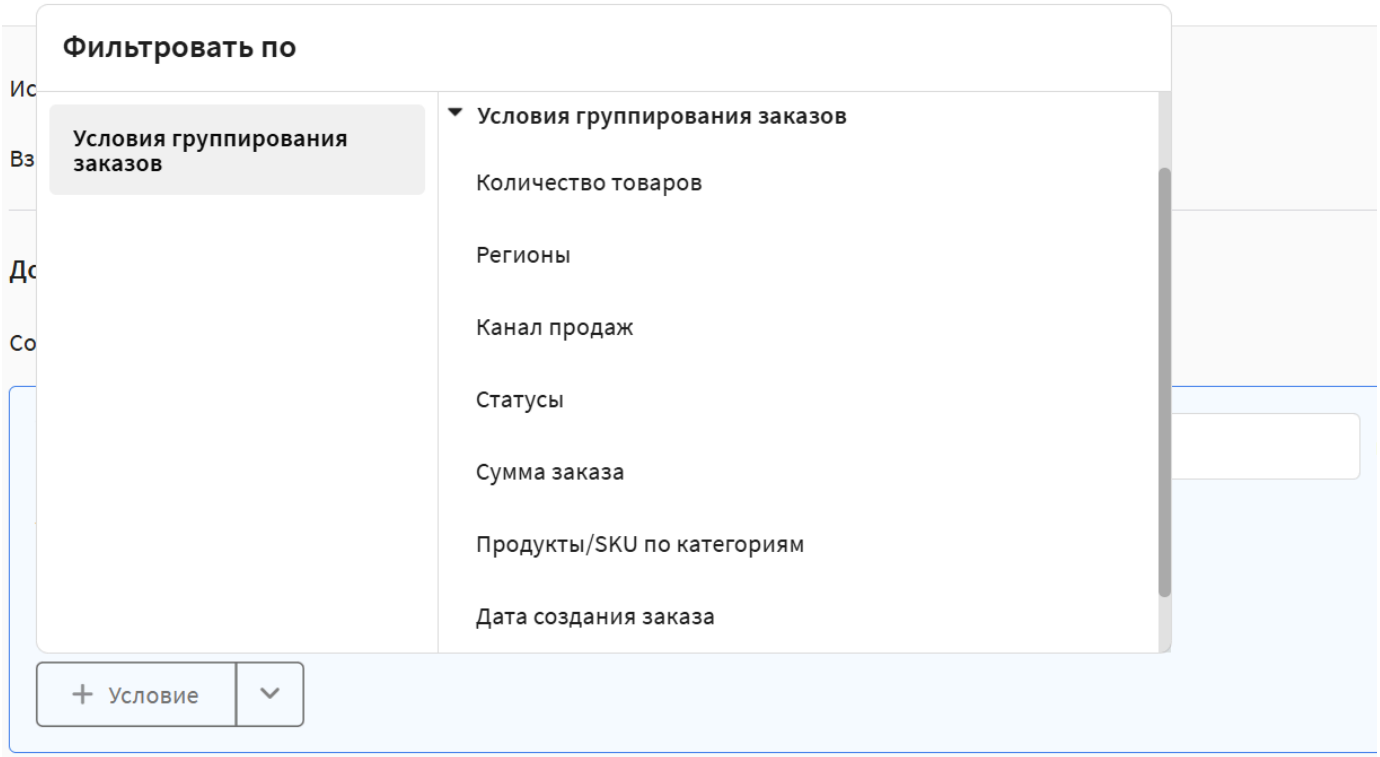
For deeper analysis of purchasing behavior, rules based on average values are available. These conditio (or only those that meet additional parameters) over a certain period of time.

Available operators include checks for a period (specific dates) and for the last X days (relative to the cu intervals, you can set an “in range” or “not in range” condition:

- **Average check — the average cost of one order (sum of all orders / number of orders)**
- **Average check for period in range / not in range**
- **Average check for last X days in range / not in range**
- **Average sum of purchases — the average cost of one item in orders (sum of all orders / total number**
- **Average check from segment A for period in range / not in range — the average cost of an order cont product segment (sum of orders containing products from segment A / number of such orders)**
- **Average check for orders from segment A for last X days in range / not in range**
- **Average sum of purchases for orders from segment A for period in range / not in range — the average specified segment (sum of cost of all items of products from segment A / number of these items)**
- **Average sum of purchases for orders from segment A for last X days in range / not in range**

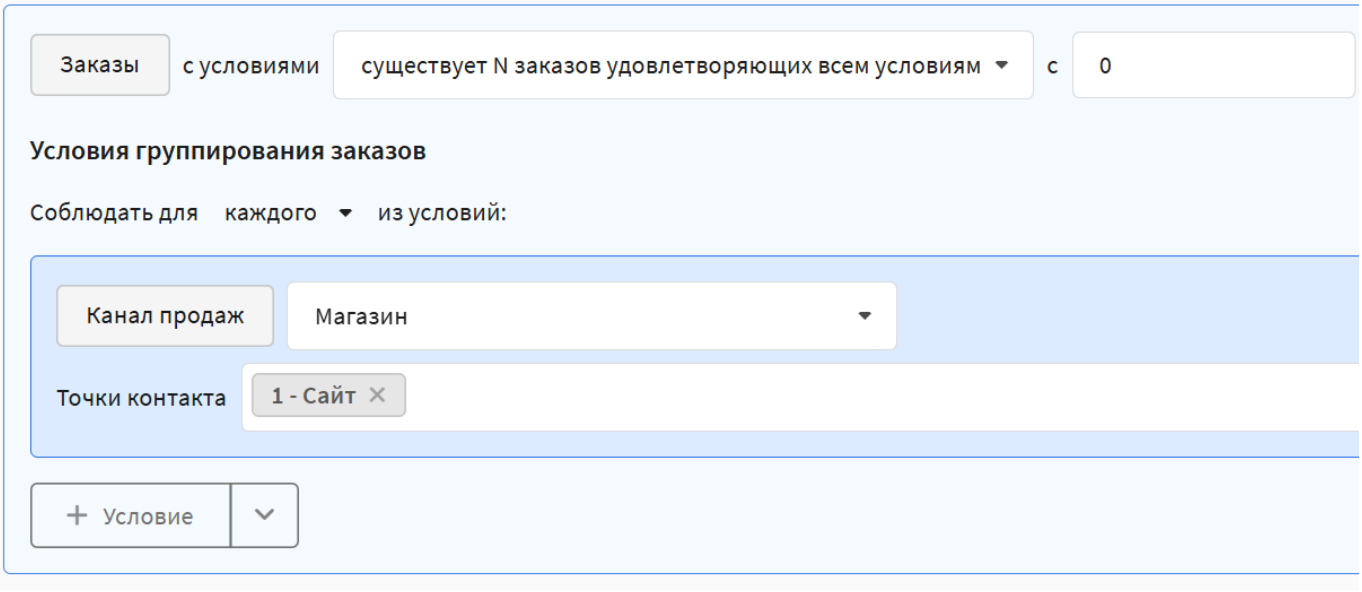
Order Filtering Terms

In any of the rules described above, you can set additional conditions to specify which orders to take int are available:



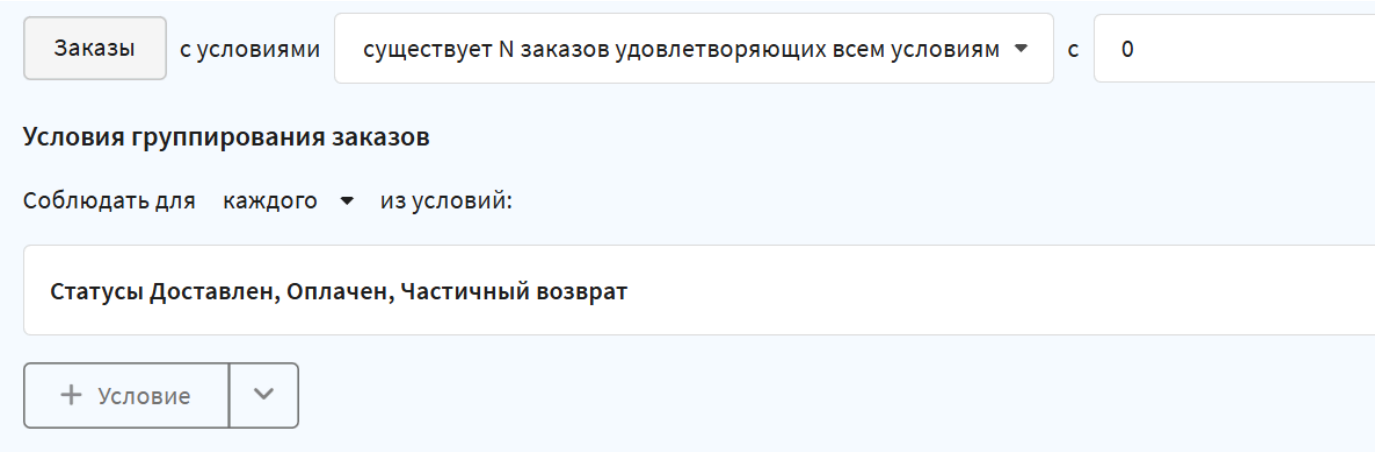
Sales Channel

Selects orders made in one sales channel. Multiple contact points can be specified.



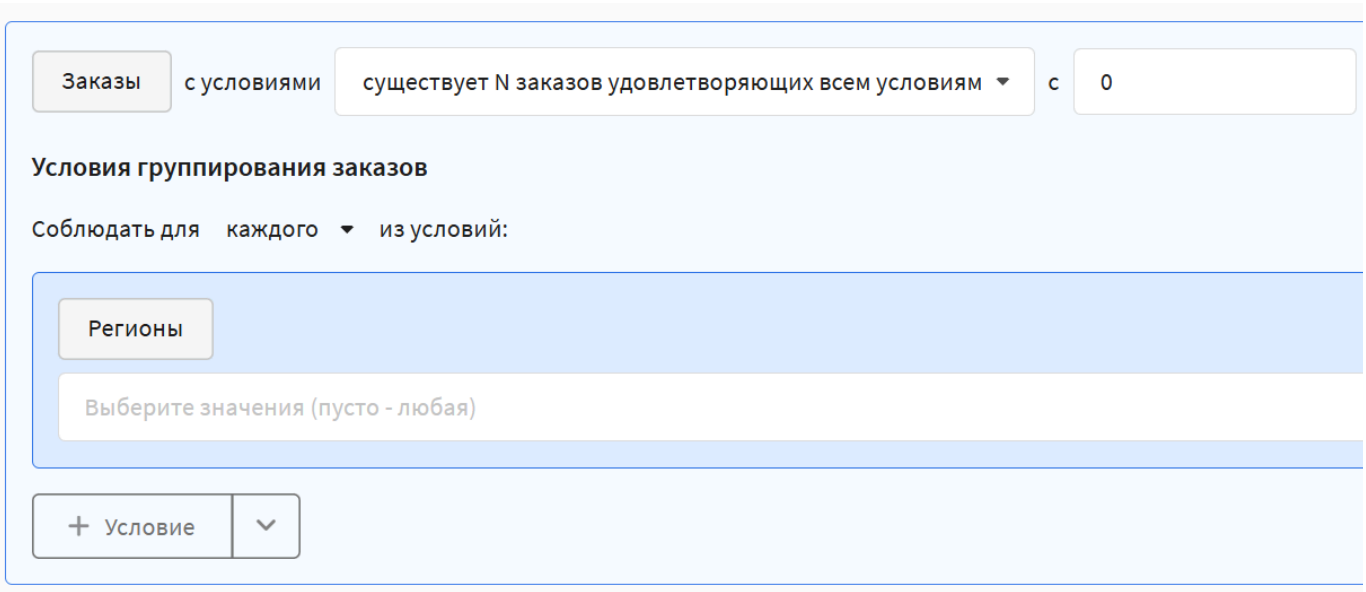
Statuses

Selects orders with the specified statuses: *new, delivered, paid, canceled, partial return*.



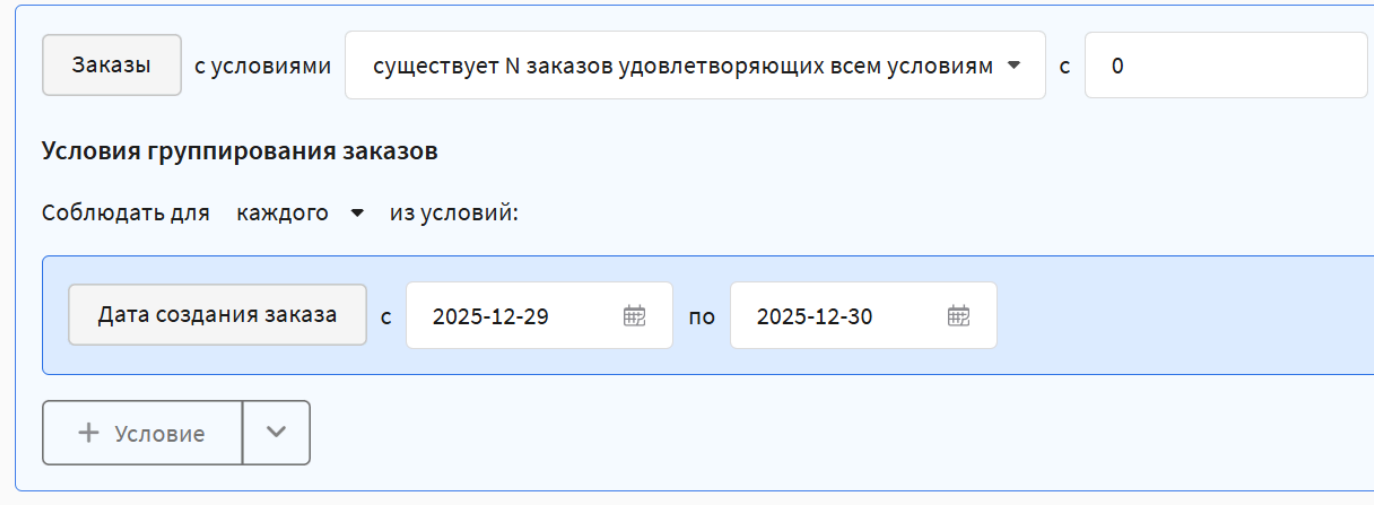
Regions

Selects orders made in specific regions.



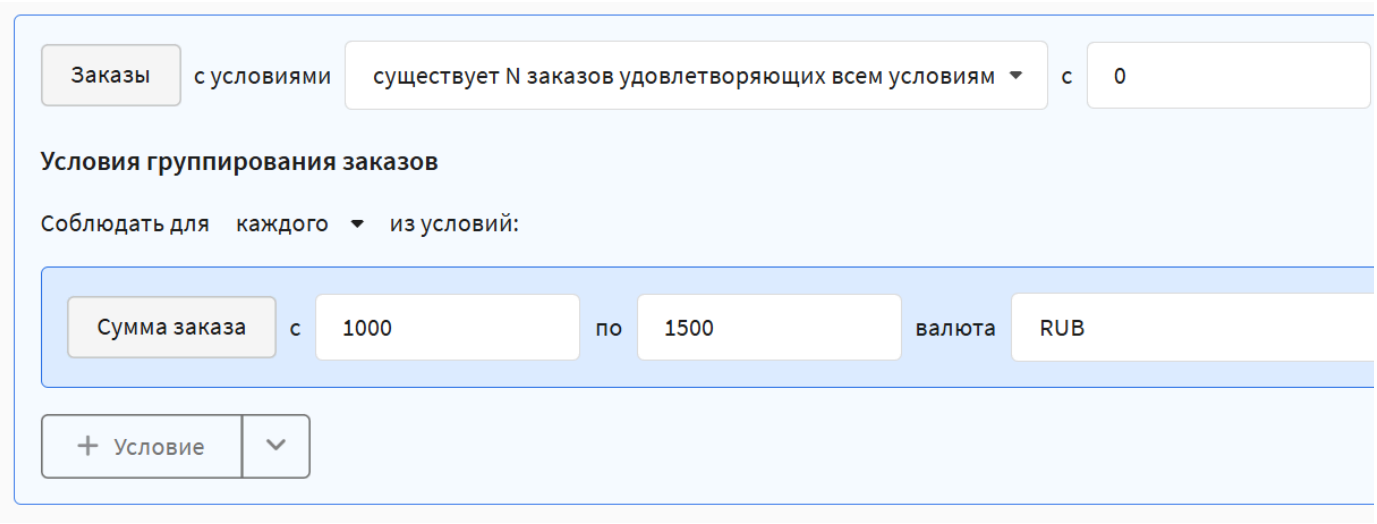
Date Created

Selects orders registered during the specified time period.



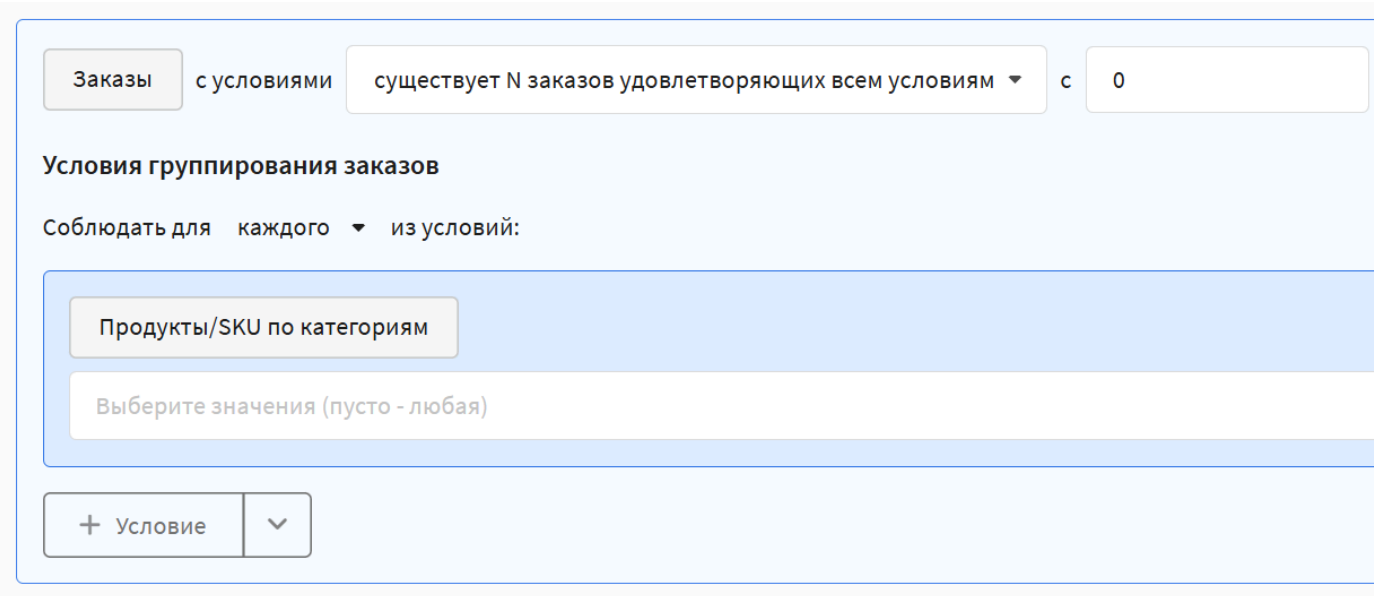
By Order Amount

Selects orders with a specific cost. Calculation currency can be specified.



Products/SKUs by Category

Selects orders containing products from specific categories.



Products by Name

Selects orders containing specific products — by name or SKU.

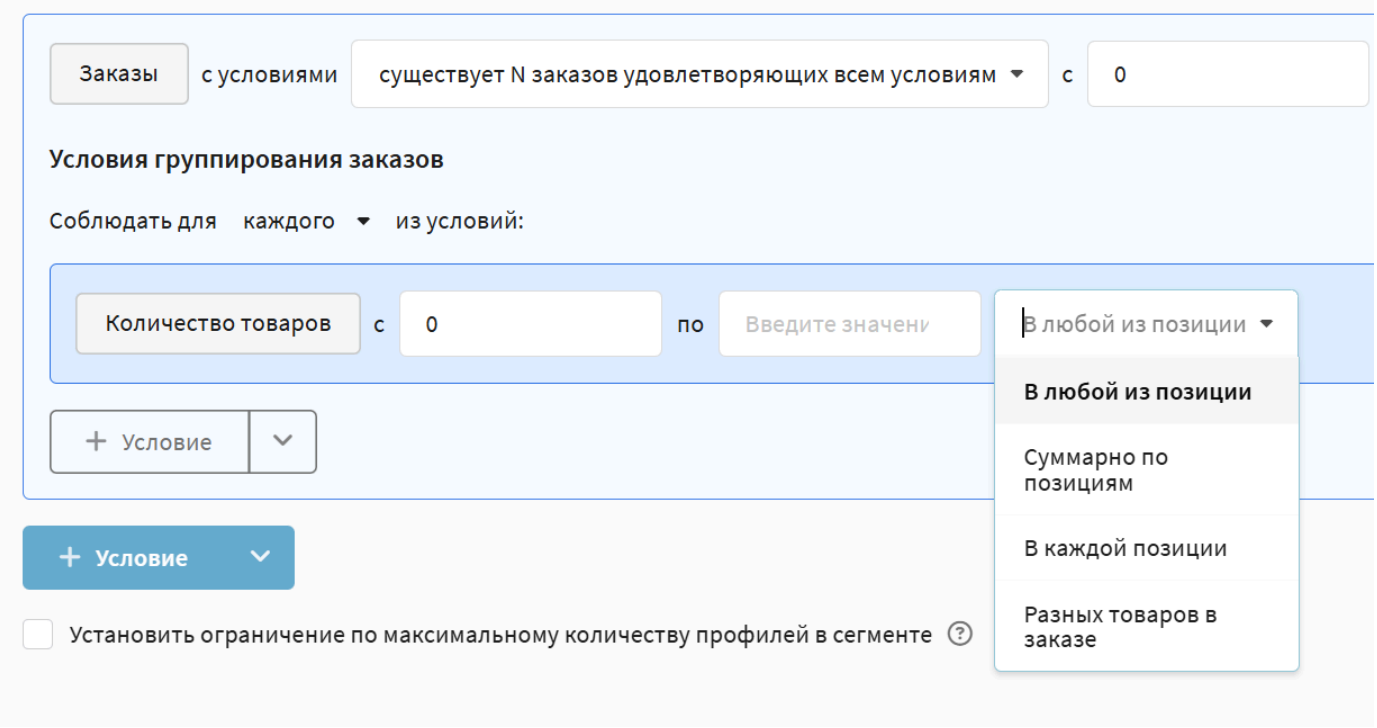
Product Segments

Selects orders containing products included in the specified product segment.

By Quantity of Goods

Selects orders with a specific quantity of products. Available calculation methods:

- **in any of the items — any order item has the specified quantity.**
- **summed across items — all order items together have the specified quantity.**
- **in each of the items — each order item has the specified quantity.**
- **different products in order — number of items in the order (1 product = 1 item).**



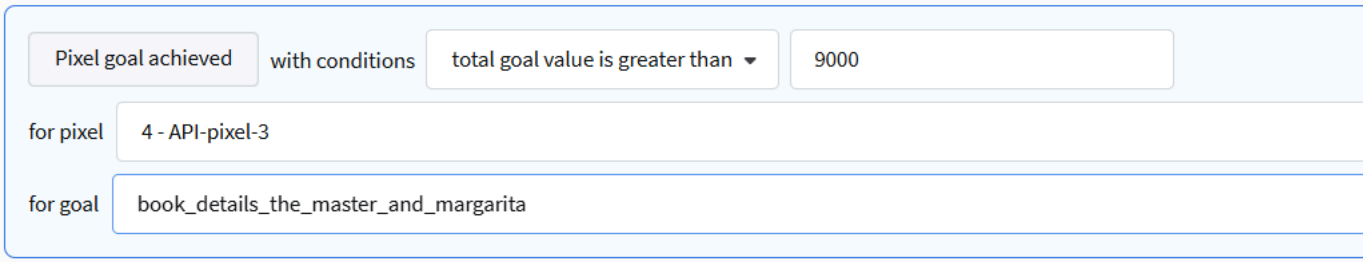
Goal/Pixel Behavior

Goal Achieved

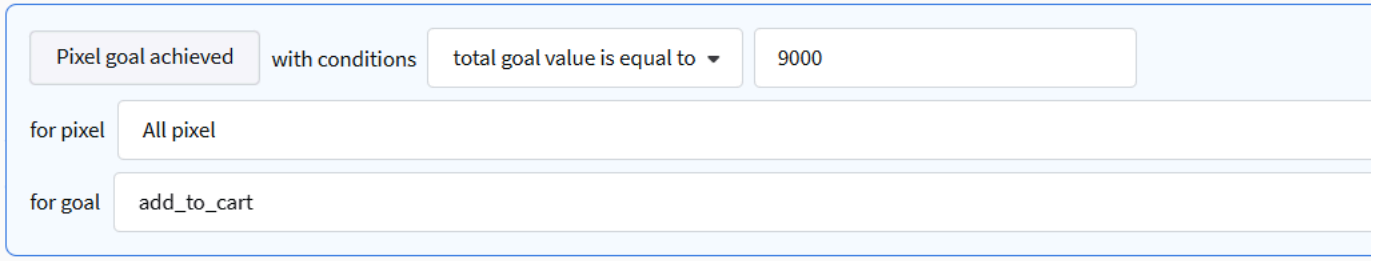
Segmentation based on subscriber actions on external resources (websites, apps) tracked using Altcraft

Available operators:

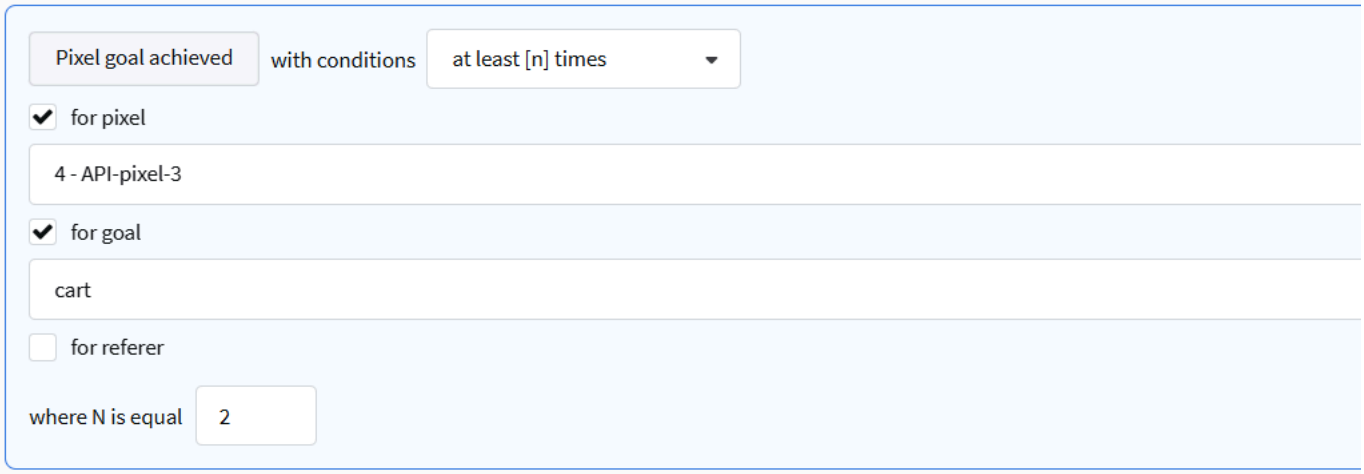
- **Exists / Does not exist — checks for the presence or absence of goal achievements.**
- **Total goal value — calculates the sum of the cost of achieved goals. Operators: greater than / less th time, for a period, or for the last [x] days.**



- **Average goal value — calculates the arithmetic mean of the cost of achieved goals. Operators: great specified for all time, for a period, or for the last [x] days.**



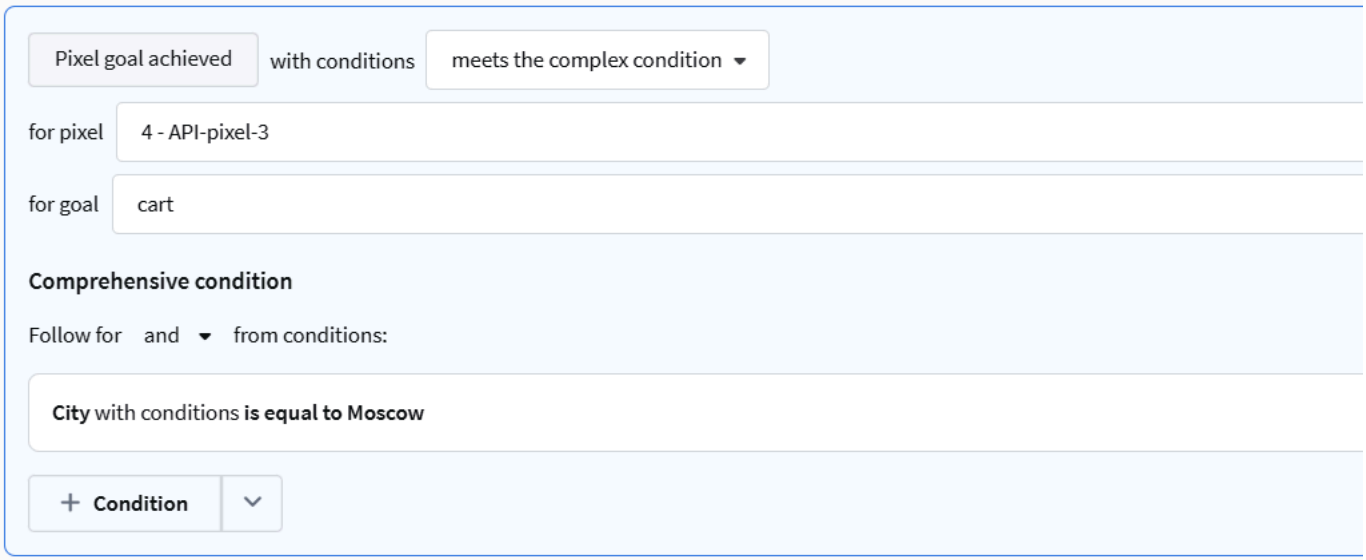
- **At least [n] times / Not happened — selection based on the number and frequency of actions perform (traffic source).**



- **Meets / Does not meet complex condition — extended selection based on additional data passed wh**

Rules can be configured for selection by:

- **Goal value (cost)**
- **Data from the `pixel_data` object (string, numeric, date, boolean)**
- **Profile device and geo-data (browser, OS, country, city, etc.)**
- **Sent messages and promo codes**
- **UTM labels**
- **URL Referer and Origin**
- **Profile order data**

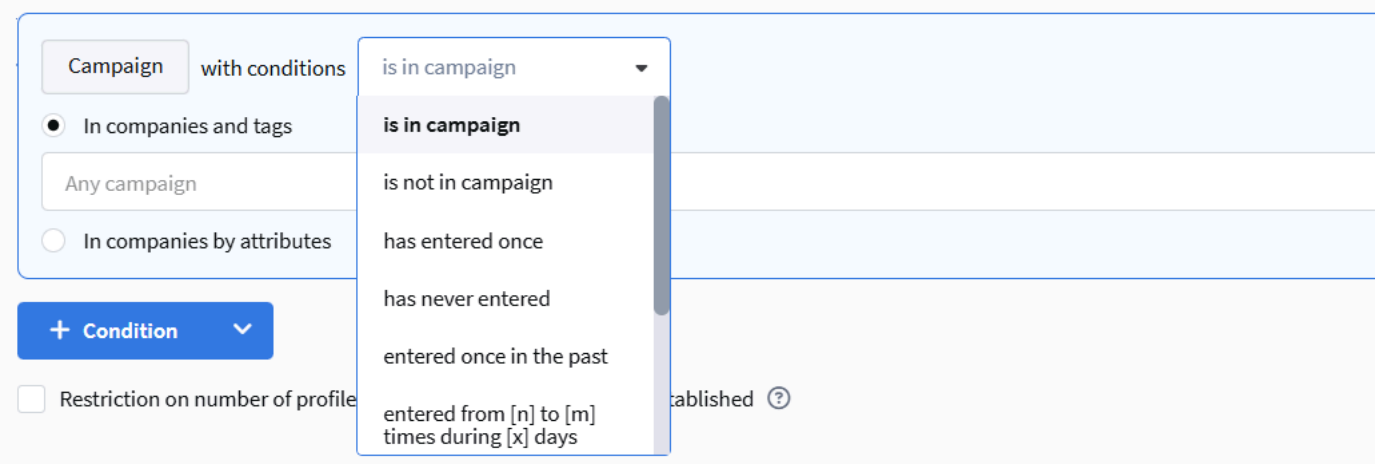


Campaigns

Segmentation by participation in a campaign.

- **currently in campaign — the profile is currently in the campaign.**
- **not in campaign — the profile is currently not in the campaign.**
- **entered at least once — the profile has ever been in the campaign.**
- **never entered — the profile has never been in the campaign.**

- **currently not, but entered at least once — the profile is currently not in the campaign but was previo**
- **entered at least [n] times in the last [x] days — the profile entered the campaign the specified numb**
- **entered at least [n] times in period — the profile entered the campaign the specified number of times**
- **never entered in the last [x] days — the profile has not entered the campaign in the last X days.**
- **never entered in period — the profile has not entered the campaign in the selected period.**

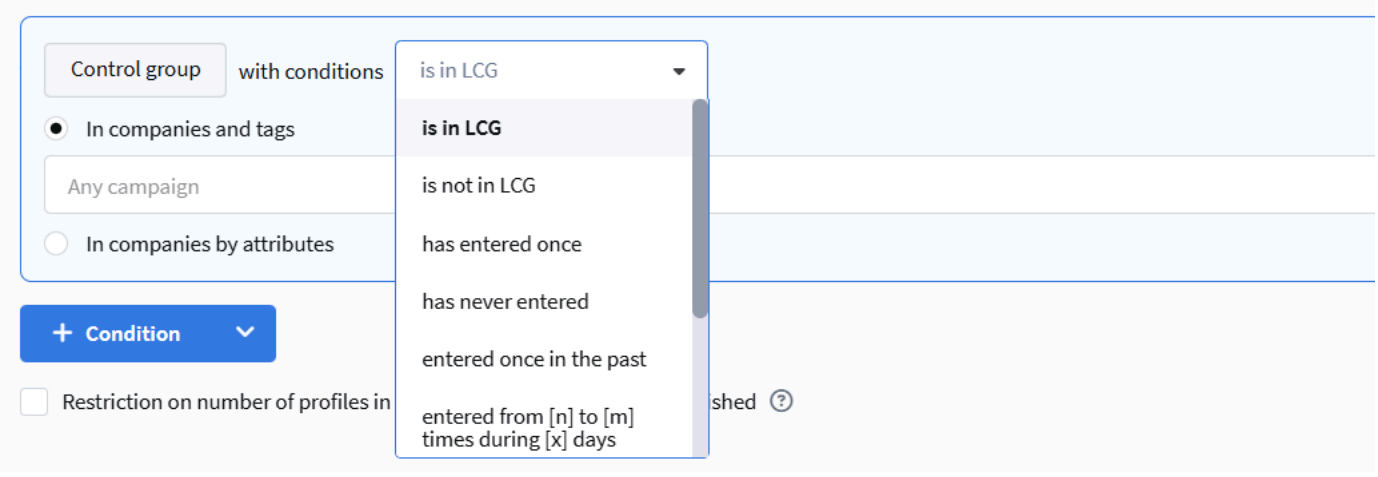


Control Group / Test Group

Segmentation by participation in test or local control groups.

Available operators:

- **is in GCG — the profile is currently in the global control group.**
- **is not in GCG — the profile is currently not in the global control group.**
- **has entered once — the profile has ever been in the control group.**
- **has never entered — the profile has never been in the control group.**
- **entered once in the past — the profile is currently not in the control group but was previously.**
- **entered at least [n] times during [x] days — the profile entered the control group the specified numb**
- **entered at least [n] times during period — the profile entered the control group the specified number**
- **never entered during [x] days — the profile has not entered the control group in the last X days.**
- **never entered during period — the profile has not entered the control group in the selected period.**



Segmentation by Activity of the channel

The history of subscriber actions within sent messages is formed based on events in various communica this activity allow segmentation by engagement (e.g., by opens and clicks) and effective setup of retarge
For most rules of segmentation by message activity history, the following types of selection operators a

- **Exists/Does not exist — selection based on the presence of at least one action of the selected type f**
- **Not happened — selection based on the absence of at least one action of the selected type. Dependi additionally specify the campaign, message template, time period, or number of days from the curre**
- **At least [n] times — selection based on the presence of the specified number of events for subscribe can additionally specify the campaign, message template, time period, or number of days from the c**

NOTE

How "in the last x days" is calculated
"x days" means the number of calendar days counted backwards from the current day inclusive:

- **1 day — the current day only**
- **2 days — the current day + the previous day**
- **7 days — the current day + 6 previous days, etc.**

Each day is a calendar day — from 00:00:00 to 23:59:59.9999 according to the account's timezone (TZ).

TIP

In all rules except Exists, you can select message templates or campaigns — within which the action w several templates or campaigns in one rule. Action in any of these objects will count — they are combir

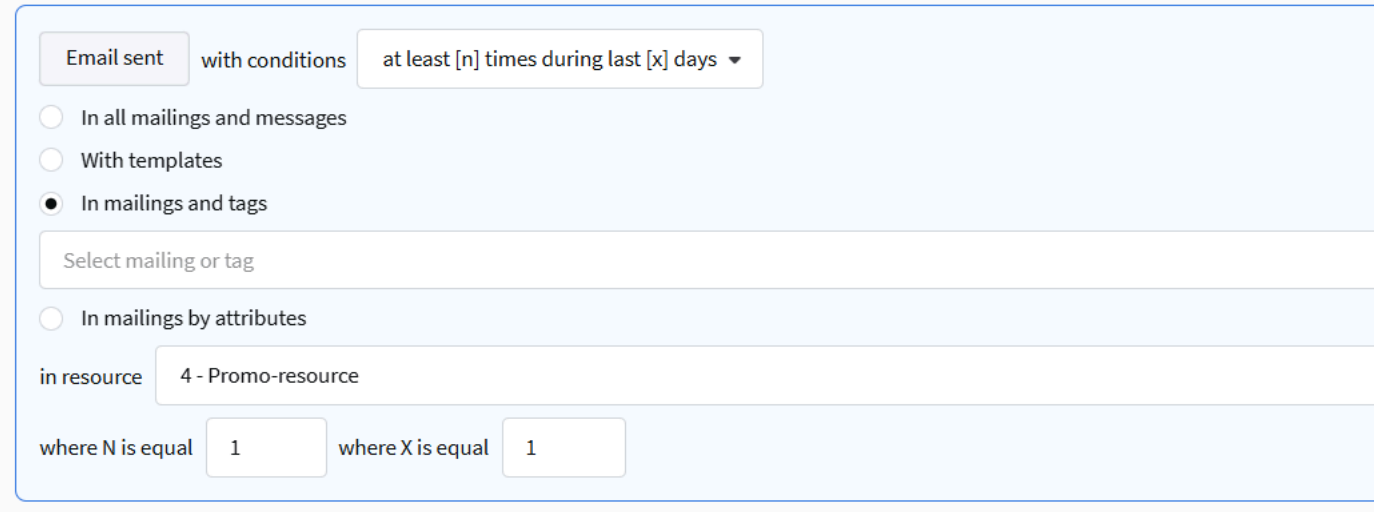
Common Events for All Channels

For most communication channels, similar event types are available. Below are common event groups wi they apply.

Message Sent

The event is registered when a message is formed and redirected to the sending module (mail server, SM platform).

- **Available in: Email, SMS, Push, Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Vib**



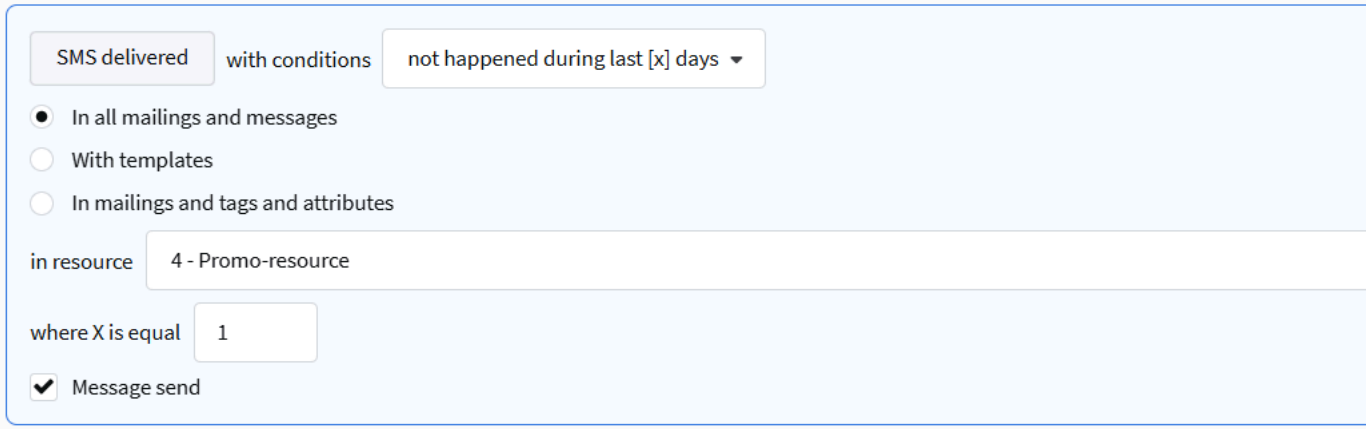
Message Delivered

The event is registered when the platform receives confirmation of message acceptance for delivery fro

- **Available in: Email, SMS, Push, Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Vib**

TIP

Allows determining which subscribers' messages were not delivered due to temporary channel unavail



Message Opened / Read

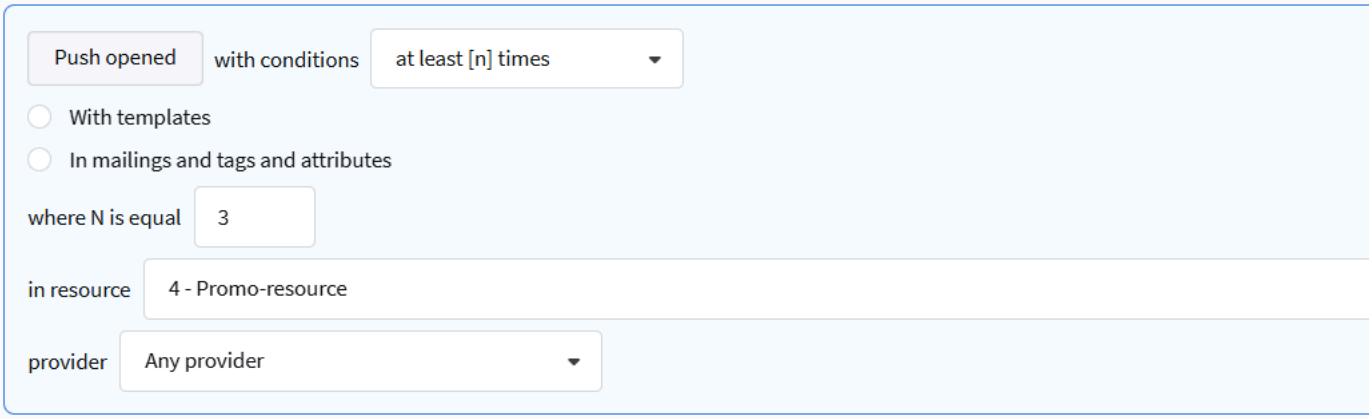
The event is registered when a user opens a message.

- **Opened: Email, SMS, Push.**
- **Read: Email, Push, WhatsApp, Viber, VK Notify.**

For Email, opening is tracked using a pixel in the email body.

TIP

Allows obtaining the exact number of unique opens and calculating the Open Rate (OR) for a specific re



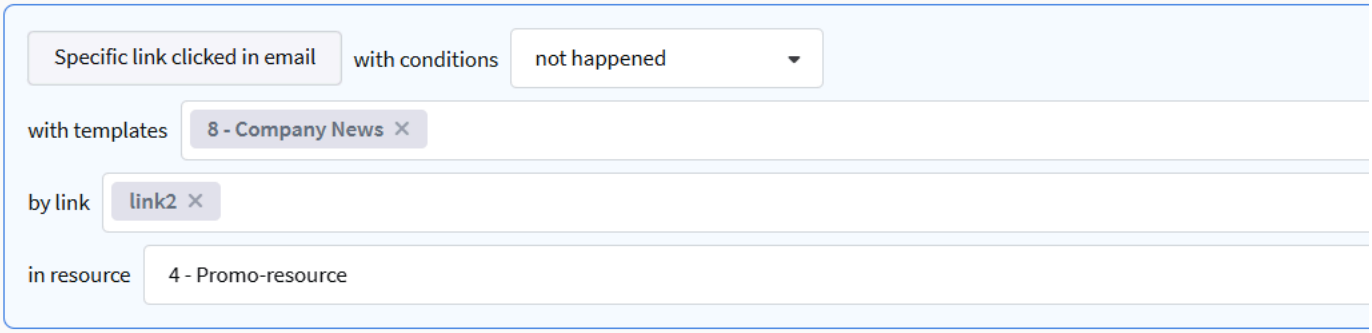
Clicks in Messages

Various events related to clicks on links within messages.

- **Any click in message: Email, SMS, Push, VK Notify. Selects recipients who clicked on any link in the**
- **Specific link click in message: Email, SMS, Push, Viber, WhatsApp, Telegram Bot, MAX Bot, VK Notif templates and links for selection of clicks.**
- **Unsubscribe click in message: Email, SMS, Push, VK Notify. Selects recipients who unsubscribed via**

TIP

A click on an unsubscribe link has a separate event type and is not included in the selection by the rule



Unique Clicks in Messengers (Telegram, MAX, WhatsApp, Viber, VK Notify)

For messengers, detailed click statistics are available:

- **Click — each link click is recorded.**

- **Unique click — the event is registered if the recipient clicked on at least any link in the message (1 u**
- **Unique link click — unique click on a specific link (one event is recorded for each link, even with mul**

Non-delivery and Sending Errors

Events related to problems during sending or delivery.

- **Non-delivery: Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Viber, VK Notify.**
- **Sending policy violation: Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, VK Notify.**
- **Hard bounced (permanent delivery errors): Telegram Bot, MAX Bot.**

Available operators for these events: Exists/Does not exist, Not happened, At least [n] times.

Subscription Management in the Channel

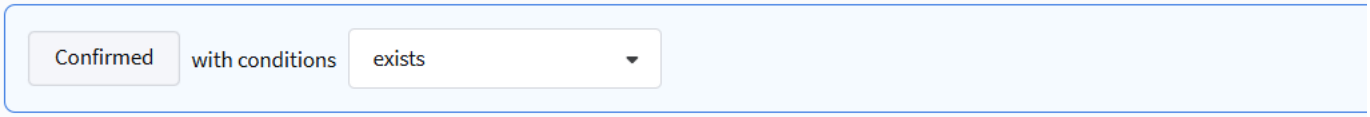
Events related to subscription and unsubscription from communication in a specific channel.

- **Subscribed to channel: Telegram Bot, MAX Bot, WhatsApp, Viber, VK Notify.**
- **Unsubscribed from channel: Telegram Bot, MAX Bot, WhatsApp, Viber, VK Notify.**
- **Subscription confirmation: Telegram Bot, MAX Bot, Email (Click on Confirm link).**

Channel-Specific Events

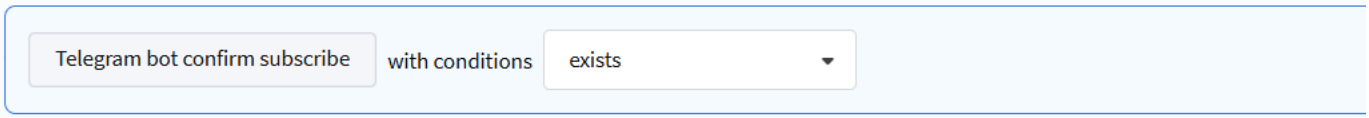
Email

- **Email Read — the `{read}` pixel triggers after 12 seconds of the email being open.**
- **Click on Confirm link — subscription confirmation via Double opt-in.**



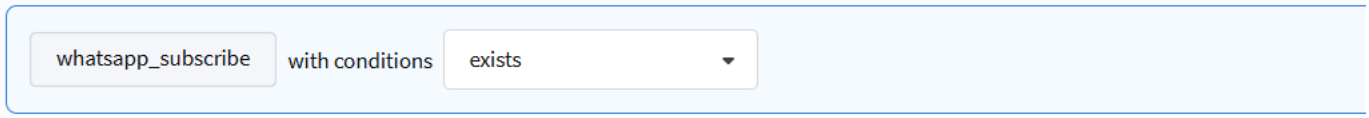
Telegram Bot

- **Telegram bot subscription confirmed — confirmation of subscription to the bot.**



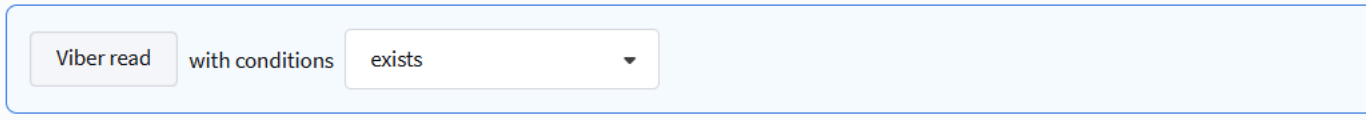
WhatsApp

- **WhatsApp read — message read event.**



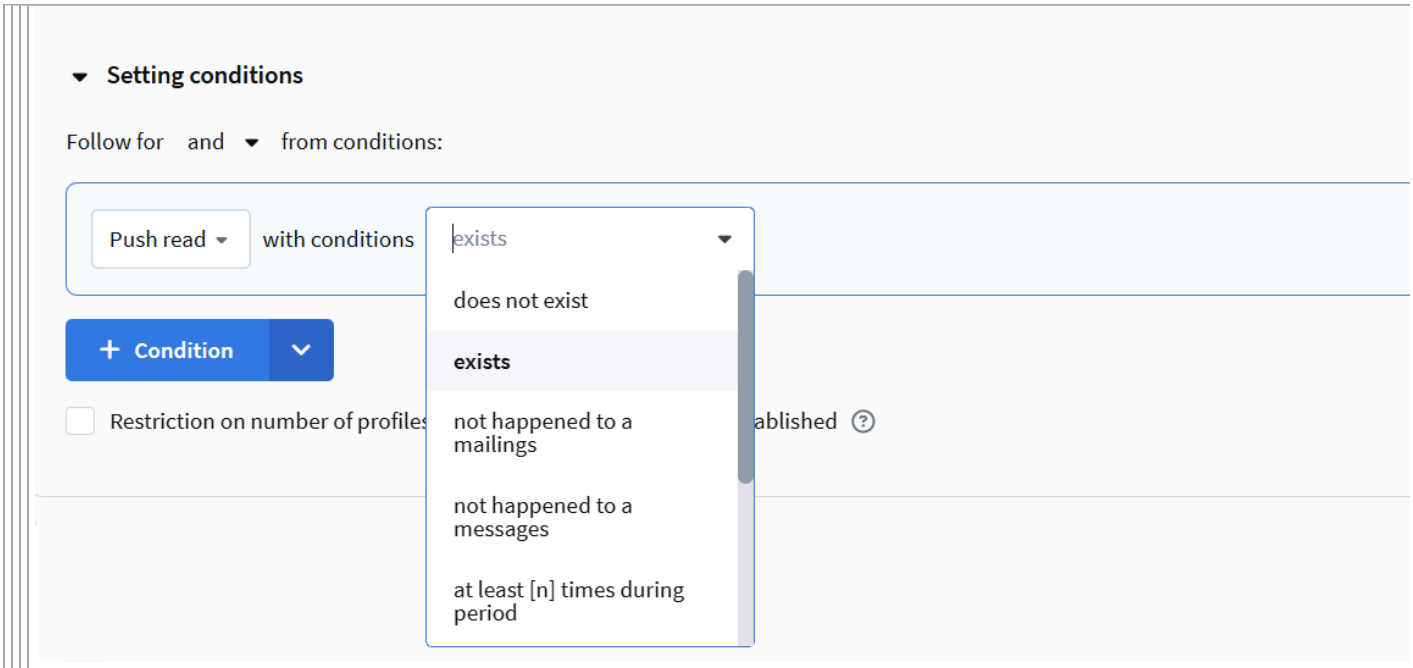
Viber

- **Viber read — message read event.**



Push

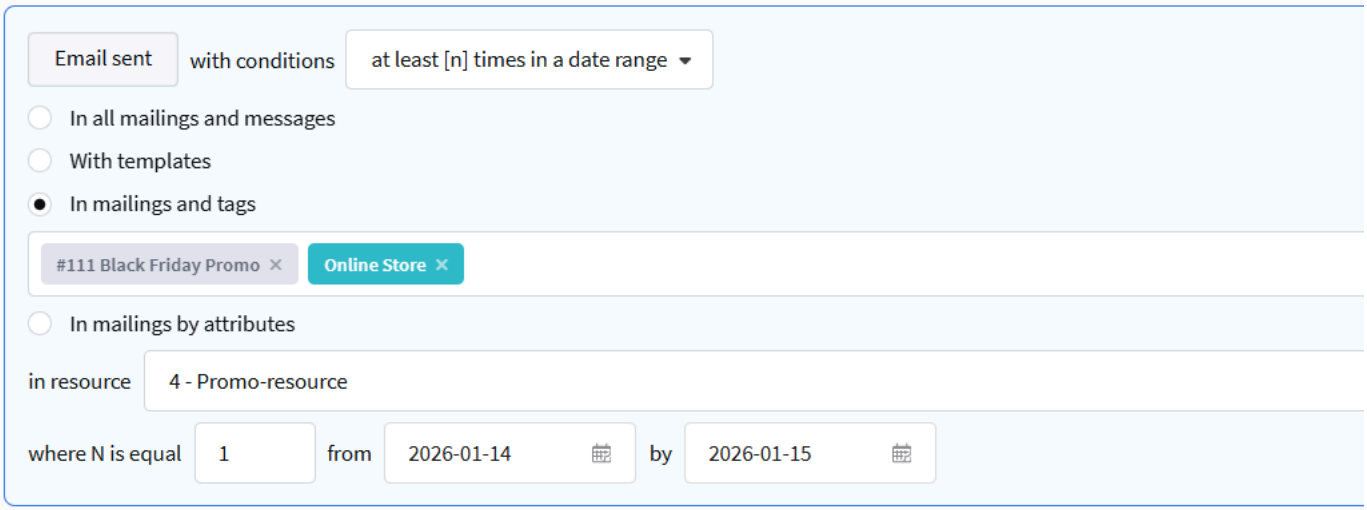
- **Push read — the event is registered when a recipient reads a push notification in the notification cer communication when notification content is duplicated for deferred viewing: the platform tracks not also the fact of reading.**



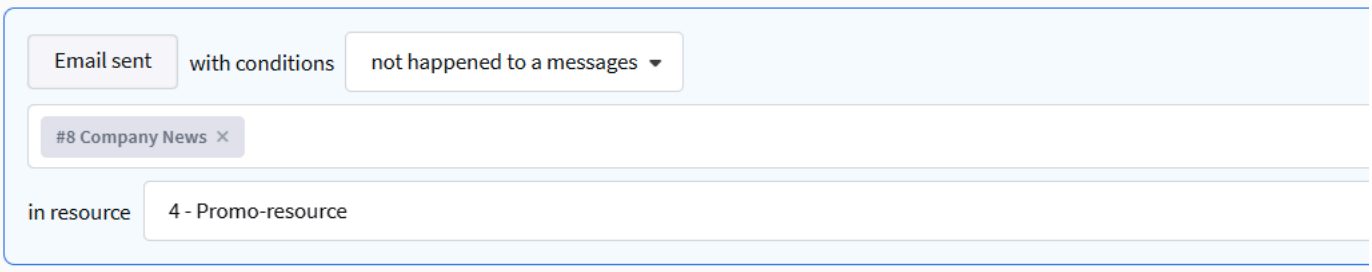
Checking an Event in a Message or Campaign

For precise checking of whether an event occurred in a specific campaign or message template, use cor

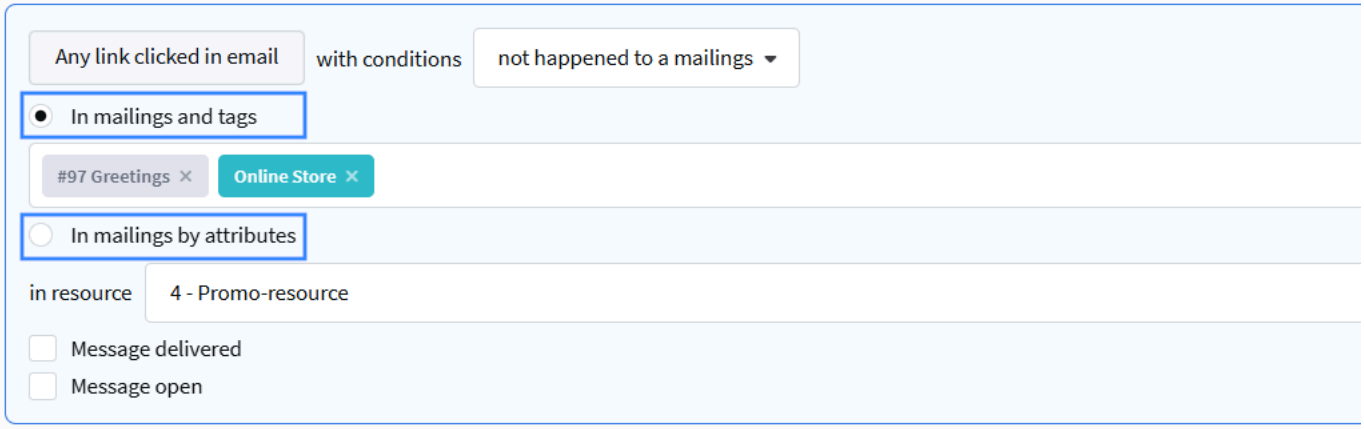
- **At least [n] times in messages / in campaigns — the profile will be included in the segment if the acti templates or campaigns.**



- **Not happened to messages / campaigns — the profile will be included in the segment if the action di objects.**

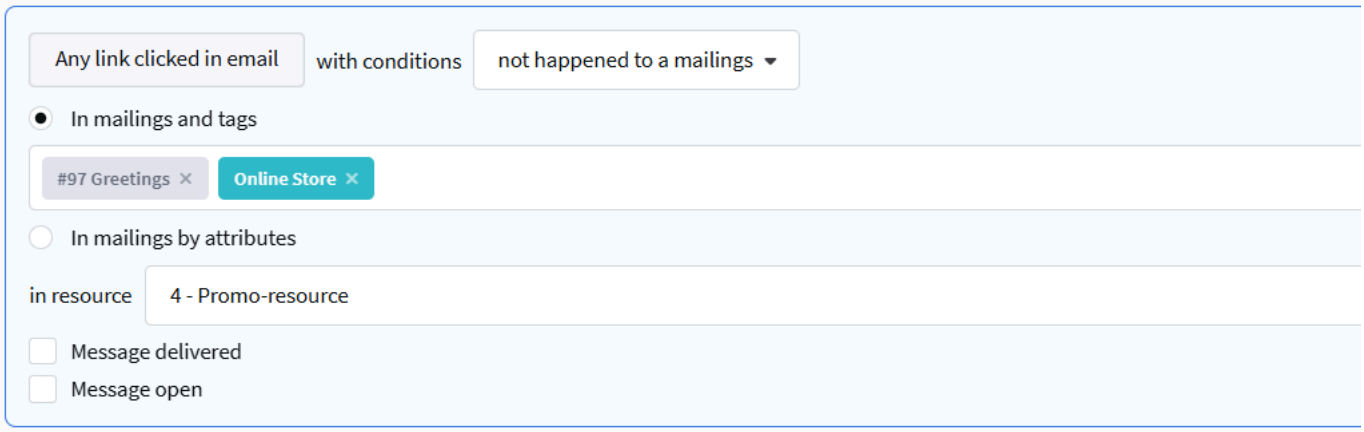


For reverse conditions, you can add a check for a preceding event (e.g., for a click — a check for delivery

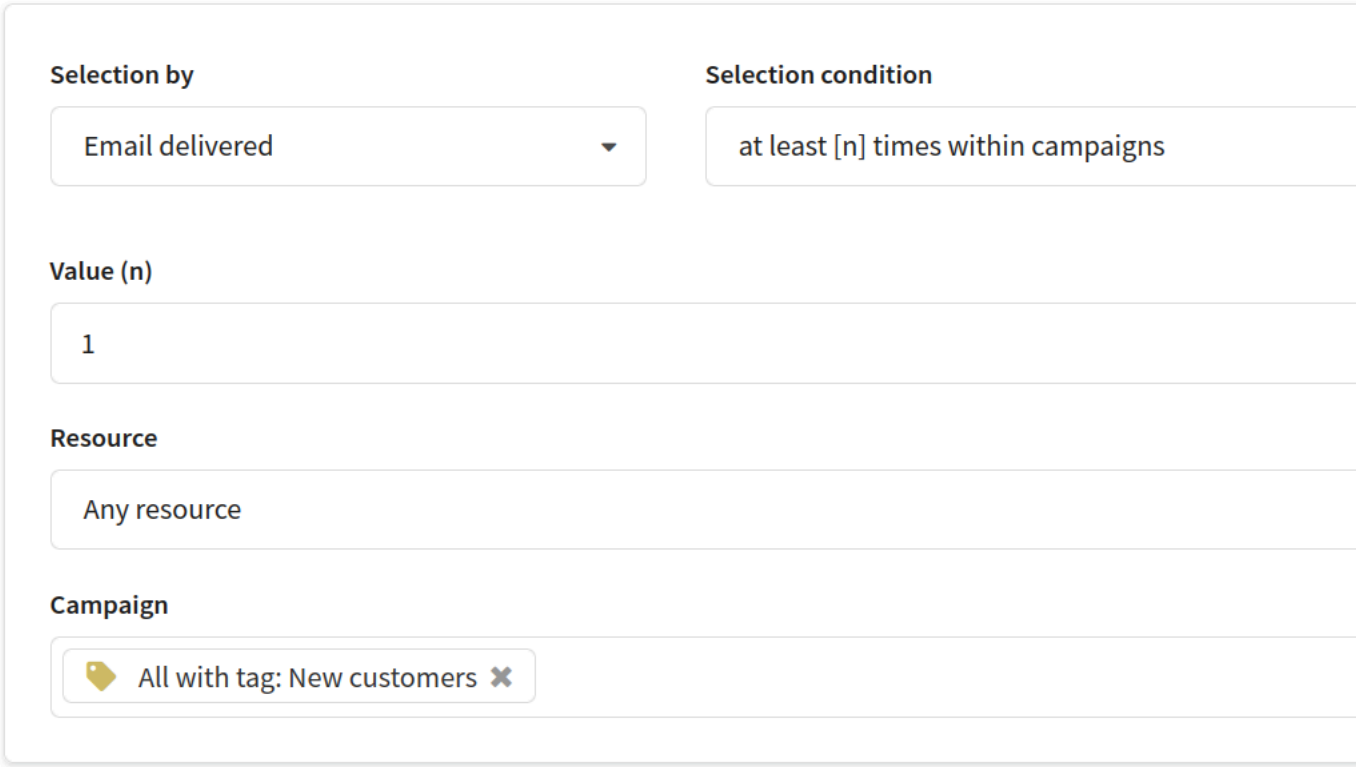


Selecting Campaigns, Messages, or Templates by Tags or Attributes

When segmenting by events in a campaign or message, you can also filter by tags and by attributes. Selection of objects for checking is available not only individually but also by tags or by attributes.



In the example below, profiles that clicked in any campaign tagged “Online store” will be selected:



** Meta, the organization that owns Instagram, Facebook, and WhatsApp, is recognized as extremist and*

** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.*

Segmentation by external data

Segmentation rules can request data for profile selection from an external source. This allows you to au and reduce the number of repetitive conditions in a segment.

External sources can include:

- **Web service APIs**
- **External SQL databases**
- **Uploaded files**
- **Manually entered lists**

Segmentation by External data

Description

Segmentation rules can appeal to external sources for profiles selection data. This information can be st

- **Web service API**
- **Uploaded file**
- **Data input form (in the rule)**

This allows you to make automatic segmentation by mane fields, and reduce similar conditions in the se

HTTP queries to external service API

If you select Is (not) in a field of post url responce operator in the rule, you can perform selection depen Platform can take data in JSON, XML, and PIPE formats.

JSON segmentation rule example

Let's take situation, when information is given in JSON format. MIME header will be given, like:

`"Content-Type": "application/json"`

and data in this format:

```
{
  "segment": {
    "emails": [
      "john@example.org",
      "suzy@example.com",
      "rick@example.net",
      "andy@example.edu"
    ]
  }
}
```

It is necessary to set API endpoint URL, and path to necessary field in JSON, in the segmentation rule:

Segmentation rule example for XML

Let's take situation, when information is given in XML format. MIME header will be given, like:

`"Content-Type": "application/xml; charset=UTF-8"`

and data in this format:

```
<names>
  <name>Andy</name>
  <name>Dave</name>
  <name>John</name>
  <name>Suzy</name>
</names>
```

It is necessary to set API endpoint URL, and path to necessary field in XML object, in the segmentation r

Segmentation rule example for PIPE

PIPE is the format of data, divided by line breaks (written in one column). Let's take situation, when infor header will be given, like:

`"Content-Type": "text/plain; charset=UTF-8"`

and data in this format:

```
+78002000100
+78004000200
+78006000400
+78008000600
```

It is necessary to set API endpoint URL in the segmentation rule:

Uploaded file

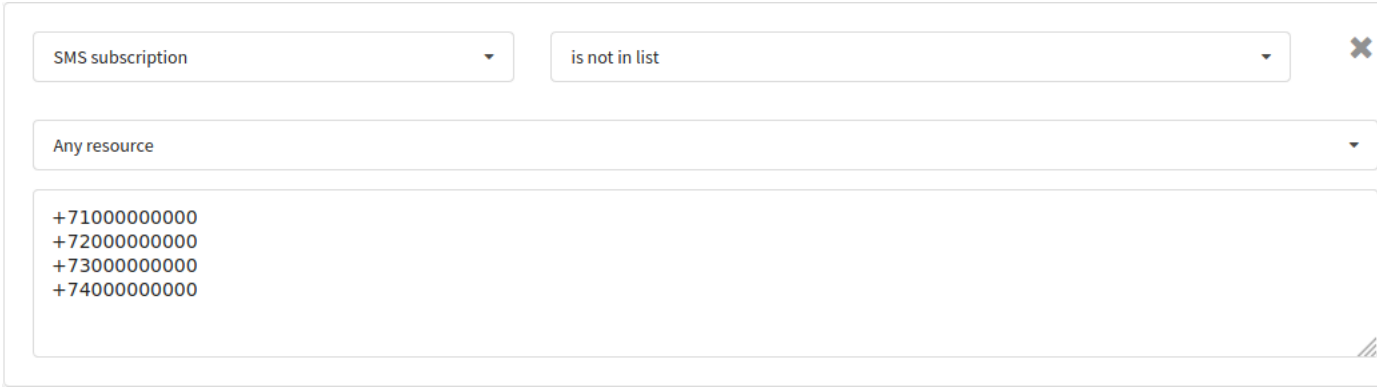
If you select Is (Is not) is file in a column segmentation rule, form for uploading CSV (comma separated v

```
Email,Name
john@example.org,John
suzy@example.com,Suzy
dave@example.net,Dave
andy@example.edu,Andy
```

After uploading the file, you can select column in the drop-down menu:

Information from buffer

If you select Is (Is not) in list segmentation rule, data in PIPE format will be inserted manually into the sp
PIPE is the format of data, divided by line breaks (written in one column).



Segmentation by External SQL tables

The ability to connect external SQL databases allows for flexible profile segmentation:

- **You don't need to store all data in Altcraft Platform**
- **No regular synchronization is needed — data is always up to date**
- **You can use data for entities not available in Altcraft Platform**
- **It works regardless of the platform or system you use to store data; all you need is access to execut**

Database connectors for external databases are created in the Altcraft administrative panel.

> TIP
>
> **If you are using a cloud solution — send the database connection credentials to our support service: su clarify questions related to connecting to other external services and databases there.**

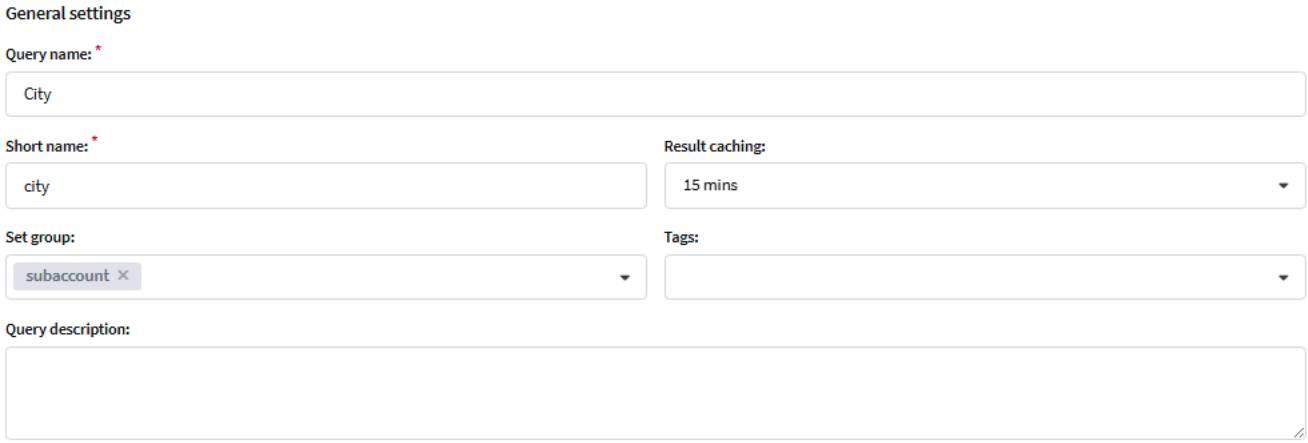
Creating a database query

Queries can be configured both in the administrative panel and in the user panel. The difference is that i view the profiles that were selected by your query.

Editing queries in the user panel can be disabled by configuration setting in main.json if necessary. To d `"DISABLE_QUERY_EDITOR": true` **.**

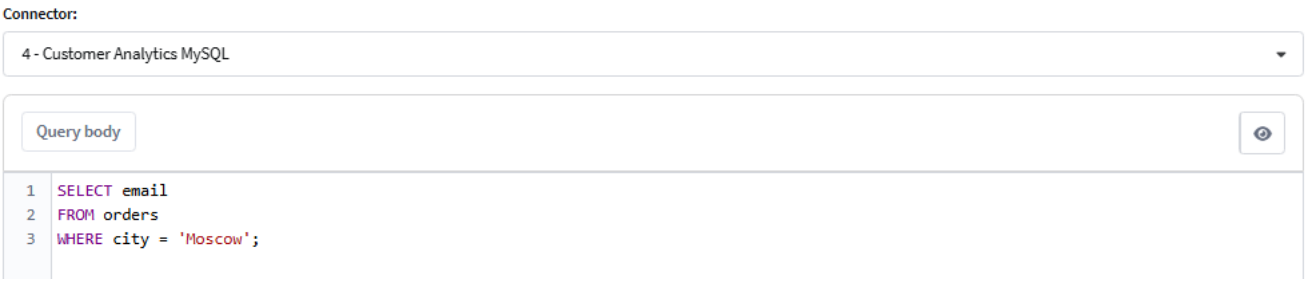
![img] (/img/attachments/9408217/segmentation_queries_en.png)	**Queries are configured in the **Data —> Segmentation Queries** section. Go to the section and press the **+ Cre**

In the main settings, specify the query name, short name (unique for each query, used in platform variab period, as well as the query group, tags, and description:



> INFO
>
> **Note: the result caching period only works when calculating segments. Automation scenarios use a sep own settings.**

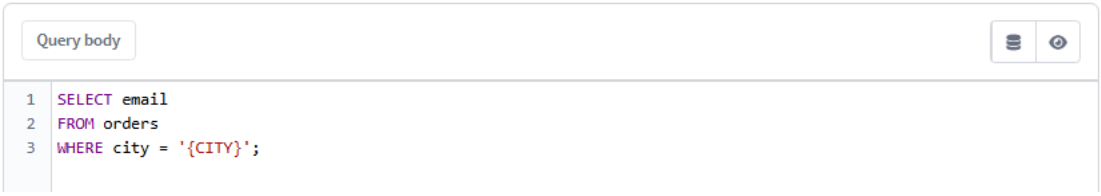
For the segmentation query to work correctly, it must select a column with data that already exists in Al Altcraft has a customer identifier in the profile database, you need to form a query so that this identifier columns:



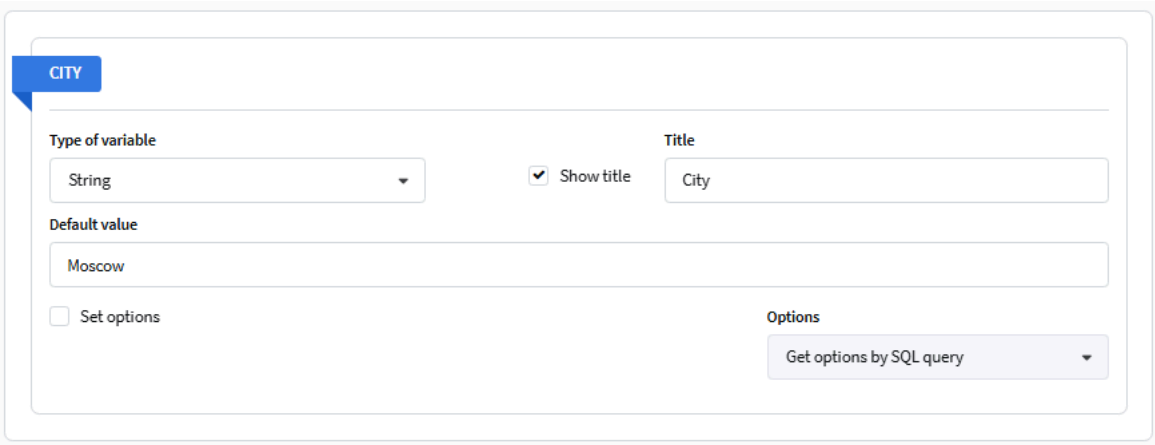
These queries will be used inside the segment. You can create an unlimited number of such queries to re can select by one or several parameters, combining queries, using subqueries, JOINs, UNIONs, etc.

Be sure to select the column that will be used to search for profiles; use the update button to get a new

You can add parameters to the query so that it changes when used inside the segment. To do this, use a name in curly braces, for example `{CITY}`:

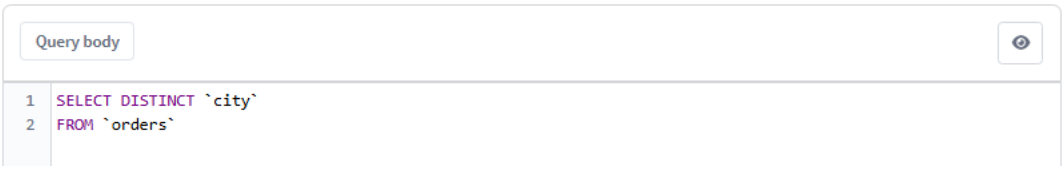


Once you specify it in the query, a form with parameter settings will appear:

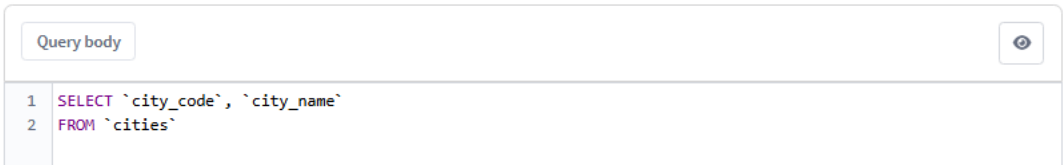


Here you set the field type, its default value, and the title that will be displayed when creating the segme

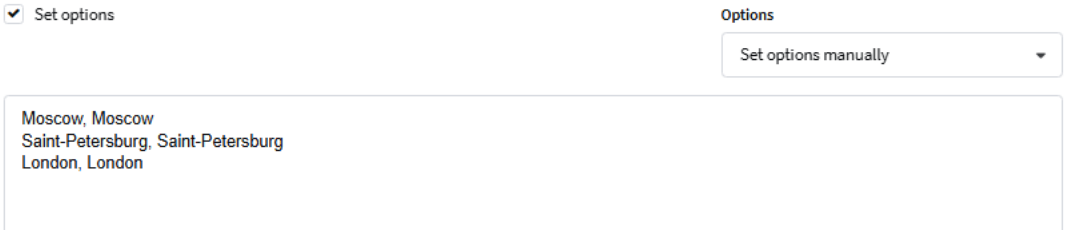
If we want the segment field to be selected from a list, we use the Set options setting. When enabled, w list of options or also select it from SQL:



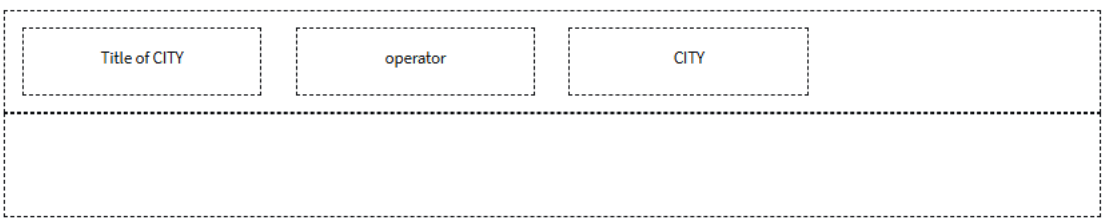
If identifiers are used, the SQL query must return two columns: the first is used as the identifier, and the



With a manually specified list, the identifier and options can be listed separated by commas:

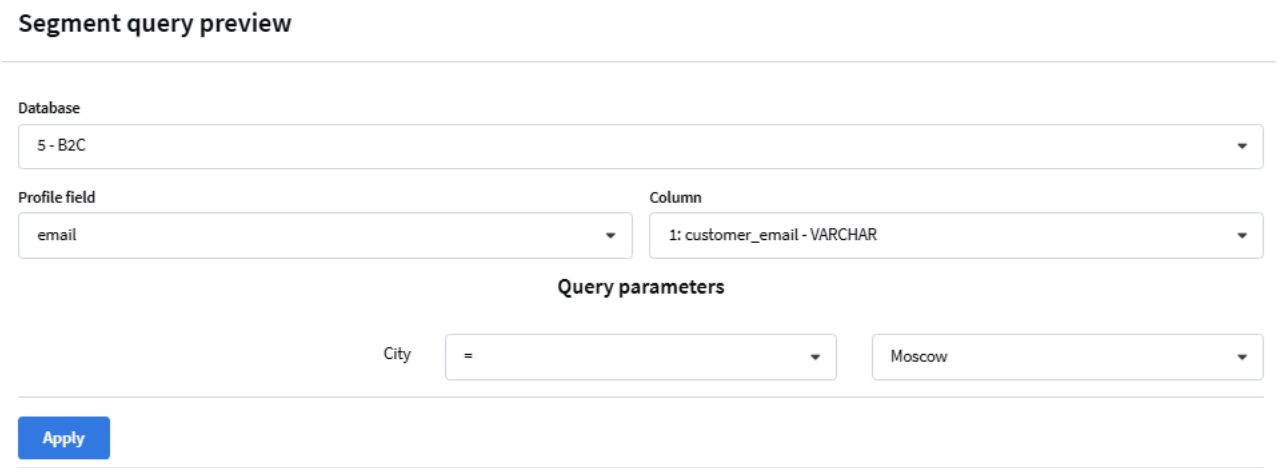


For all parameters, the order of their display during segment construction is configured. The order can b elements. Headers are positioned separately for more flexible display.



Use the database selection preview button to see how the query form will look in the segment and what

Specify the database and profile field for selection, as well as the column if there are several. In the Que be generated as it will appear in the segment. Pressing the Apply button will run the query and immediat query condition:



After saving the query, it can be used in segments.

Variable Types

When creating a query parameter, you can choose one of the types. The chosen type determines how th query.

String

The value is substituted as text.

```
SELECT email FROM users WHERE city = '{CITY}'
```

For example, to find profiles from Moscow, set the `{CITY}` parameter to `Moscow` when configuring the segr parameter title, such as “City”. The following query will be sent to the database:

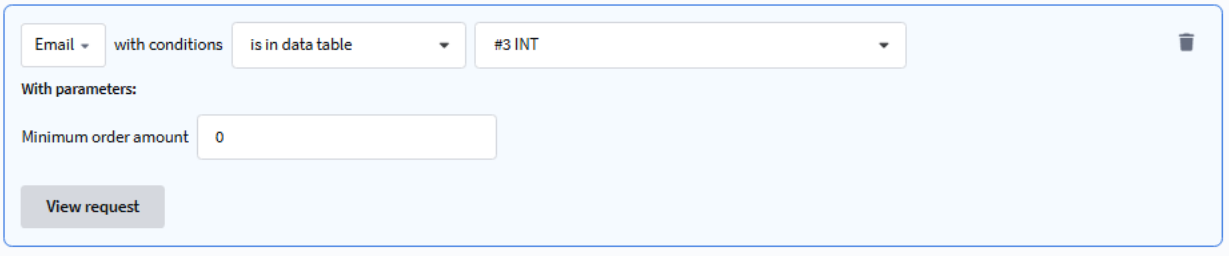
```
SELECT email FROM users WHERE city = 'Moscow'
```

As a result, profiles with Moscow specified in the `city` field will fall into the segment.

Integer and Floating Point Number

The value is substituted as an integer if the variable type is “integer”, or as a fractional number if the va Number”:

```
SELECT email FROM orders WHERE order_amount >= '{AMOUNT}'
```



For example, to find profiles with an order amount of at least 1500, set the `{AMOUNT}` parameter to `1500`. Y parameter title, such as “Minimum Order Amount”. The following query will be sent to the database:

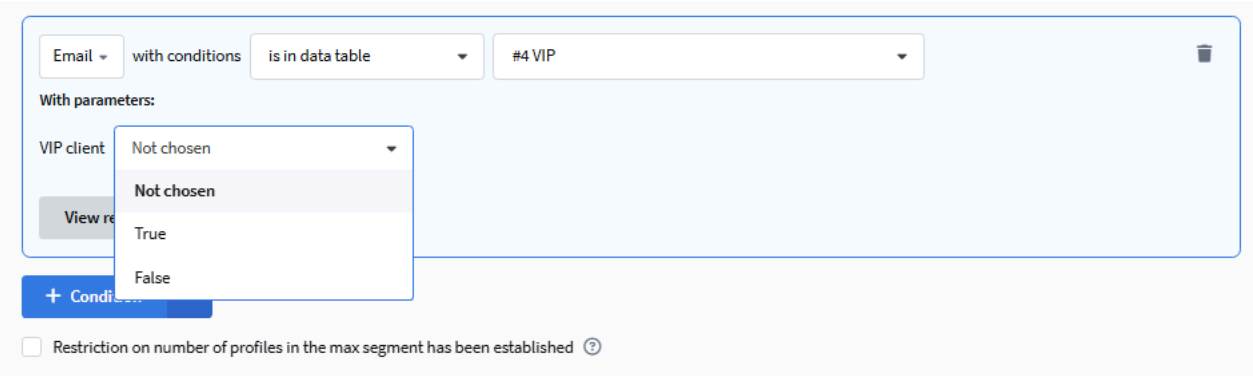
```
SELECT email FROM orders WHERE order_amount >= 1500
```

Profiles with an order amount greater than or equal to the specified amount will fall into the segment.

Boolean

The value is substituted as `true` (True), `false` (False), or `null` (Not selected).

```
SELECT email FROM orders WHERE is_vip = {IS_VIP}
```



For example, to find VIP clients, set the `{IS_VIP}` parameter to `True`. You can also set a descriptive paran following query will be sent to the database:

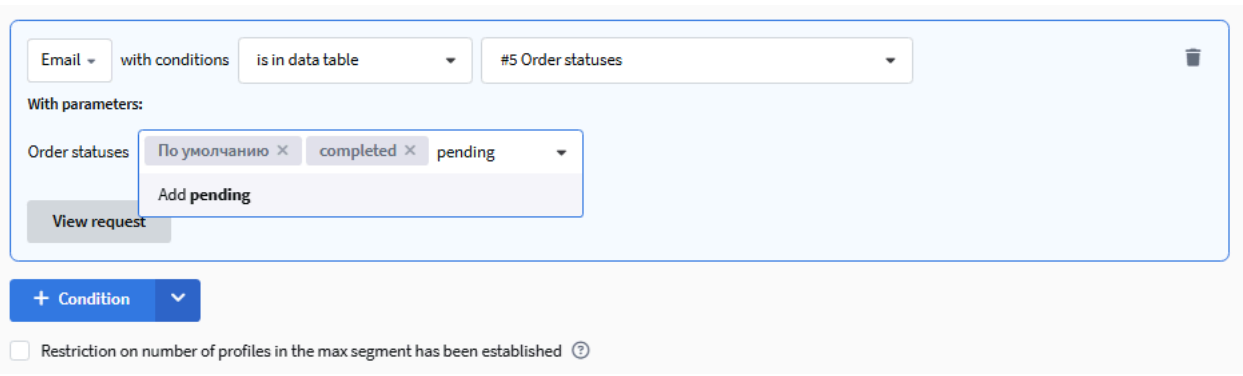
```
SELECT email FROM orders WHERE is_vip = true
```

As a result, profiles with VIP status will fall into the segment.

Array

Used for working with array-type columns in PostgreSQL. Values are substituted as an array.

```
SELECT email FROM orders WHERE statuses && ARRAY[{status_list}]
```



The variable `{status_list}` has the type “Array”, Array Element Type — “String”. When configuring the seg For example, the `orders` table has a `statuses` column of type `text[]`, which contains arrays of order statu have at least one of the statuses `completed` or `pending` in the array. In the segment, we select both values the database:

```
SELECT email FROM orders WHERE statuses && ARRAY['completed', 'pending']
```

As a result, profiles that have `completed` or `pending` in the `statuses` array will fall into the segment.

INFO

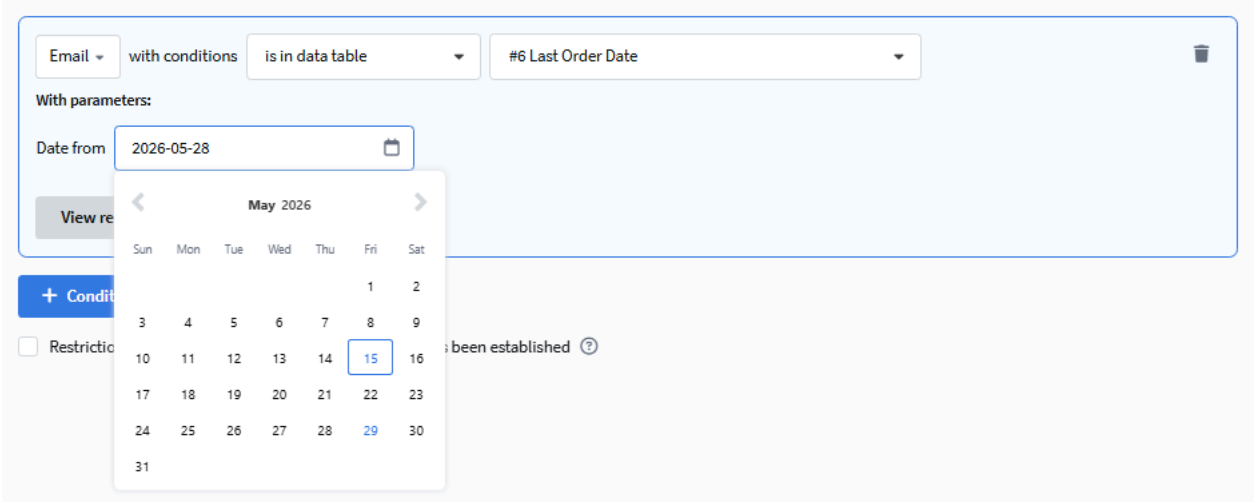
The “Array” type only works for PostgreSQL and requires that the column in the external table has an a This type is not supported for MySQL and ClickHouse.

To save the query, you must set at least one default value for the parameter.

Date

Used for filtering by dates. The value is substituted in the format that your database expects.

```
SELECT email FROM orders WHERE order_date >= '{date_from}'
```



The variable `{date_from}` has the type “Date”. When configuring the segment, we select a date in the cale
For example, we need to find profiles that placed an order after a certain date. In the segment, we selec
following query will be sent to the database:

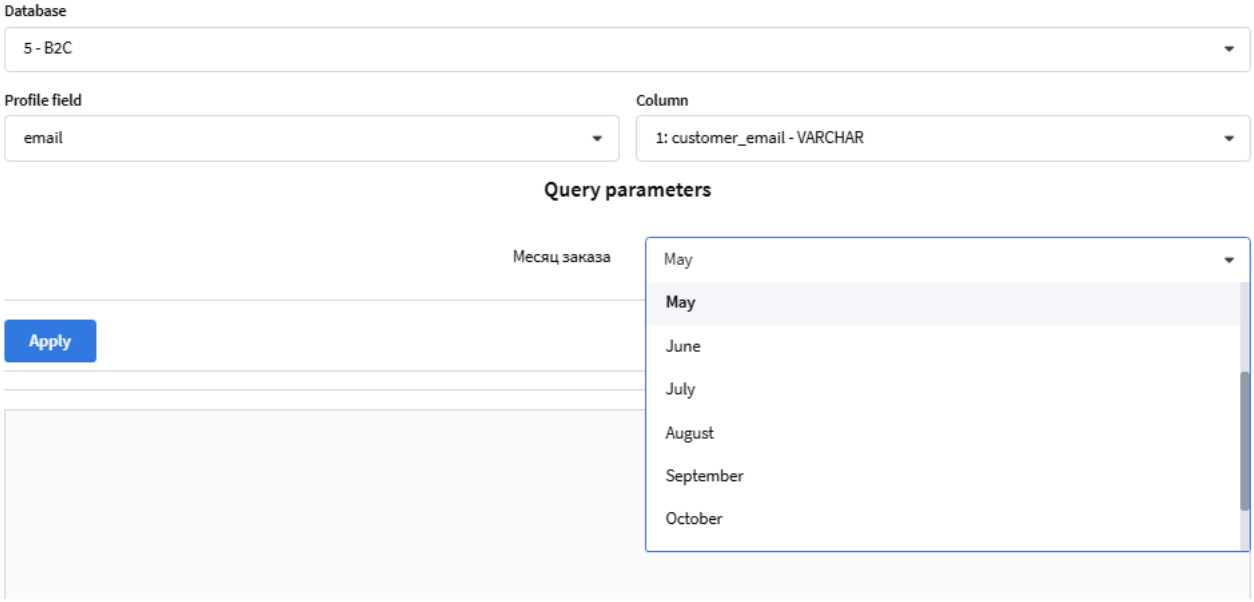
```
SELECT email FROM orders WHERE order_date >= '2024-05-28'
```

As a result, profiles with orders from the selected date and later will fall into the segment.

Month

Used for filtering by month from a date. The month number is substituted into the query.

```
SELECT email FROM orders WHERE MONTH(order_date) = '{month}'
```



The variable `{month}` has the type “Month”. When configuring the segment, we select a month from the d
For example, we need to find profiles that placed an order in May. In the segment, we select “May”. The
database:

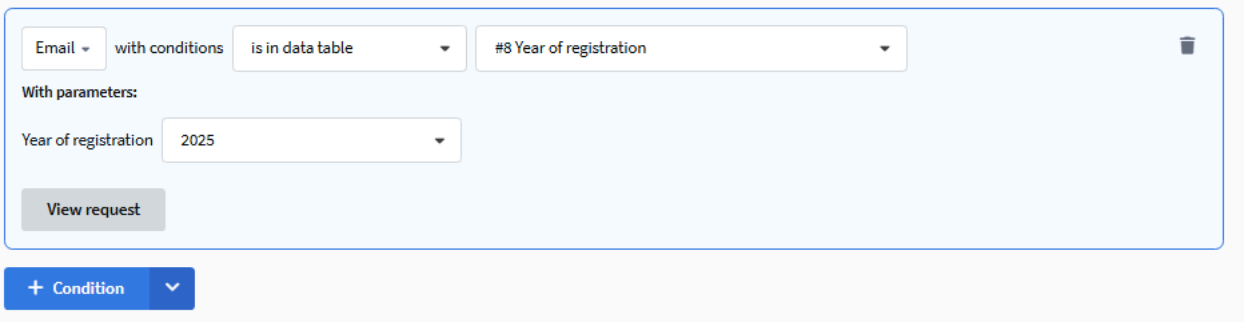
```
SELECT email FROM orders WHERE MONTH(order_date) = 5
```

As a result, profiles with orders in May will fall into the segment.

Year

Used for filtering by year from a date. The year is substituted into the query.

```
SELECT email FROM orders WHERE YEAR(order_date) = '{year}'
```



The variable `{year}` has the type "Year". When configuring the segment, we select a year.
For example, we need to find profiles that placed an order in 2025. In the segment, we select `2025`. The f database:

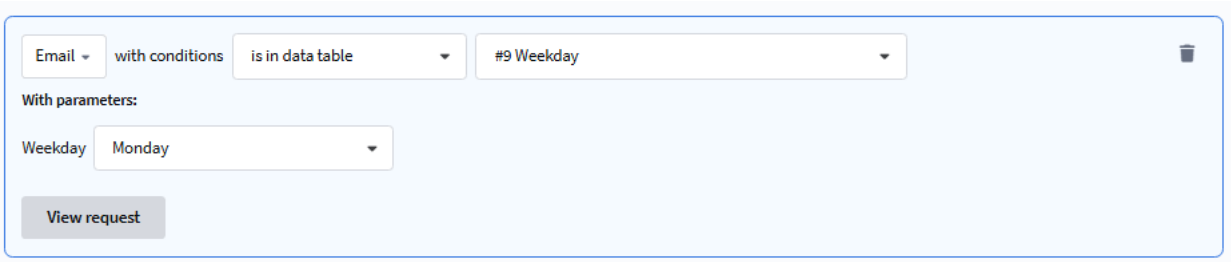
```
SELECT email FROM orders WHERE YEAR(order_date) = 2025
```

As a result, all profiles with orders in 2025 will fall into the segment.

Weekday

Used for filtering by day of the week. The day number is substituted into the query using the platform's n

```
SELECT customer_email FROM customer_orders WHERE WEEKDAY(order_date) = '{weekday}'
```



The variable `{weekday}` has the type "Weekday". When configuring the segment, we select a day of the we
Correspondence of days of the week and values substituted into the query:

Day of week	Value
Sunday	**1**
Monday	**2**
Tuesday	**3**
Wednesday	**4**
Thursday	**5**
Friday	**6**
Saturday	**7**

For example, we need to find profiles that placed an order on Tuesday. In the segment, we select "Tuesd to the database:

```
SELECT customer_email FROM customer_orders WHERE WEEKDAY(order_date) = 3
```

INFO

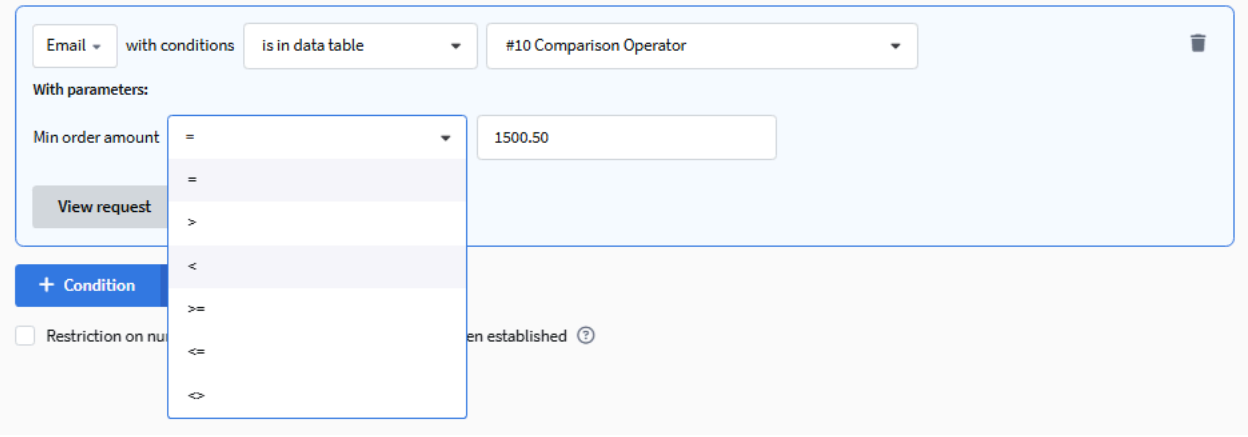
If your database uses a different day-of-week numbering, convert the value using the `CASE` operator. For returns 0 for Monday, 6 for Sunday. To map the platform numbering to MySQL numbering, use the followi

```
SELECT customer_email FROM customer_orders
WHERE WEEKDAY(order_date) = CASE '{weekday}'
    WHEN 1 THEN 6   -- sunday
    WHEN 2 THEN 0   -- monday
    WHEN 3 THEN 1   -- tuesday
    WHEN 4 THEN 2   -- wednesday
    WHEN 5 THEN 3   -- thursday
    WHEN 6 THEN 4   -- friday
    WHEN 7 THEN 5   -- saturday
END
```

Comparison Operator

A comparison operator is substituted into the query. Quotes are not used for the "Comparison Operator"

```
SELECT email FROM orders WHERE order_amount {operator} '{MIN_AMOUNT}'
```



The variable `{operator}` has the type "Comparison Operator". When configuring the segment, it allows sel dropdown list.

Available operators:

Operator	Description
`=`	**Equal**
`>`	**Greater than**
`<`	**Less than**
`>=`	**Greater than or equal**
`<=`	**Less than or equal**
`<>`	**Not equal**

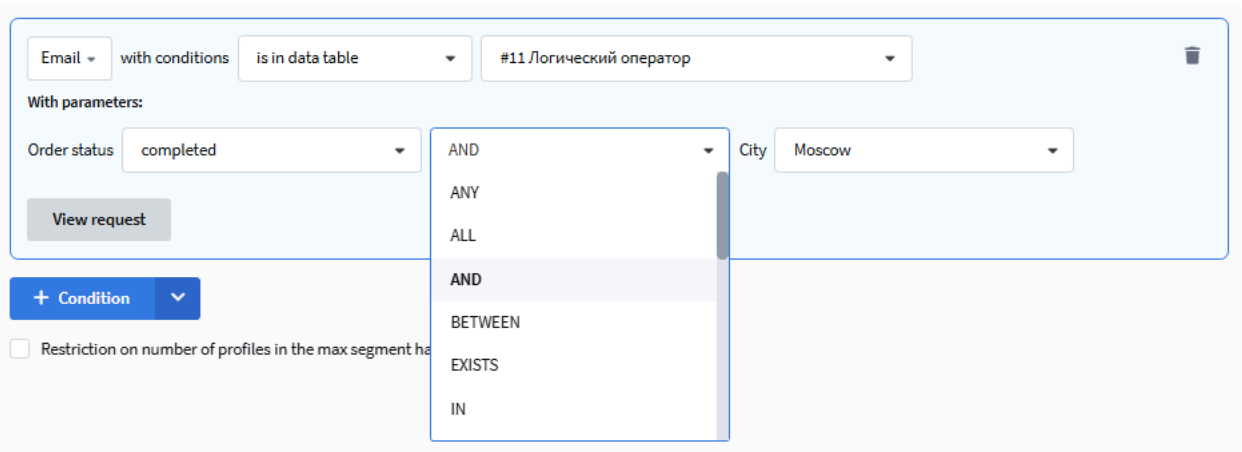
For example, we need to find profiles with an order amount greater than 1000. In the segment, we select `1000`. The following query will be sent to the database:

```
SELECT email FROM orders WHERE order_amount > 1000
```

Logical Operator

When selecting this variable type, the user will be offered to choose an operator from a list when configu are not used.

```
SELECT email FROM orders WHERE status = '{STATUS}' {logical} city = '{CITY}'
```



The variable `{logical}` has the type "Logical Operator". When configuring the segment, we select an oper

Available operators:

Operator	Description	Usage Example
`AND`	**Both conditions must be true**	`status = 'active' AND city = 'Moscow'`
`OR`	**At least one condition is true**	`status = 'active' OR city = 'Moscow'`
`NOT`	**Negation of condition**	`NOT (status = 'active')`
`IN`	**Value is in the list**	`city IN ('Moscow', 'Kazan')`
`NOT IN`	**Value is not in the list**	`city NOT IN ('Moscow', 'Kazan')`
`LIKE`	**Contains**	`city LIKE '%Mos%'`
`NOT LIKE`	**Does not contain**	`city NOT LIKE '%Mos%'`
`BETWEEN`	**In range (including boundaries)**	`amount BETWEEN 1000 AND 5000`
`NOT BETWEEN`	**Outside range**	`amount NOT BETWEEN 1000 AND 5000`
`EXISTS`	**Exists**	`EXISTS (SELECT 1 FROM orders WHERE user_id`
`NOT EXISTS`	**Does not exist**	`NOT EXISTS (SELECT 1 FROM orders WHERE use`
`ANY` / `SOME`	**Comparison with at least one value from subquery**	`amount > ANY (SELECT amount FROM orders)`
`ALL`	**Comparison with all values from subquery**	`amount > ALL (SELECT amount FROM orders)`

For example, we need to find active profiles from Moscow. In the segment, we select the operator `AND`, s query will be sent to the database:

```
SELECT email FROM orders WHERE status = 'active' AND city = 'Moscow'
```

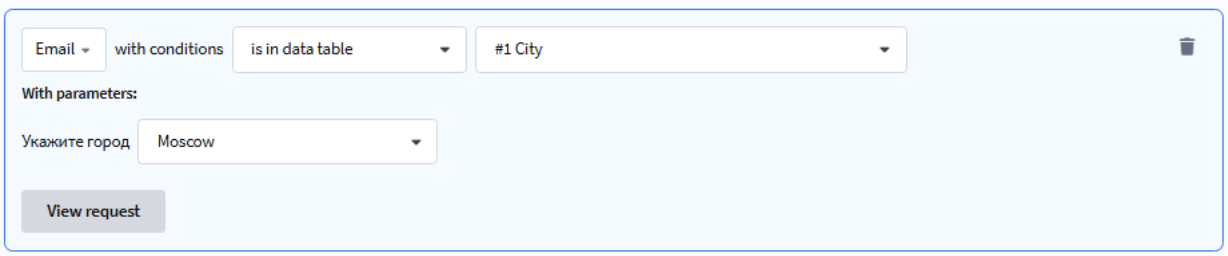
> INFO
>
> **All parameters, except “Comparison Operator” and “Logical Operator”, must be enclosed in single quote operators and logical operators are written without quotes.**

Using an external database query in a segment

Go to the Data — Segments section. Create a new segment or open an existing one for editing.
Add a new rule and select the identifier field or the field by which the profile is searched by the query. Th

- **Is in data table — if you want to include everyone who fits this condition.**
- **Is not in data table — if you want to include everyone who does not fit this condition.**

Then select your query to the table and fill in the query parameters form:



The conditions is in data table / is not in data table can be applied repeatedly in the segment, including i *AND* if you want each of them to satisfy the conditions, or with *OR* if you want any of them to satisfy the

Now you can save the segment.

Recommendations for segmentation by external data

Segmentation by external data sources is a powerful tool, but if configured incorrectly, it can significant following recommendations will help you avoid typical performance issues.

Minimizing data volume

External data sources should be designed to return as little data as possible. The fewer records the plat segment will be calculated.

Basic rules:

- **Return only the column with the profile identifier used for matching. Additional columns are unneces transferred data.**
- **Add basic filtering directly to the query or API URL whenever possible. For example, if you only need to the SQL query. This will reduce the amount of data the platform receives from the external databa calculation.**
- **For HTTP requests to APIs — the external service should return only a list of identifiers, not full obje**
- **For uploaded files — the file should contain only the needed column with identifiers.**
- **Use the result caching parameter when creating a SQL segmentation query. If data changes infreque up repeated segment calculations.**

Combining conditions in a single SQL query

When you need to filter profiles by multiple conditions from one external SQL database, it is better to wri conditions rather than creating multiple queries and joining them in the segment with "AND".

Each separate segmentation query is a separate call to the external database. The more queries, the lon

Example. Suppose you need to select clients who have an active contract and are in a specific region.

Inefficient approach — two separate segmentation queries:

1. **Query "Contracts":** `SELECT id FROM contracts WHERE status = 'active'`
2. **Query "Regions":** `SELECT id FROM customers WHERE region = 'Moscow'`

Then in the segment these two conditions are joined with "AND":

```
Is in data table (query "Contracts")
AND
Is in data table (query "Regions")
```

The platform will execute two separate queries to the database, get two lists of identifiers, and intersec

Efficient approach — one segmentation query with combined conditions:

```
SELECT customers.id
FROM contracts
JOIN customers ON customers.id = contracts.customer_id
WHERE contracts.status = 'active' AND customers.region = 'Moscow'
```

In the segment — one condition:

```
Is in data table (query "Contracts and regions")
```

The platform executes one query, and the intersection happens on the database side — this is faster.

Recommendations:

- **If conditions relate to one external database — combine them into a single SQL query using JOIN, su**
- **If conditions relate to different databases — it is impossible to combine them into one query, and the sequentially. In this case, try to make each query return as few records as possible.**
- **Use query parameters to create one universal query instead of several similar ones. For example, on better than separate queries for each region.**
- **Try to make all queries in the segment reference the same profile field (e.g.,** `customer_id`**). The platfor only if they are bound to the same profile field and use the same connector.**

Automatic query combination

The platform has an optimization that combines multiple segmentation queries to one external database executing each query separately, downloading results, and intersecting them inside the platform, the pla (intersection), `UNION` **(union), or** `EXCEPT` **(exclusion) operations — and executes it on the external database**

How it works. If the segment has multiple "Is in data table" conditions that use the same connector and the platform can combine them:

```
Is in data table (query "Contracts")
AND
Is in data table (query "Regions")
```

Instead of two separate queries, the platform will form one:

```
(SELECT id FROM query1) INTERSECT (SELECT id FROM query2)
```

If one of the conditions is "Is not in data table", the platform uses `EXCEPT`**:**

```
(SELECT id FROM query1) EXCEPT (SELECT id FROM query2)
```

INFO

The optimization is enabled by the platform administrator (parameter `SEGMENT_SQL_DATA_GROUP_OPTIMIZATION`). documentation.

Limitations:

- **Only queries from the same connector and the same profile field are combined.**
- **If there are other segment conditions (not from the external database) between queries, the combin construct like `Query AND (Query OR Condition)` cannot be combined because the non-query condition "bre**

Segment-inside-segment expansion

If a segment condition uses another segment (the "In segment" operator), the platform can expand the n external database queries in the general grouping for combination.

Example. The "VIP" segment contains the condition `Is in data table (query "VIP")`. The parent segment co

```
Is in data table (query "Active clients")
AND
In segment (segment "VIP")
```

Without expansion, the platform will execute the "Active clients" query separately, calculate the "VIP" se results.

With expansion, the platform will "see" the "VIP" query from the nested segment and combine both queri `INTERSECT`.

For expansion to work, all conditions must be met:

- **The "In segment" operator is used. If "Not in segment" is used, standard expansion will not work (se**
- **Expansion works only for dynamic or updatable segments, as they store segmentation conditions. Ex segments — they contain only a ready-made list of profiles.**
- **Queries of the nested and parent segments use the same connector and are bound to the same profi**
- **Group condition operators match — if the parent uses "AND", the nested must also use "AND" (simila**

INFO

Segment-inside-segment expansion is enabled by the platform administrator (parameters `SEGMENT_SQL_SIS` More details in the administrator documentation.

"Not in segment" expansion

A separate optimization is implemented for conditions with the "Not in segment" operator. It allows expa transforming the negation: each operator inside the nested segment is replaced with the opposite one (" "does not contain", "greater than" → "less than or equal", etc.), after which the queries can be combined

For expansion to work, all conditions must be met:

- **The condition uses the "Not in segment" operator.**
- **The nested segment is not static — dynamic or updatable.**
- **Queries of the nested and parent segments use the same connector and are bound to the same profi**
- **All operators in the nested segment have an inversion pair (e.g., "equal" ↔ "not equal", "contains" ↔ operator has no pair — expansion will not work.**

Operators without inversion

Subscription status: "Unsubscribed", "Hardbounced", "Complainer", "Invalid", "Suspended", "Not confirm

Automation scenarios: "was at least once", "was never", "currently in scenario", "currently not in scenar

Forms: "filled", "did not fill", "filled during selected period", "filled within the last [x] days relative to curr

Dates: "same as today".

Channel activity: "At least [n] times", "Not a single time".

Campaigns and LCG: "not in LCG".

Loyalty and orders: "in range".

Relations: "Presence / absence of direct relation", "Presence / absence of reverse relation".

INFO

"Not in segment" expansion is enabled by the platform administrator (parameters `SEGMENT_SQL_SIS_NOT_EXPA` details in the administrator documentation.

"NOT" conditions on external data marts

Segmentation conditions on external data marts with the “NOT” operator (e.g., “Not in data table”) lead t
database. This happens because the system needs to check each profile for absence in the external sele

Recommendations:

- **Minimize the number of “NOT” conditions on external data.**
- **Whenever possible, replace negative conditions with positive ones. For example, instead of “Not in d**
  an inverted query on the external database side.
- **If a “NOT” condition is necessary, make sure the external data mart returns the most compact resul**

For more on the “NOT” operator in segments, see Recommendations for working with segments.

Message templates

Description

Message template is a form for creating content for your marketing mailings. Within a single template, y different communication channels. Templates in the platform support several types of dynamic content, websites and SQL databases, as well as data passed in API requests to launch mailings. The message e personalization variables, functions, logical expressions and cycles to your template. You can also custo for each specific group of subscribers.

You can attach files to the template to send to subscribers and stop lists to block email addresses. Tem to confirm subscription or unsubscribe.

Where to start?

![img] (/img/attachments/3114224/9408085.png)	Managing message templates Creating and editing message templates . Content editors for Email, Push and SMS versions. General and advanced settings. Pers
![img] (/img/attachments/3114224/9408086.png)	Templates fragments Fragments are parts of message templates, which are stored separately and can be used for sections like header, footer, social li may contain plain text or HTML code for email.
![img] (/img/attachments/3114224/9408087.png)	Variables and template functions You can use variables, logical expressions, loops and functions in the templates. In this section you can find description of the sy
![img] (/img/attachments/66503/76808521.png)	Dynamic content in messages You can insert dynamic content into your emails, such as HTML, APIs, JSON content, and data from external SQL tables. You can re
![img] (/img/attachments/76808516/76808521.jpg)	Importing a template from a third-party service If you already have a message template that you're used to, you can easily transfer it to the Altcraft platform and use it in your n guide.

Managing message templates

Description

![img] (/img/attachments/3114225/9407904.png)	**Message template is a set of content for your marketing mailings. One template combines content settings for all communication creating content for your mailings.**

Managing templates

All the Message templates available in your account or your access group are displayed in the list. You c ID, or group templates by tags. You can also display only drafts or ready-to-send templates. Click on the preview of the message. Here you can check how the message looks in different channels and on differe

> **TIP**
>
> **If you use Double opt-in — double subscription confirmation — such templates will have Confirmation ta more about Double opt-in subscriptions [in this article.]**

For every template, quick actions menu is available, you can use it to edit, clone, remove template - or se button at the top part of the page to add a new template.

For each template, general information is collected. Whether there is a:

- **Confirmation link: a template with such a link is required when using a [Double opt-in] subscription.**
- **External content: the use of links to [dynamic JSON content] in the template.**
- **Suppression list when unsubscribing: if the subscriber clicks on the unsubscribe link in the message selected [suppression list]. A mailing that uses the template will block sending an email to a subscrib**

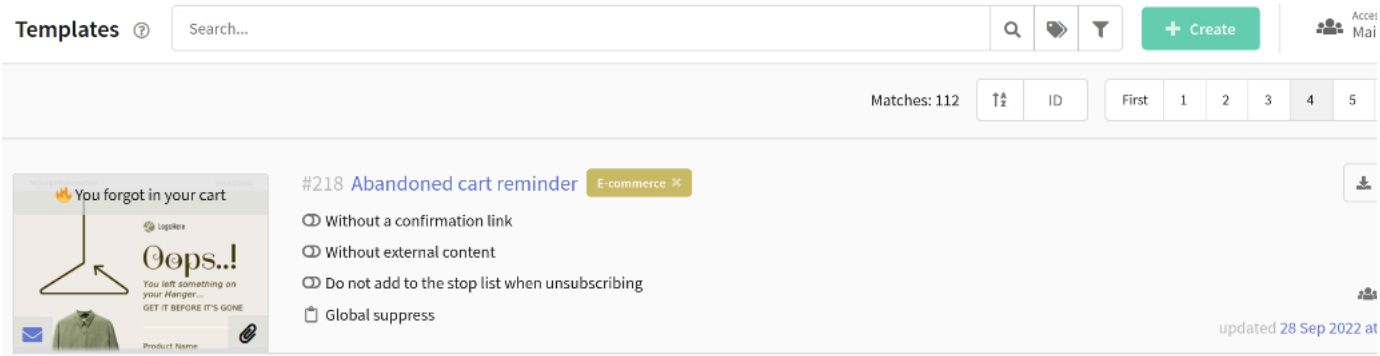
On the right there is a menu of quick actions:

— go to the summary report by template.

— clone the template.

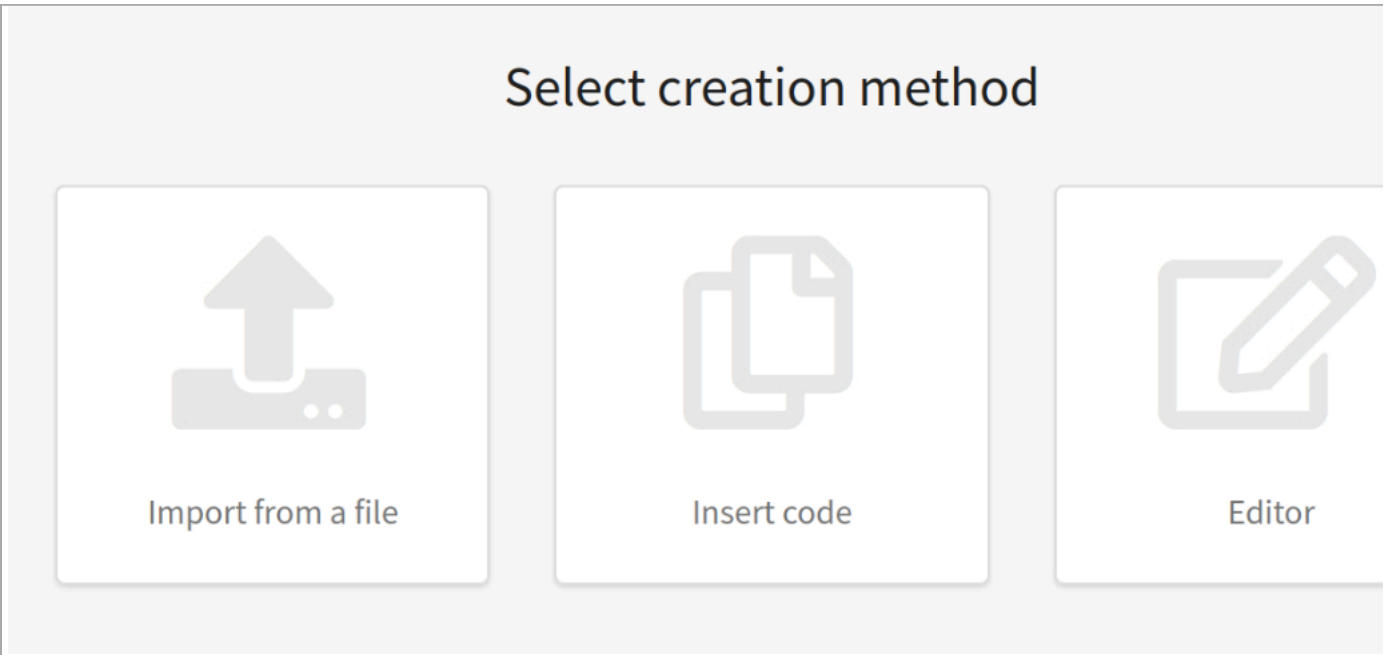
— delete the template.

To add a new template, at the top of the page, click the + Create button. If you need to edit an existing t



Next, you need to choose how to create a template:

- **Import from a file: download a ready-made template as a zip archive or html file.**
- **Insert code: Adds the copied html code to the text box.**
- **Editor: create a template in the editor.**

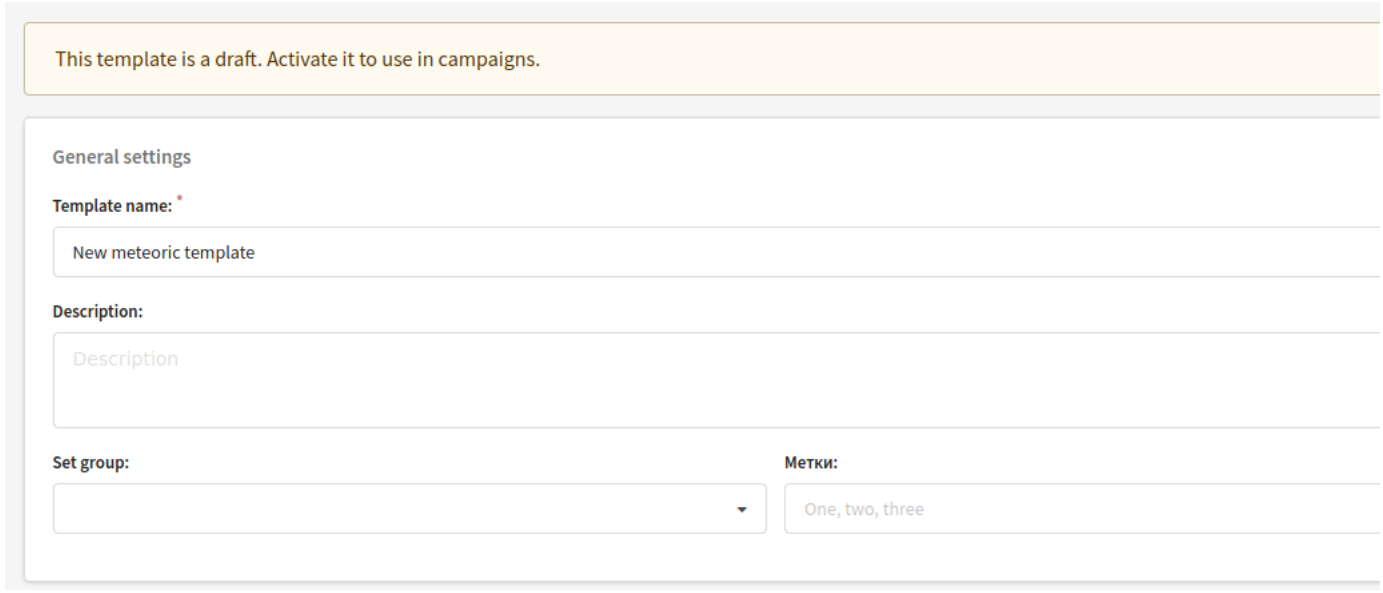


TIP

If you need to move a template from a third-party service, use import from a file or paste code.

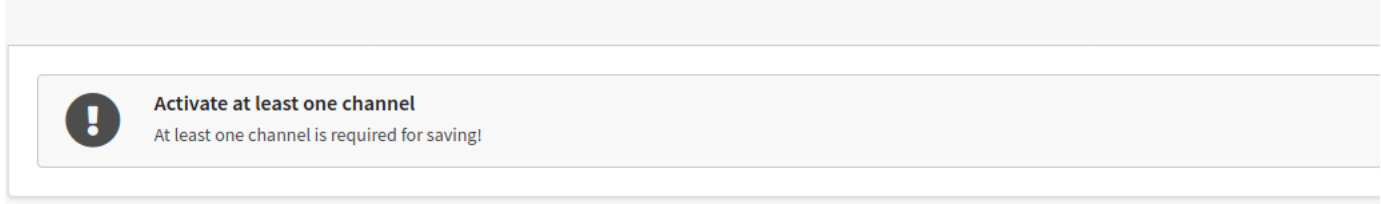
General settings

In general settings set template name and description, add Tags and, if necessary, set access groups.
If you want to organize content moderation, you can mark template as a Draft. Thus junior and outsource and their senior colleagues can then verify it by lifting the Draft restrictions.



Content editors

Message template combines message editors for all available communication channels. It will help you c messages, that make content personalization easier. Press button with “plus” sign to add content for the



Content editor for the selected communication channel will open in a new tab:

Email

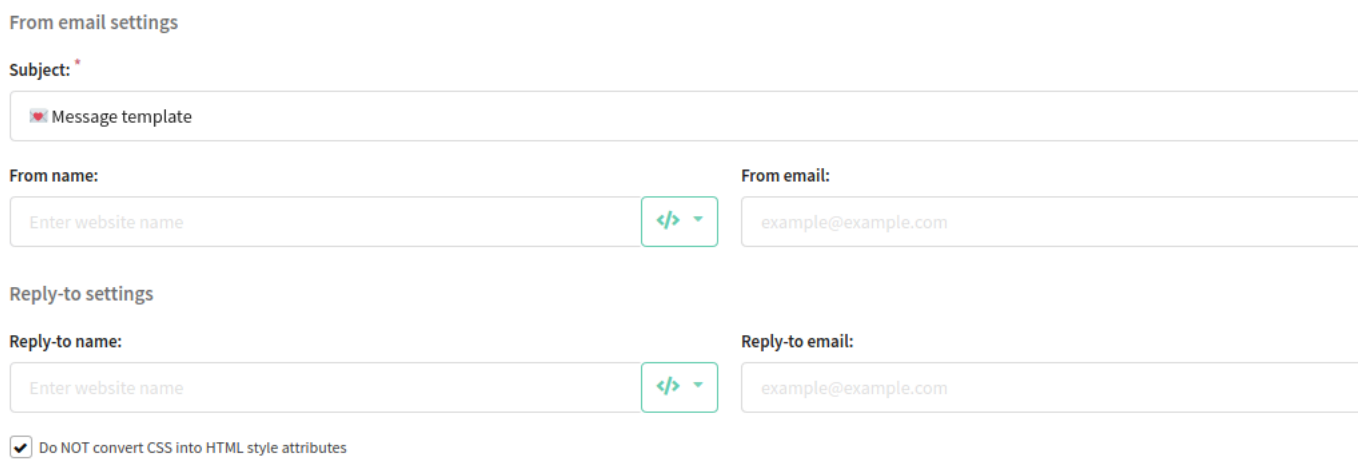
For email channel fill in the following fields:

- **Subject — letter subject, that will be displayed to the customer. You can use personalization variable**
- **From name — your company, department or mailing name. Here you can also add personalization and**
- **From Email — your sender domain address, from which letter will be sent to the customer.**
- **Reply-to name — name for replies to the message. Personalization is available: for example, you can name.**
- **Reply-to email — Email address for replies.**

By default, Altcraft does not convert styles, included into HTML version of the message, into line tags (D attributes). But, if you remove this option, styles will be added as attributes to message elements:

```
<h1 style="font-size:36px;">Hello!</h1>
<h2 style="font-size:28px;">Nice day, isn't it?</h2>
```

This will provide better compatibility with Outlook and other mail clients, that do not support `<style\>` **ta**



CAUTION

From name and From Email fields overwrite values, configured for Email channel both in Resource and F

HTML version editor

Most people see HTML version of the email in their inboxes. Editor supports HTML, CSS and personalizat available to make formatting and work with objects easier:

Formatting:

B

— bold

I

— italic

~~S~~

— ~~strikethrough~~

— bullet list

— numbered list

— table

Service functions:

— pixels for tracking of letter opening and reading. If you do not add pixel during message editing, the sy

— external HTML content. You can also set variable source address manually `{[http://example.com]}`**.**

— links builder (see more on the email editor tab).

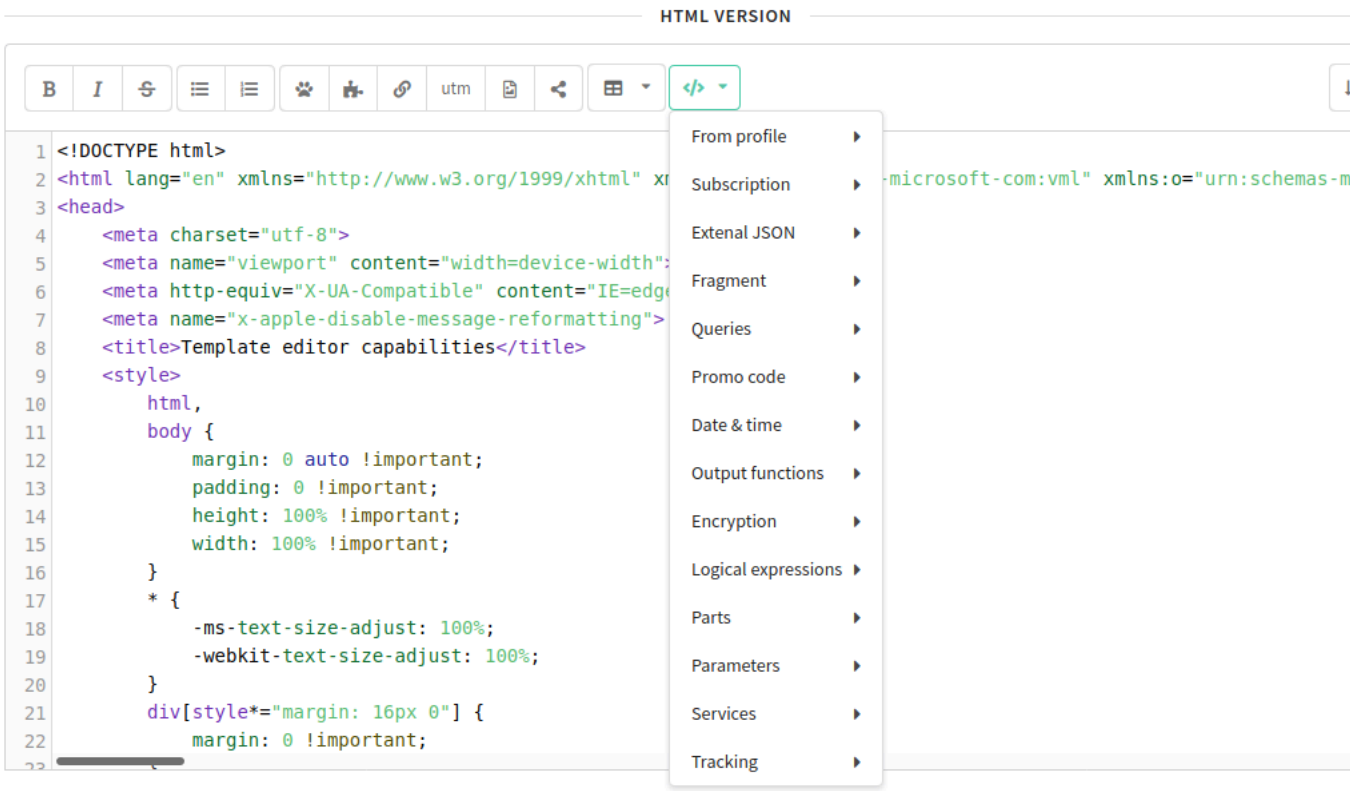
utm

— UTM tags.

— images.

— social buttons.

— variables and editor functions menu. Using this button, you can add Customer profiles' data, promo co dynamic content and service parameters into the message.



TIP

You can also toggle word wrapping or enable fullscreen mode.

Link builder

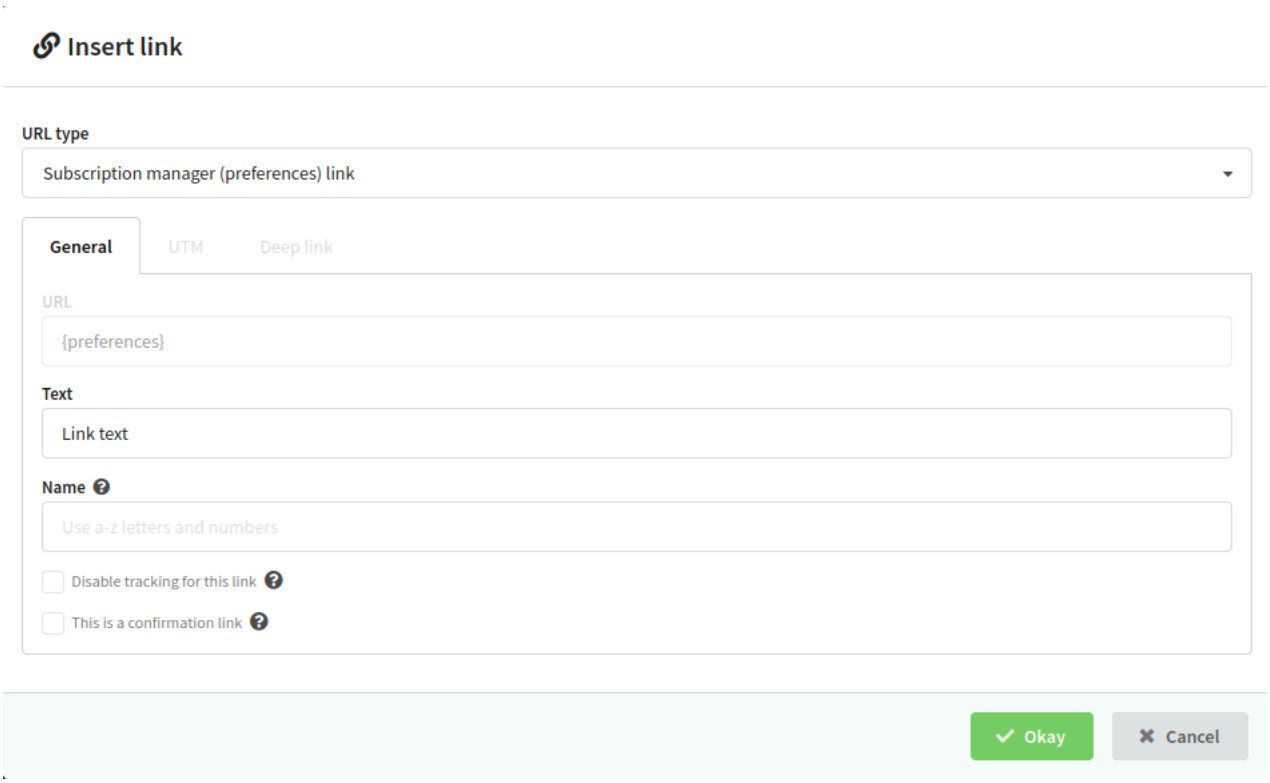
Link builder

Press Insert link button in HTML or plain text version editor to insert a new link. If you want to edit an ex the link editor button:

You can choose URL type depending on its purpose:

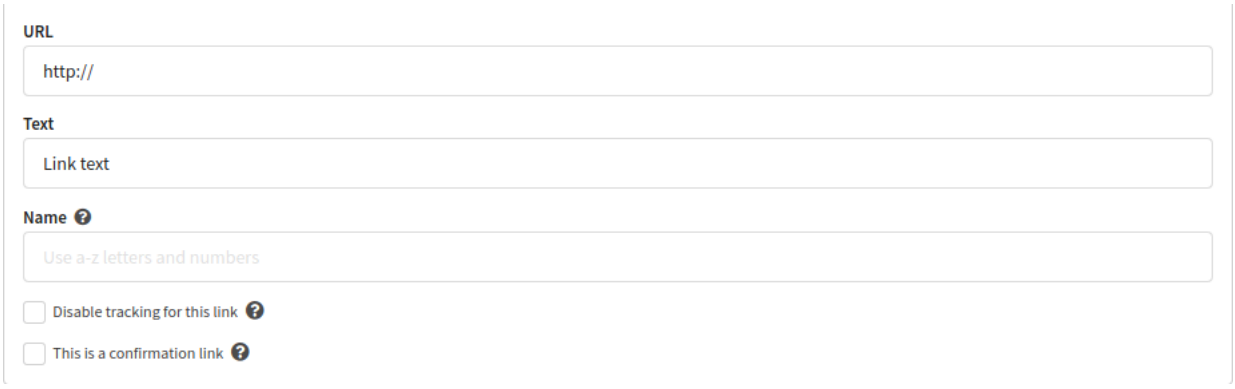
- **Enter a custom link — link to any page, except unsubscription and confirmation links.**
- **Unsubscription link — link to a system or custom link to stop resource subscription. Subscription go affect global status profile and other subscriptions.**

- **Subscription manager (preferences) link — link to the system page of subscription management. Cus resubscribe or unsubscribe to mailings.**
- **Suppression link — link for adding customer into suppression list. Using this link, the customer can b**
- **Global unsubscription link — changes global profile status to Unsubscribed, and completely stops an**
- **Web version link — link for a web browser copy of the letter, that was sent to the subscriber. If the cu showing the letter, it would be available via this link in any browser.**



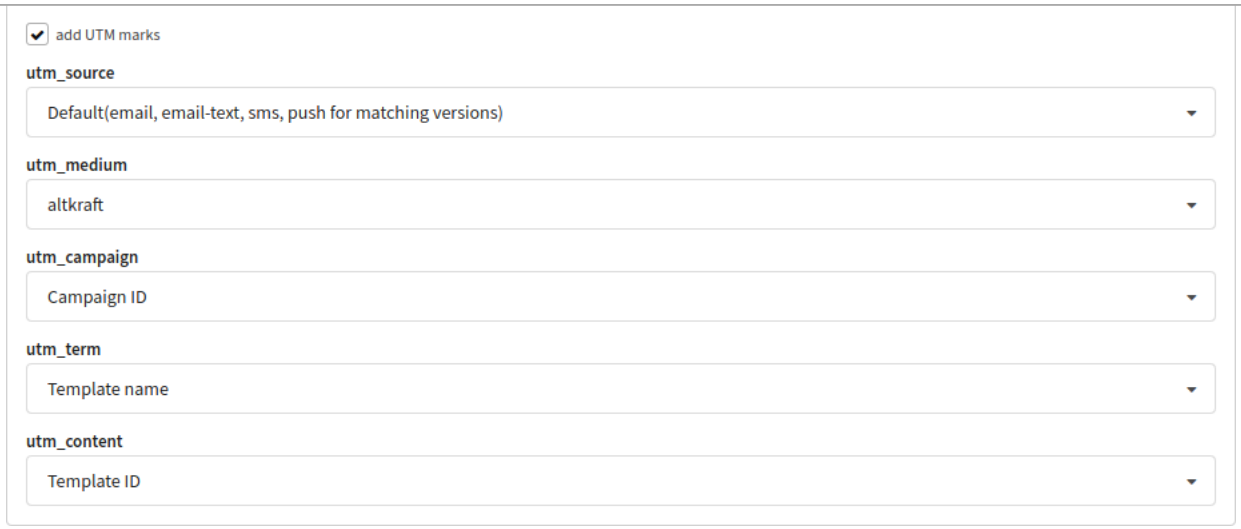
If you want to add your link, enter URL in general settings. This field will be pre-filled for system links. Al and Link name — it is used in reports, segmentation and during triggers and scenarios launch.

Also, you can disable redirection to tracking module when clicking on the link, or you can make any link

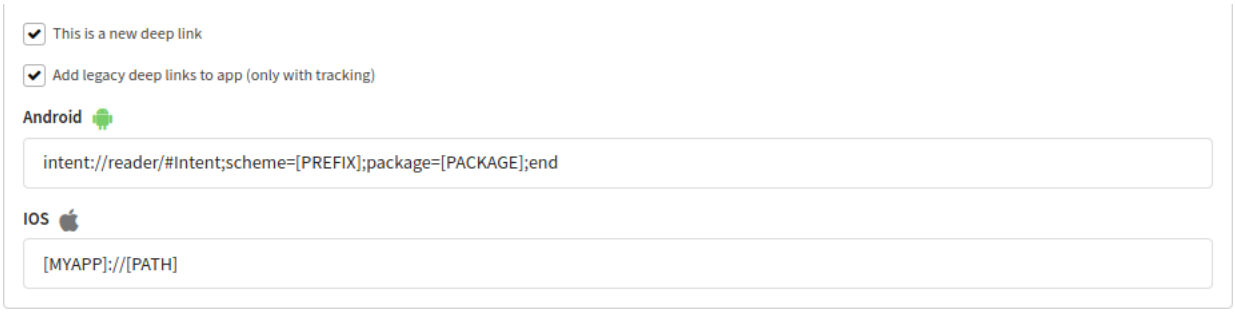


You can set UTM tags for every specific link in the message. You can use a button in content editor to se

You can use variables, like mailing name or database name, as UTM tags.



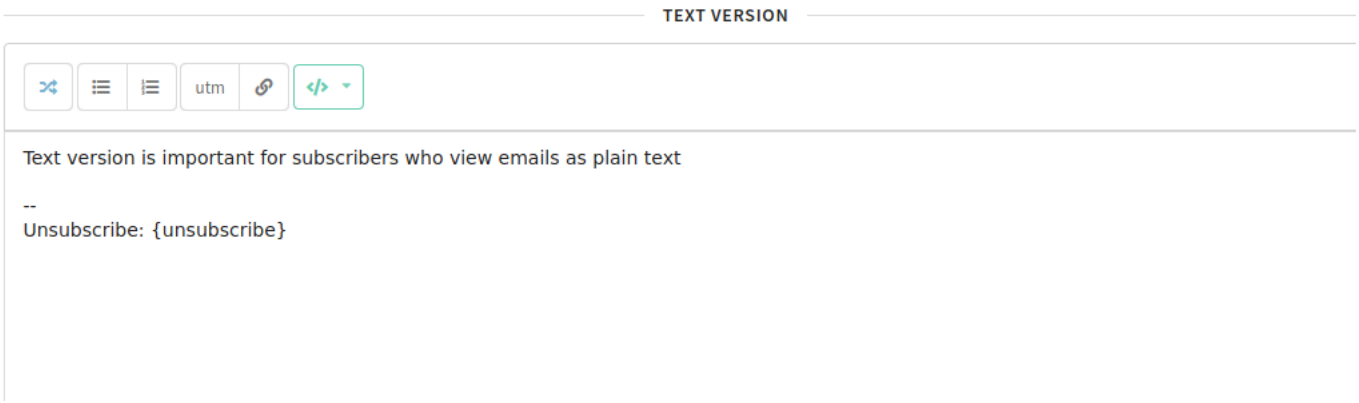
If you want to use the link for redirecting to mobile application, you need to activate This is a new deep l
Legacy deep link can be added only if you use tracking.



Plain text version

Message text version is shown in email instead of HTML, if email provider — or email client — blocks HT

TIP
Messages with text version have better delivery rate. Make sure you fill it in for the messages.



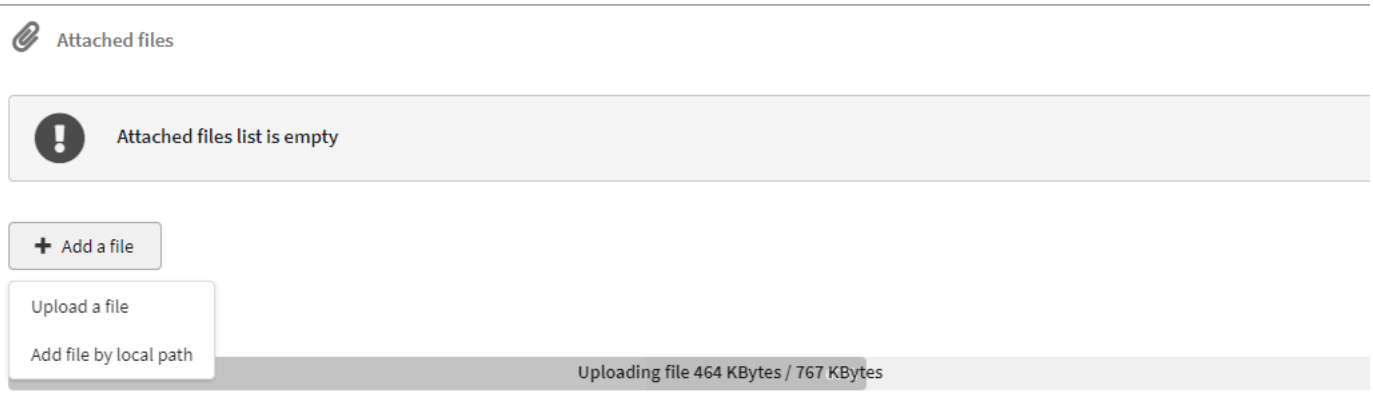
You can use

button to generate text version based on HTML. Text version also supports lists, UTM tags, links and var

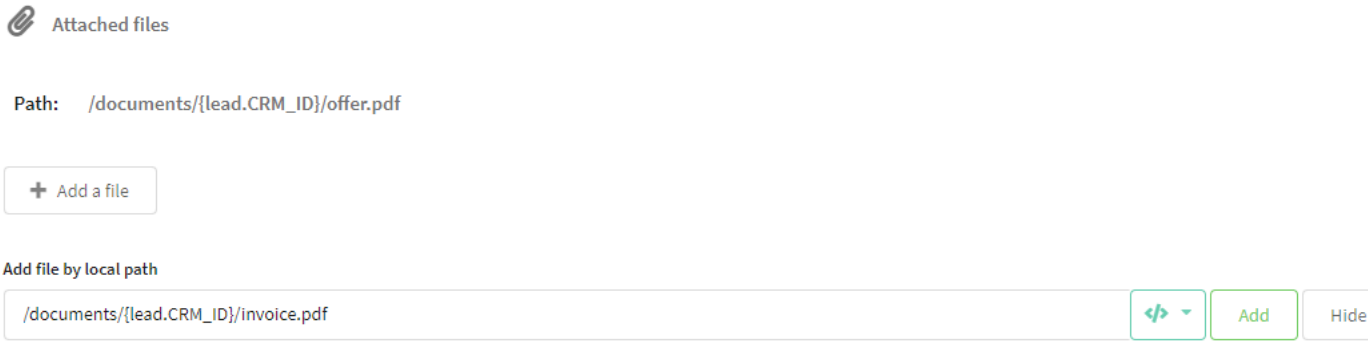
Email attachments

To attach files to your email message use the section in the lower part of the form. There are 2 ways:

- **you can upload a file from your computer file system:**



- **or specify a local (server, where the platform is installed) path to the file. Here you can use profile va file path and then press Add:**



To delete an uploaded file or a saved file path press an X in the right part of the form.

SMS

Enter SMS text into the text field. The editor will automatically count the number of symbols and messag variables in the message body:

</>

— variables and functions menu. Using this button, you can add customer profiles data, promo codes, loc content and system links into the message.

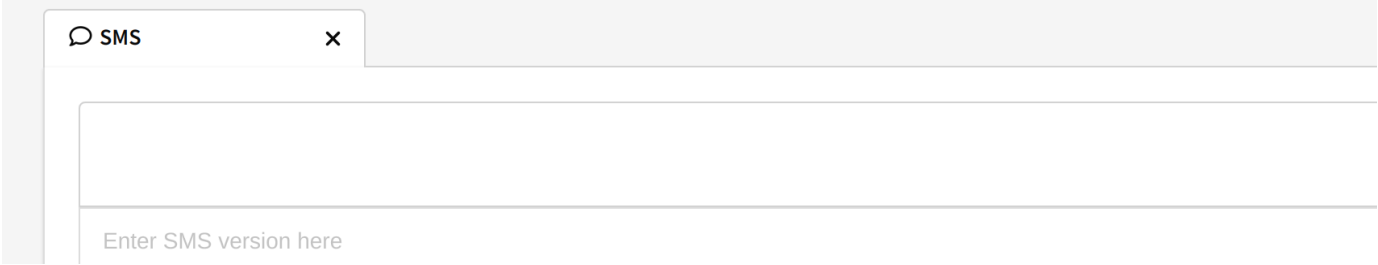
utm

— UTM tags settings.

— links builder.

Link builder

Press Insert link button to insert a new link. If you want to edit an existing link — select it and then pres



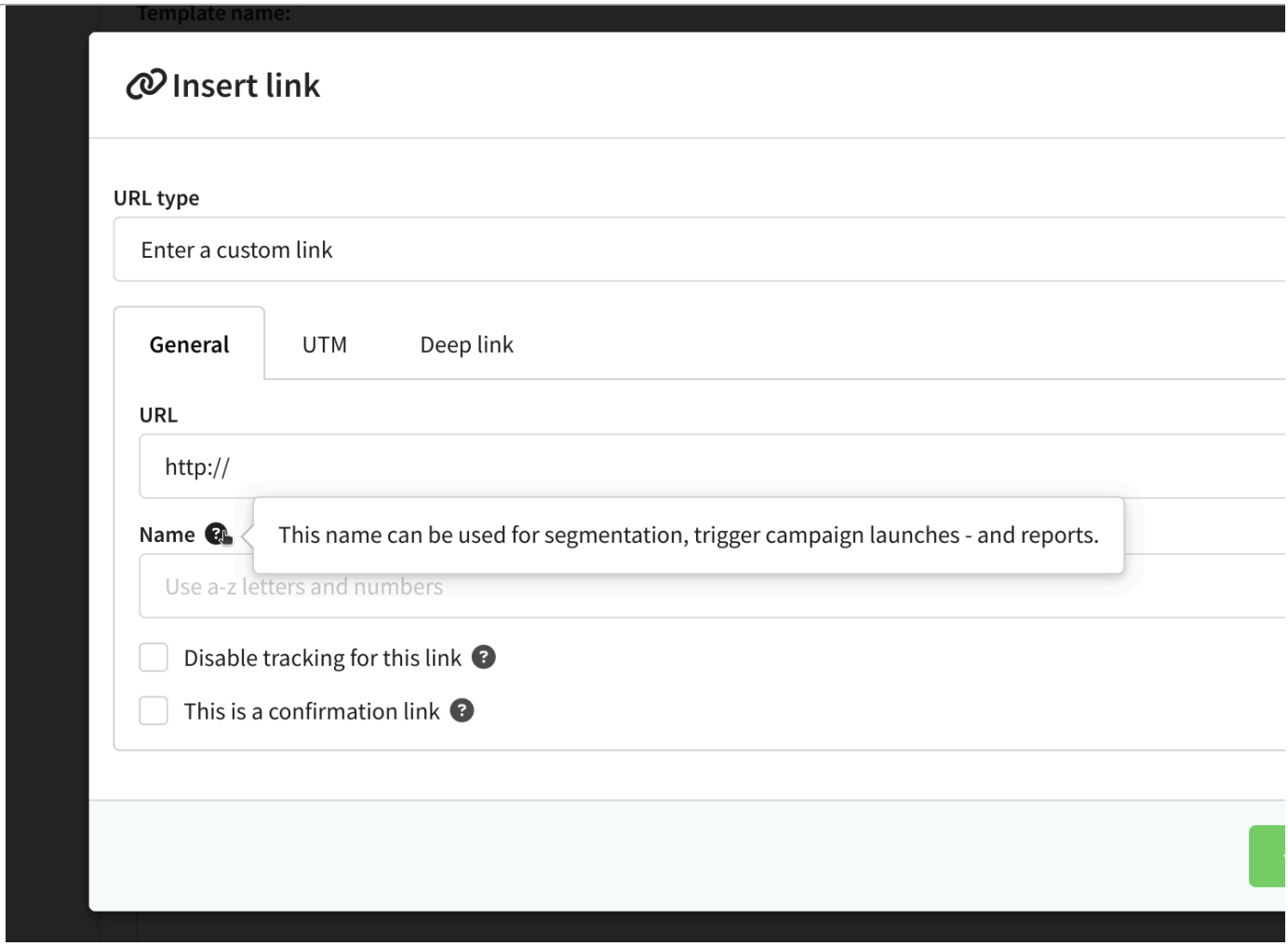
There is only one URL type for SMS. Press Enter a custom link to create a link to any page, except unsub

To add your link, enter URL in general settings. Also, enter Link name — it is used in reports, segmentati launch.

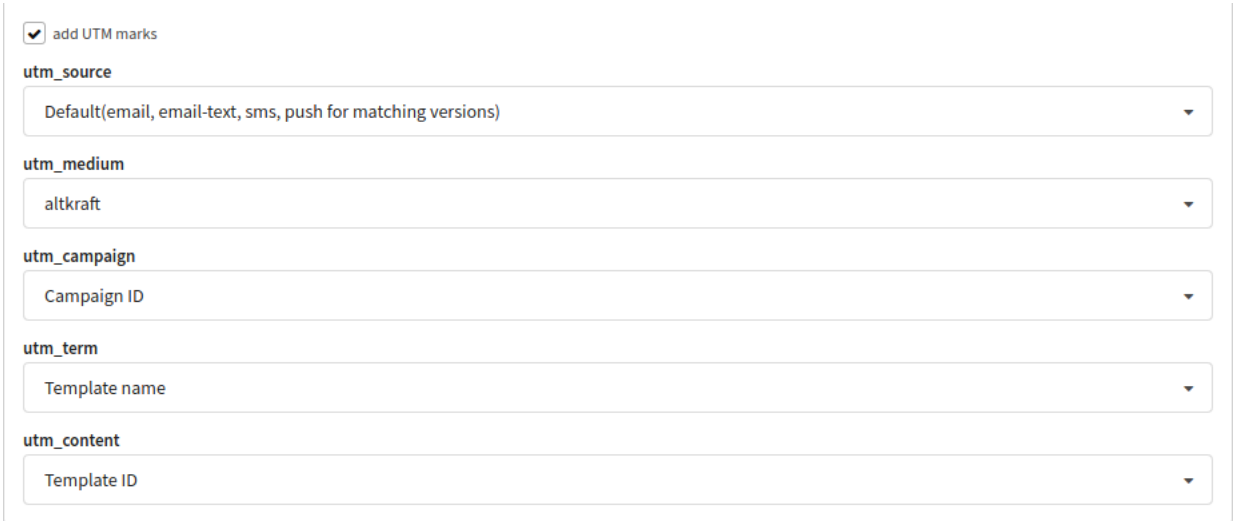
CAUTION

For further usage in scenarios and triggers, you must use the Link Name in the corresponding field, not

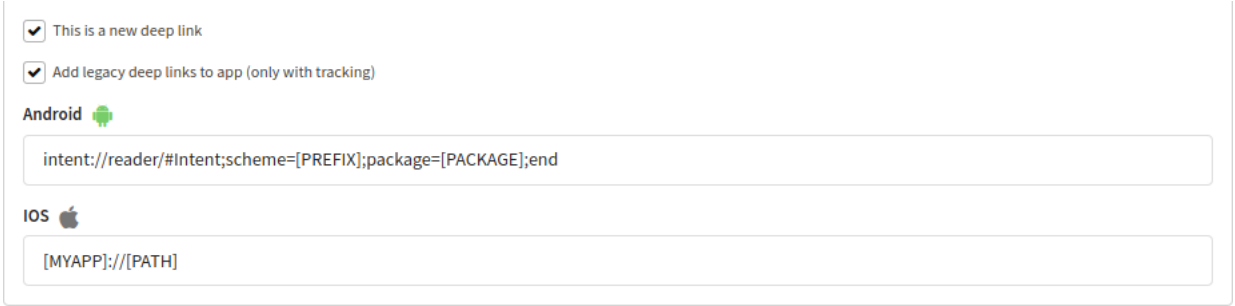
Also, you can disable redirection to tracking module when clicking on the link, or you can make any link



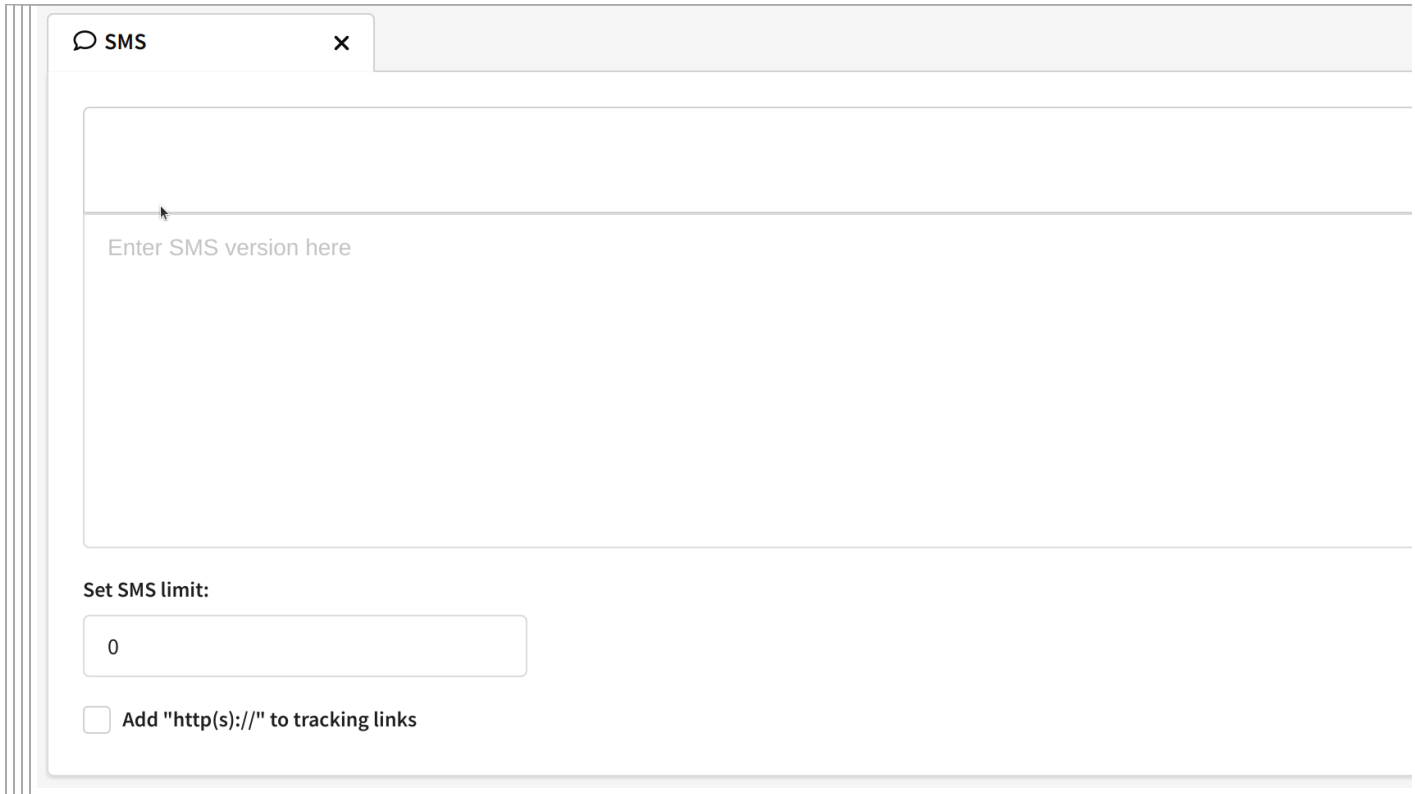
You can set UTM tags for every specific link in the message. You can use a button in content editor to se
You can use variables, like mailing name or database name, as UTM tags.



If you want to use the link for redirecting to mobile application, you need to activate This is a new deep l
Legacy deep link can be added only if you use tracking.



Also, you can limit the amount of SMS sended and add “http://” into the URLs:



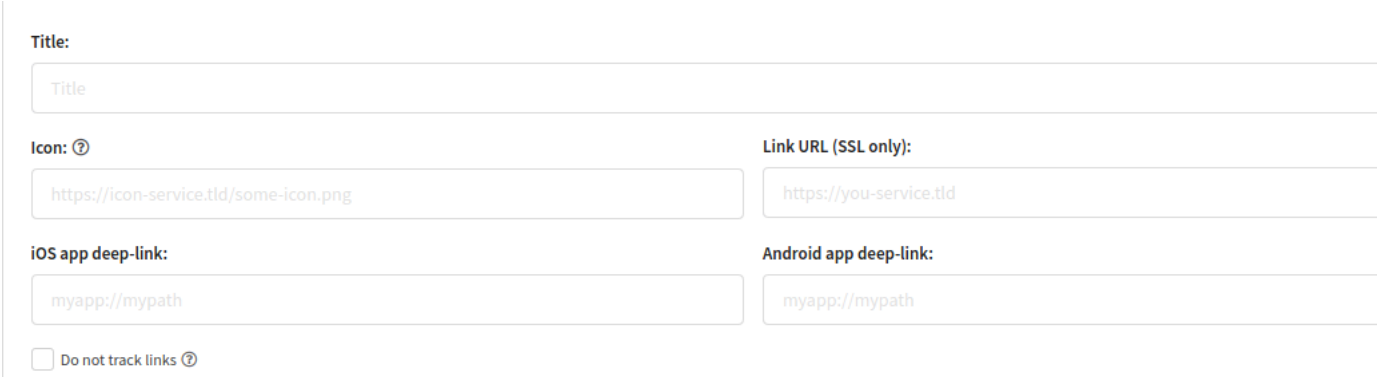
CAUTION

In SMS messages, angle brackets are used to denote parameters (for example, the name parameter wo in angle brackets will not be shown in the message.

Push

For push notifications, you can configure:

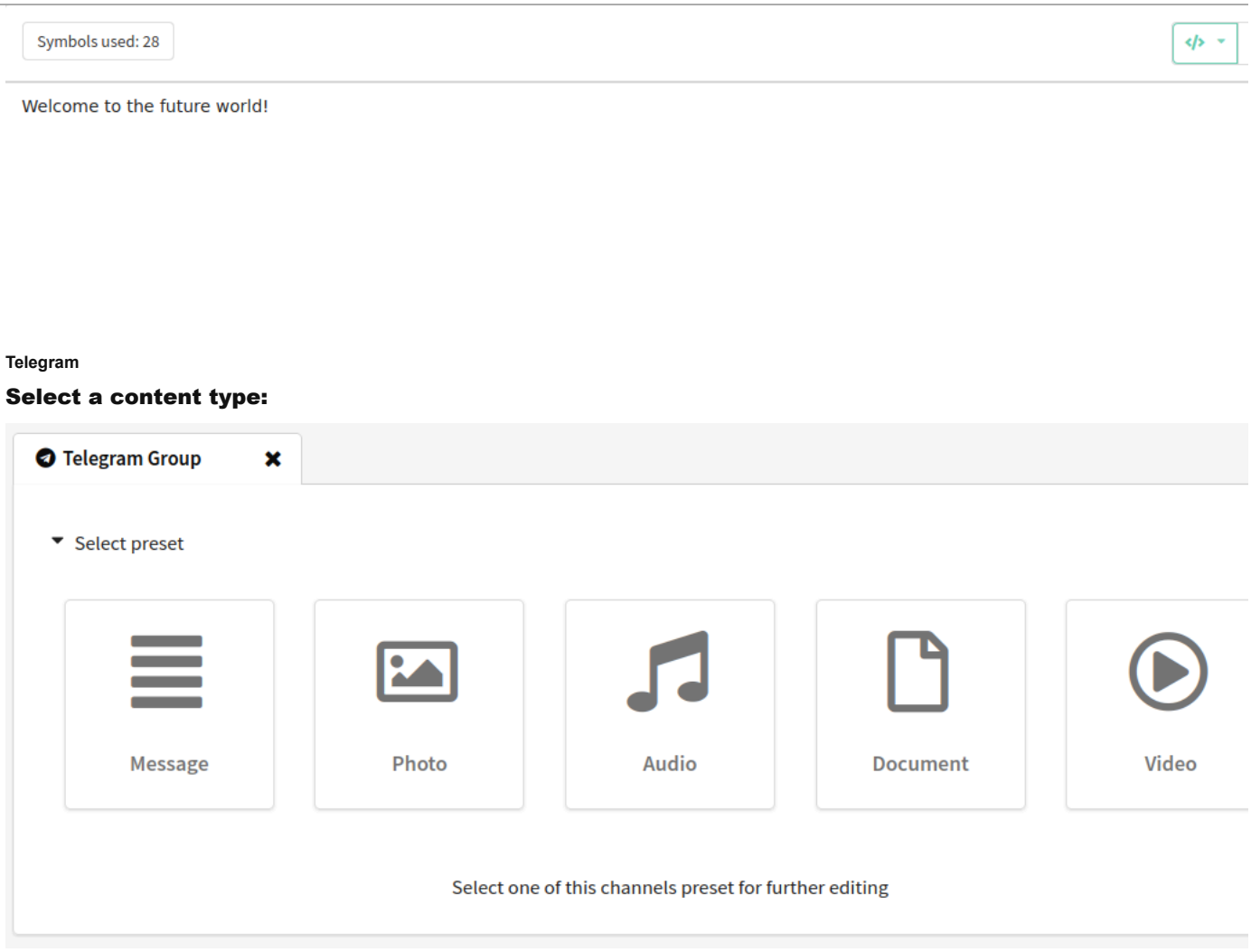
- **Title — variables menu is available for the title of your notifications.**
- **Link URL — link to follow when a customer clicks your push. You can use only encrypted protocol (S available: for example, you can insert promotion codes. When customers clicks on this link, he will b discount calculation.**
- **Icon — Image for display in the notification. Push notifications in Apple Safari support only one imag in Push sending resource.**
- **iOS app deep link — link for iOS applications.**
- **Android app deep link — link for Android applications.**



Enter push message text into the text field. The editor will automatically count the amount of symbols. Y body. Also, you can set UTM tags for the link in the message.

— UTM tags settings.

— variables and functions menu. Using this button, you can add customer profiles data, promo codes, loc content and system links into the message.

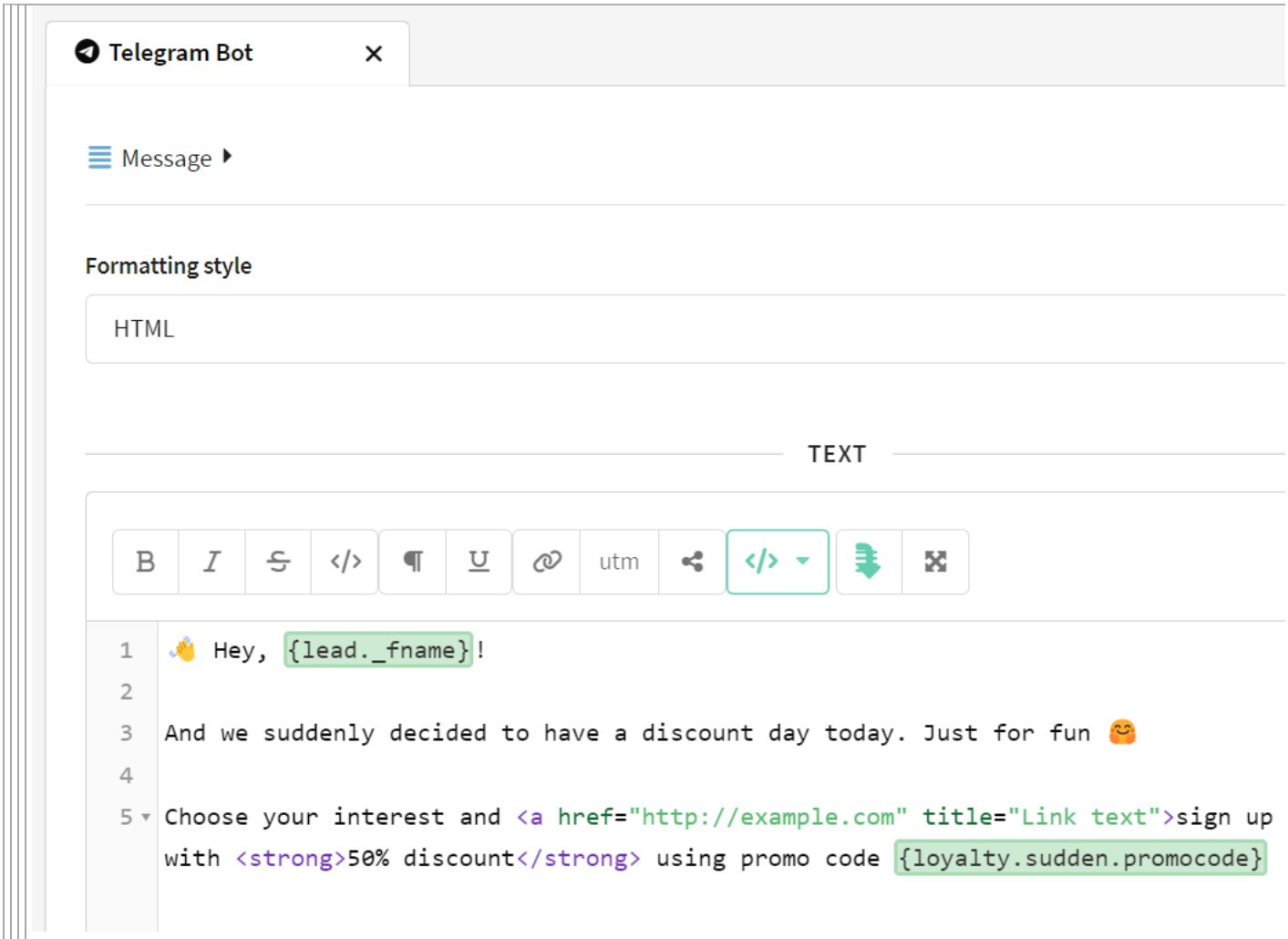


Message

Select a formatting style. For fine-tuning conten it's recommended to use HTML. Enter the text of your m

TIP

The maximum length of one message is 4096 symbols. You can also add dynamic variables to the mess pulled from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend monito



If you need to add a button to your message, enter a name and specify the link that the user will follow v add only two buttons.

Photo

To create engaging content, you can add an image—click the "Upload image" button and choose an imag images in Telegram:

- **Formats: PNG, JPG, JPEG, GIF.**
- **Maximum size: 10 MB.**
- **The sum of the width and height of the photo should not exceed 10,000.**
- **The aspect ratio of width to height should be no more than 20.**

You can also add a description to the image (up to 1024 characters).

Audio, Video, Document

You can add files in two ways:

- **Upload a file from your device.**
- **Add a file via the local path. Choose the directory where the file is stored on the server where the pl path, you can use client profile variables. Enter the file location and click "Add."**

Requirements for audio files:

- **Formats: MP3 and M4A.**
- **Maximum size: 5 MB.**

Requirements for documents:

- **Formats: Any document format.**
- **Maximum size: 5 MB.**

Requirements for videos:

- **Formats: MP4.**
- **Maximum size: 5 MB.**

You can also add a description to files of any type (up to 1024 characters).

> TIP
>
> **All links in templates for Telegram Bot and Telegram Group are shortened.**

WhatsApp

WhatsApp* uses 2 types of templates to communicate with users:

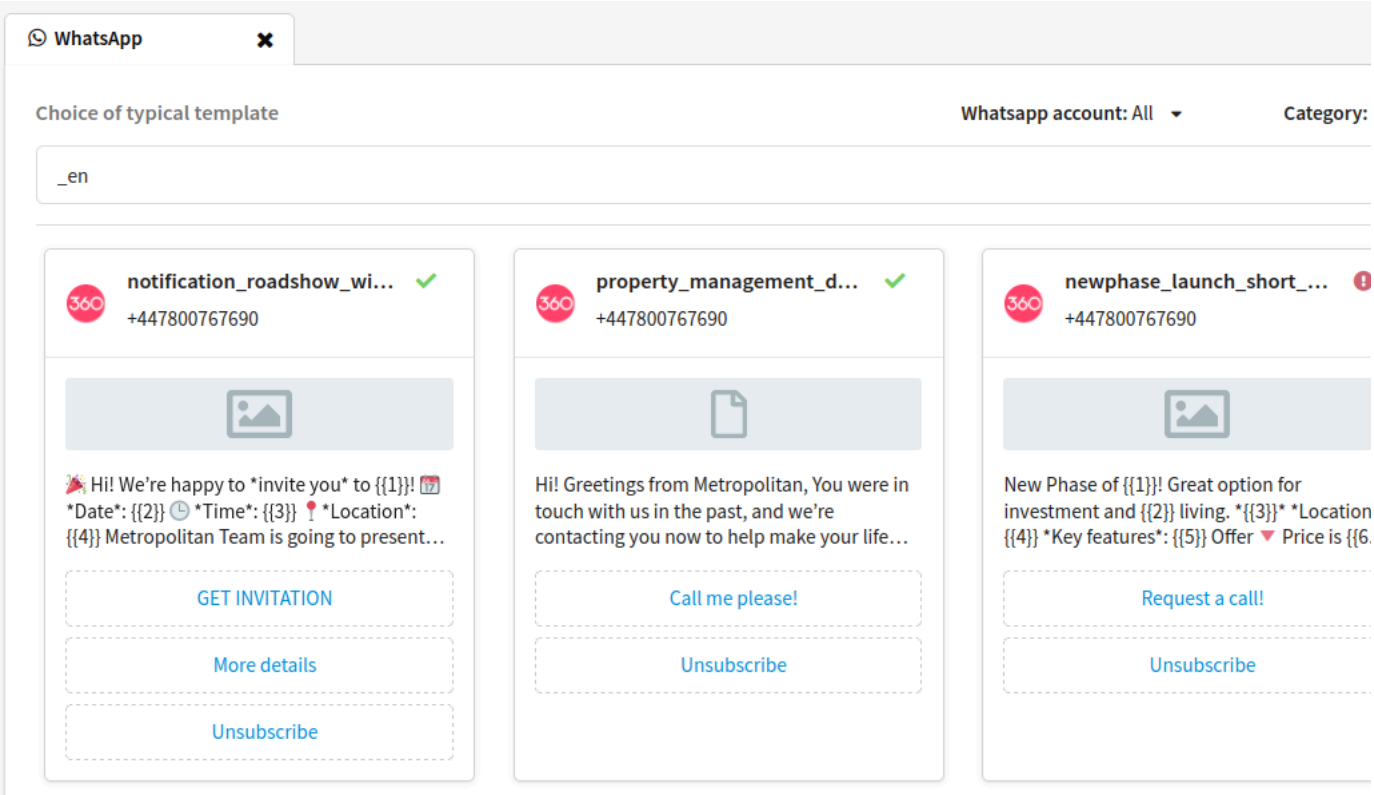
- **Typical template is an HSM (Highly Structured Message) template pre-approved by Facebook*. This t business-initiated conversation. You have to use such templates to initiate a conversation with an en period of 24 hours since the last interaction.**
- **Custom template is a free-form message. This type of template is required to start a user-initiated c where a user messages a business. In this case, a business can reply to a user within the 24 hour cu**

> CAUTION
>
> **Currently, only typical templates (HSM) can be sent via Altcraft Platform.**

You must first create message templates in your personal provider account.

> TIP
>
> **You can find more information about creating templates in your provider's documentation:**
>
> - **Edna: https://docs-pulse.edna.ru/docs/template/whatsapp/operator-template**
> - **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

All templates available in connected WhatsApp* accounts through edna or 360dialog are automatically a displayed in the editor. To search for the desired template in the platform, you can enter a query in the s account/category filter in the upper right corner.



> CAUTION

Mailings can only use templates approved by Facebook*.

The template status is displayed to the right of its name:

— template approved;

— template rejected;

— the template is being moderated;

— the template has not been sent for moderation.

Select the template you want to use in the mailing. To start creating a message based on a WhatsApp* t display the template structure:

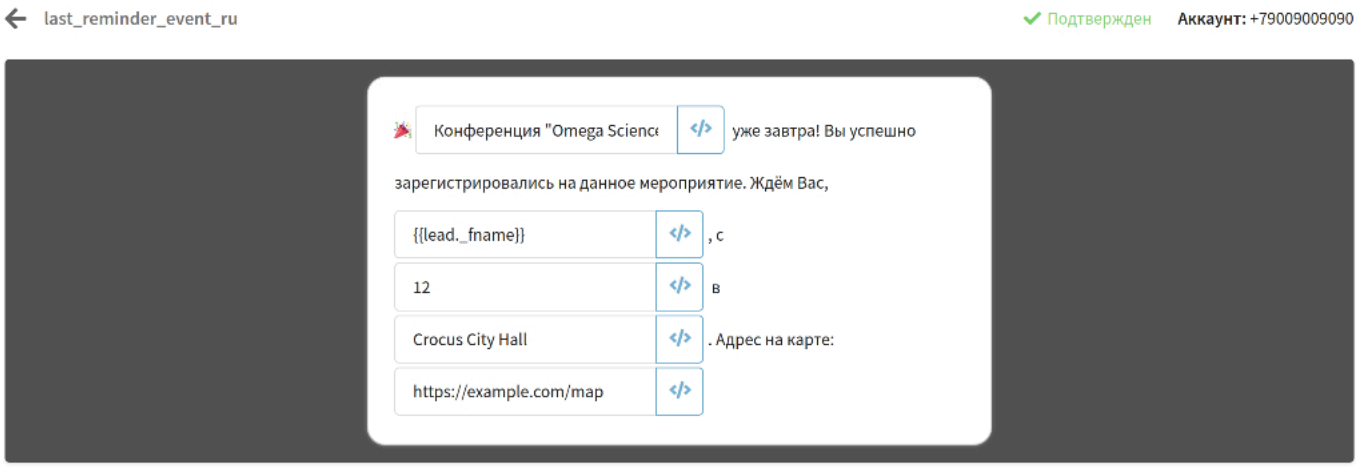
- **Header — the block can contain a welcome text or a media file;**
- **Body — the main text of the message with dynamic variables (required block);**
- **Footer — additional text;**
- **Button — block with buttons to add interactivity.**

You can edit template elements such as variables, media files, and buttons.

Variables

A text block consists of fixed text and variables. Variables are template text that changes for each new name, date and time, link, etc.).

You need to manually enter variable values or add Altcraft variables. To open the list of variables, press entered text will be automatically added to the template.



TIP

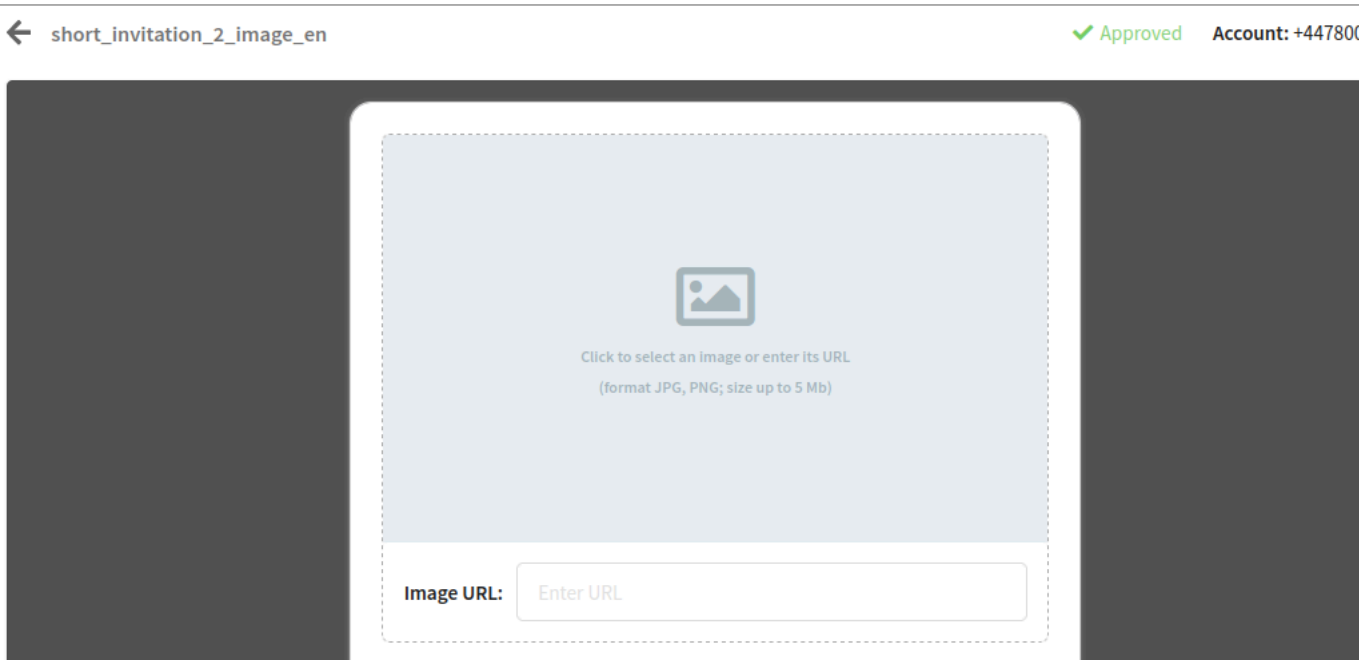
Links embedded in WhatsApp* variables are shortened.

Media files

Available media files:

- **image (JPG/PNG, up to 5 MB, recommended image size 640x640)**
- **file (any valid MIME type, up to 16 MB)**
- **video (MP4, 3GPP, up to 100 MB)**

To add an image, click on the upload area and select an image from the platform gallery or enter the URI



If you need to add a file or video, click on

and select the file on your device or enter an URL.

Buttons

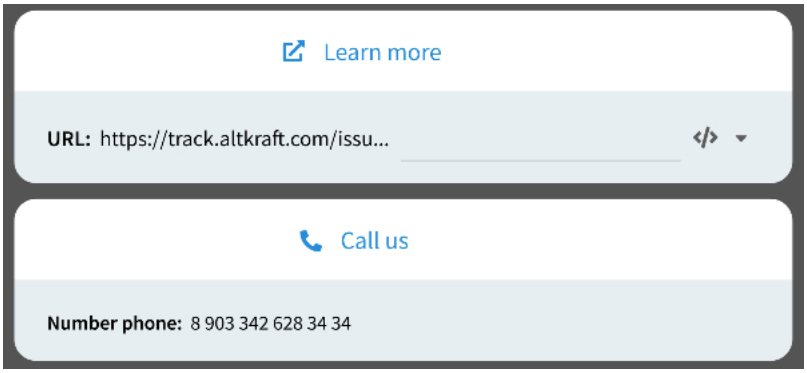
WhatsApp* has two types of buttons for HSM templates: Call-to-action and Quick-reply.

Call-to-action button

The CTA can be a URL button or a Phone button. There can be only two buttons of this type in the templ

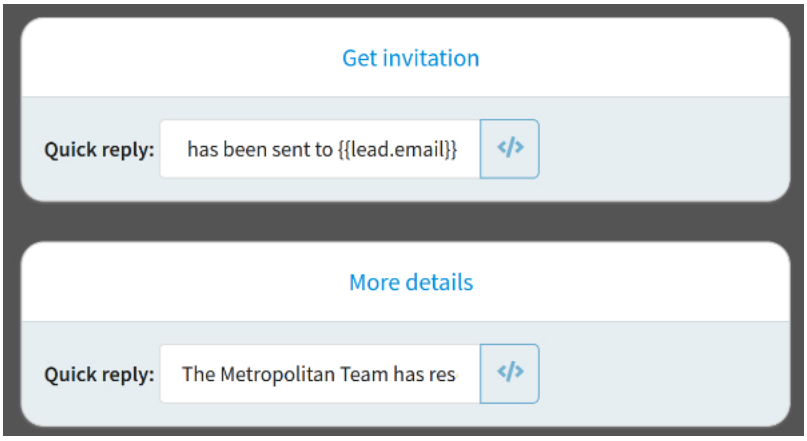
When clicking on the URL button, the user goes to the specified page. If a parameterizable URL is specif must be filled in.

When a user clicks on the Phone button, the number is dialed. The phone number cannot be edited.



Quick-reply button

When clicking on such a button, the user receives a quick response specified in the template. A templat of this type.

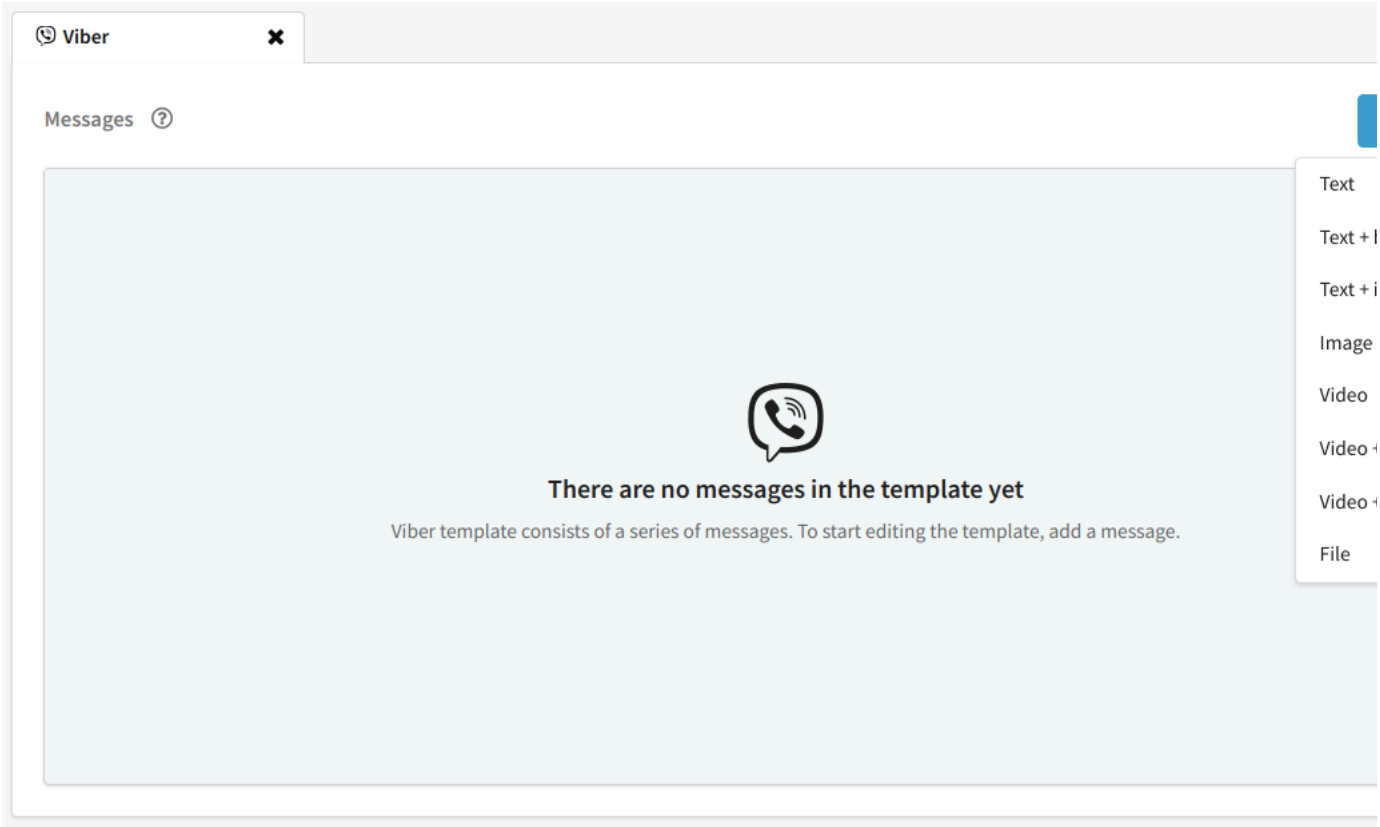


TIP

It is possible to save the template only after filling in all variables, buttons, and adding a media file.

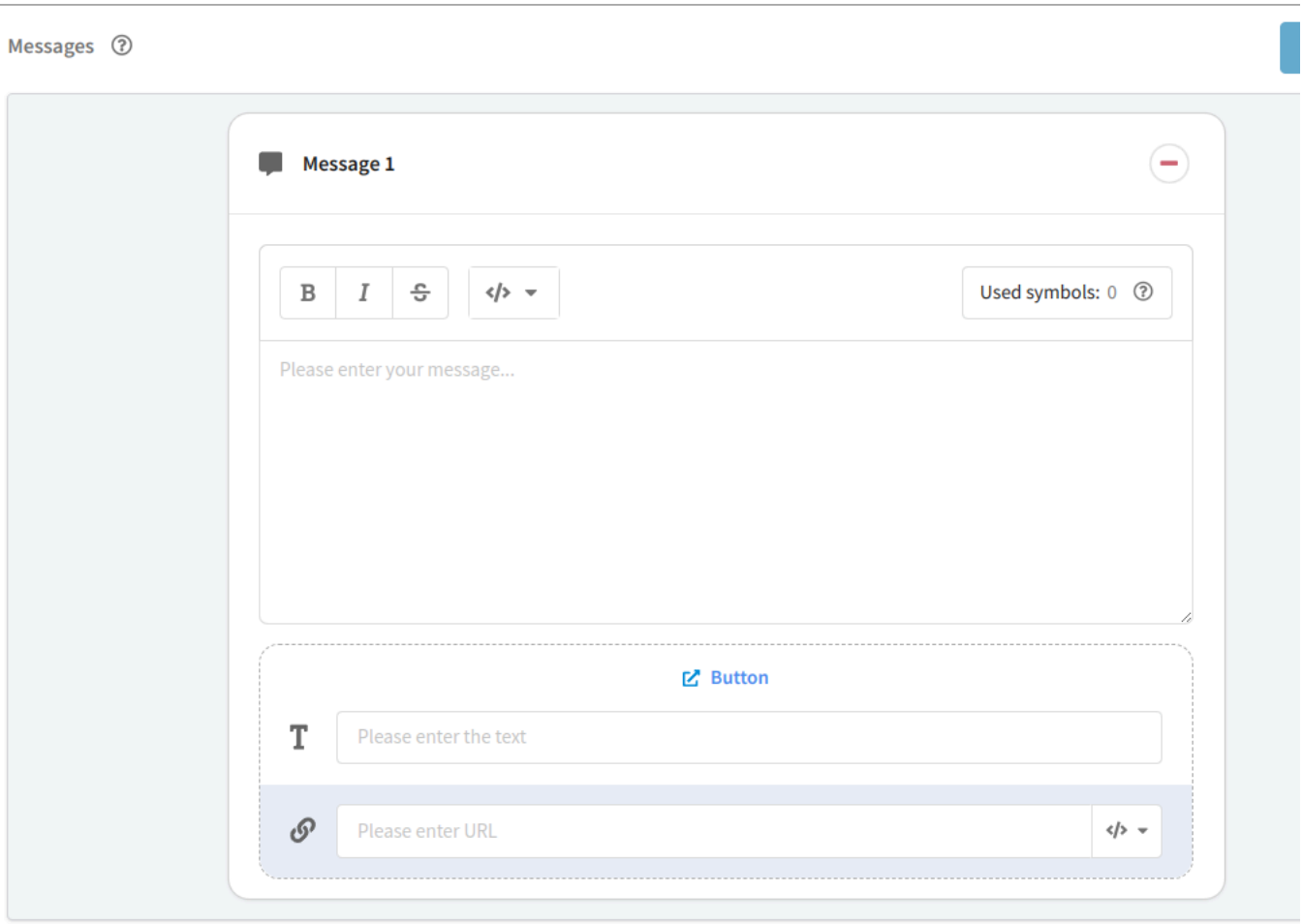
Viber

A Viber template consists of a series of messages. To begin editing a template, add a message using the



The maximum number of messages in a series is 5. If multiple messages are used in the template, each
To remove a message, click

.



The following types of messages are available:

- **Text**
- **Text + button**
- **Text + image + button**
- **Image**
- **Video (not available with MTS OmniChannel)**
- **Video + text (not available with MTS OmniChannel)**
- **Video + text + button (not available with SMS Traffic and MTS OmniChannel)**
- **File (not available with SMS Traffic and MTS OmniChannel)**

Text

Enter your message text in the editor using Markdown format. For convenient formatting, the following fu panel.

B

— bold,

I

— *italic*,

~~S~~

— ~~strikethrough~~,

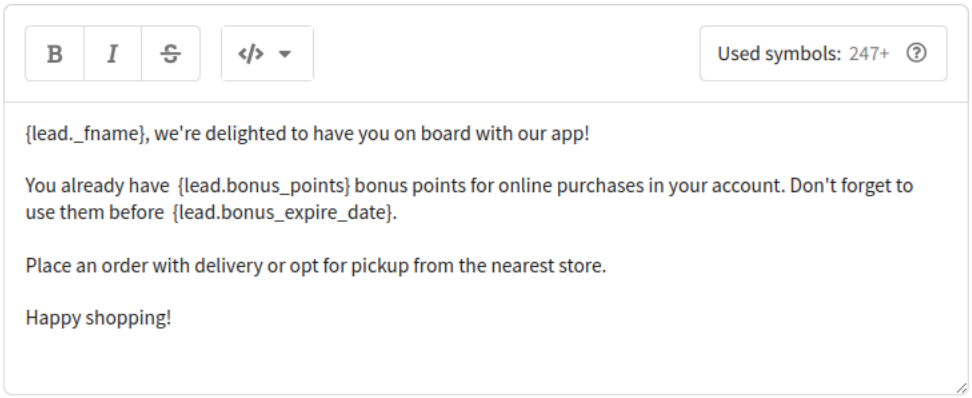
</>

— a list of [variables and functions] to personalize your message.

CAUTION

Please note that Markdown formatting (e.g., `"**text**`) is displayed in the preview mode, but the messa formatting (text").

Also, you cannot add links in Markdown format in the “Text” block.



TIP

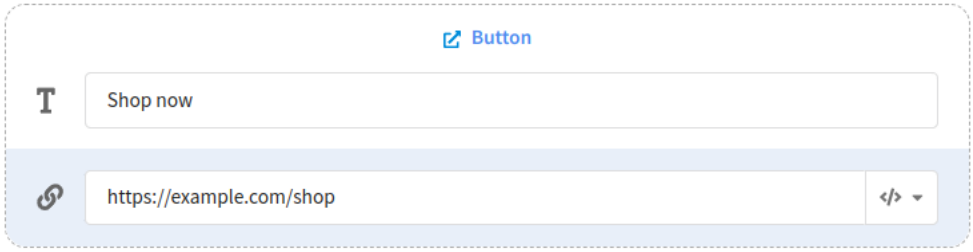
You can add emojis to Viber messages. To insert an emoji, you need to use its code. For example, to ad
codes for all available emojis at this link.
Please note that when counting the number of symbols in a message, each symbol in the emoji code is
Emojis are not displayed in preview mode.

INFO

On the right-hand side, you'll find a symbol counter. The maximum length of one message is 1000 symb
from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend monitoring th

Button

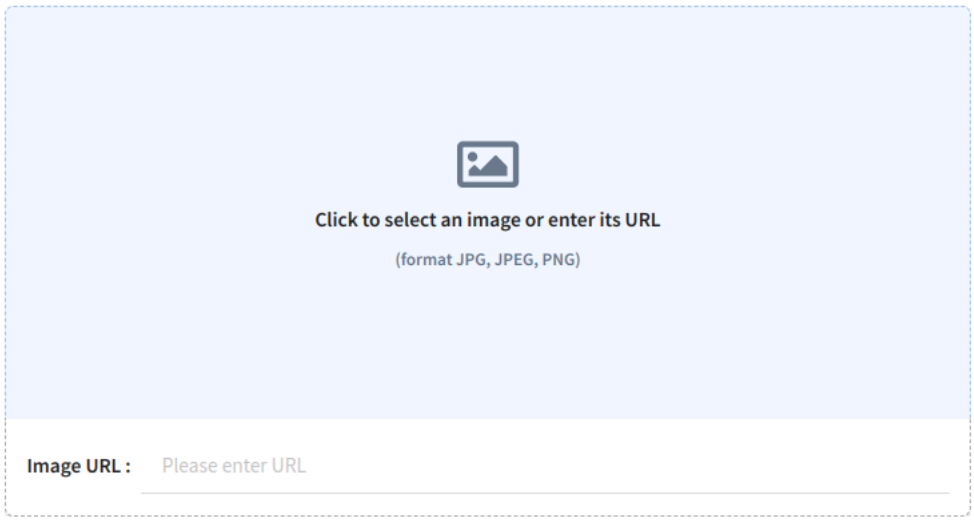
Specify the text that will be displayed on the button and the link for redirection when clicked. You can al



Image

Supported formats are jpg and png, with a maximum size of 5 MB.
You can add images in two ways:

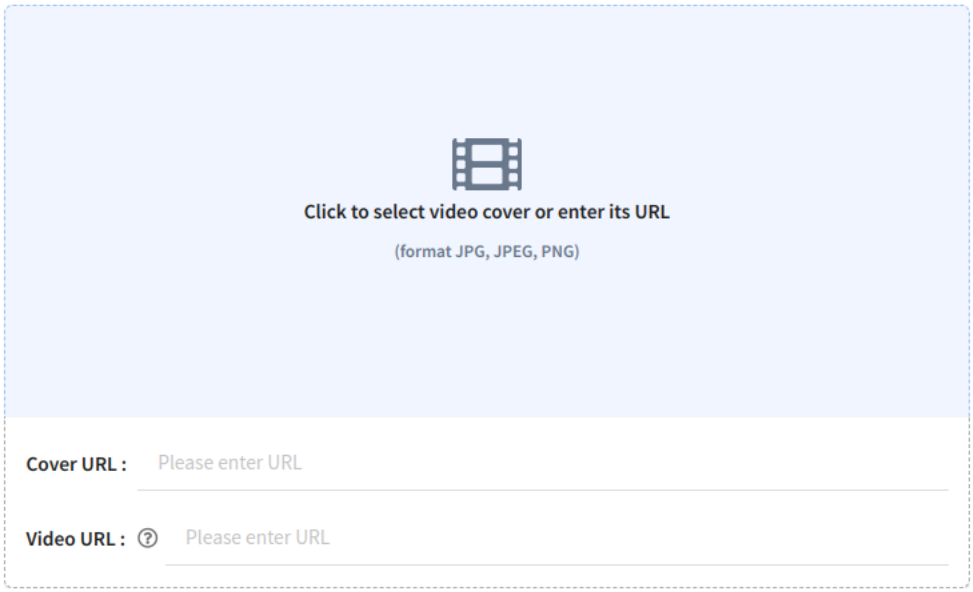
- **From the platform's gallery. Click on the upload area to open the gallery. Choose the desired image o**
- **From an external source. Provide a link to the image.**



Video

Supported formats are mp4, 3gpp. Codec: H.264. Maximum size: 5 MB.

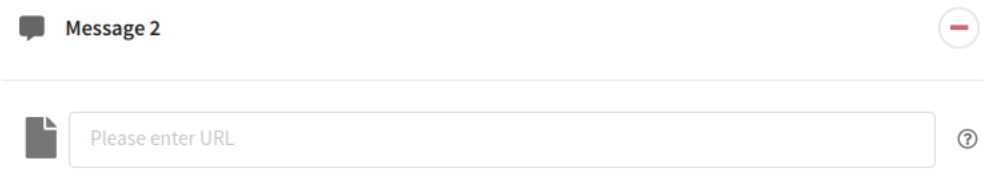
To add a video, specify its URL. You also need to set a cover for the video. You can choose an image fron link to an image.



File

Supported formats are doc, docs, rtf, odt, odf, txt, pdf, eps, xls, xlsx, ods, csv. Maximum file size: 200 MB (including the extension).

To send a file, provide a link to it.

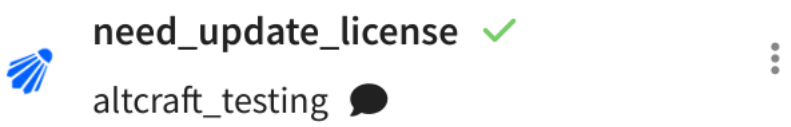


Notify

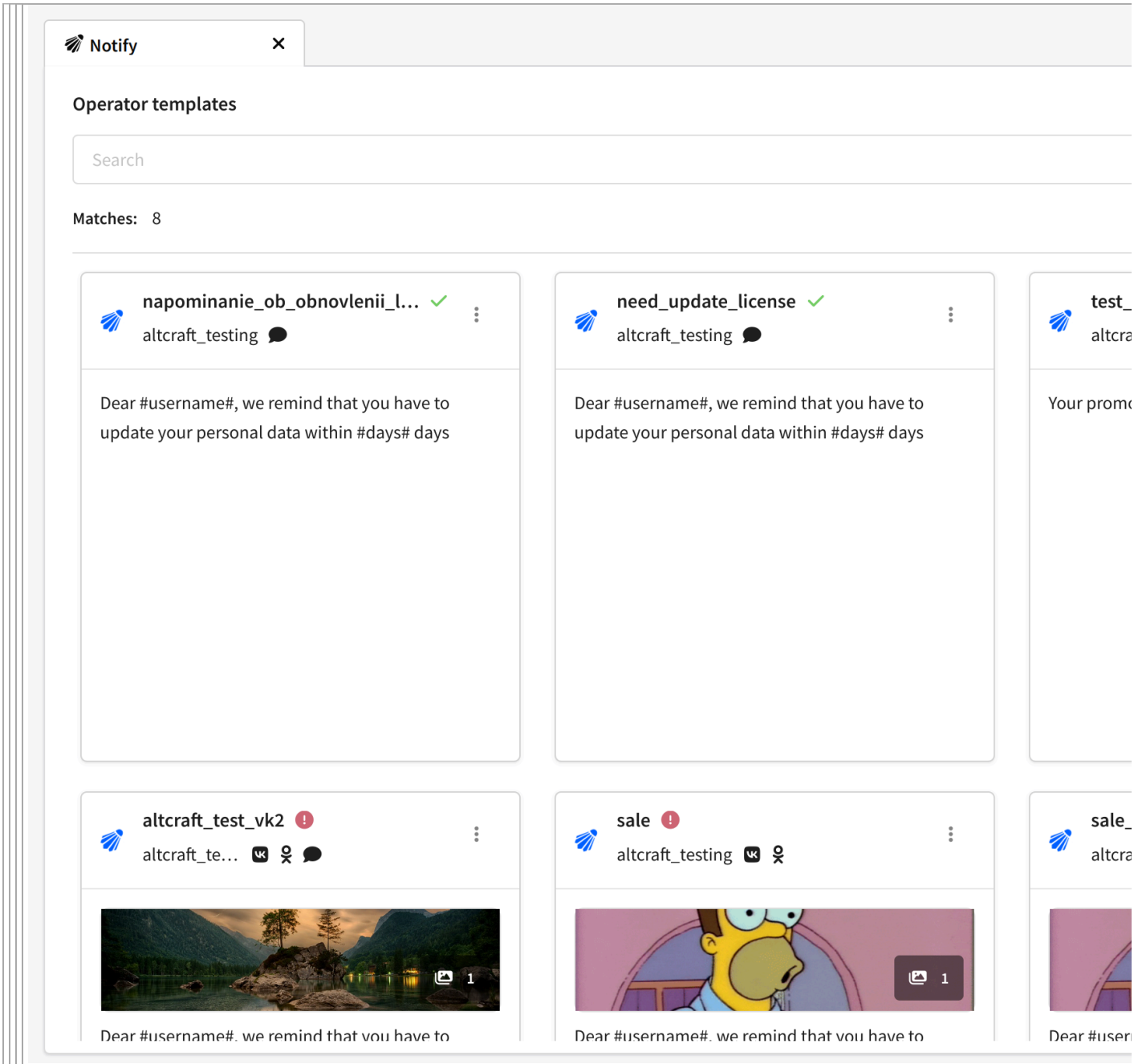
Using the Notify channel you can send messages via Vkontakte, Odnoklassniki or SMS messages. Vkont and SMS messages are customized in two different sends that follow each other.

CAUTION

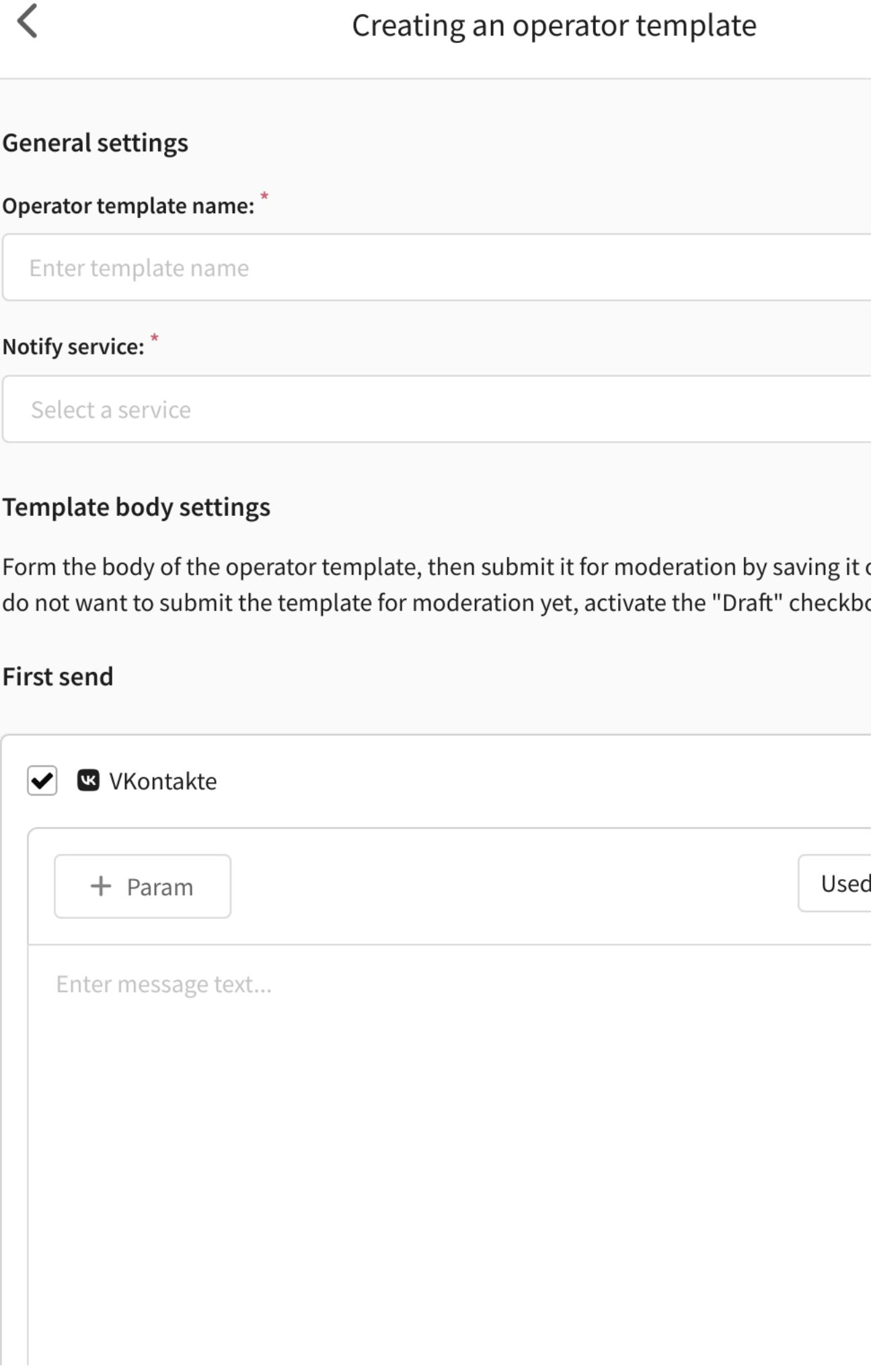
Only templates that have passed moderation can be used for mailing. They are marked with a special si

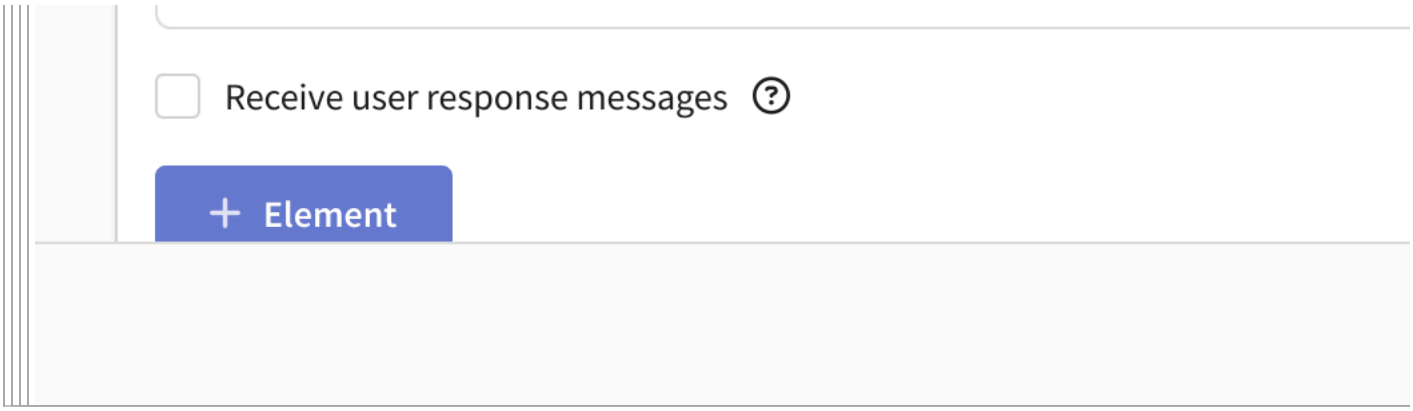


When you open the template editor for Notify, you will be shown the Operator templates of your account:



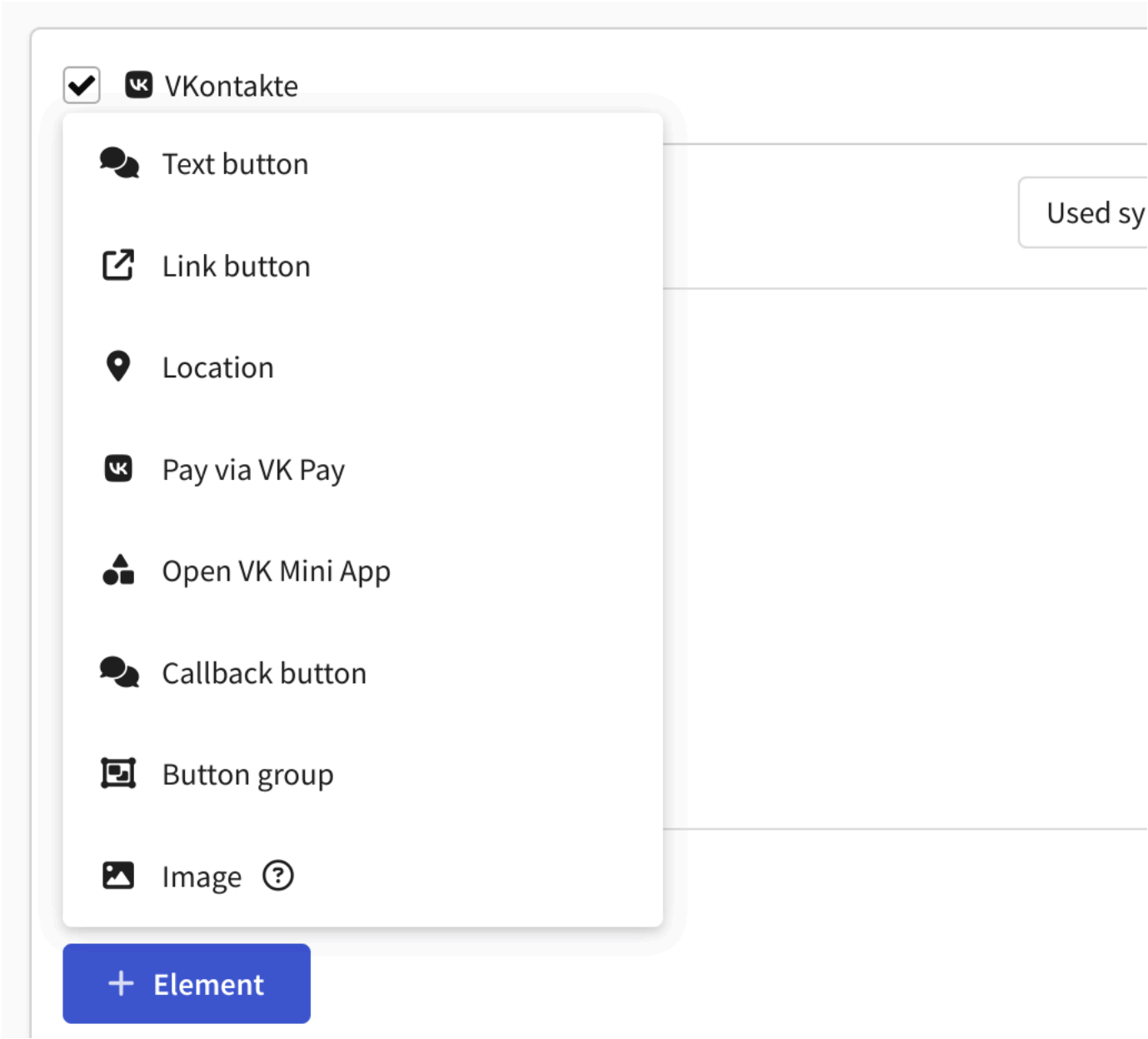
You can use one of the existing ones or create a new one. To create a new template, click the appropriat window will open:


Creating an operator template
General settings
Operator template name: *
Enter template name
Notify service: *
Select a service
Template body settings
Form the body of the operator template, then submit it for moderation by saving it
do not want to submit the template for moderation yet, activate the "Draft" checkbo
First send
VKontakte
+ Param
Used
Enter message text...


Receive user response messages
+ Element

In this window you should specify the name of the template, the service configured in your personal cabi and the time to wait for the message to be delivered and read. The last parameter defines the time befor is started.

You can attach an element to a “Vkontakte” message:

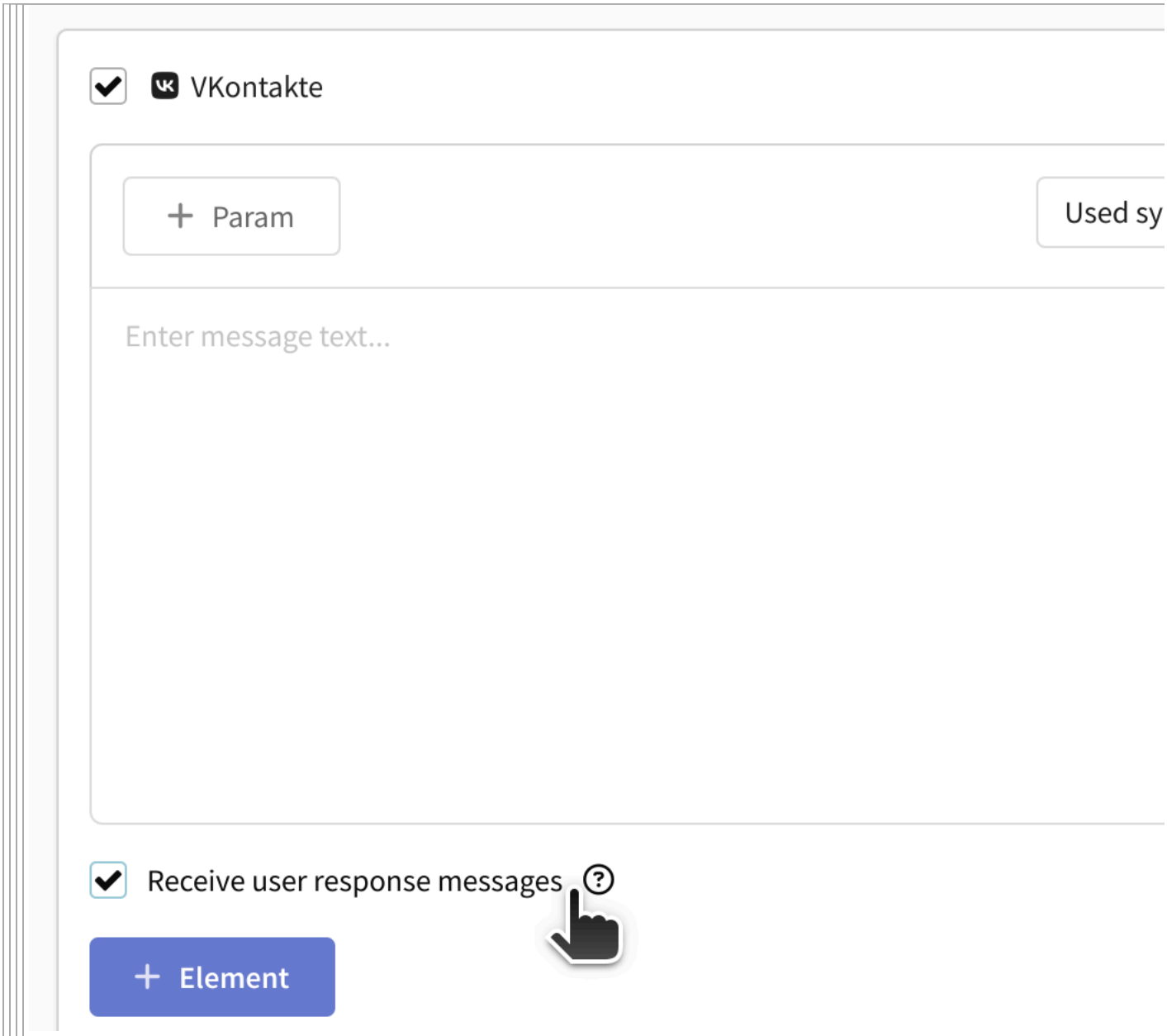


- **Text button - sends a message with the text.**
- **Link button - opens that link.**
- **Location - by clicking, it sends the location into a dialogue with the bot/conversation.**
- **Pay via VK Pay - opens the payment window VK Pay with predefined parameters.**
- **Open VK Mini App - opens the specified application VK Mini Apps.**
- **Callback button - allows without sending a message from the user to receive a notification about pre necessary action.**
- **Button group - cobmination of the buttons.**
- **Image - up to 5 images with a total weight not exceeding 50 MB.**

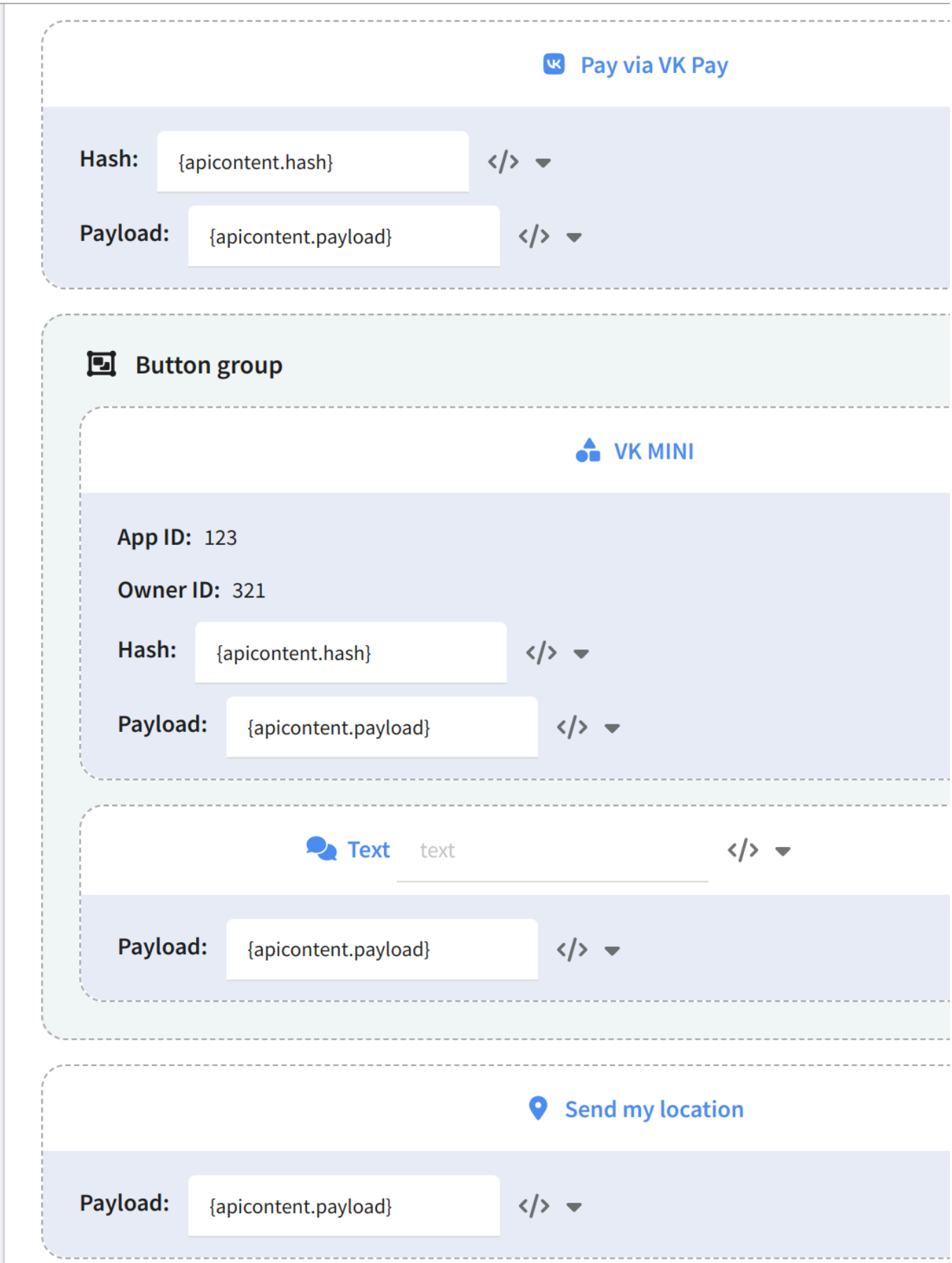
TIP

Read more about elements in Notify documentation.

You can also enable the customer to send reply messages in chat with the company. To do this, activate settings:



Once you save the template, it will be sent to moderation in the service. If you don’t want to send it for n
After selecting an approved template, you can insert platform variables into it using the drop-down menu

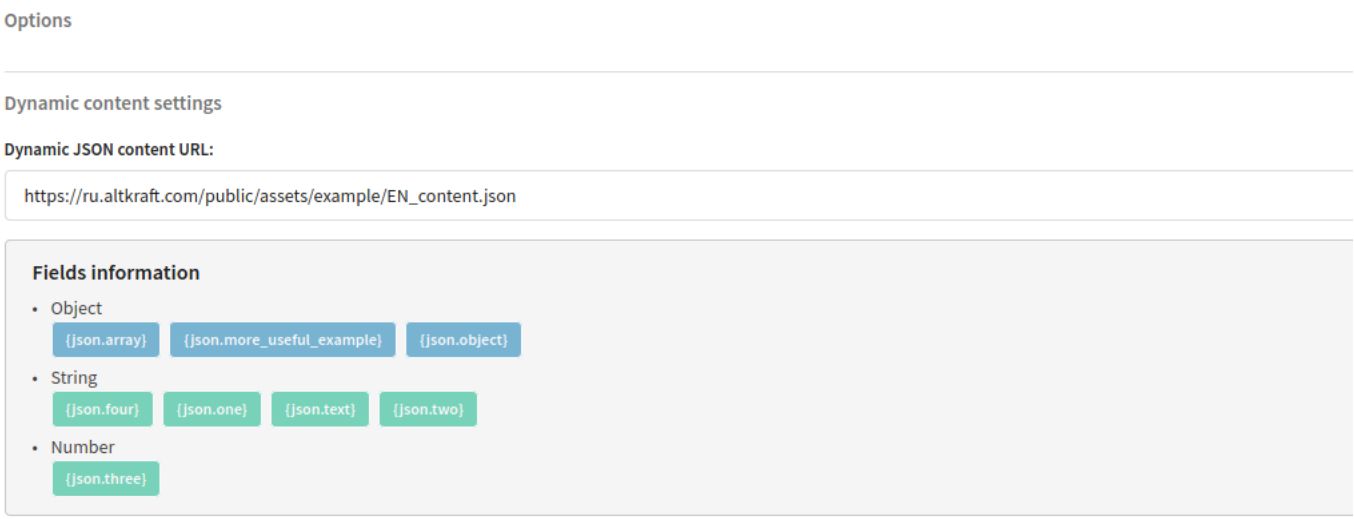


Don’t forget to save the template in the platform after making all the changes. You can now use this tem

Message options

In Options block you can set some extra parameters for the template.

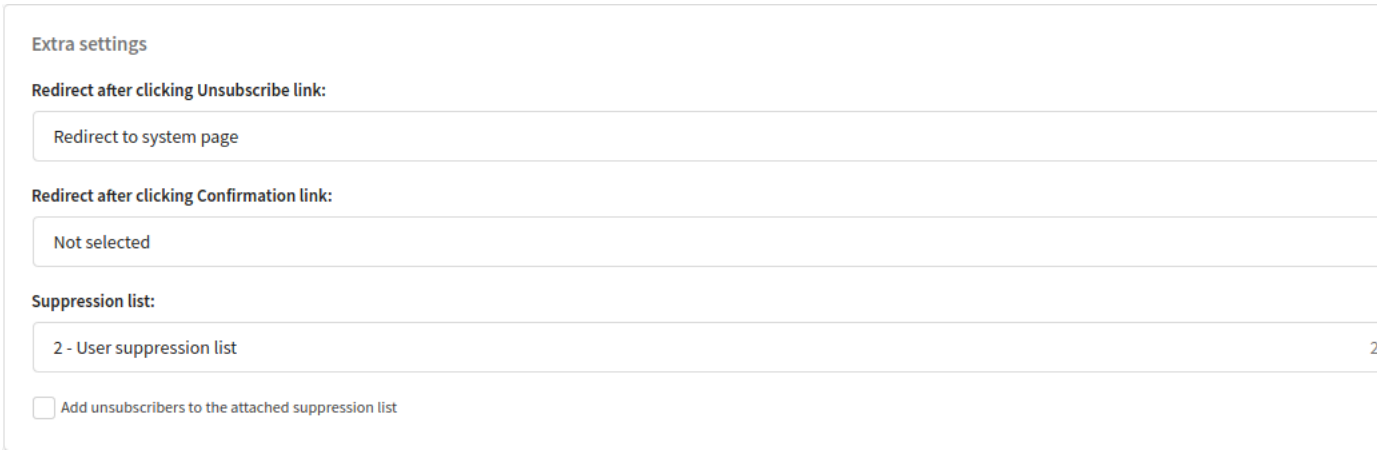
In Dynamic content settings you need to set data source for JSON content in the messages. If you have will be shown:



In Extra settings, you can select custom unsubscription and confirmation links for your message. For exa customers to a personal page or subscriptions management page.

> **TIP**
>
> **Custom links for unsubscription and confirmation can be created by master user of the account. You ca links in the template options. More information is available in Settings section.**

Also, in Extra settings, you can select a custom suppression list, that you would like to use with this tem easily restrict messaging for those customers, who unsubscribe from similar mailings. Also, you can tog attached suppression list option to automate the process.



Personalization placeholders and Parts

In the Content tab you can set default values for profile variables you use in the template. If you use cus can be empty for some profiles, a default value like “Dear customer” will be used.

> **TIP**
>
> **Content tab appears, when you use variables with profile data or parts of the message into the templat**

Message Parts are similar to fragments, but they can be used only in templates wherein they were creat used in several templates - e.g. for a header and a footer.

> **TIP**
>
> **To get started with parts, add `{part.example_part}` to your template code. After that, the “Content” tab wi where you can customize parts of the template.**

> **TIP**
>
> **It is convenient to use parts with logical expressions for more compact message code. For example, he depending on customer’s score (custom integer field):**

```
{if lead.score gte 9000}
{part.part_1}
{else}
{part.part_2}
{end}
```

For every part, you can choose content type: HTML for email messages, or text — for text version of ema
Depending on content type, appropriate editor with necessary functions will be opened.

Message preview

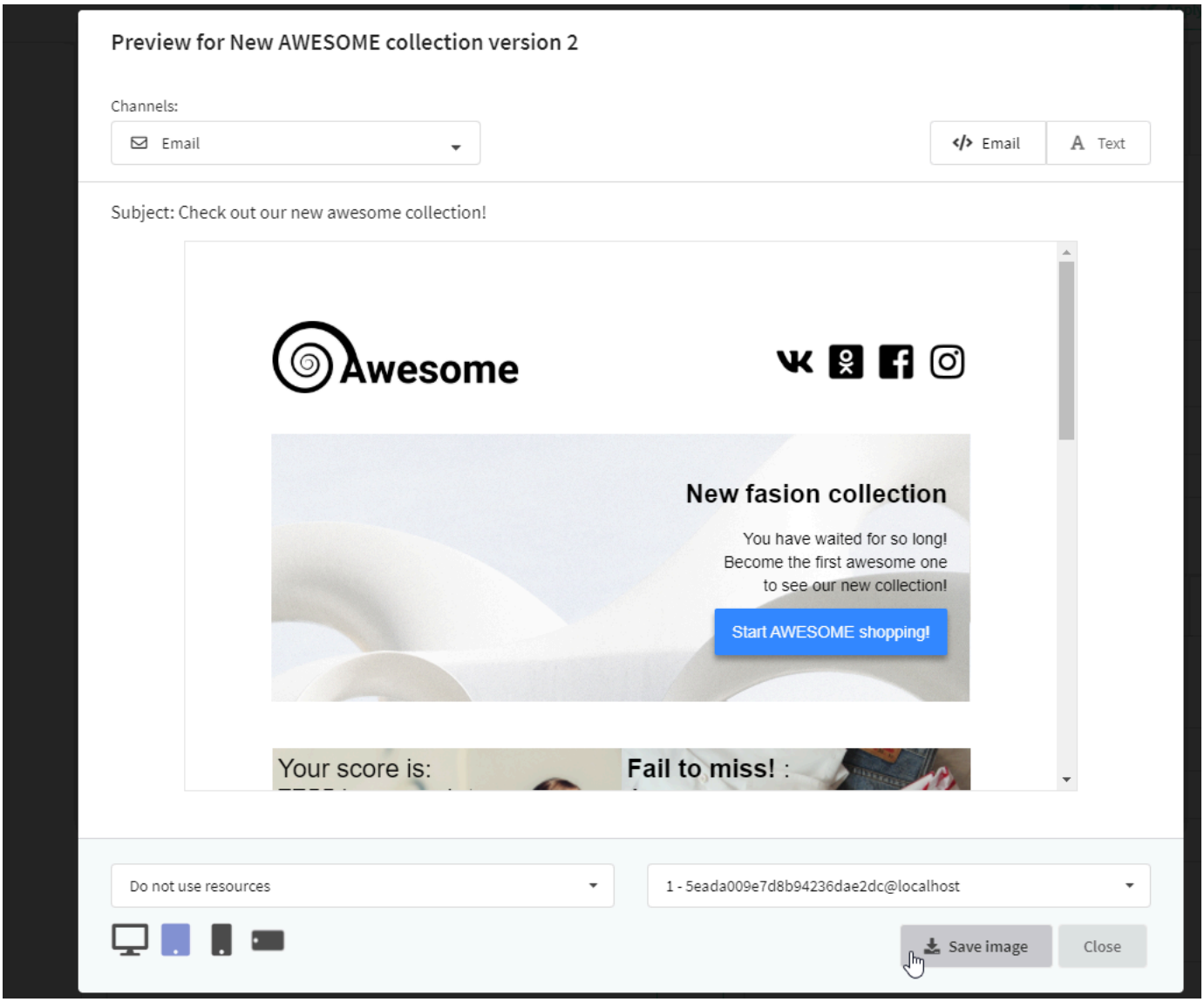
Message preview allows you to check HTML layout, and to see content personalization result, based on
activate preview:

In preview window, you can select communication channel, resource and customer profile you want to u
and text version preview is available.
In the bottom part of the page, there are buttons for switching desktop/mobile display mode, and person

TIP

Choose the resource, within which you want to send the message - and select the profile in the drop-do
search in the database: segmentation rules are used, and you can immediately choose the profile with

The preview is available for desktops, tablets and cellphones both in portrait and landscape orientation:



You can save previewed message as an image to use it in any way you need. To download the image to y
button.
Follow these rules for correct email template preview export:

1. **CSS styles such as** `text-decoration`, `text-decoration-line`, **and** `text-decoration-color` **should be set directly**

```
<div>
    <span style="text-decoration: line-through;">Text</span>
</div>
```

2. **If you are using fonts imported from an external source (e.g., Google Fonts), it is recommended to re property to fonts installed on the system.**

```
<!-- Correct display in the preview: -->
<span style="font-family: Roboto, Arial, Tahoma, sans-serif">Text</span>

<!-- Incorrect display in the preview: -->
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900
```

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a*

Template fragments

Description

![img](/img/attachments/3114274/9408048.png)	**Fragments are message blocks, which are used throughout several messages. For example, a fragment can be used as header or f any content, that you are going to send within several mailings.**

TIP

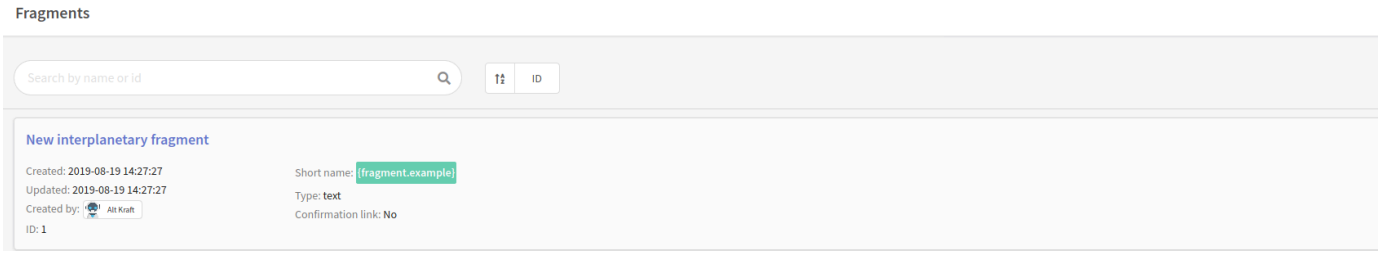
Fragments are similar to template parts, but they can be used throughout several templates, when part where created. You can read more about parts in templates managing article.

For example, emails often contain links to social networks pages:

Fragments have been developed to make repeated message sections easy and fast to use — and thus to

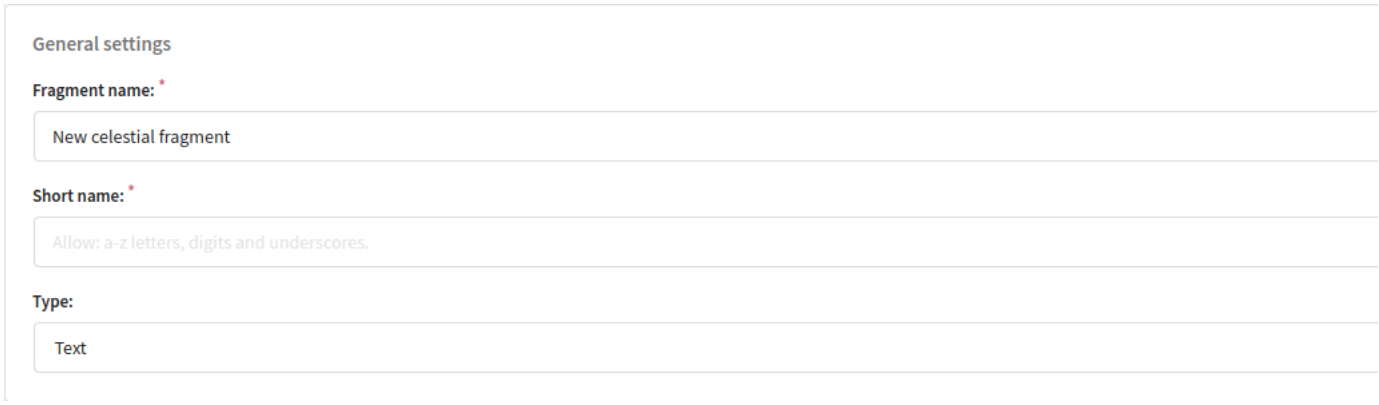
Managing fragments

Select Templates —> Fragments in the main menu to start working on fragments. A list of existing fragm you can edit, clone or delete the fragment, — or set access group for it. If you need a new fragment — pr page:

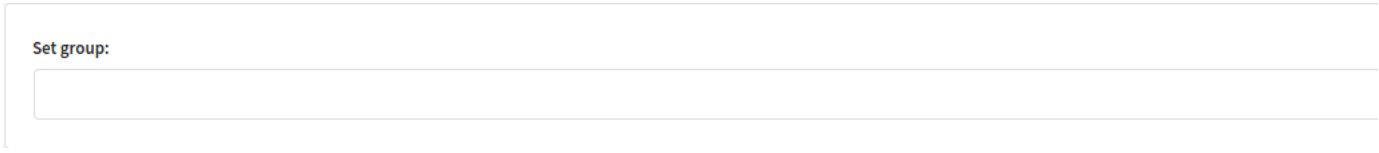


General settings

In general settings, you need to set Fragment name, that will be shown in the list, and Short name. Short fragments into the templates manually, like `{fragment.<short name>}`, or using variables menu (`</>`). Also, yo HTML or text:



At the bottom part of the page, there is a block, using which you can set access group for the fragment:



HTML fragments

HTML fragments are suitable for email messages, and custom channels that support HTML. The editor su personalization variables. To make formatting and working with objects easier, next functions are availa

Formatting:

— bold

I

— *italic*

— ~~strikethrough~~

— bullet list

— numbered list

— table

Service functions:

— pixels for tracking of letter opening and reading. If you do not add pixel during message editing, the sy

— external HTML content. You can also set variable source address manually `{[http://example.com]}`.

— links builder.

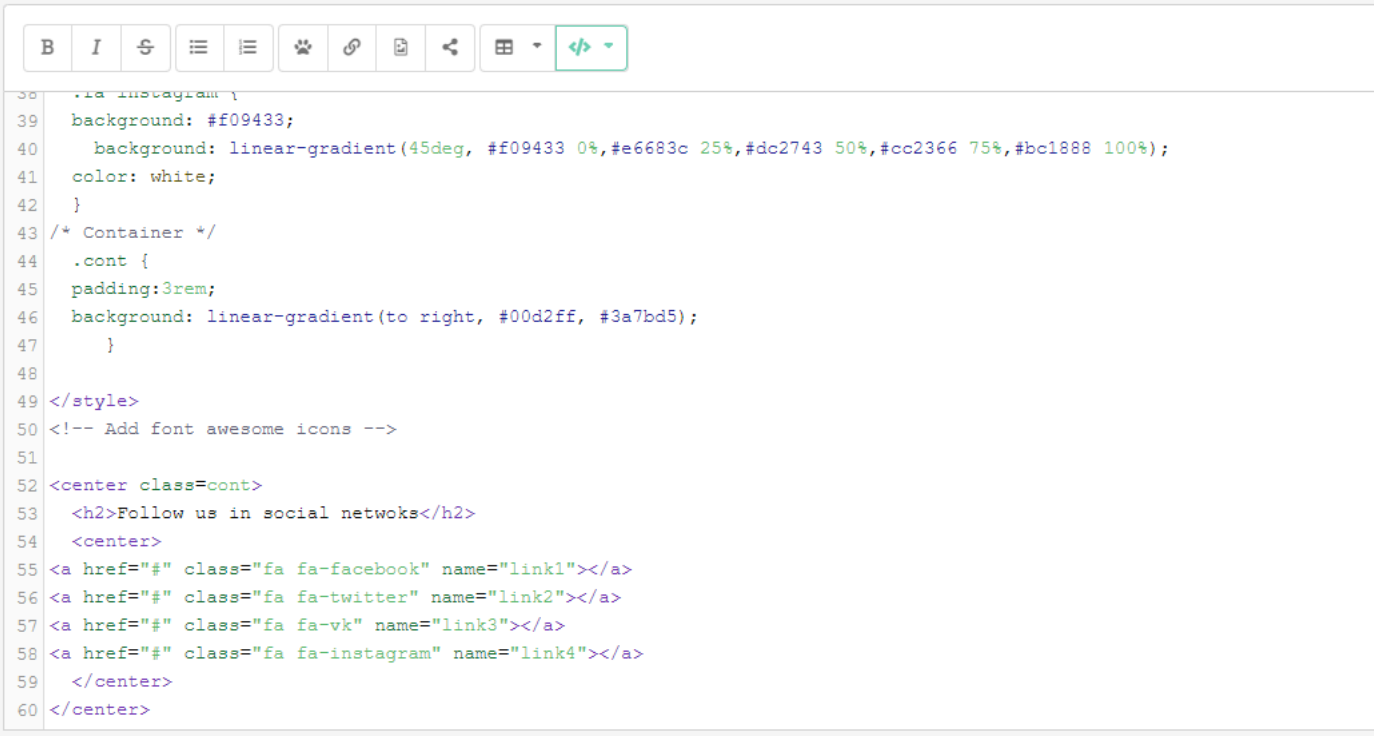
utm

— UTM tags.

— images.

— social buttons.

— variables and editor functions menu. Using this button, you can add Customer profiles' data, promo co dynamic content and service parameters into the message.



```
  .fa instagram {
  background: #f09433;
    background: linear-gradient(45deg, #f09433 0%,#e6683c 25%,#dc2743 50%,#cc2366 75%,#bc1888 100%);
  color: white;
  }
/* Container */
  .cont {
  padding:3rem;
  background: linear-gradient(to right, #00d2ff, #3a7bd5);
     }

</style>
<!-- Add font awesome icons -->

<center class=cont>
  <h2>Follow us in social netwoks</h2>
  <center>
<a href="#" class="fa fa-facebook" name="link1"></a>
<a href="#" class="fa fa-twitter" name="link2"></a>
<a href="#" class="fa fa-vk" name="link3"></a>
<a href="#" class="fa fa-instagram" name="link4"></a>
  </center>
</center>
```

Plain text fragments

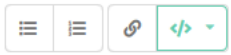
Text fragments can be used in all types of content. The following functions are available in the editor:

— bullet list

— numbered list

— links builder.

— variables and editor functions menu .



Thank you for taking part in the action {lead.event_name}!
Your promocode: {loyalty.event.promocode}.

Using fragments in templates

Fragments can be added into templates using variables menu (</>). Also, you can add them manually us
Fragments can be displayed depending on some conditions and subscriber data.

TIP

If you want to display one of several fragments in a random order, choose Random fragment and specify use:

```
{randomfragment[fragment.header_1 fragment.header_2 fragment.header_3]}
```

Image gallery

Description

Gallery provides an opportunity to save and organize images for using while working with marketing tem

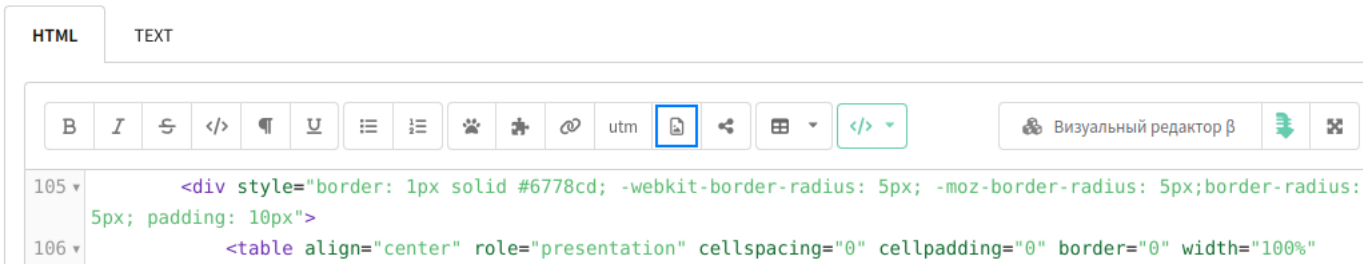
- **HTML and AMP versions of Email templates;**
- **Push templates (banners and icons).**

Images are loaded into messages and pop-ups from the platform server through the tracking domain.

Work in the gallery

To open the image gallery, go to the template editor and click on

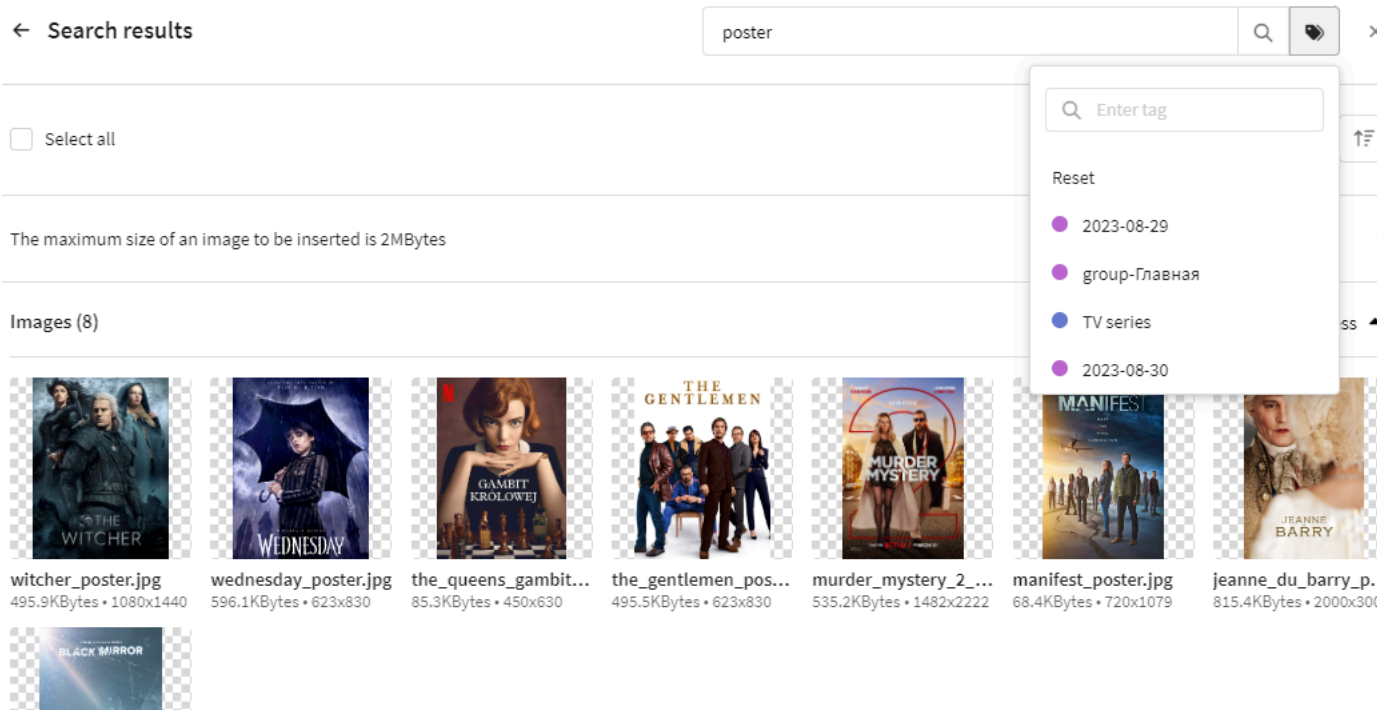
the icon. For example, in the Email message editor, you can go to the gallery using the button in the tool



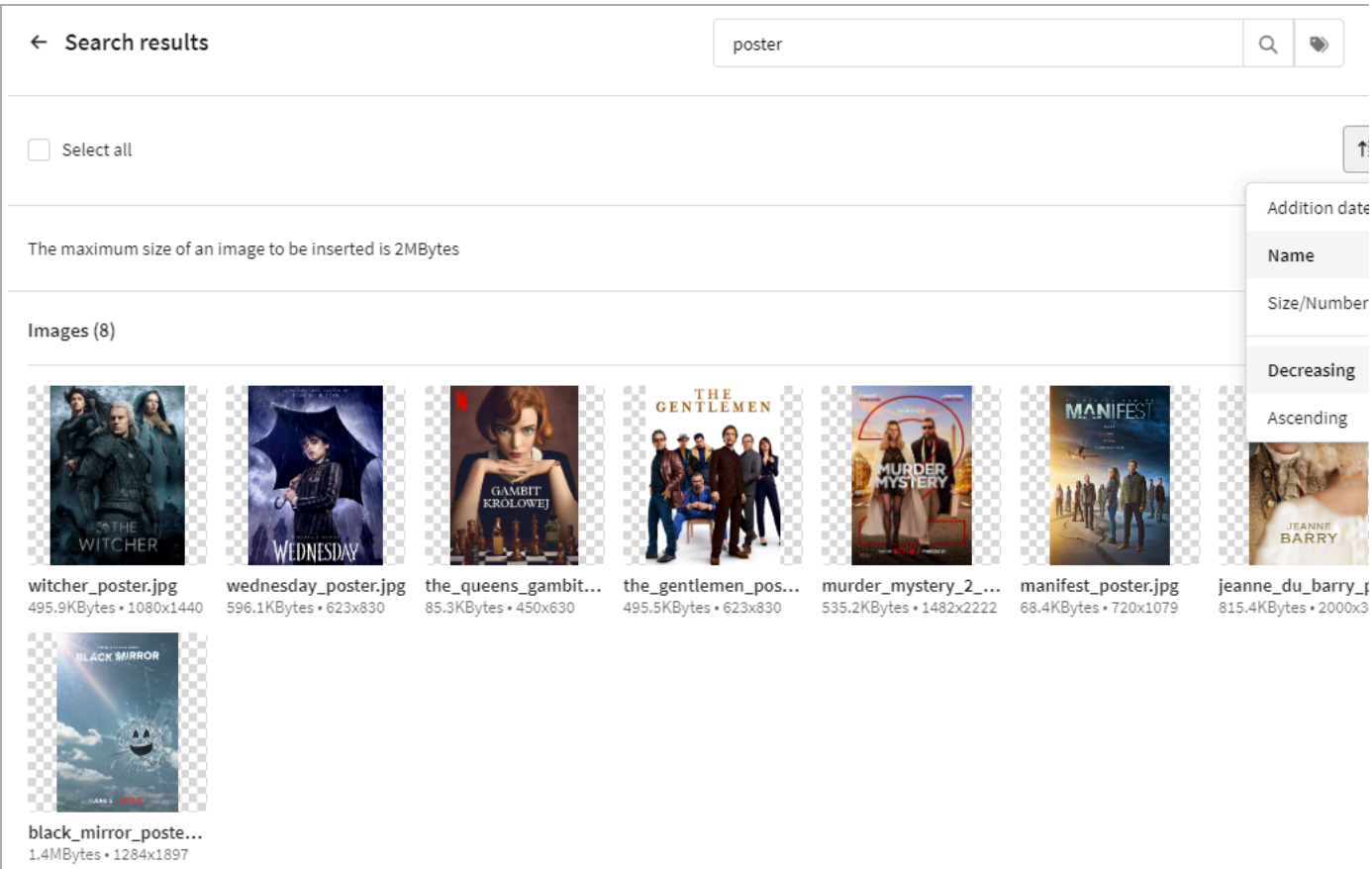
The gallery is a modal window, visually divided into 3 blocks:

- **recent (all images added in the last month);**
- **directory;**
- **images.**

You can search by file/directory name or attached tags. To exit search mode, press ← Search results:



You can also sort objects in the gallery by date added, by name, or by size:



You can select one, several, or all objects in the gallery and do one of the following:

— move the object to another directory.

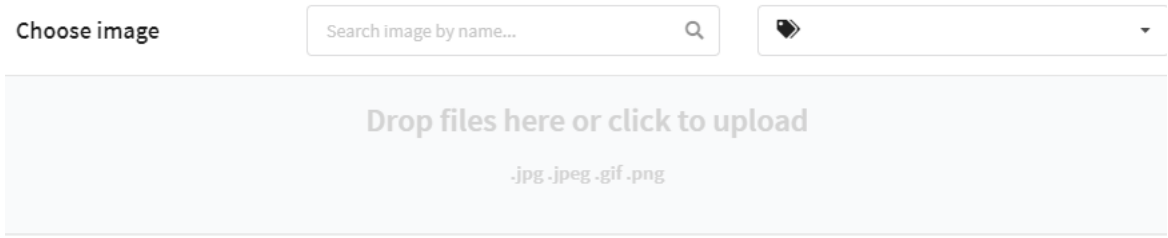
— download an object (only for images).

— delete an object.

Working with images

Uploading an image

To add an image to the gallery, tap + Image and select the file on your device. The second way is to drag



NOTE

You can upload multiple images at once. The size of one image should not exceed 2 MB. Available forma

The upload status is displayed in the lower block. You can interrupt upload by clicking on the cross. If yo image, hover over the question mark to find out the reason.

If you close the gallery before all files have been imported, the import process will be interrupted, but al uloaded to that point will remain in the gallery.

Actions with images

To work with downloaded images, move the cursor over the image and click

. A list of available actions will open:

— insert a link to the image into the template. Convenient when you don’t need to change the formatting img tag.

— insert an image into the template in HTML format (new img tag).

— download the image to your device. Images are uploaded to a zip archive.

— move the image to another directory.

— rename the image. When you change the name, the link to the image does not change.

— remove an image from the gallery.

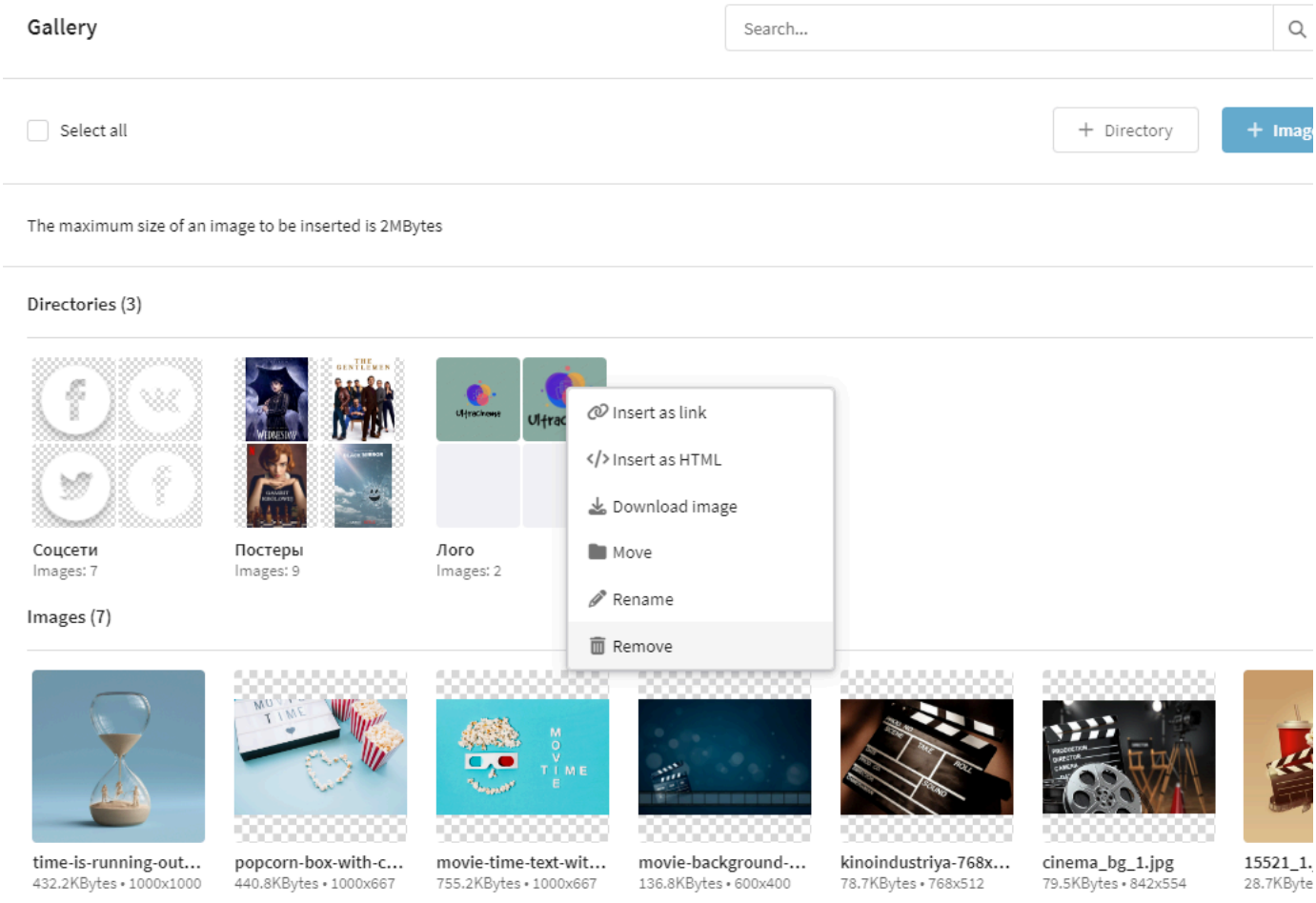
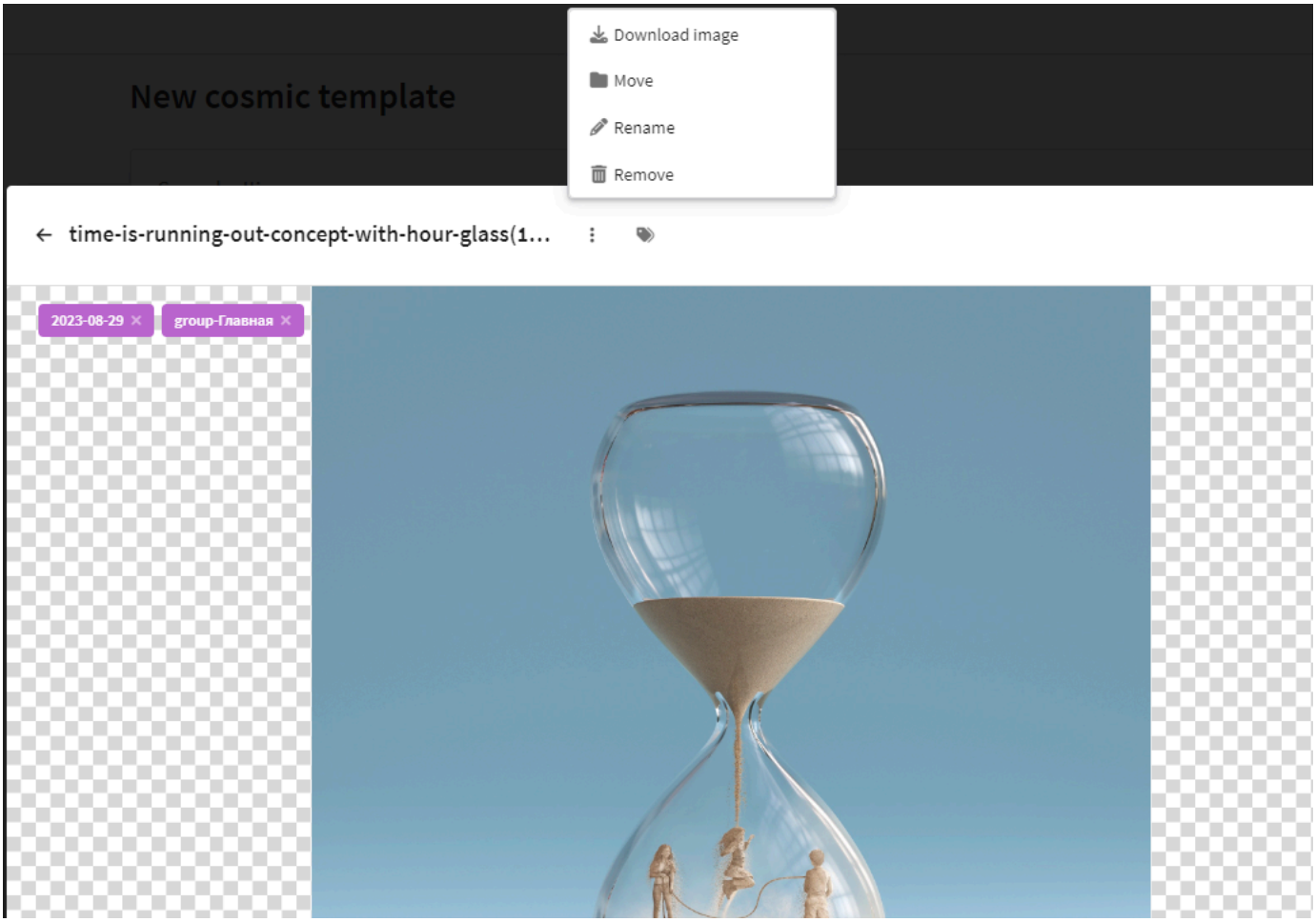


Image view mode

If you want more information about image, click on it. You will enter image viewing mode. All the actions available here. To open the list of actions, click on

:



You can also copy URL link of your image by clicking on Copy

. This link can be used, for example, when setting up an icon or banner in a push template.

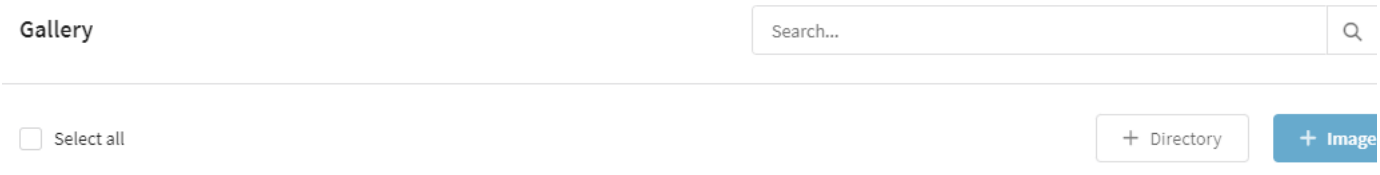
Besides, in this mode you can view tags attached to the image, as well as add new ones or remove exist automatically, for example:

- `2023-10-25` **(date tag when an image was added to the gallery);**
- `group-Home` **(name of the group in which the user worked at the time the image was added).**

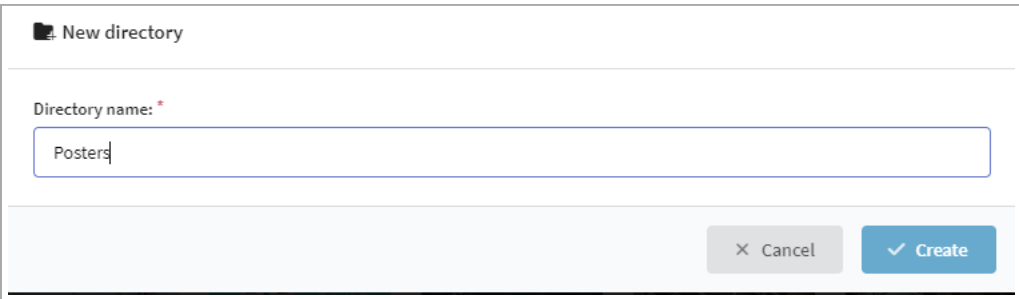
To exit image viewing mode, press ← <image_name>.

Working with directories

To more conveniently organize images in the gallery, use directories. To create a new directory, click "+



Enter a name for your directory. You can use any characters in the name, but no more than 244.



To work with a directory, move the cursor over it and click

— a list of available actions will open:

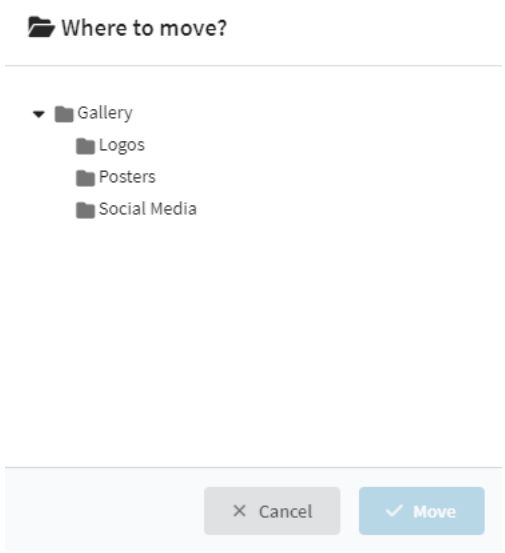
— move a directory to another directory. You cannot move a parent directory to a child directory.

— rename the directory. The directory name must be unique.

— delete the directory. All items in the directory will also be removed from the gallery.

If you want to upload images to a specific directory, go to it and click + Images or drag and drop files int

To move already added images to a directory, select the images, click Move in the list of actions and sel



Sub directories is also available. To create a sub directory, go to the parent directory and click + Directo

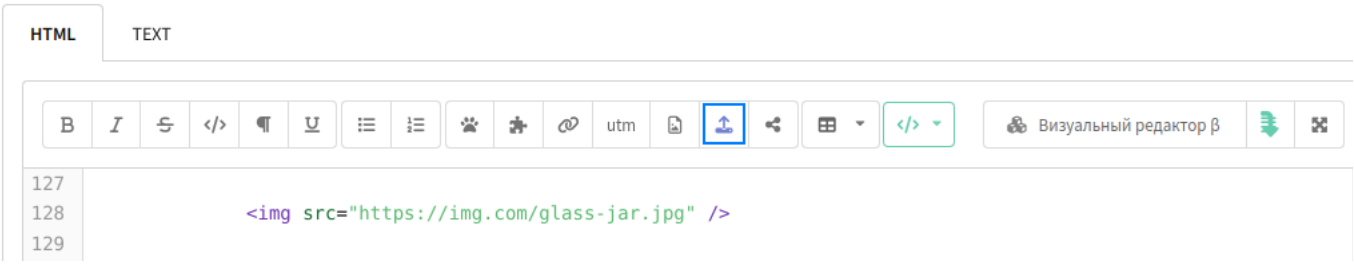
To return to the previous directory, press ← <directory name>.

Uploading images from external sources

Often templates use links to images from external sources. For example, if you designed a template in a transferred it to the Altcraft platform. In case of failures in the external service, such images may becon recommended to upload them to the platform's gallery.

Click on the button

in the editor toolbar (the button appears only after adding the URL to the `src` attri



After uploading to the platform server, the value of the `src` attribute for all external images will change, `(https://img.com/glass-jar.jpg)` will be replaced by `![img](http://{trkdomain}/383848228/1/64eefca84a8f9017e5/glass-ja` loaded into messages by the tracking domain.

Content personalization

Description

Customer profiles data can be used for creating dynamic personal messages, that do not need constant profile information from the database. It allows you to make personal human-like mailings.

> **TIP**
>
> **For example, this is the message with a daily horoscope:**
>
> **Hello, Eddie!**
>
> **Daily horoscope for Scorpio.**
>
> **This is the greatest day to consider your options, study economic trends, and analyze what you want fr message, using profile data and the link to a server generating horoscopes. This is called dynamic HTM**

```
Hello, {lead._fname}!

Daily horoscope for {lead.Sign}.

{[http://127.0.0.1/horoscopes?sign={lead.Sign}]}
```

This way, you will make one letter for all 12 zodiac signs, that does not require any time afterwards.

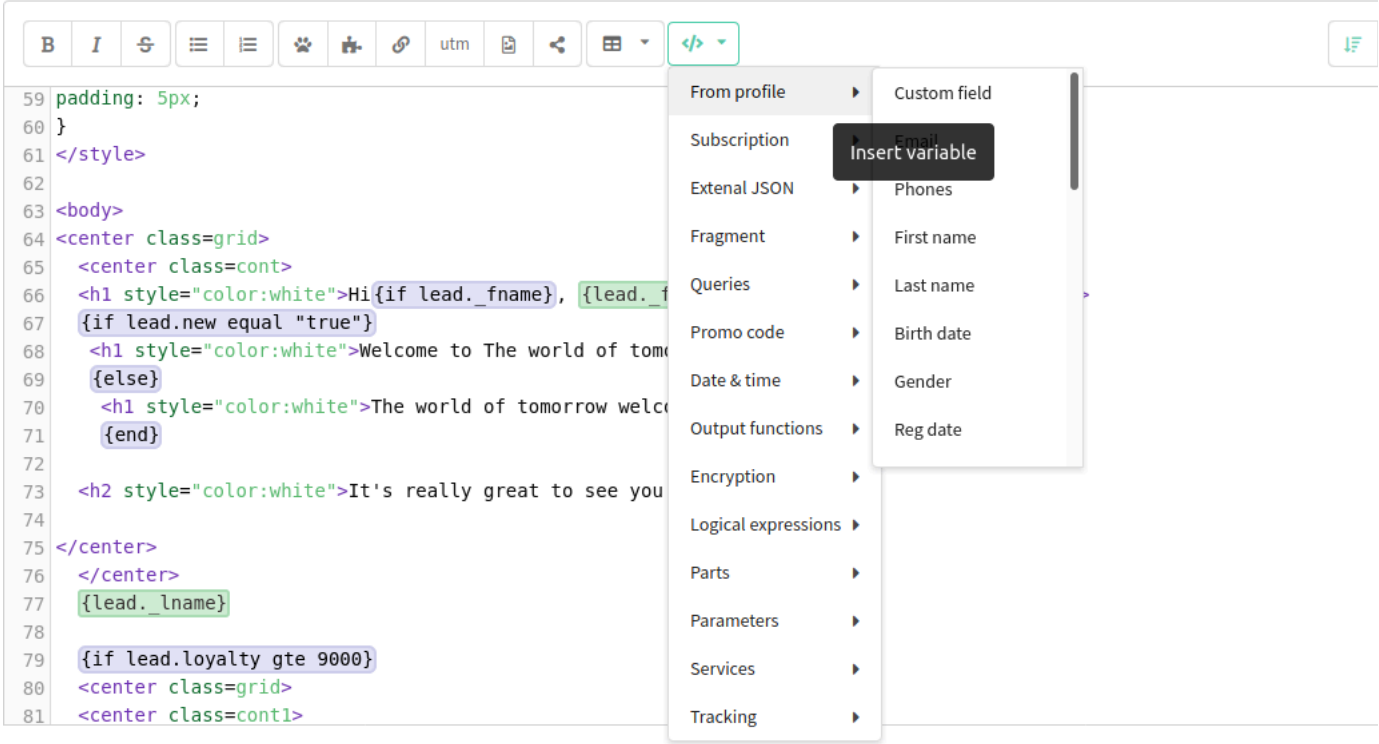
Customer data in message templates

To add data from customer profiles, open variables menu (`</>`) in template content editor, and select on

- **From profile — customer data from profiles database.**
- **Subscription — information about profile subscriptions to different resources.**
- **Promo code — promo code from a certain loyalty program.**
- **Parameters — additional data, like Database, Mailing, Template, Resource name.**

To add data from additional (custom) fields of the client profile, add `{lead.Custom_Field_Name}` into the code, of the additional field that matches the one in the profile database. In our example (see above), it is `{lea`

You can also add data from the additional field of the "Object" type. Nesting can be of any type. For exa

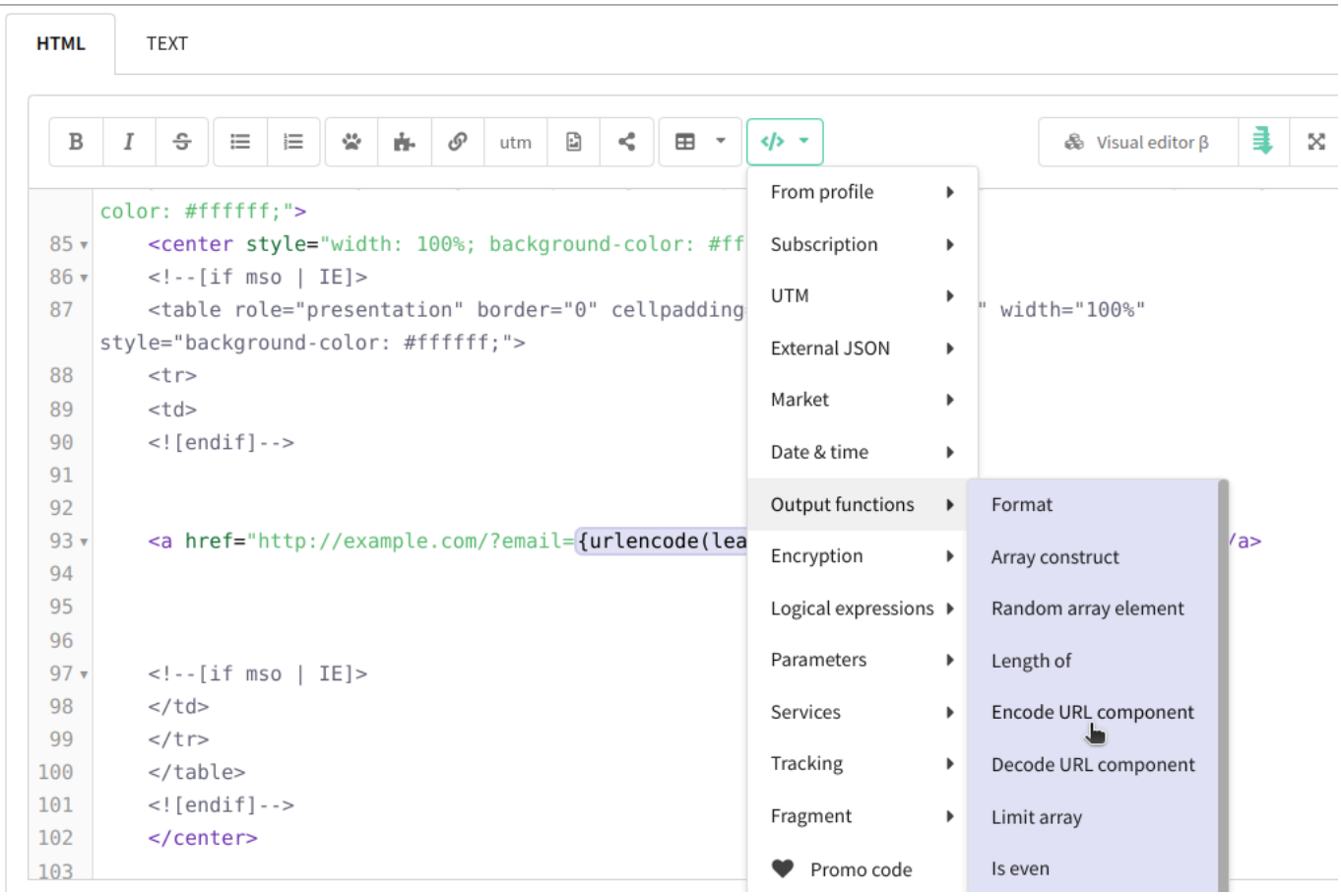


Using variables in URL parameters

URL addresses only allow Latin letters, numbers, and a limited set of special symbols. However, when us profile fields into a template, certain characters may be invalid for URLs, such as spaces or non-Latin ch the template will not work.

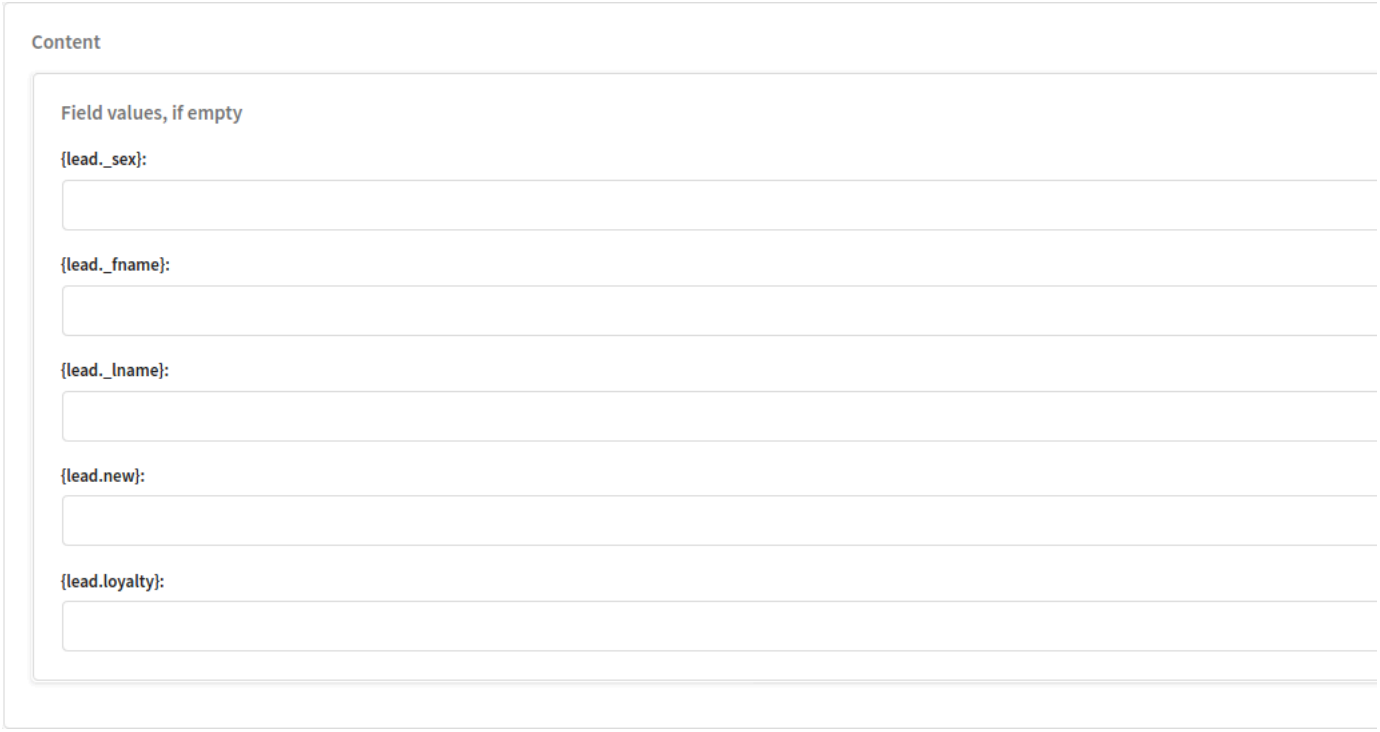
To avoid this situation, it is important to encode the parameters in the URL. This can be done using the you're inserting the variable `lead.email` into a link, use the following format:

`https://example.com/?email={urlencode(lead.email)}`.



Placeholders for missing data

In the template editor Content tab you can set default values for the profile variables used. They will be i data for a customer.



On-click customer data update & subscription

Customer data can be also updated dynamically whenever your customer clicks a specific link in your m any link: field_name should be written the same as in profile database:

```
update-field_name="value"
```

When a customer clicks such link, the value of the customer profile field in the Database will be changed

```
<a href="http://www.example.org" name="link" update-clicked_link="true">Update clicked_link!</a>
```

Adding tags

To add tags to a customer's tag-type field, use the `addtags-field_name="value"` attribute. Tags are added to t

```
<a href="http://www.example.org" addtags-interests="sports,travel">Add interest tags</a>
```

Multiple tags are separated by commas.

Removing tags

To remove tags from a customer's tag-type field, use the `removetags-field_name="value"` attribute.

```
<a href="http://www.example.org" removetags-interests="sports">Remove the "sports" tag</a>
```

Multiple tags to remove are also separated by commas.

> NOTE: General notes
>
> - **Field names in `update-`, `addtags-`, `removetags-`, and `subscribe-to` attributes are case-sensitive. For exa `interests` will target different fields.**
> - **For `addtags-` and `removetags-`, the field must be of the "Tags" type.**
> - **If the field is not found in the database, the operation will be skipped.**

Subscribing to a resource

To subscribe your customer to a different resource on click - use the following link attribute:

```
<a href="http://www.example.org" subscribe-to="1">Subscribe me to resource #1</a>
```

Creating tables based on array elements

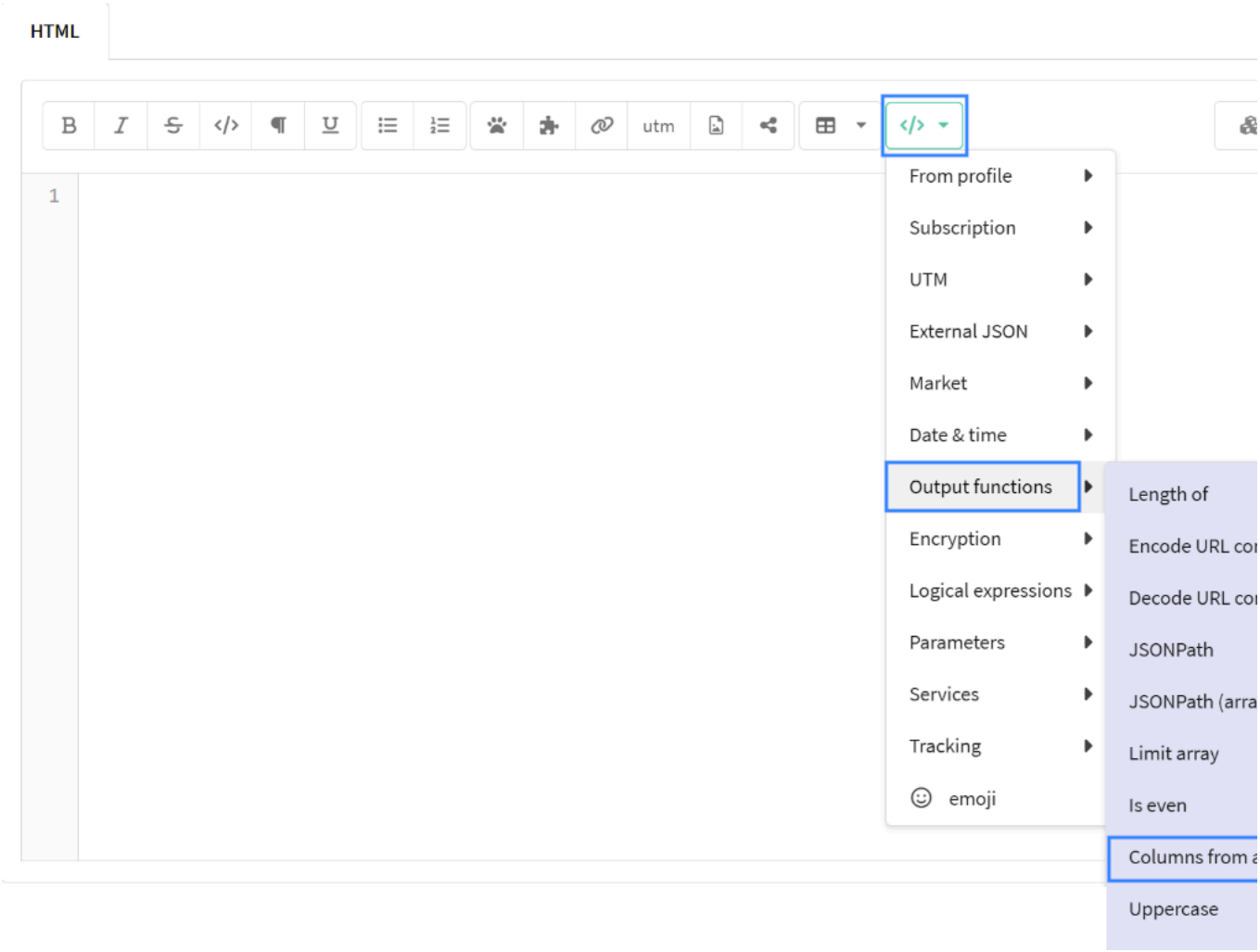
Description

Creating tables based on array elements is one of platform features you can use if you transfer dynamic objects from the database, and then distribute the elements of this array in the table as the situation rec

TIP

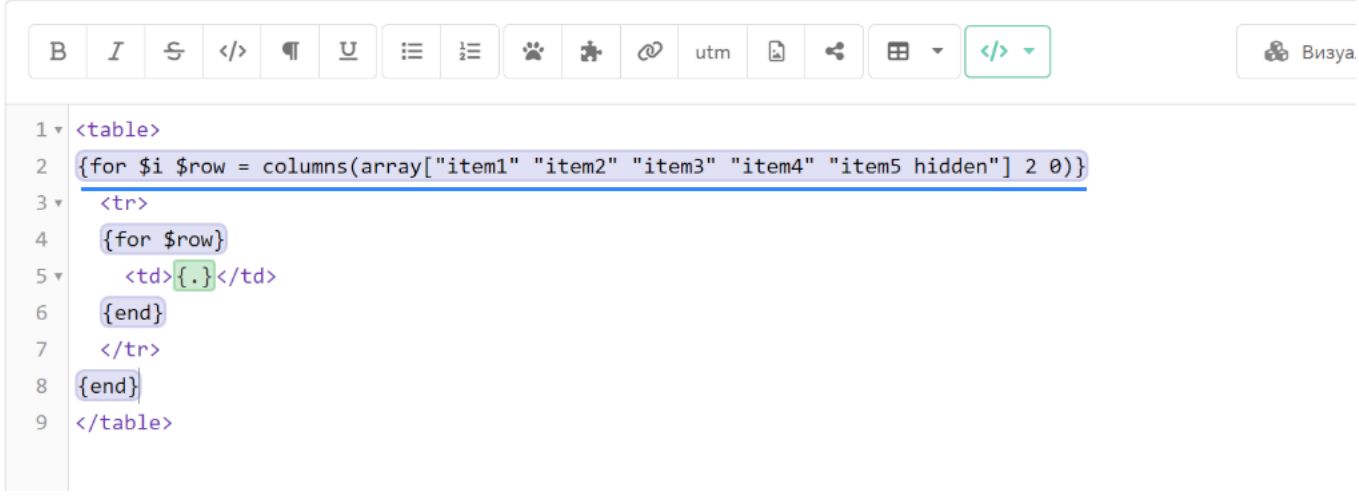
This feature is available when transferring content in any way: API content, JSON content, content fron

If you want to create tables based on array elements, in the HTML version tab click: Insert variable —> C array.



The code example will appear in the editor window. Look at the line of code:

```
{for $i $row = columns(array["item1" "item2" "item3" "item4" "item5 hidden"] 2 0)}
```



Array is an object from which we take content for our table. There is content for the certain table ceil in “item2”, “item3” и etc).

“2 — it says how many columns there are in the table. 0” — it’s the amount of elements that will not be a first element of the array).

As a result, in our table all elements of the array will be distributed into two columns and none of them v

Use cases

Let’s take a look at one of the cases for this option.

For instance, an airline’s website dynamically changes content for each user. The content depends on th see on his screen 5 flight offers that he may be interested in. These offers are shown in the interface as information on cities, dates and ticket prices. We want to not only display these blocks, but also add a s we will offer the user to choose the city of flight himself (offer):

```
![img](/img/attachments/37847460/37847465.png)
```

Our steps:

- **Create an array where we add information about all five offers ("item1", "item2", "item3", "item4", "it**
- **Separately add the first element of the array (item1).**
- **Add a block with an offer.**
- **Add the remaining elements of the array, distributing them into 2 columns. The first element will not we want to skip one element.**

```
<table>
  <tr>
    <td colspan="2">{for limit(array["item1"] 1)}{.}{else}{cancel}{end}</td>
  </tr>
  {if lead.offer}
  <tr>
    <td colspan="2">{lead.offer}</td>
  </tr>
  {end}
{for $i $row = columns(array["item1" "item2" "item3" "item4" "item5" "item6 hidden"] 2 1)}
  <tr>
  {for $row}
    <td>{.}</td>
  {end}
  </tr>
{end}
</table>
```

So, creating tables based on array elements is a flexible mechanism that allows you to easily change the message.

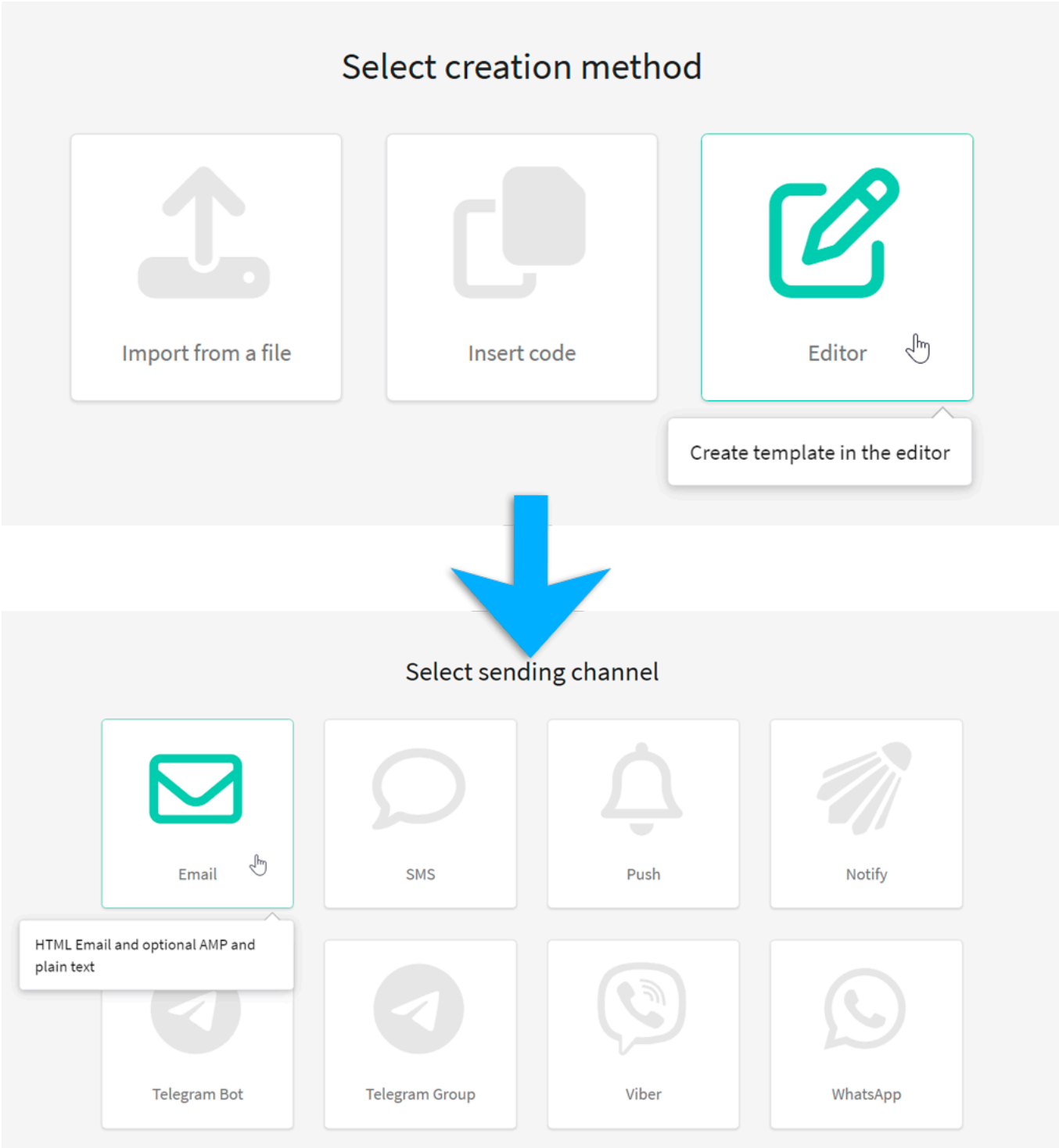
Block editor for email template

Block editor for email template is a tool for creating, editing and customizing marketing email templates designed for marketers and designers who want to quickly assemble professional emails with an easy-to elements and design customizations.

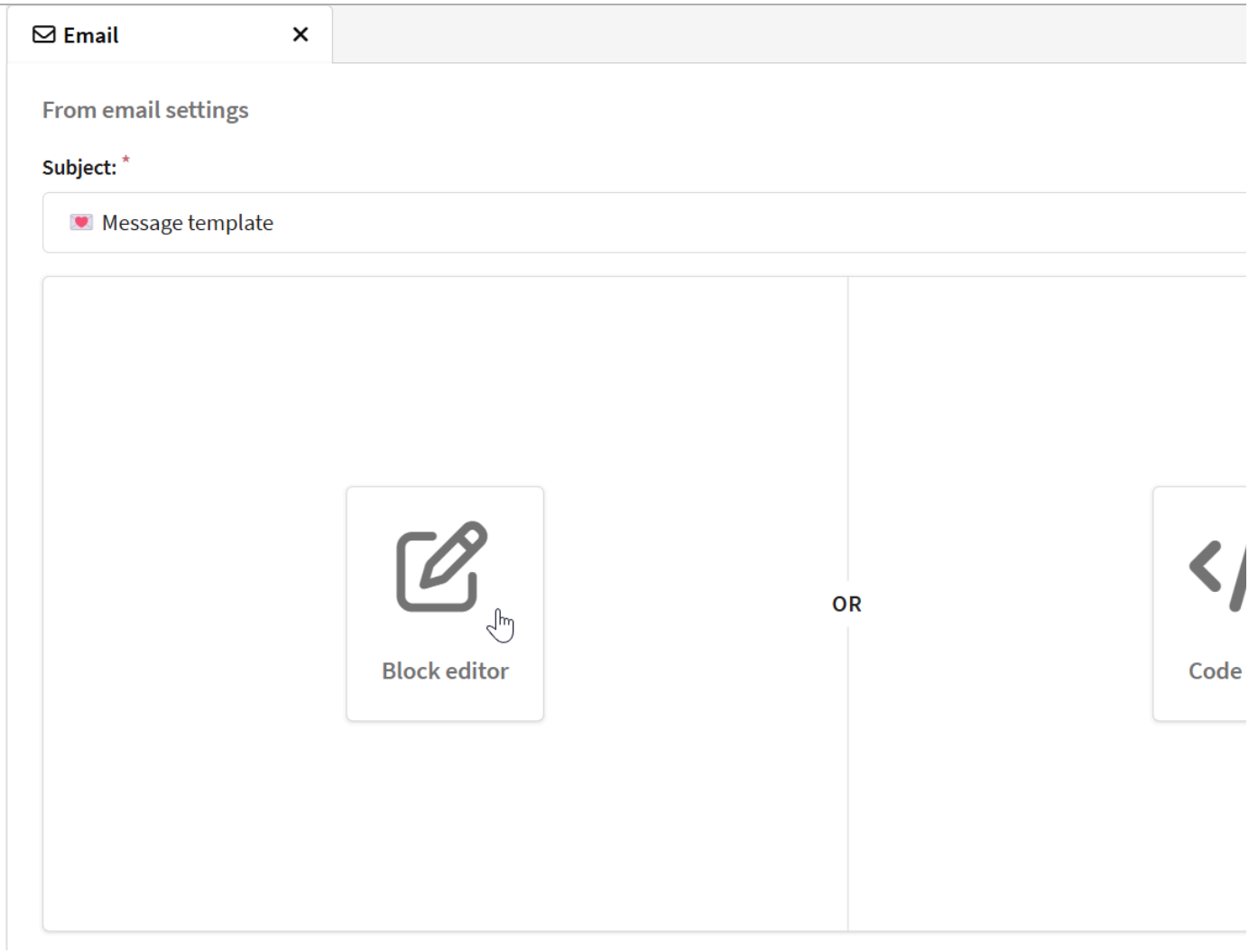
Detailed instructions, usage examples and tips on how to use the editor are available in [a documentatio](#)

Getting started

In the main menu, open the Templates section and create a new message template. Select Editor —> Em



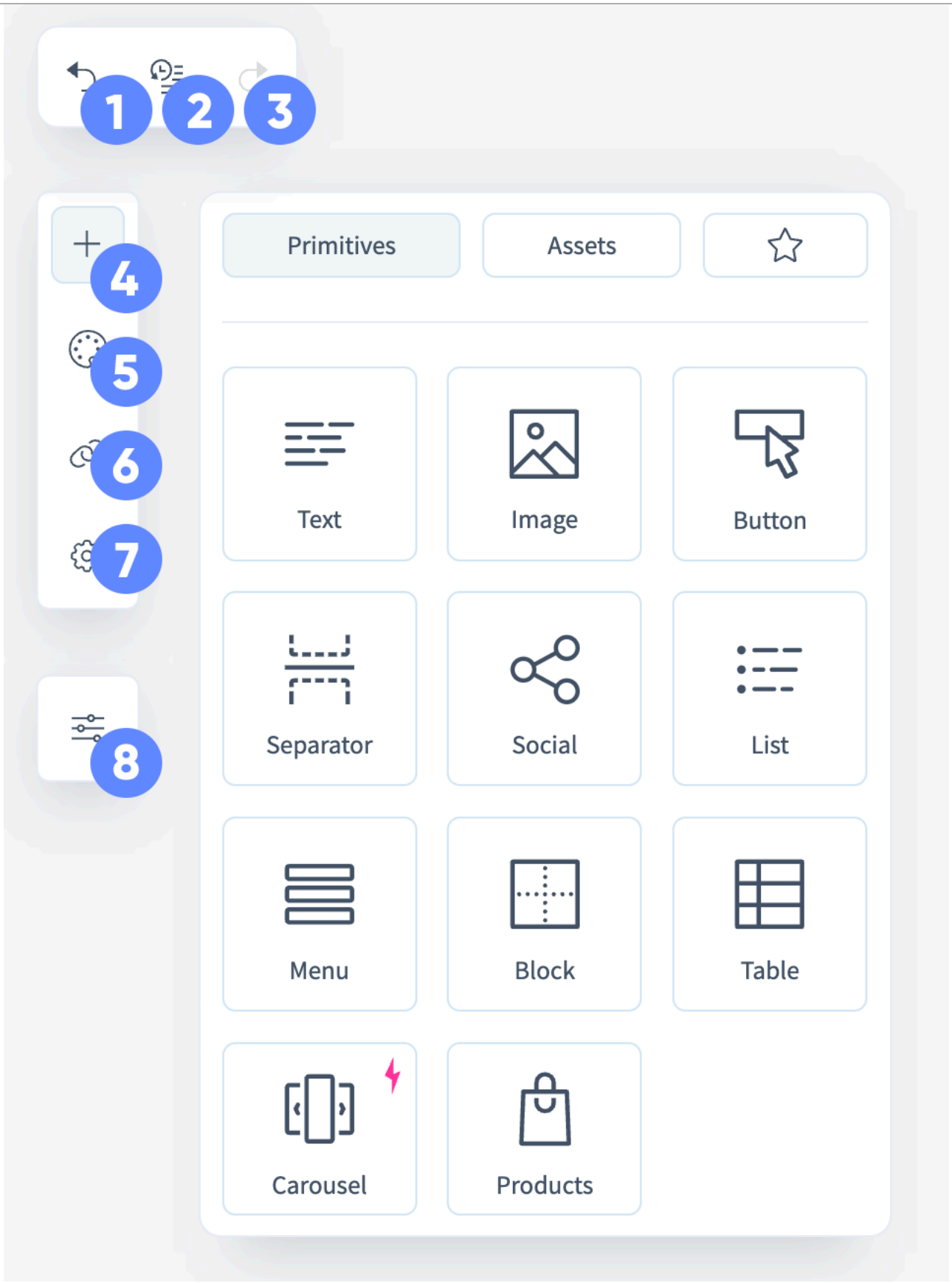
You will see an email template configuration window, select Block Editor.



Editing templates and interface

Toolbar

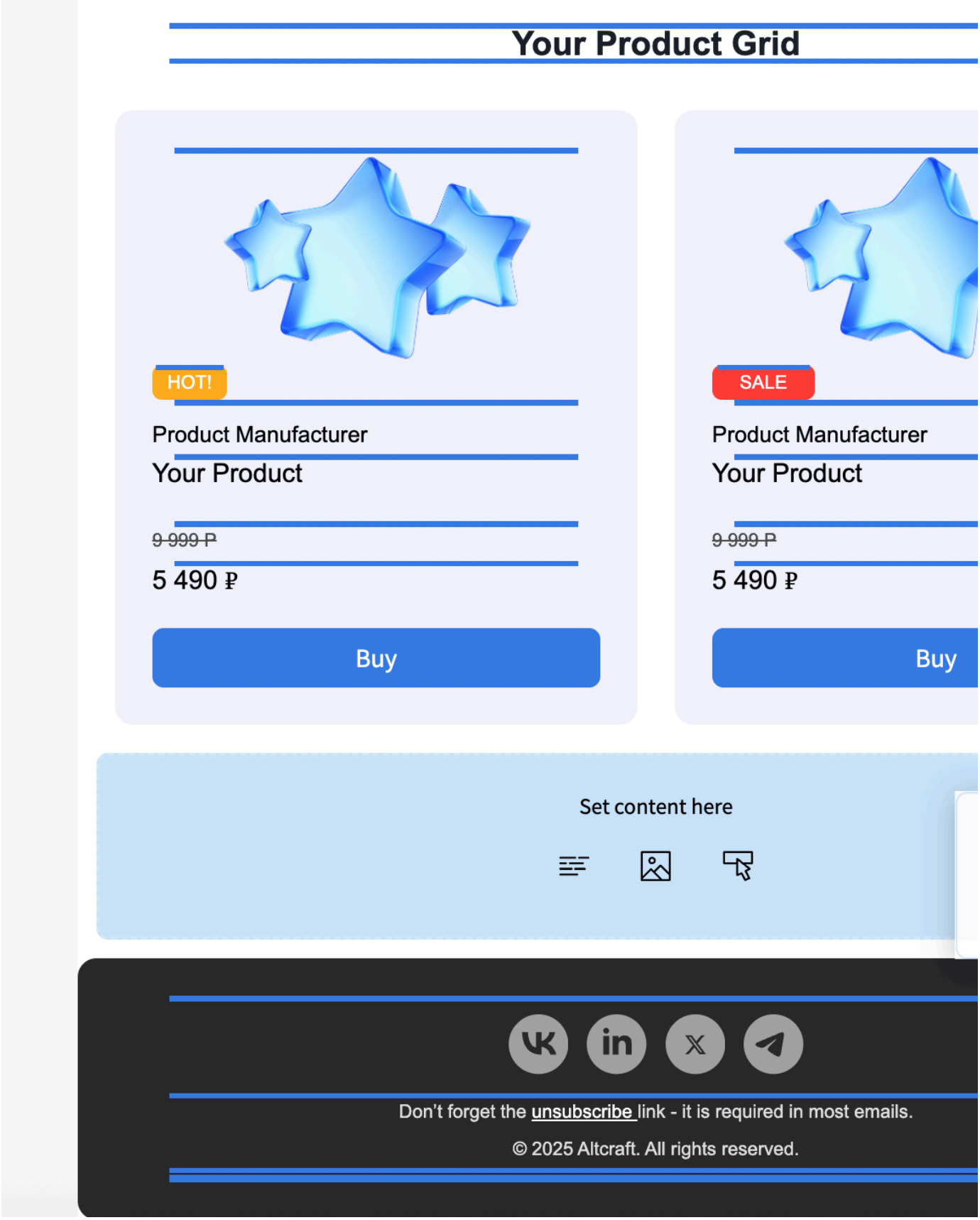
The toolbar contains the following controls:



1. Undo — cancels the last performed action;
2. Action history — opens the list of all changes and allows you to revert to any previous version;
3. Redo — repeats the previously undone action;
4. Add — opens the menu of primitives and ready-made blocks for inserting into the template;
5. Styles — panel for customizing the design of the template;
6. Link Map — displays all links used in the template for review and editing;
7. Parameters — theme and preheader settings for the email;

8. Options — general editor parameters (interface theme and property panel position).

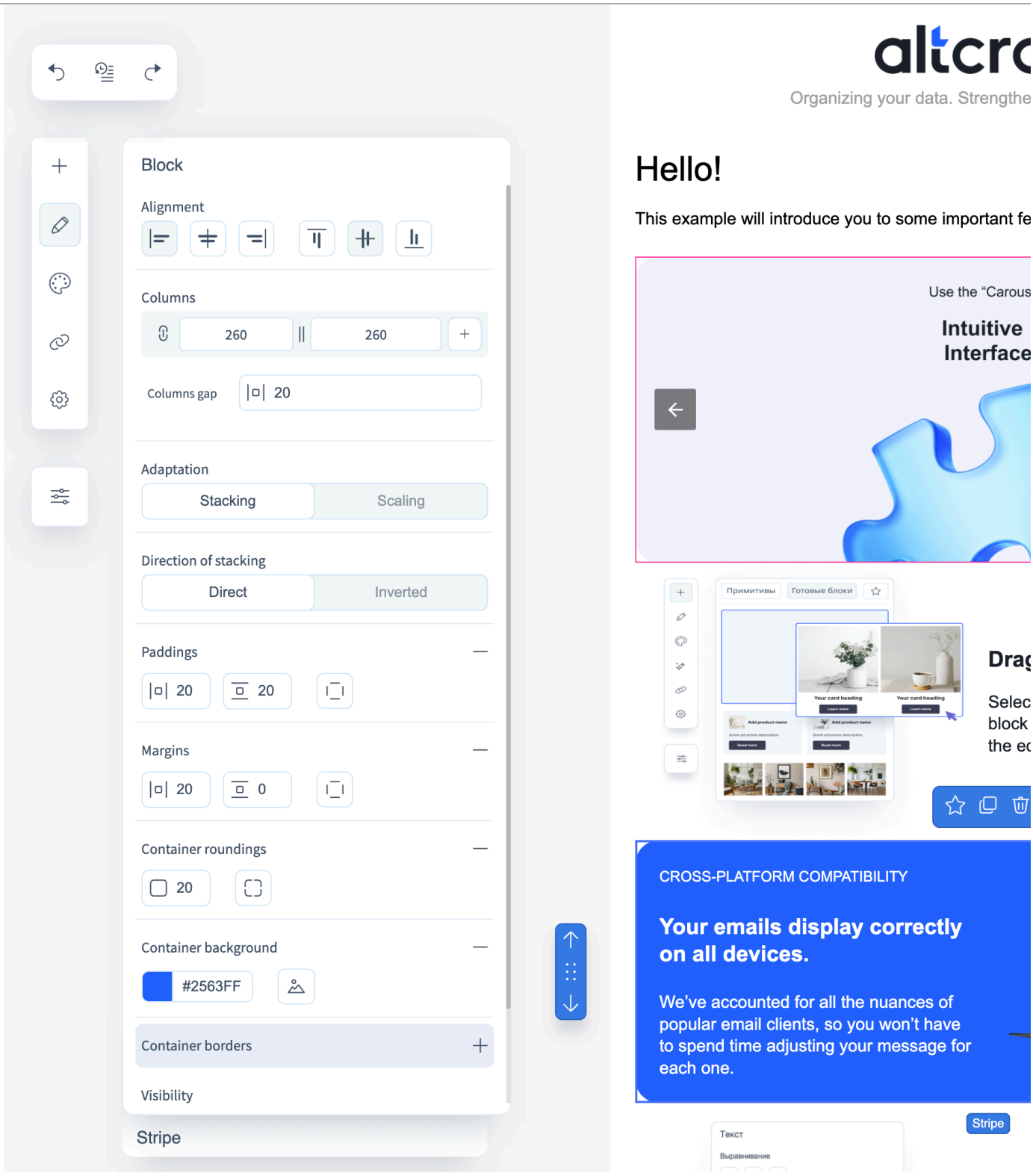
Editing Area



The main part of the editor where the email is assembled. Blocks can be dragged here from the toolbar. modified directly.

Blocks and Elements

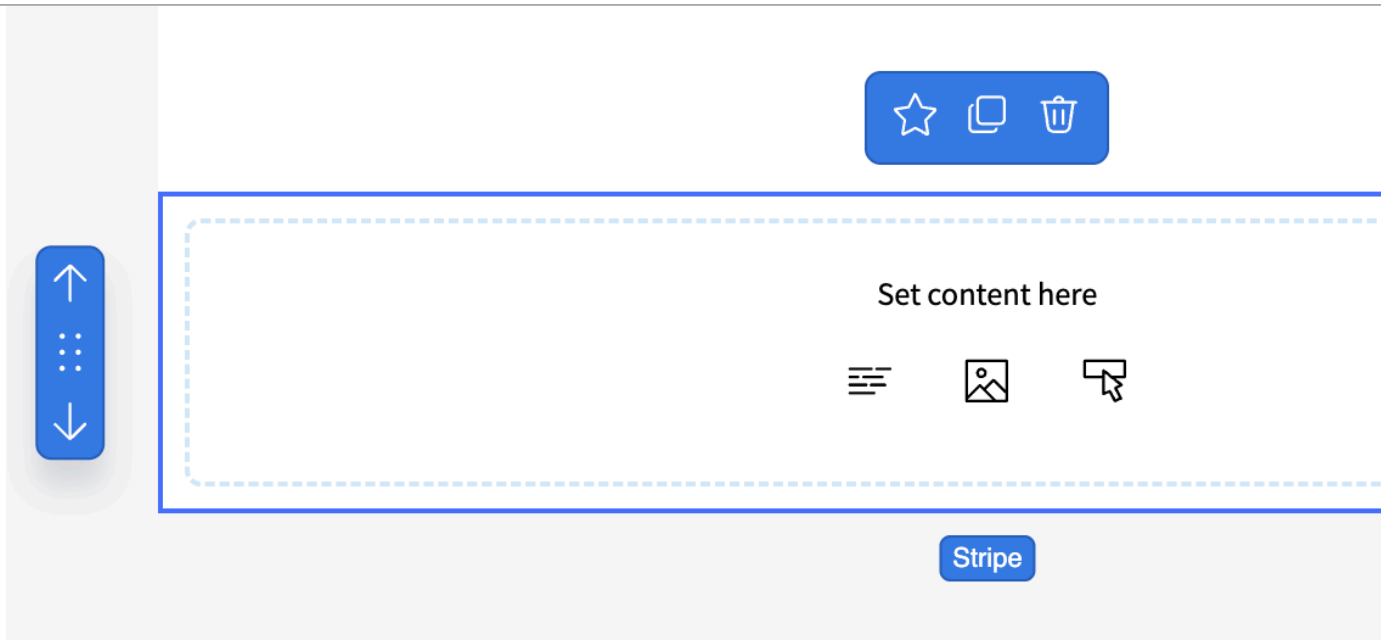
Block



For a content block, the following parameters can be configured:

- **Alignment — alignment of the block to the left, right, or center of the email;**
- **Columns — set the number of columns, their width, and spacing;**
- **Adaptation — available if the block has more than one column. Defines how the columns appear on n vertically, maintaining size;**
- **Direction of stacking — sets the column order if vertical stacking is selected;**
- **Paddings — inner spacing between block content and its borders;**
- **Margins — outer spacing between the block’s borders and the email layout;**
- **Container roundings — controls corner rounding;**
- **Container background — choose a background color or upload an image;**
- **Container border — set the color, thickness, and border style;**
- **Visibility — determines visibility in different versions of the email (HTML/AMP/everywhere, desktop/n**

You can add content to an empty block by clicking the respective icons inside the element:



Column properties can also be customized:

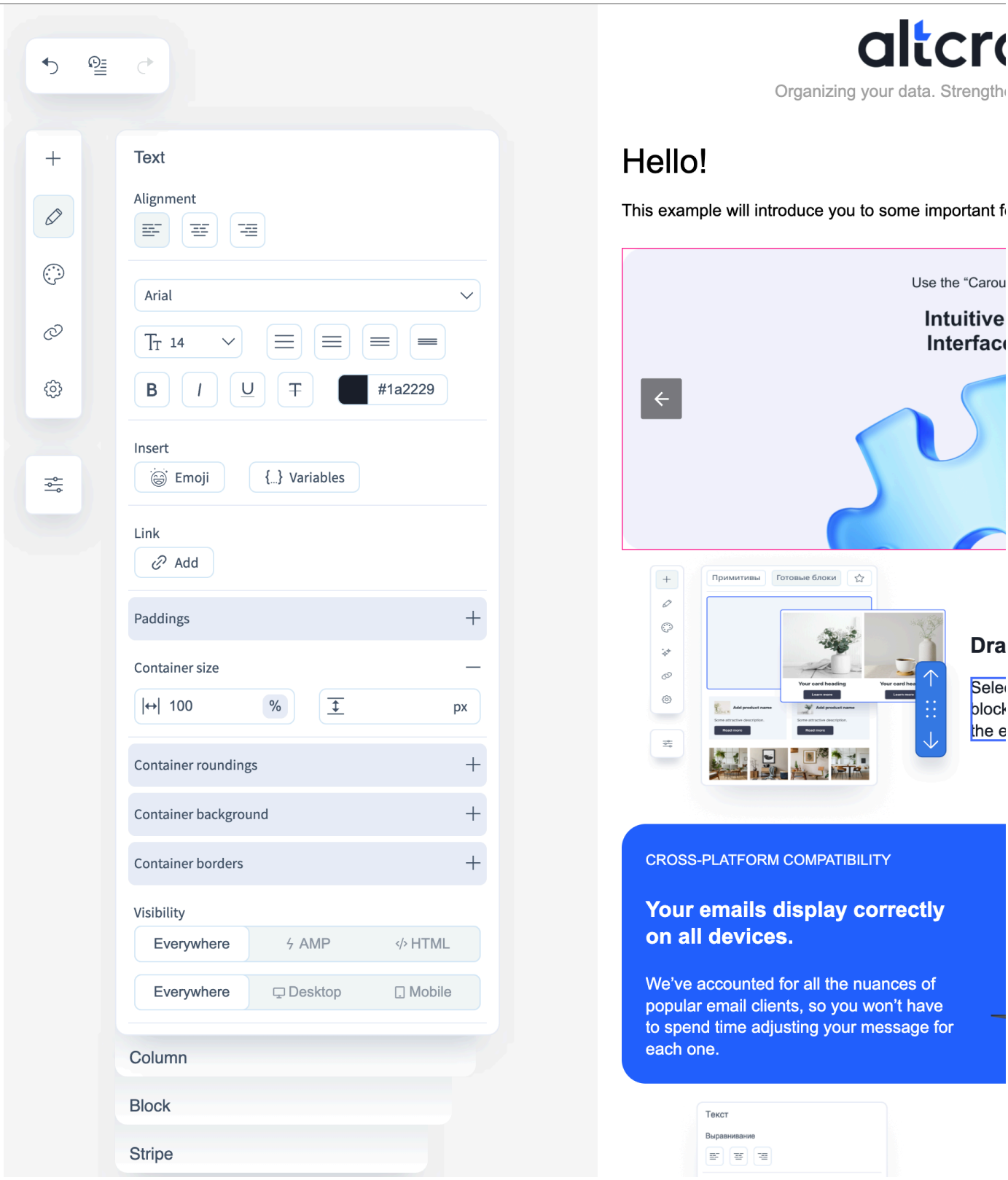
- **Alignment;**
- **Padding;**
- **Column background;**
- **Column rounding;**
- **Column border;**
- **Column size.**

For the row containing the block, you can configure:

- **Alignment;**
- **Background;**
- **Content width;**
- **Content rounding;**
- **Content borders;**
- **Padding.**

Some row parameters affect the appearance of the entire email.

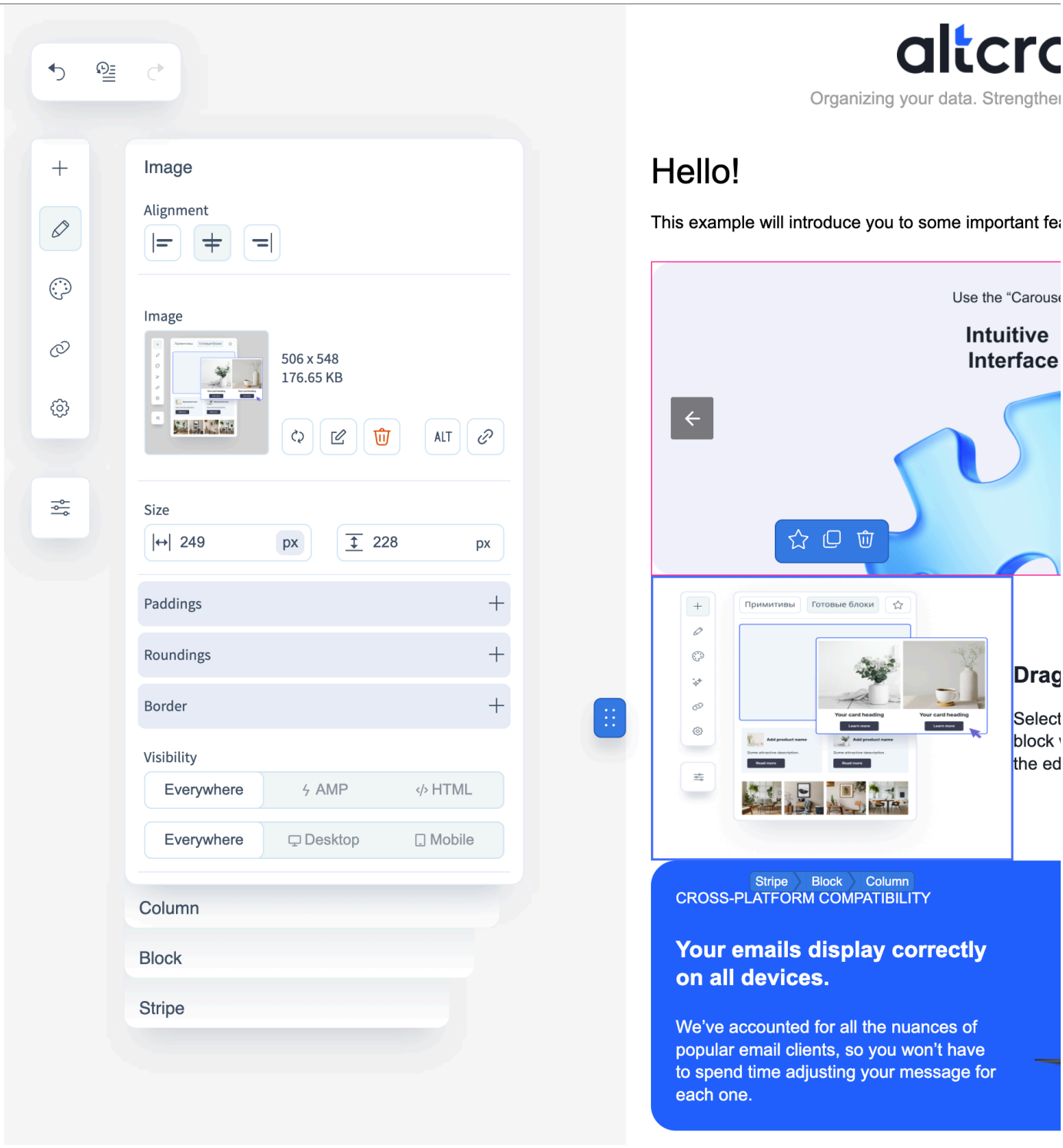
Text



Configurable properties for a text block:

- **Alignment of text within the container;**
- **Font, size, style, and color;**
- **Insertion of emojis and variables (variables supported by popular email services);**
- **Insert links in text;**
- **Padding inside the block;**
- **Size, rounding, background, and border of the container;**
- **Visibility — determines block visibility in different email versions (HTML/AMP/everywhere, desktop/m**

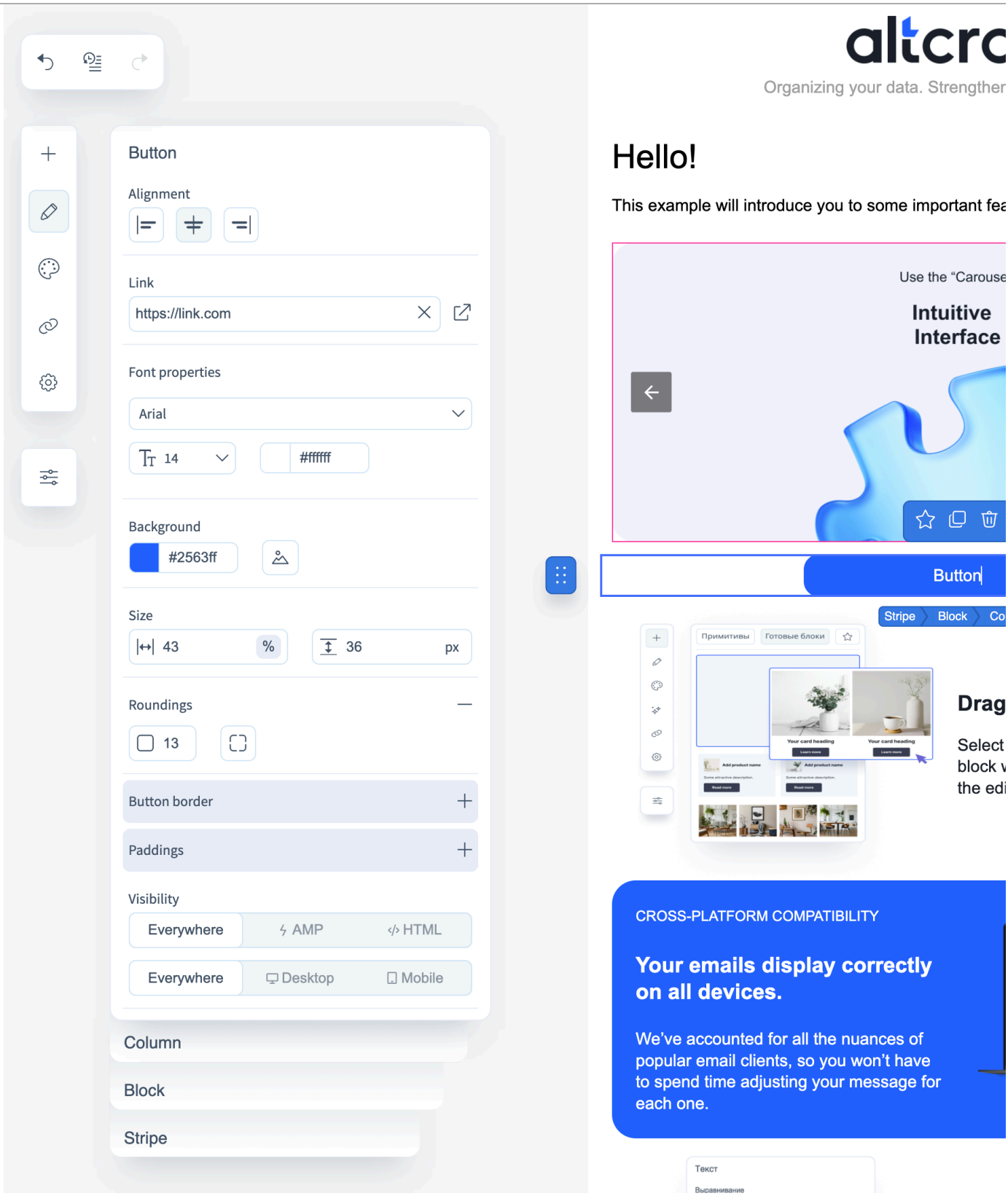
Image



Configurable properties for an image block:

- **Alignment of the image within the container;**
- **Image size;**
- **Padding within the block;**
- **Container rounding and its borders;**
- **Visibility — determines block visibility in different versions of the email (HTML/AMP/everywhere, des**

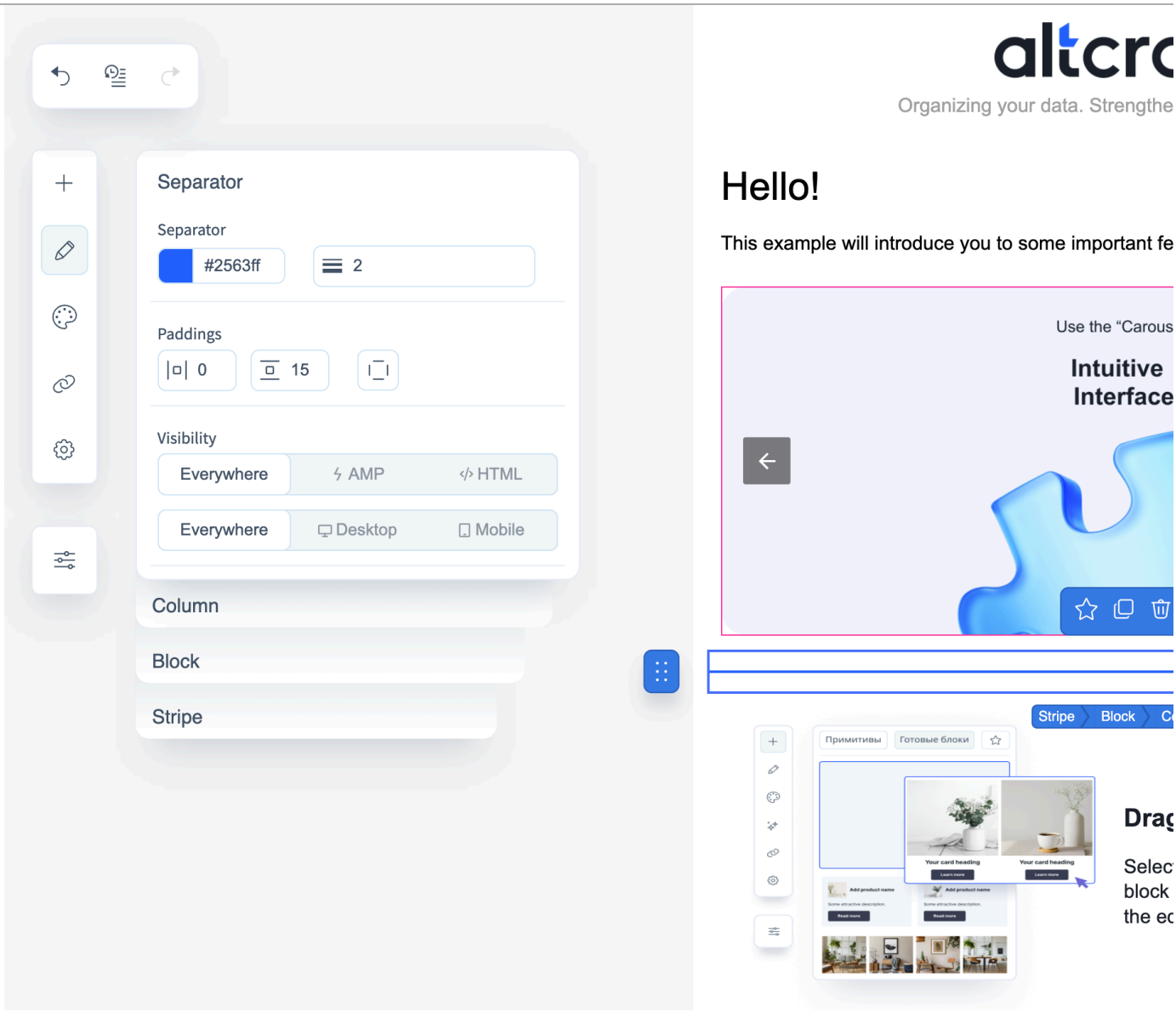
Button



Configurable properties for buttons:

- **Alignment of the button within the container;**
- **Link to follow when clicked;**
- **Font, size, and color of the text;**
- **Background of the button (you can upload an image);**
- **Size, style, and corner rounding of borders, as well as padding within the container;**
- **Visibility — determines block visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

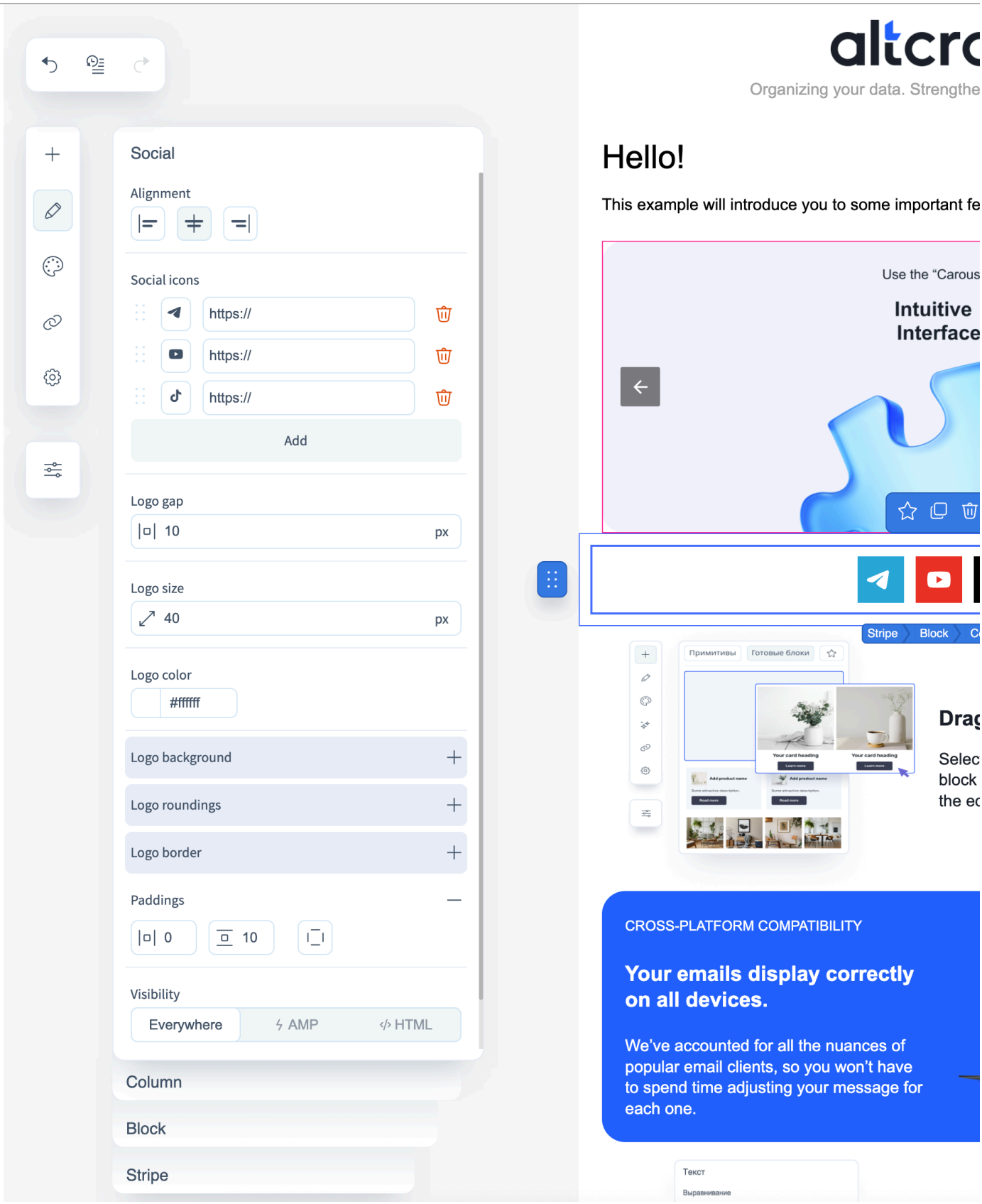
Separator



A separator is an empty block with a horizontal line inside. Configurable options:

- **Color and thickness of the separator;**
- **Padding inside the container;**
- **Visibility — determines visibility in different email versions (HTML/AMP/everywhere, desktop/mobile).**

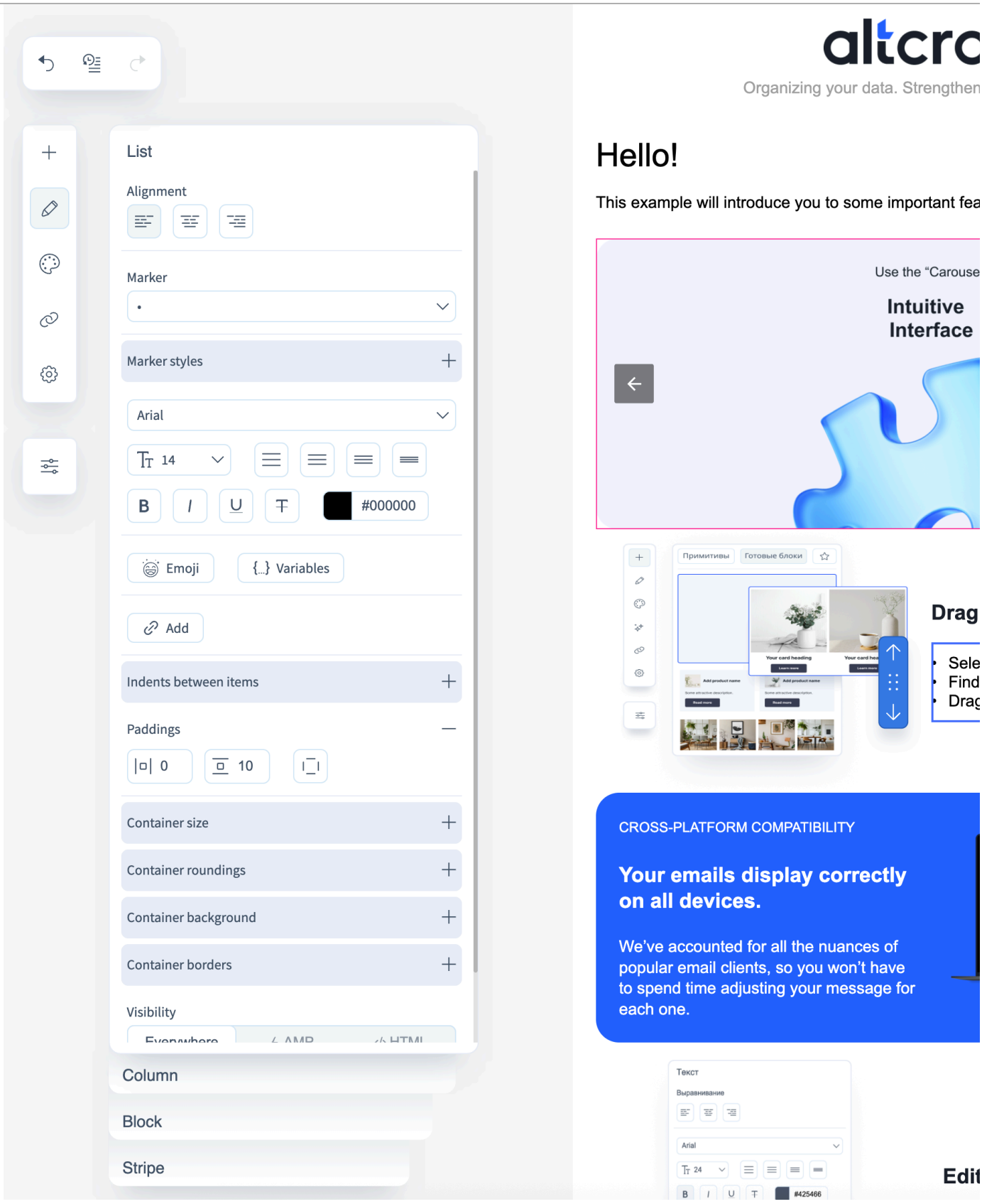
Social Networks



Social Networks — a block with clickable icons of popular social networks. Configurable options:

- **Alignment within the container;**
- **Social network settings — add or remove icons, change their order, and configure click links. You ca logo colors for each network;**
- **Spacing between logos, logo size;**
- **Logo color and background color — customizable per network;**
- **Padding inside the container, corner rounding, and borders of logos;**
- **Visibility — determines block visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

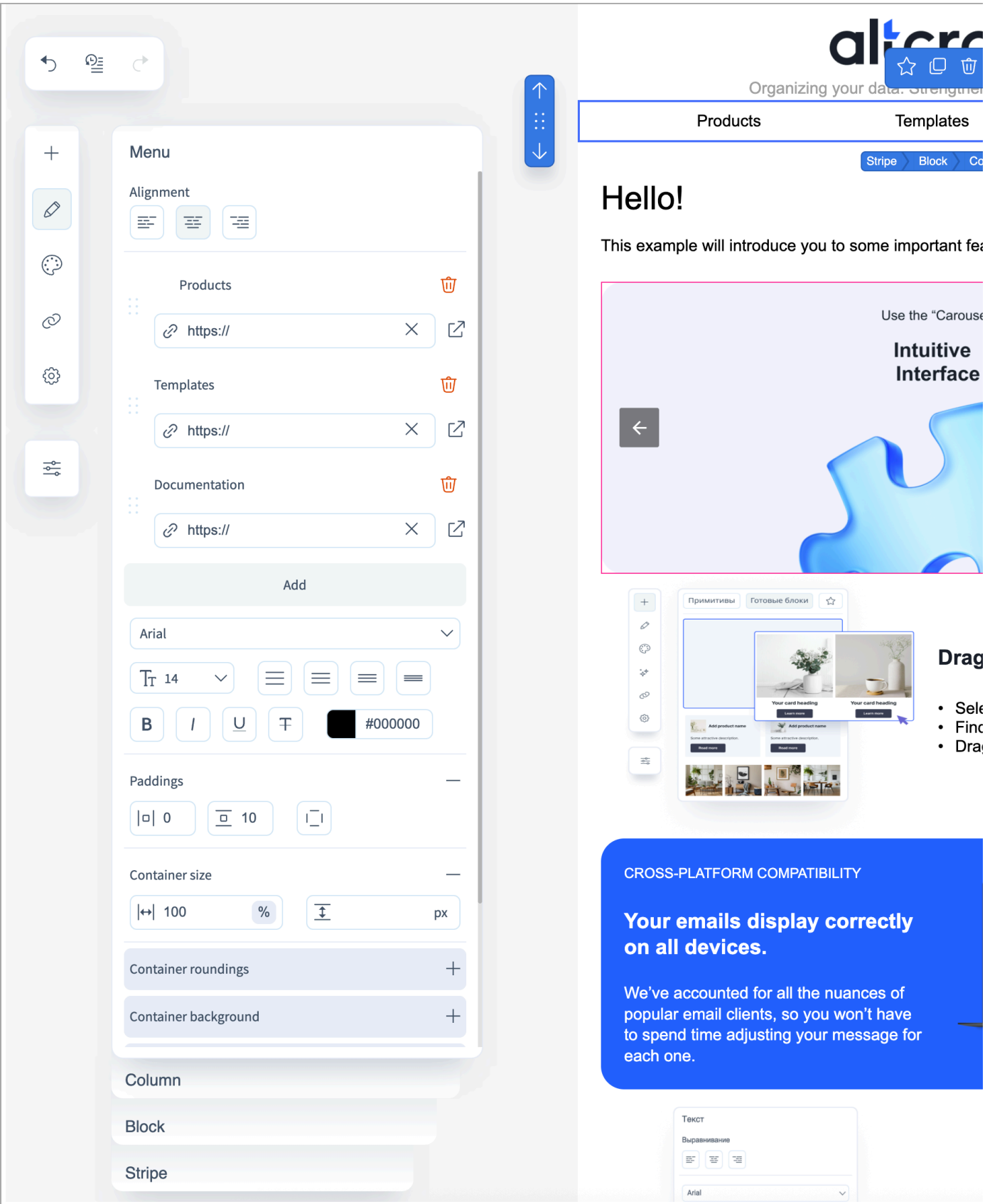
List



Configurable properties for lists:

- **Alignment within the container;**
- **Marker style, as well as size, color, and font (if a numeric marker is selected);**
- **Container size and background, border style and rounding, plus padding inside the container;**
- **Text properties — size, color, font, style, and insertion of variables and emojis. Links can also be ins**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

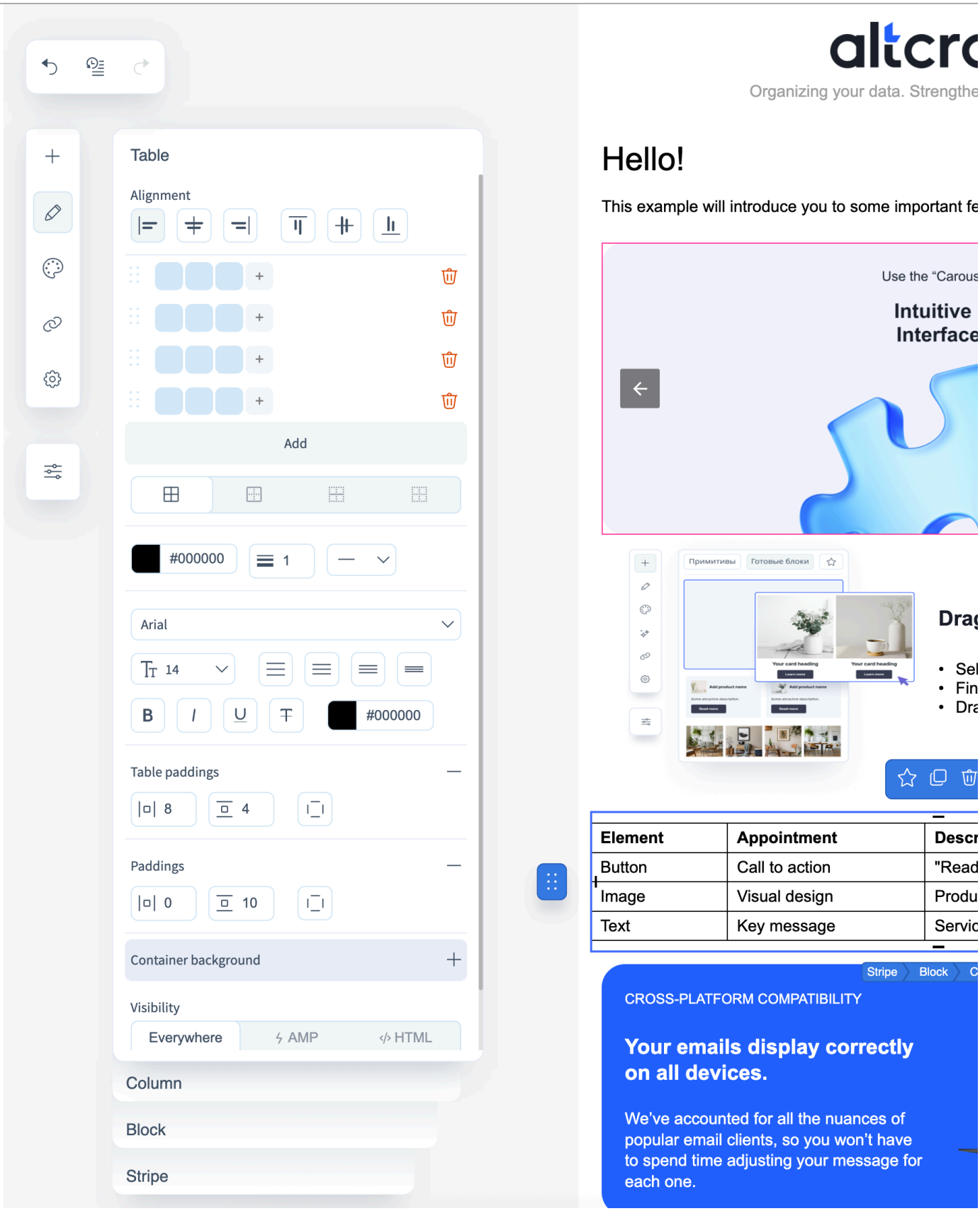
Menu



Configurable properties for menus:

- **Alignment within the container;**
- **Menu items and their order — you can specify both the text and a link for each;**
- **Text properties — size, color, font, style;**
- **Container size, background, border style, rounding, and padding;**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

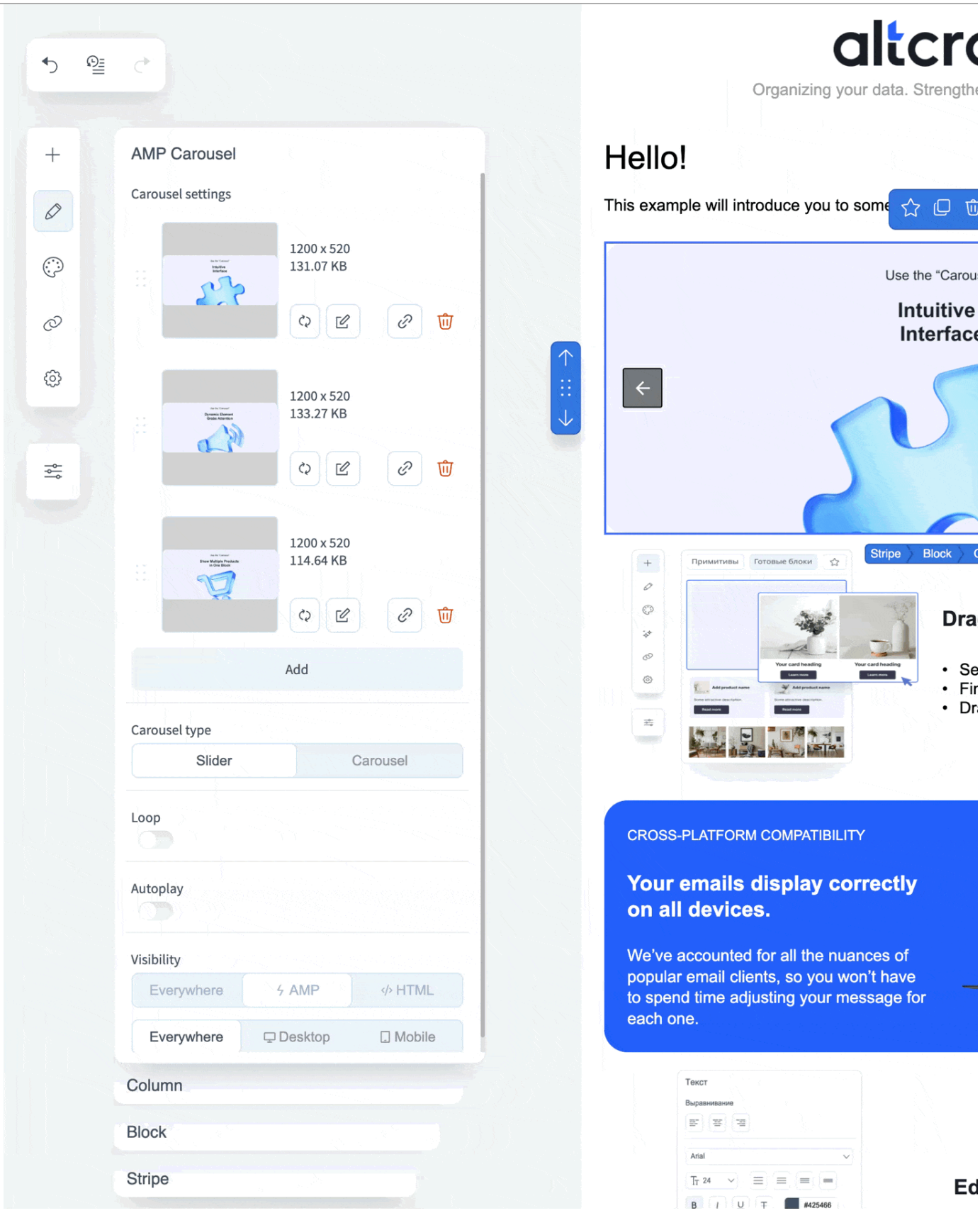
Table



Configurable properties for tables:

- **Text alignment within the table;**
- **Table size and border style/thickness between cells;**
- **Text properties — size, color, font, style;**
- **Padding inside cells;**
- **Padding inside the container;**
- **Container background;**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

Carousel

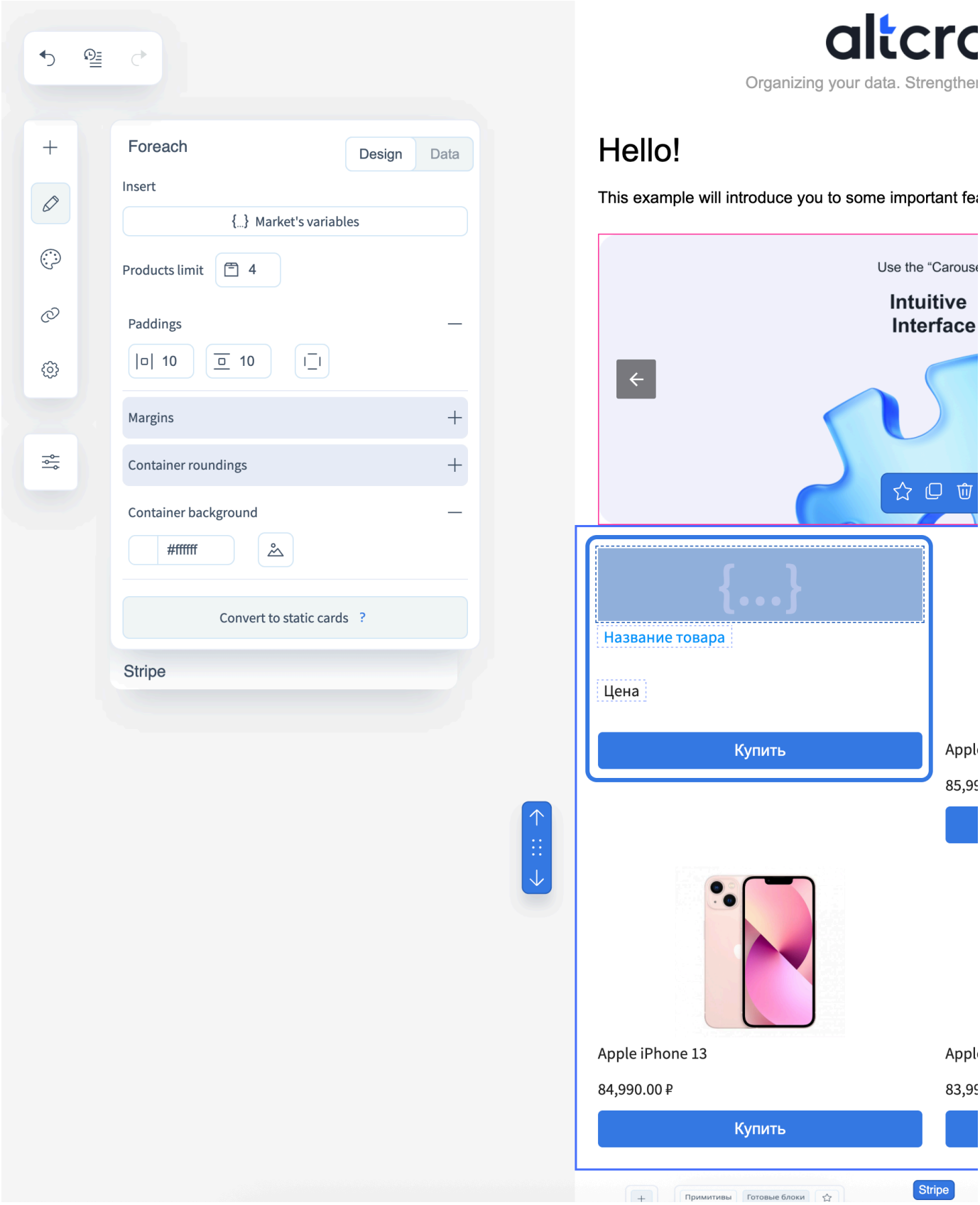


Carousel — an AMP component consisting of several scrollable images. Configurable options:

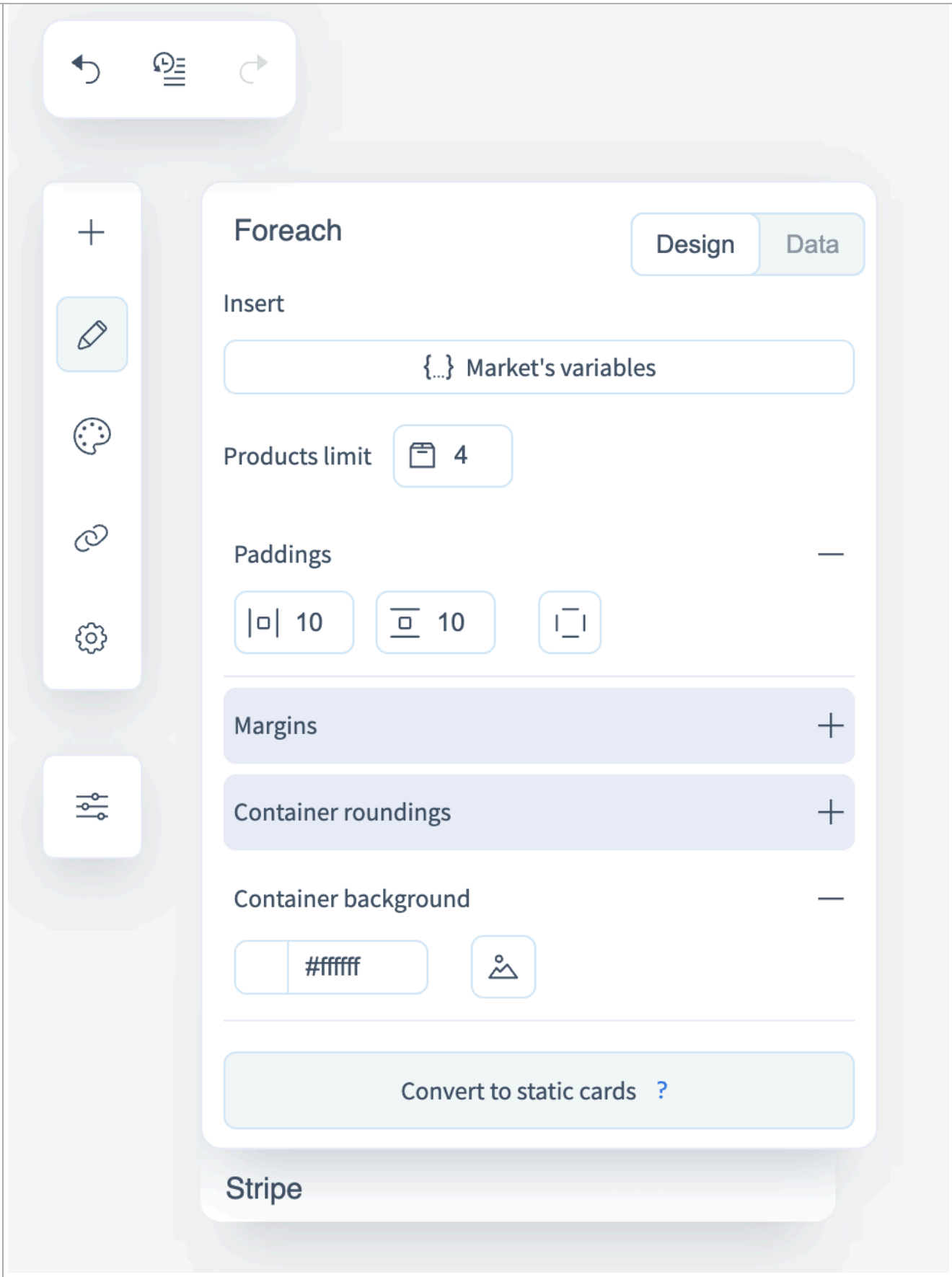
- **Images included in the carousel. You can upload from your device, choose from the Pixcraft gallery,**
- **Carousel type — slider carousel (displays one image) or classic carousel (displays multiple images);**
- **For slider carousels, you can enable looping and auto-scroll after a set delay;**
- **For classic carousels, configure the height of the images;**
- **Visibility — determines visibility in different versions (desktop/mobile). Since this is an AMP compone emails.**

Products

The Products block is used for dynamically generating product sections in newsletters — for example, se recommended products synchronized with up-to-date data from the CDP.



Design

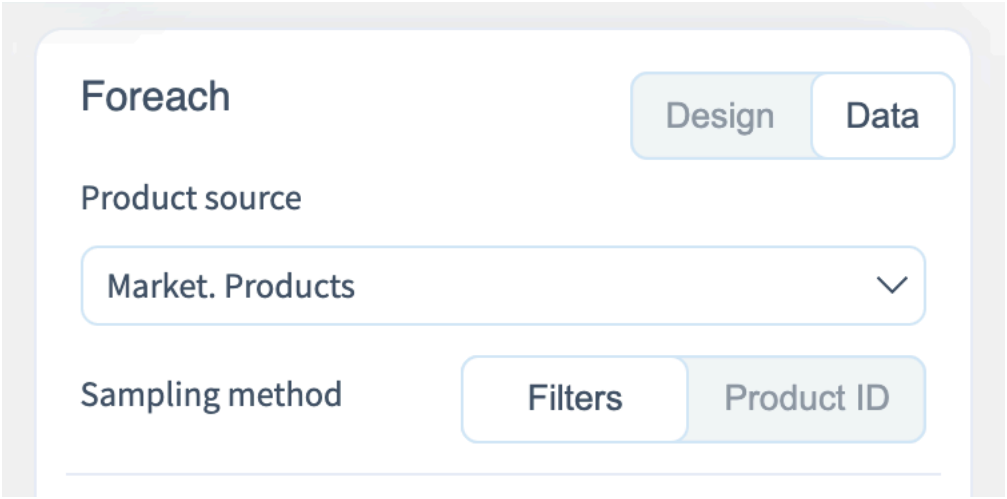


The Design tab controls the visual appearance of product cards and includes the following properties:

- **Market variables — define which elements are displayed in the card: name, price, image, button, etc.**
- **Number of products — sets how many product cards are displayed per row.**
- **Padding — defines inner and outer spacing around the cards.**
- **Container rounding — sets the corner radius of the cards.**
- **Container background — defines the color or image of the card background.**

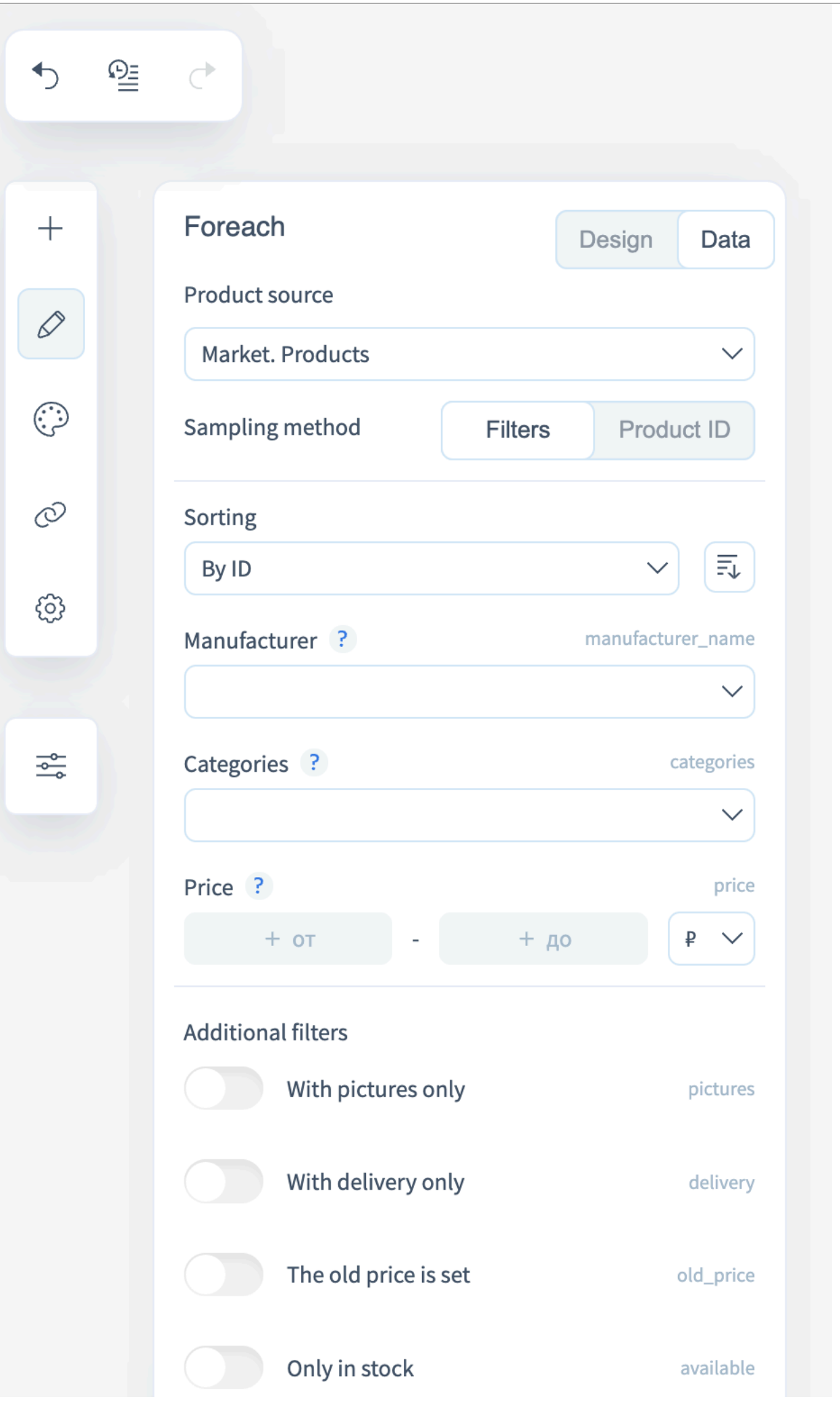
All changes made in this tab are applied to all cards within the block.

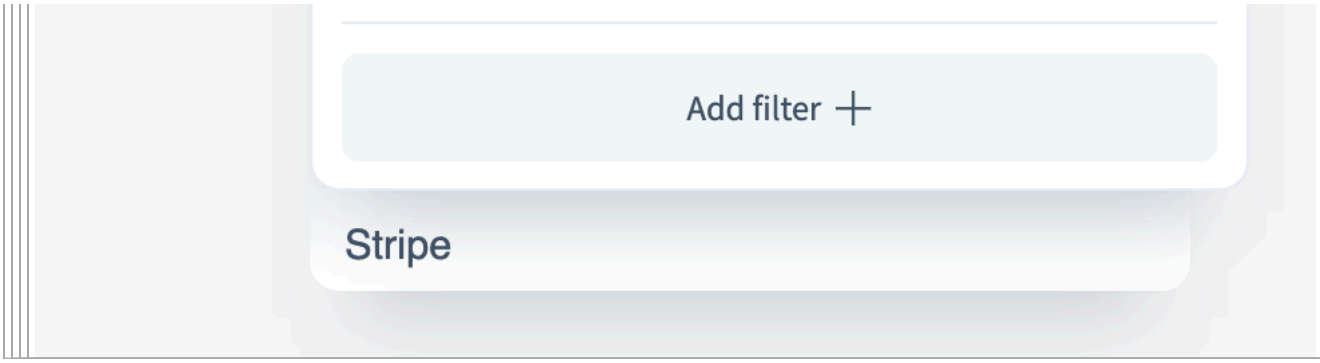
Data



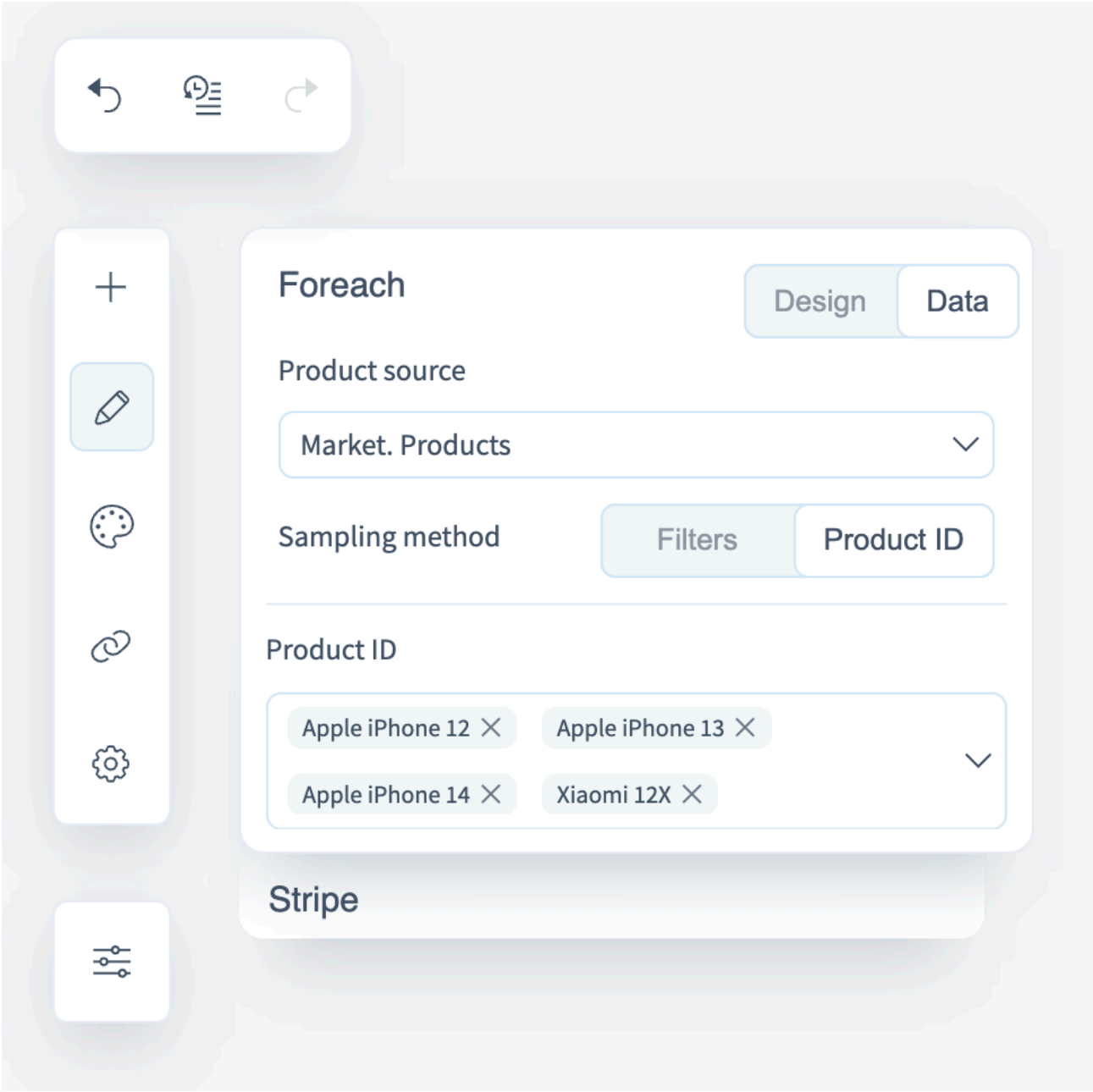
The Data tab is used to configure product sources and data selection parameters from Altcraft CDP. It in

- **Source — select the data source from which products are loaded.**
- **Selection method — defines how products are retrieved:**
  - **Filters — dynamic product selection mode based on defined criteria (sorting, manufacturer, cate available: only with images, only with delivery, has old price, in stock only.**


Foreach
Design
Data
Product source
Market. Products
Sampling method
Filters
Product ID
Sorting
By ID
Manufacturer
manufacturer_name
Categories
categories
Price
price
+ от
-
+ до
₽
Additional filters
With pictures only
pictures
With delivery only
delivery
The old price is set
old_price
Only in stock
available

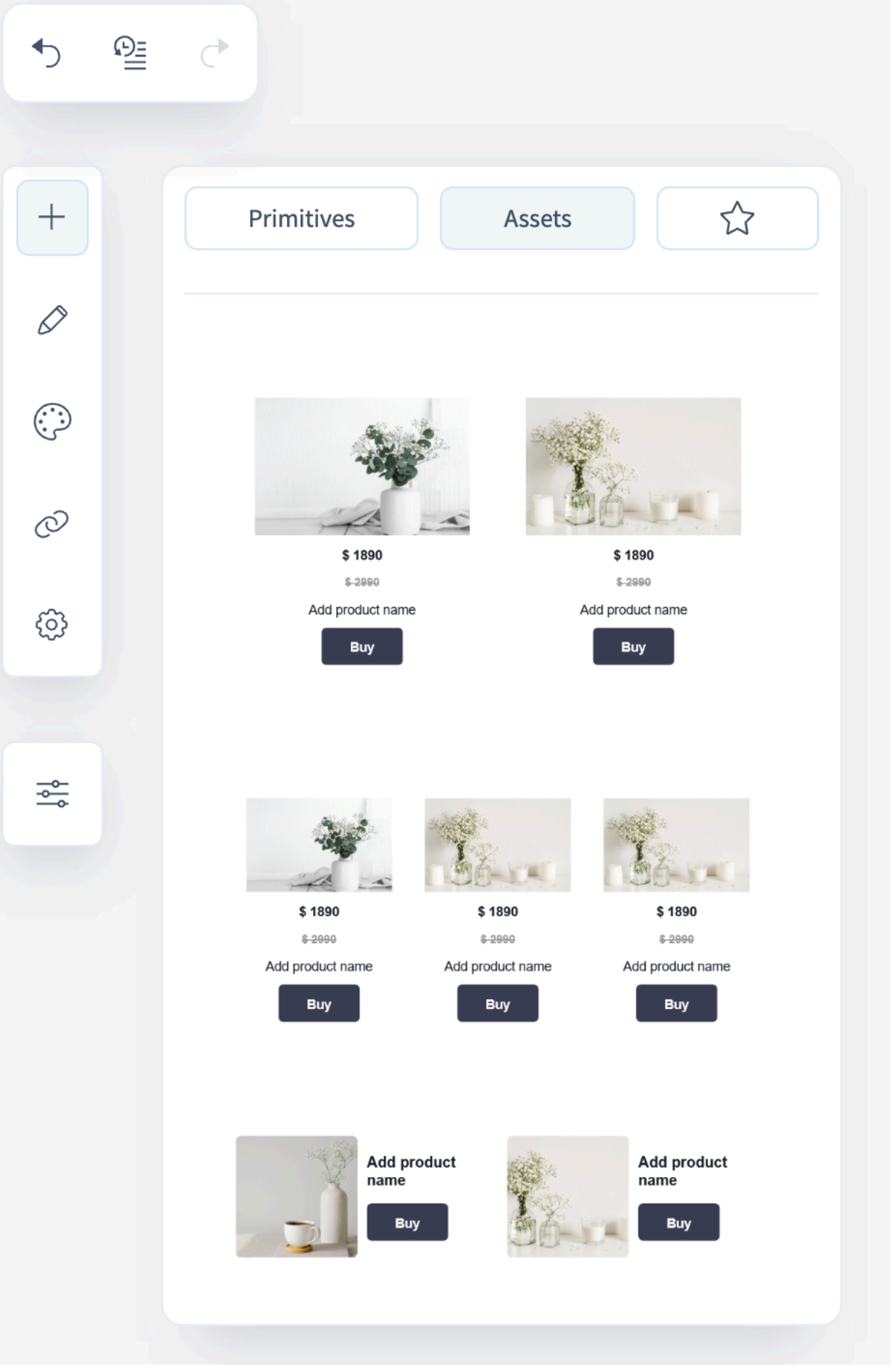

Add filter
Stripe

- **Product ID — manual selection mode by specifying specific product identifiers. Suitable for displayin depend on inventory updates.**



Assets

A collection of pre-designed blocks for various purposes. Using them, you can assemble a basic email la



Saved Blocks

If you frequently use a specific component or want to apply consistent styled blocks across different em

Primitives

Assets

Use the “Carousel”

Intuitive
Interface

$ 1890

~~$ 2990~~

Add product name

Buy

$ 1890

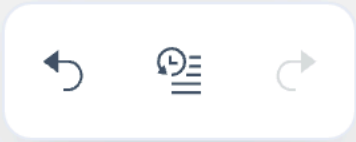
~~$ 2990~~

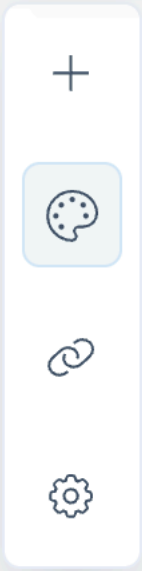
Add product name

Buy

To save a block, click the “star” icon.
After saving, it will always be available in the editor on the respective tab (to the right of the “Ready-Mad

Styles





Styles

Texts | Buttons

Text colors

#a3a3a3

#000000

#1a2229

#ffffff

Text styles

Arial / 14px / 20px
Arial, sans-serif

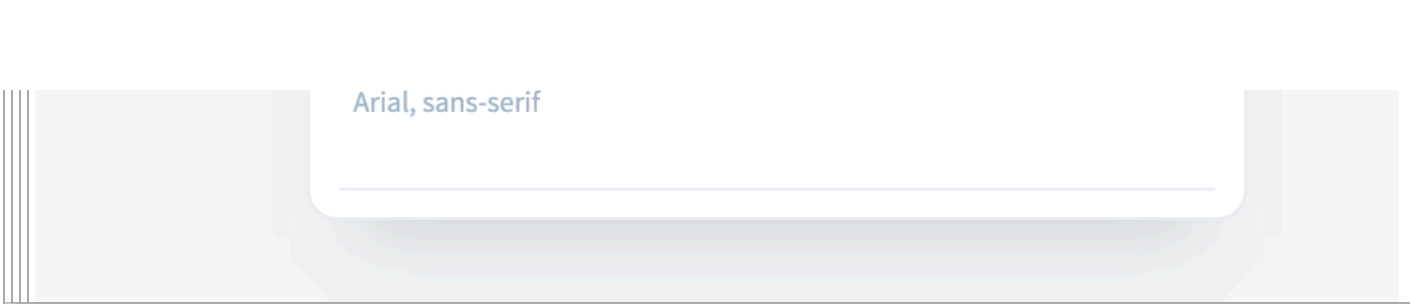
Arial / 28px / 21px
Arial, sans-serif

Arial / 14px / 21px
Arial, sans-serif

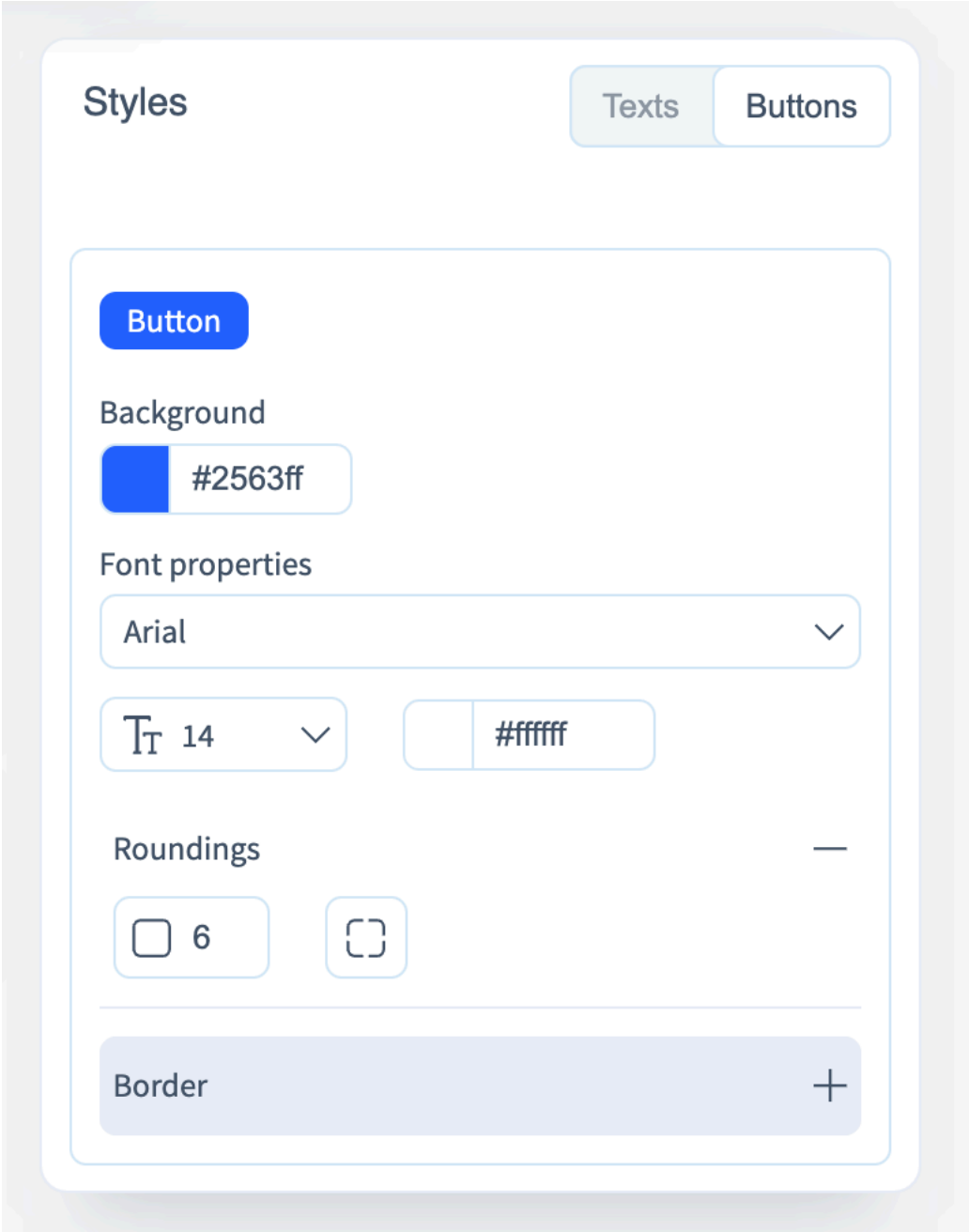
Arial / 18px / 22px
Arial, sans-serif

Arial / 12px / 20px
Arial, sans-serif

Arial / 14px / 18px

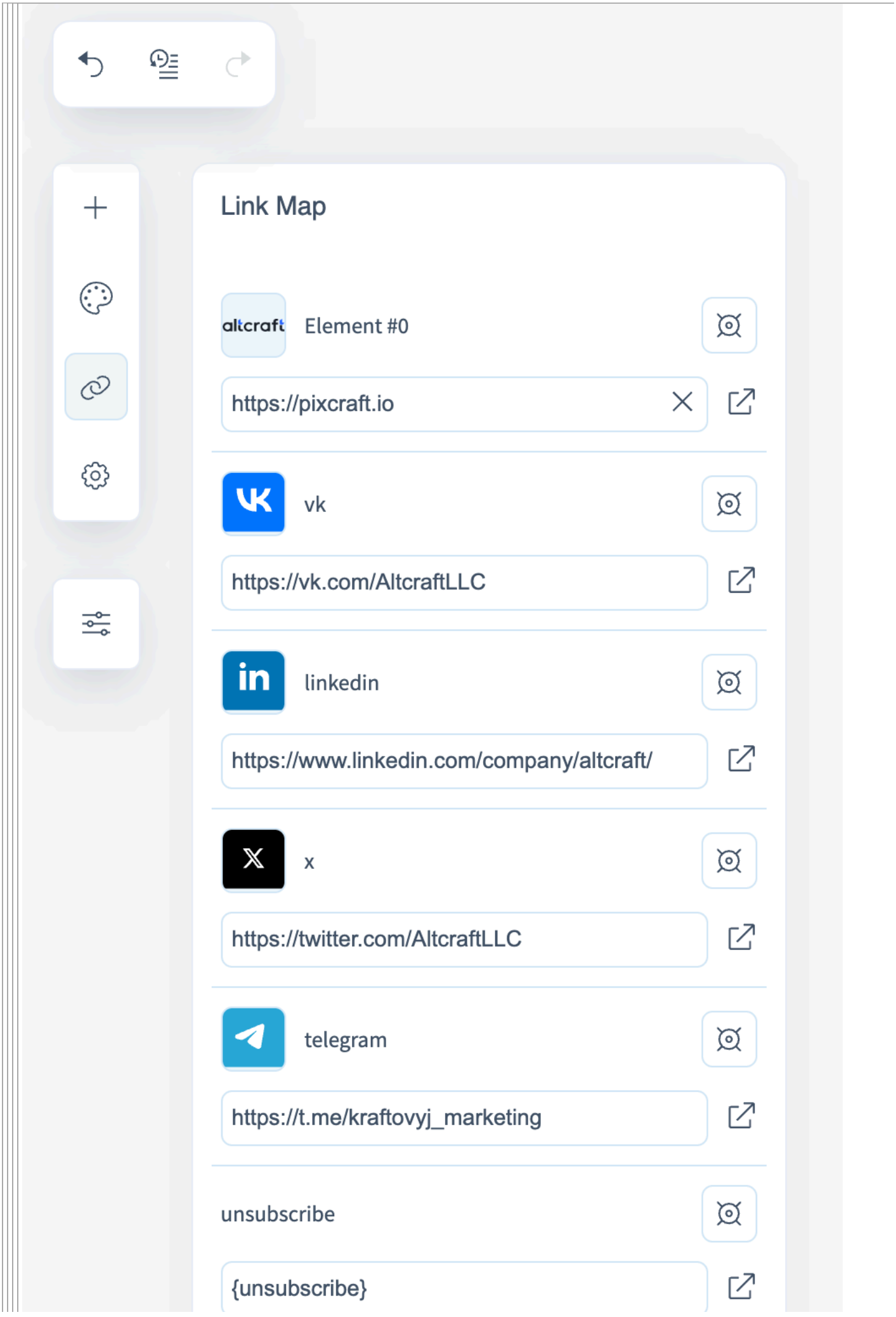

Arial, sans-serif

To maintain visual consistency and follow a unified design system, use the Styles section. Here you can used within blocks and buttons.
In the “Buttons” section, you can view a list of all buttons and their styles. From this menu, you can also button properties:



Next to each style and color, there’s a button that takes you to the corresponding element in the email w

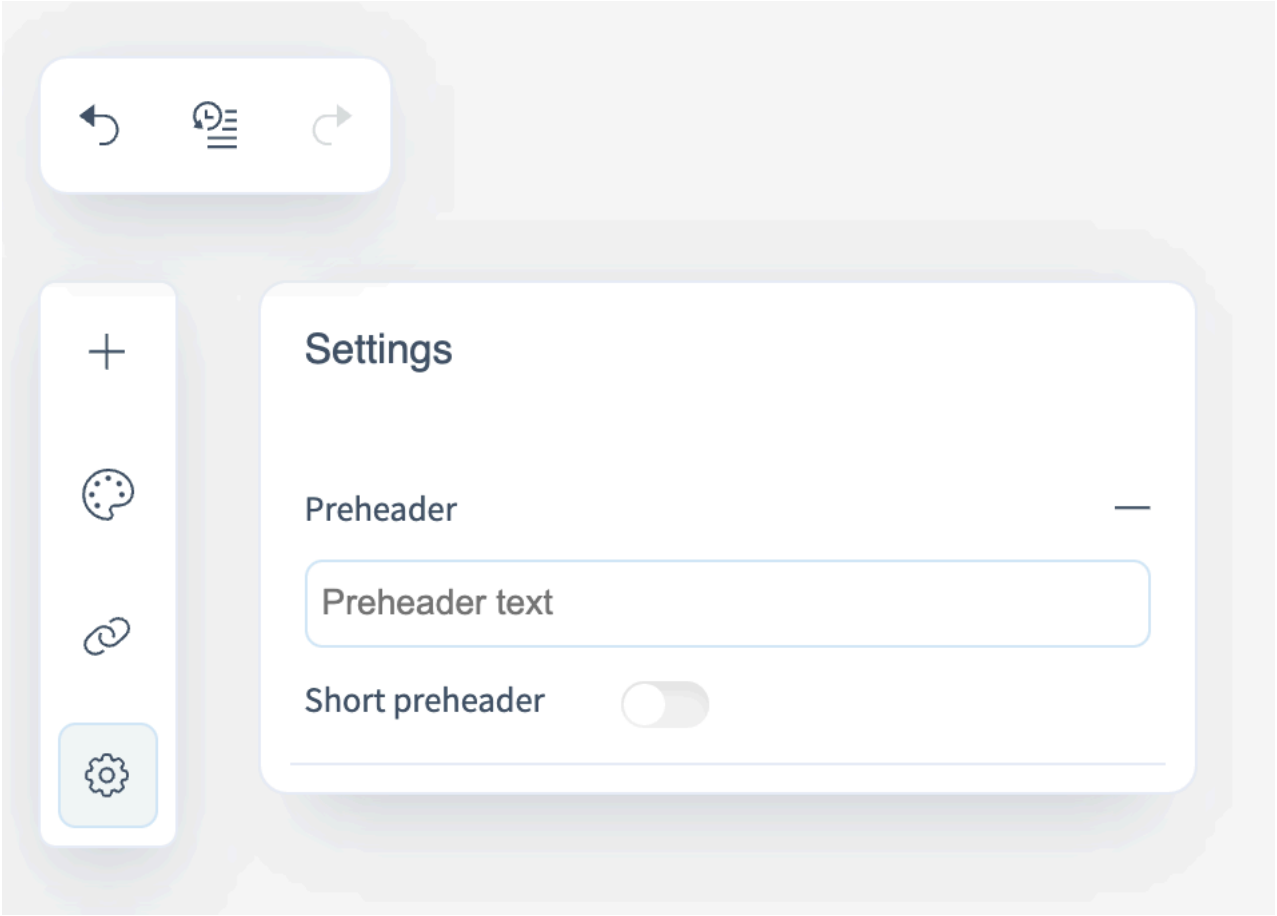
Link Map


Link Map
Element #0
https://pixcraft.io
vk
https://vk.com/AltcraftLLC
linkedin
https://www.linkedin.com/company/altcraft/
x
https://twitter.com/AltcraftLLC
telegram
https://t.me/kraftovyj_marketing
unsubscribe
{unsubscribe}

This section displays all links used in the current email. For each link, the element it's applied to is indic
The following control buttons are available:

- **Open Link — opens the link in a new browser tab to check the destination;**
- **Find Element — highlights the element in the email where the link is used;**
- **Delete — removes the link from both the list and the corresponding element on the canvas.**

Parameters

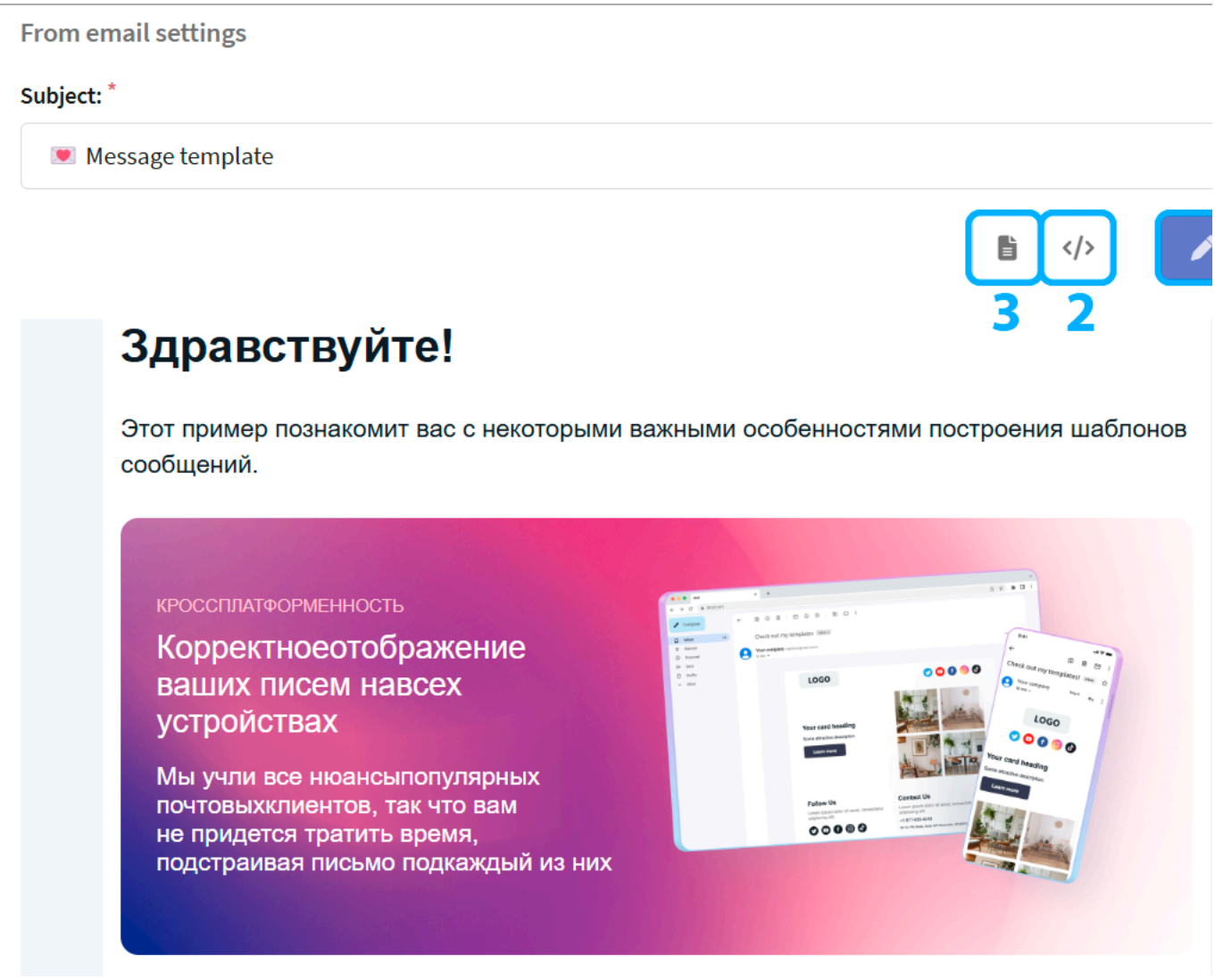


In this section, you can adjust the editor's appearance and layout according to your preferences.

- **Interface theme — choose between light and dark modes.**
- **Property panel position — defines where the properties panel appears. You can place it on the left o preference.**

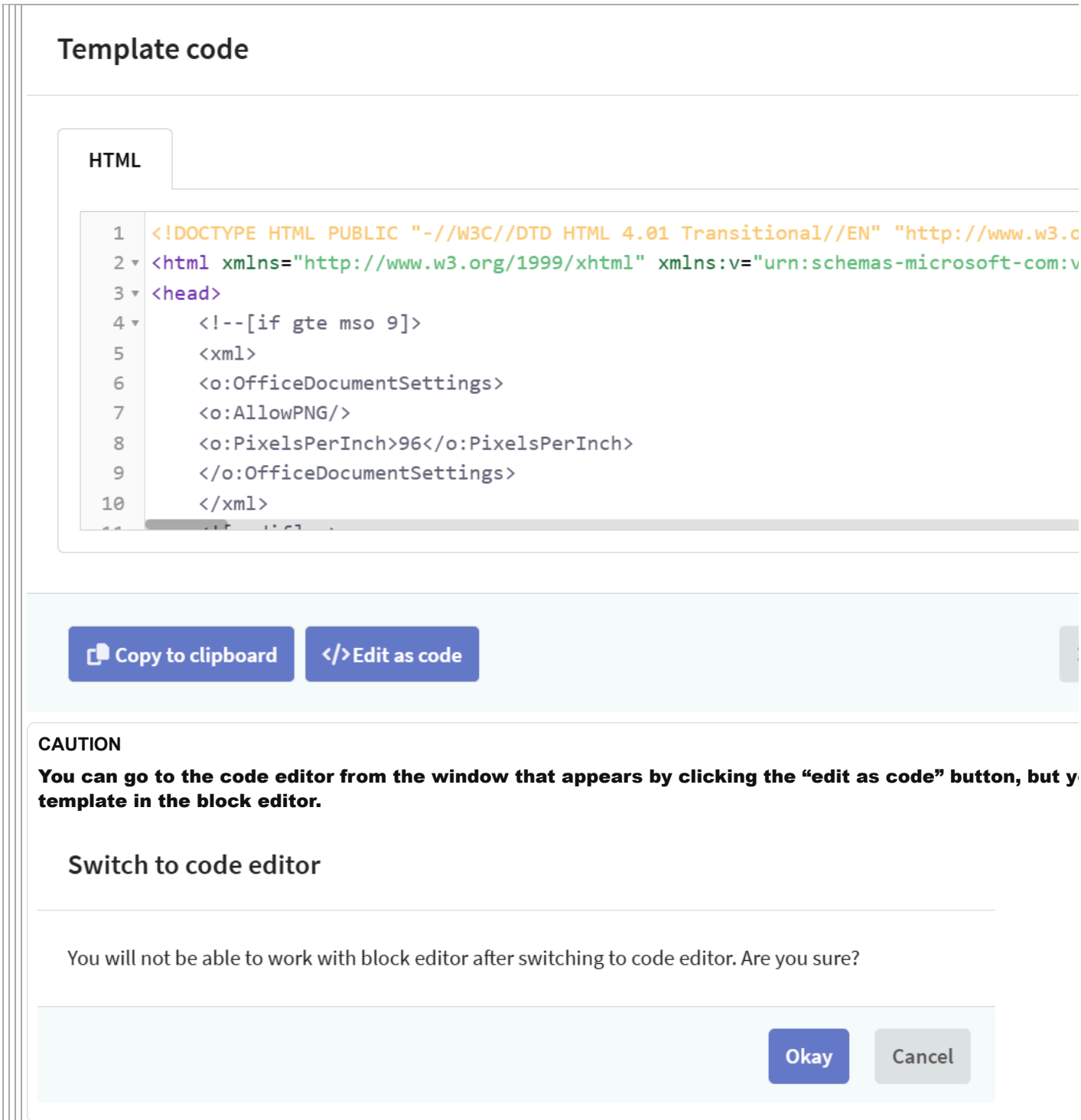
Working with the template after creation

After creating a template and applying the changes, you will have several actions available to you:



1. Edit button - allows you to return to the editor and make changes to the template;

2. Show template code button - shows HTML code of the created template:



CAUTION

You can go to the code editor from the window that appears by clicking the “edit as code” button, but y template in the block editor.

3. Edit text version button — demonstrates the text version of your template, it also allows to edit it.

Text version

Text version is important for subscribers who view emails as plain text

--
Unsubscribe: {unsubscribe}

Save

Close

Web version of an email

Web version of an email is a version of an email message that a user can view as a web page in a brows in email services. Web versions increase the chances that your message will be viewed even in case of u having a link to the web version of an email allows you to quickly and conveniently share it if necessary.

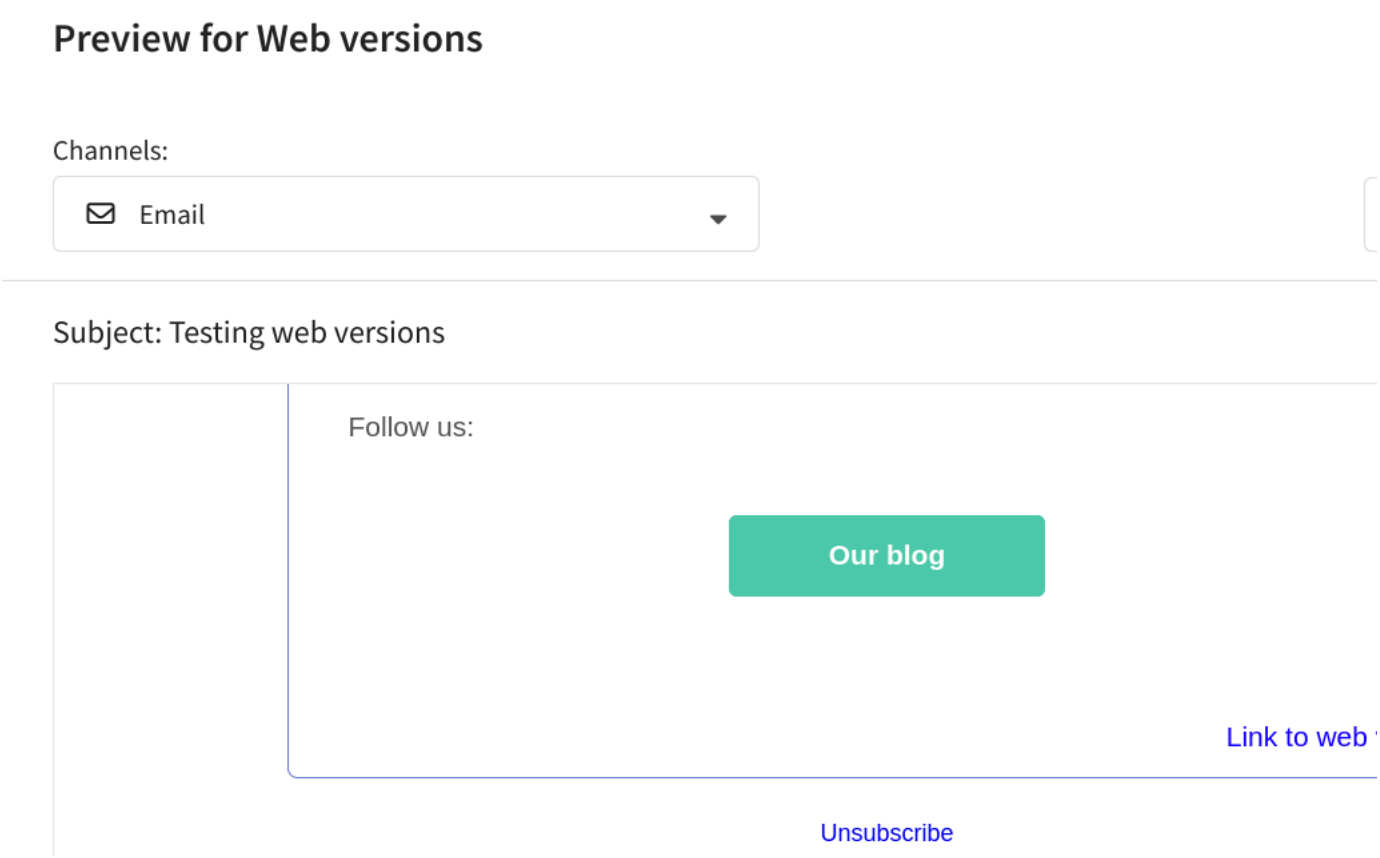
You can view the web version of an email sent to a client in the profile history. By default, the web versi period can be changed by the platform administrator in the account settings.

INFO

To enable users to save web versions, the platform administrator must change the account settings, na versions" field. After that, links to view the web version of the sent message will appear in the profile h

Create a link to the web version of the email

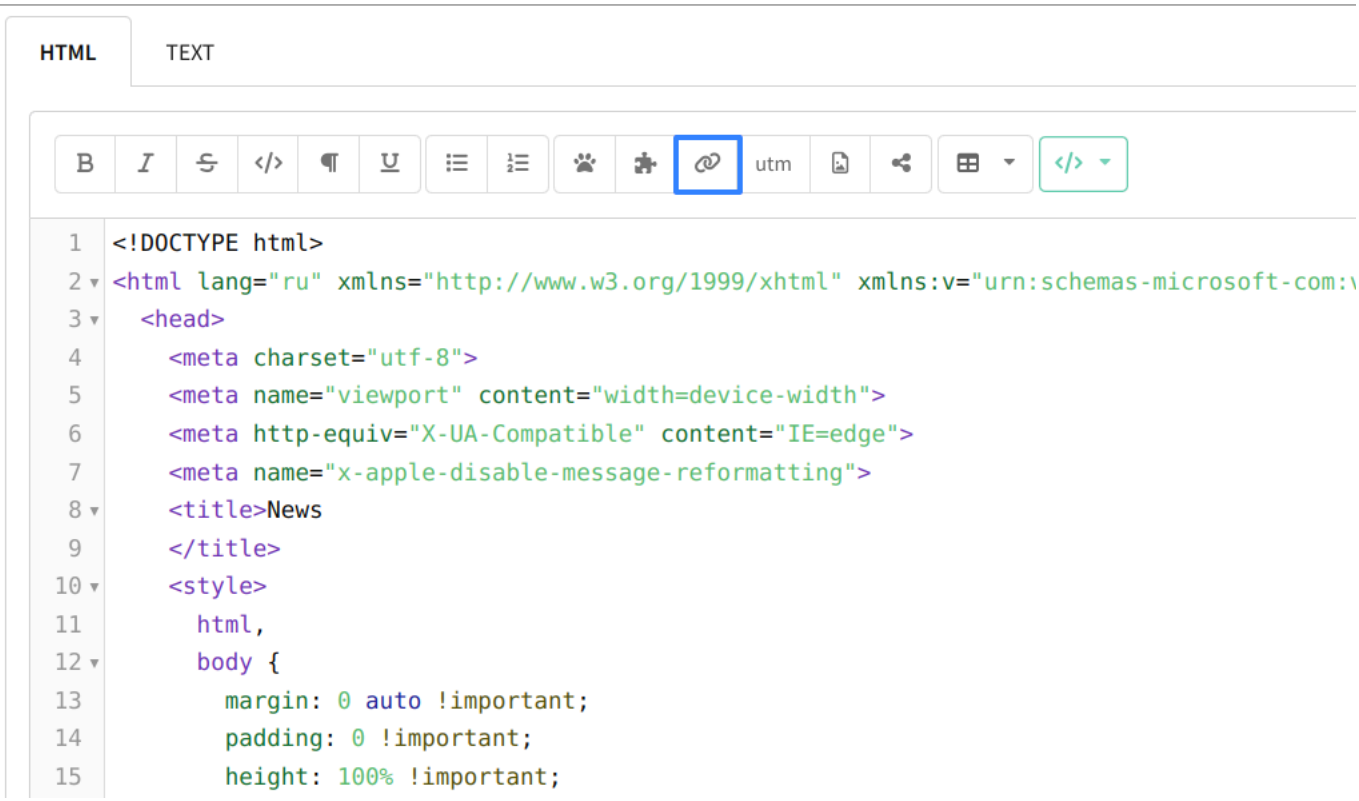
A link to the web version should be specified in the message body when creating or editing a message te



There are several ways to create this link. Let's take a closer look at them.

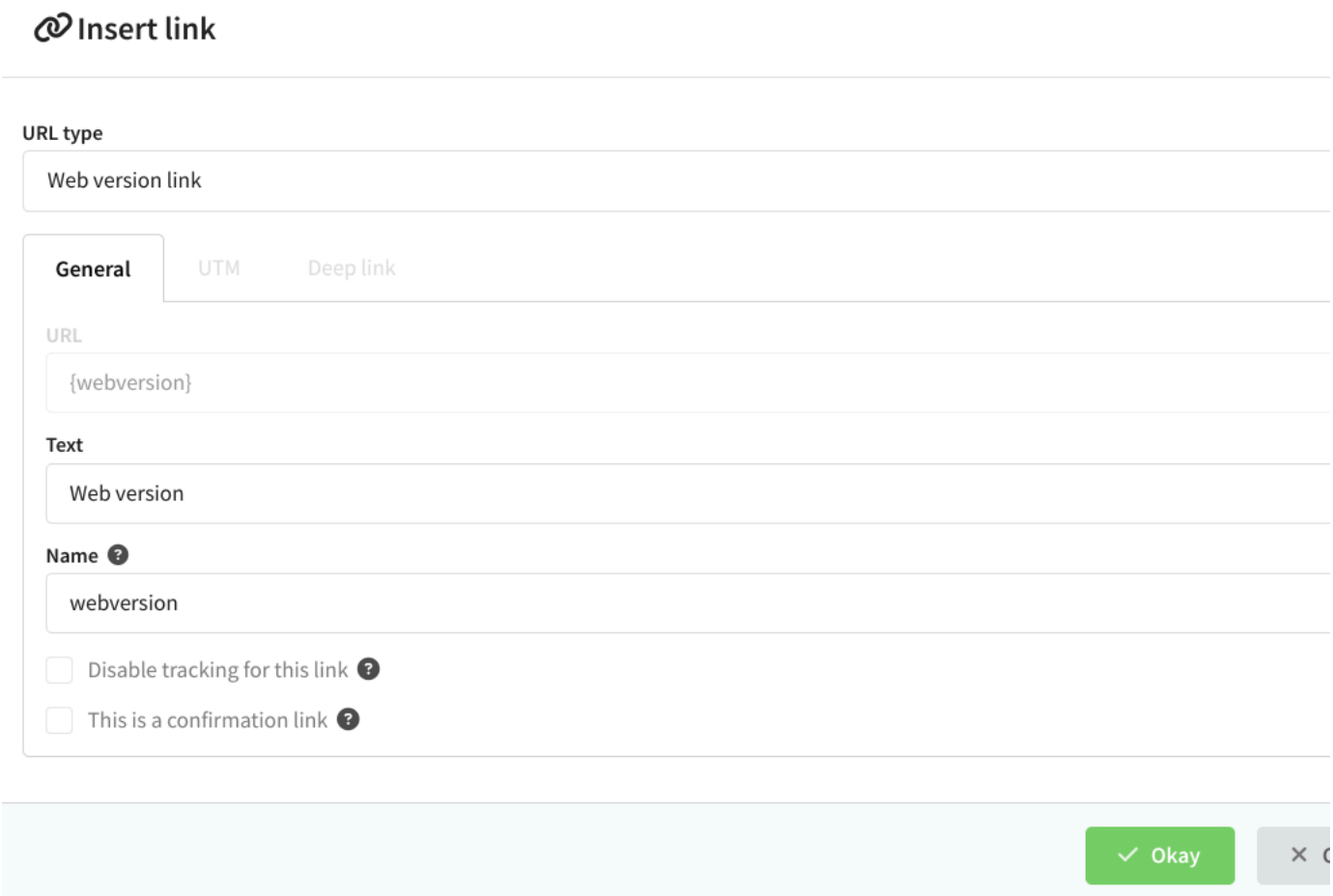
Using the link editor

Place the cursor where you want to add the link. In the toolbar, click the icon for inserting links:



In the window that opens, customize the link:

- **Select URL type - Web version link**
- **Enter link text - this is the text the user will see in the email. For example, “Open the web version of**
- **Enter link name. Later you will be able to track who used it by [building a segment]() of the customers**



Click the Okay button. The link code will appear in the message template.

NOTE

Note that the link to the web version of the email cannot be used as a confirmation link and it is not pos

Manually

The link to the web version of a message is as follows:

```
<a href="{webversion}" title="Web version of a message" name="web_version"> Open this message in the browser</a>
```

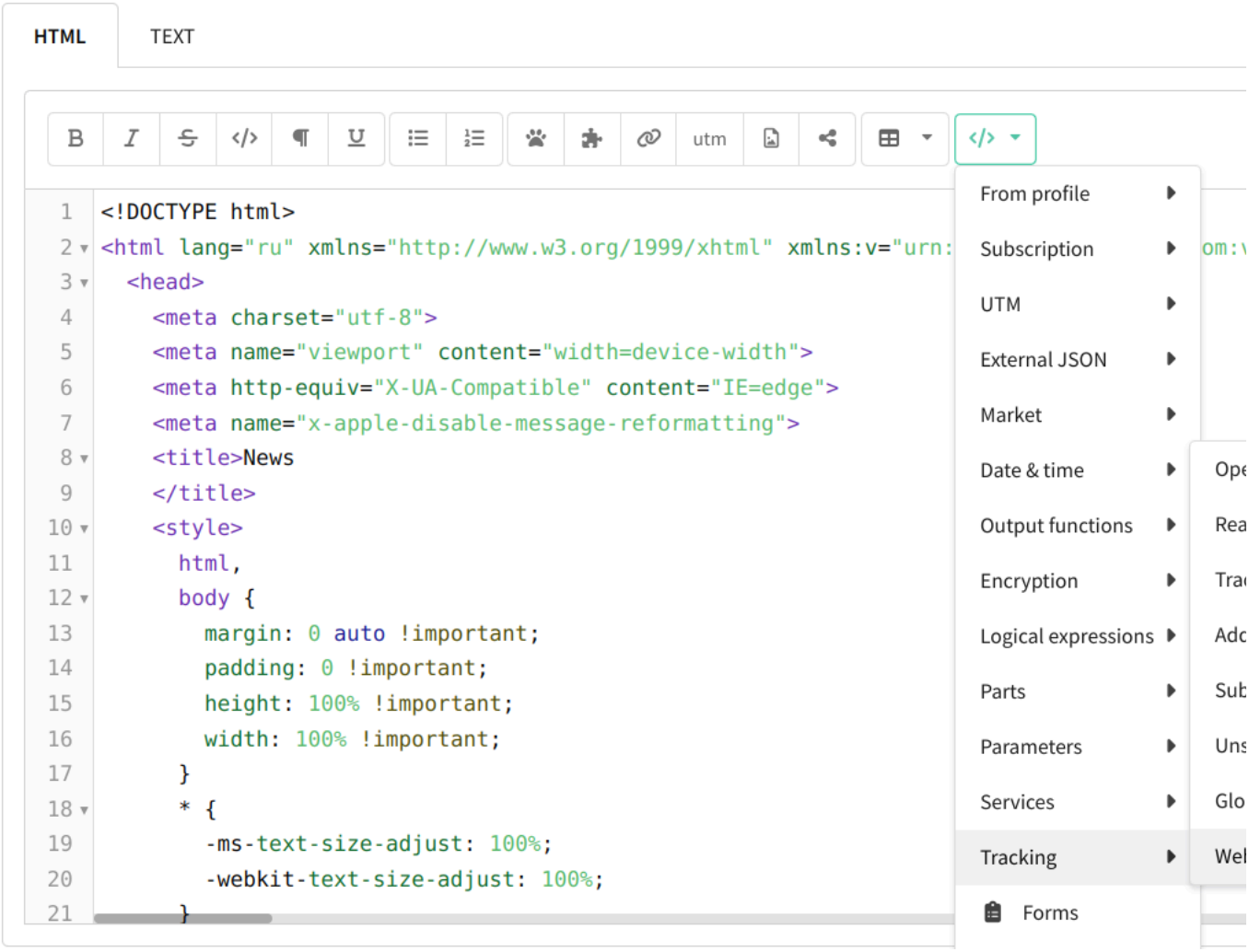
Let's look at the parts of this link in more detail:

- `<a href="{webversion}"` **- system component;**
- `title="Web version of a message"` **- the title of the link;**
- `name="web_version"` **- link name, further used to get analytics on transitions;**
- **Open this message in the browser - text displayed when pointing at the link;**
- `</a>` **- closing tag.**

You can insert the link in the place you need it.

As a variable

You can create a link to the web version from the variables menu. Open it by clicking on the toolbar. Sele



The `{webversion}` **variable will appear in the body of the template:**

HTML TEXT

```
                        <td id="ibbca8" style="padding: 20px; font-family: sans-serif; fo
                        </td>
                      </tr>
                    </tbody>
                  </table><span id="ite6yd" style="font-family: Helvetica, Tahoma, Arial,
                  <a href target="_blank" id="im3vpj" style="font-family: Helvetica, Tahc
                  </span>
                </td>
              </tr>
            </tbody>{webversion}
          </table>
        </div>
        <table align="center" role="presentation" cellspacing="0" cellpadding="0" border=
          <tbody>
            <tr>
              <td id="inc7hn" align="center" style="padding: 20px; font-family: sans-seri
                <a href="{unsubscribe}" title="Текст ссылки" data-name="link2">Unsubscri
              </td>
            </tr>
          </tbody>
        </table>
```

Next, it is necessary to add additional parameters to this link. How to do this is described in detail in the a result, your link will look like this:

```
<a href="{webversion}" title="Web version of a message" name="web_version"> Open this message in the browser</a>
```

AMP Email

Description

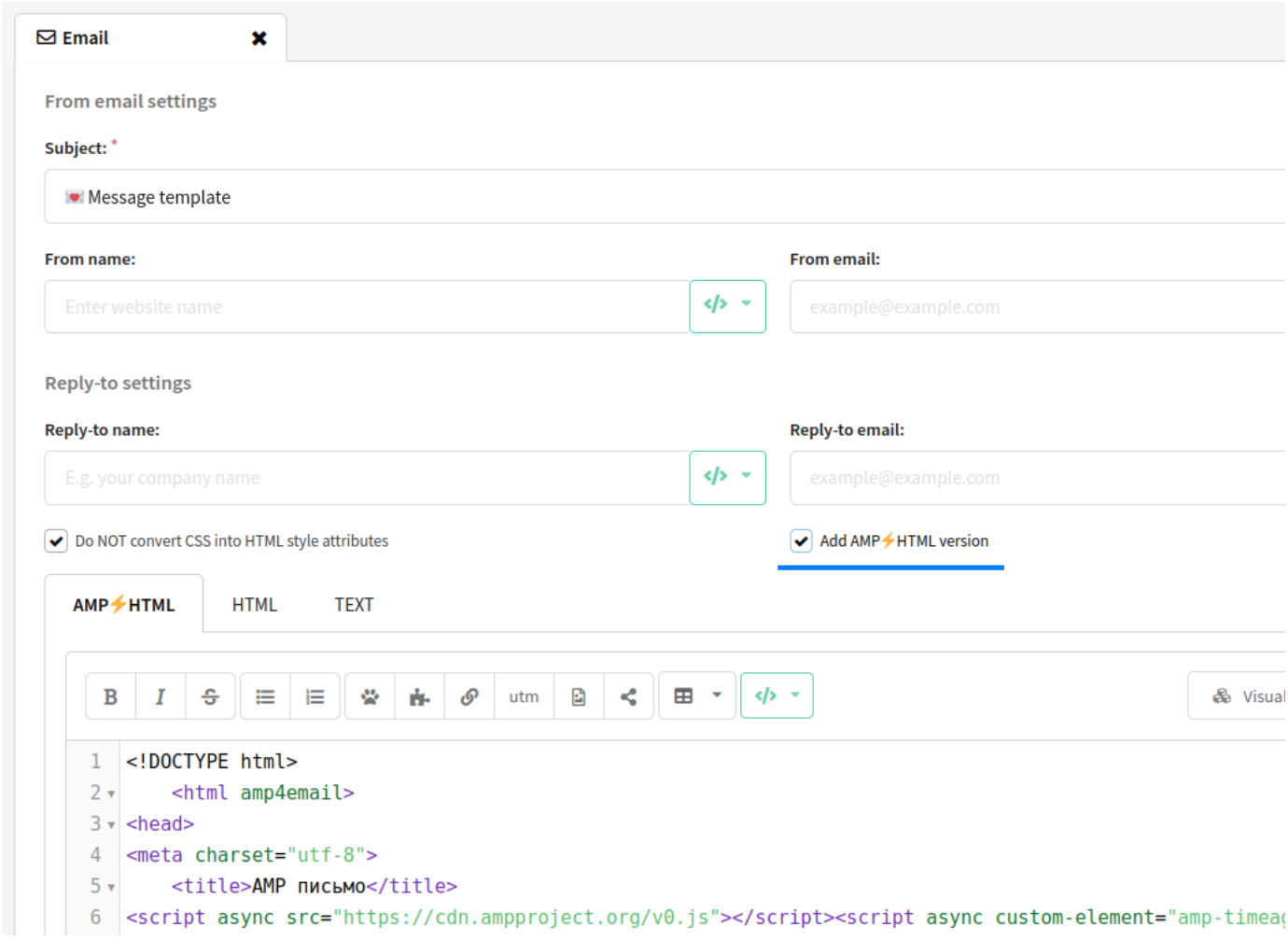
AMP technology allows you to add JavaScript components to emails and make them interactive. In an AM buy a product, or confirm registration without going to the site.

Email clients require sender registration to display AMP emails to users. Follow this guide to get your se supporting clients.

> TIP
>
> **This guide will give you everything you need to know to get started writing valid AMP Emails**

Adding an AMP version

The standard email template includes HTML and text versions of the email. To send AMP emails, select t in the template editor. Below you will see an additional tab AMP⚡HTML where you can use AMP compone



> TIP
>
> **AMP emails are currently supported by the following email clients:**
>
> - **Gmail;**
> - **Yahoo;**
> - **Mail.ru.**
>
> **If the email is delivered to an email client that does not support AMP technology, the recipient will see**

Links in AMP emails

In the AMP version of the email, instead of the `name` attribute, the `data-name` attribute is used to identify li platform will automatically insert the attribute into all links.

```
<a href="https://example.com" title="Link text" data-name="link1">Link text</a>
```

Also note that if you add an AMP parameter to the link, you don't need to wrap the link with the tracking `no-track` attribute. Clicks on such a link will not be tracked.

```
<a href="${{exampleUrl}}" title="Link text" data-name="link1" no-track>Link text</a>
```

Escaping template expressions

AMP template syntax is the same as Altcraft template syntax. To avoid incorrect code parsing, AMP tem with the $ symbol.

For example, if you are using the `<amp-list>` component, then replace the list element substitution variab

```
<amp-list id="myList" src="https://foo.com/list.json">
  <template type="amp-mustache">
    <div>${{title}}</div>
  </template>
</amp-list>
```

Importing and exporting a message template

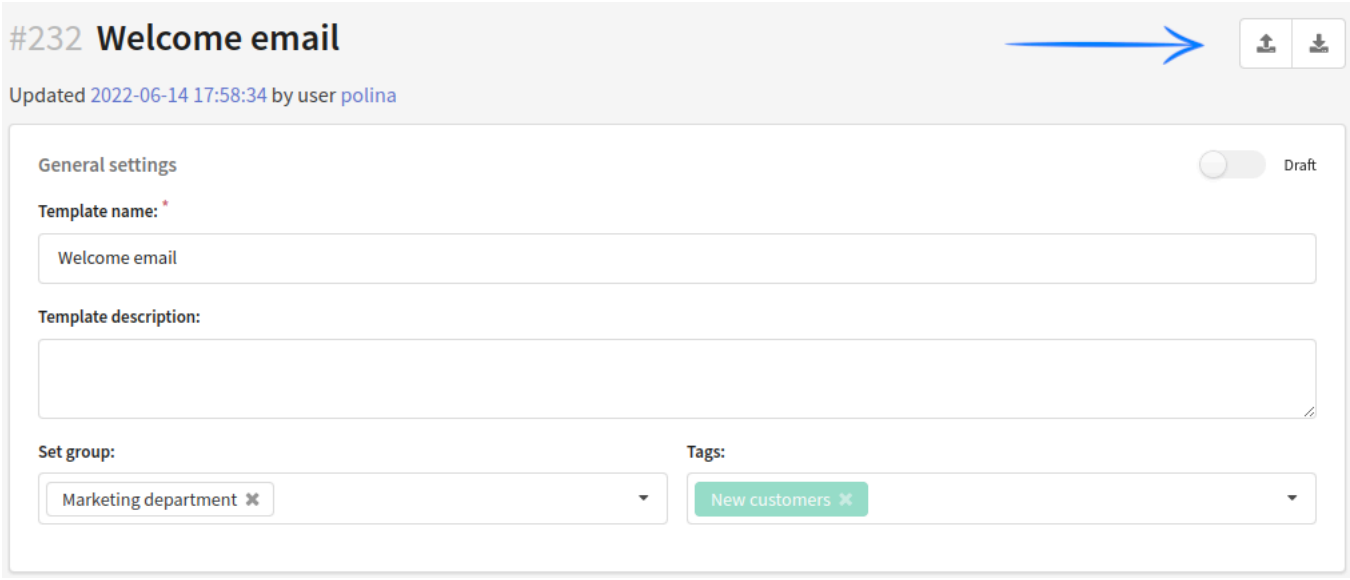
Description

This functionality allows you to upload a message template to the platform or download it to your device channel, or for several channels at once. A zip file is used for export and import.

The corresponding buttons are available in the template editing window in the upper right corner:

— template import

— template export



Template import

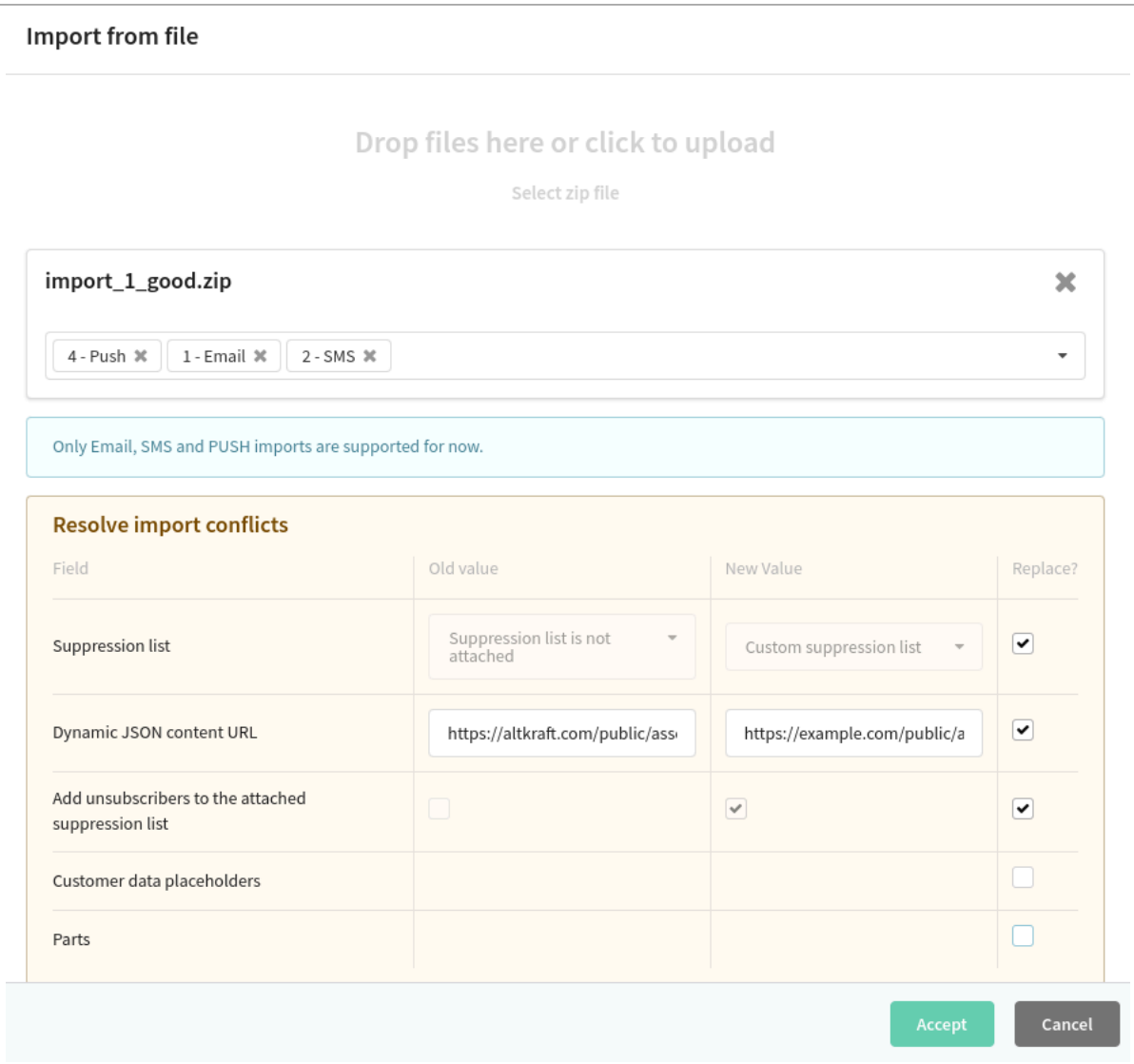
To import a template into the platform, click on the icon

. Next, click on the download area and select the zip file on your device.

Set up the import:

1. **Select the channels for which you want to import the template (Email, SMS, Push).**
2. **Resolve import conflicts. Conflicts arise if the data being imported and the data already stored in the shows the old and new value. If you want to update the data, check the "Replace" column.**

Click Accept to start the import process.



Zip file structure

A zip file is used for import. The archive must have a certain structure:

```
<catalog_name>/
    <channel_name>/
        <fieldname>.<type>
        <attach>/
            <fileid>/
                <filename>
    images/
        <image_name>
    template.json
```

Structure description:

Element	Example	Description
<catalog_name>	**template_newsletter**	**Root directory. You can use an arbitrary name.**
<channel_name>	**email** **sms** **push**	**The directory contains template data for a specific send channel. The directory name must match the channel.** **Import of Email, SMS and Push templates is available.** **Information about which files can be imported for each channel can be found here.**
images	**image_1.png**	**Directory for storing images**
template.json	—	**The file contains general data about the template (link to dynamic JSON content, additional settings, fragments an** **You will find more detailed information below.**

Zip file example

TIP

Starting from version 68, images can be stored either at the same level as the HTML file or in a folder w
images, image, img, imgs, pictures, pics. An archive containing such an email template should not inclu
template.json, as their presence may cause import errors.

Basic structure for an email channel

```
template_123/
    email/
        html.html
        text.txt
        subject.txt
    images/
        image_1.jpeg
    template.json
```

Full structure for all channels

```
template_123/
    email/
        html.html
        text.txt
        subject.txt
        amp.txt
        is_amp.txt
        is_not_inline.txt
        from_email.txt
        from_name.txt
        replyto_email.txt
        replyto_name.txt
        attach/
            023f-ad8c-4412-9bba-16ac/
                attach_file.zip
    sms/
        sms.txt
        add_sms_url_scheme.txt
    push/
        title.txt
        body.txt
        click_url.txt
        icon.txt
        is_rich_push.txt
    images/
        image_1.jpeg
        image_2.jpeg
    template.json
```

Image paths in HTML

To ensure correct image rendering in an imported template, use relative paths that match the archive st

Recommended structure:

```
template_name
├── email
│   └── html.html
└── images
    └── banner.png
```

Example of a correct path in HTML:

```
<img class="img" src="../images/banner.png" alt="Banner">
```

Description of fields for each channel

Email

File	Example	Description
html.html	`<html>Hello, world!</html>`	**HTML version of the message**
text.txt	**Hello, world!**	**Text version of the message**
subject.txt	**Best books 2022**	**Subject line**
amp.txt	`<html>Hello, world!</html>`	**AMP HTML version of the message**
is_amp.txt	**false**	**Is there an amp version in the template or not**
is_not_inline.txt	**true**	**Do not convert CSS to inline**
from_email.txt	**awesome@example.com**	**Sender's email**
from_name.txt	**Awesome Books**	**Sender's name**
replyto_email.txt	**reply_awesome@example.com**	**Reply-to email**
replyto_name.txt	**Awesome Books**	**Reply-to name**
attach (directory)	**023f-ad8c-4412-9bba-16ac/ attach_file.pdf**	**Directory for storing attached files. It contains additional directories, each containing a separate file. Arbitrary names can be used to name additional directories.**

SMS

File	Example	
sms.txt	**Hello, world!**	**Message text**
add_sms_url_scheme.txt	**true**	**Add “http://” to link**

Push

File	Example	Descripti
title.txt	**Best books 2022**	**Notification title**
body.txt	**Hello, world!**	**Notification text**
click_url.txt	**https://example.com/**	**Click-to-click link**
icon.txt	**https://example.com/images/test.jpg**	**The URL where the notification icon is located**
is_rich_push.txt	**false**	**Use rich version for push message or not**
<platform>.json	`<html>Hello, world!</html>`	**Templated data for a specific browser or OS (when using rich push)**

template.json file

This file contains data that is displayed in the “Options” and “Content” blocks.

template.json example

```
{
  "json_url": "https://altkraft.com/example/content.json",
  "suppress_id": 1,
  "is_unsub_to_supp": true,
  "field_overrides": {
    "lead": {
      "_fname": "client"
    }
  },
  "parts": []
}
```

Description of fields in template.json

Field	Example	Required	
json_url	**“https://altkraft.com/example/content.json”**	**No**	**Dynamic JSON content URL**
suppress_id	**1**	**No**	**Suppression list ID**
is_unsub_to_supp	**true**	**No**	**Add unsubscribers to the attached**
field_overrides	`{ “lead”: { _fname”: “Client” } }`	**No**	**Customer data placeholders**
parts	`[ { “name”: “part_1”, “shortname”: “part_1”, “type”: “text”, “content”: “” }, { “name”: “part_2”, “shortname”: “part_2”, “type”: “text”, “content”: “” } ]`	**No**	**Template parts**

TIP

You can export any template from your account to get an example of a zip file to import.

Importing a template from a third-party service

Description

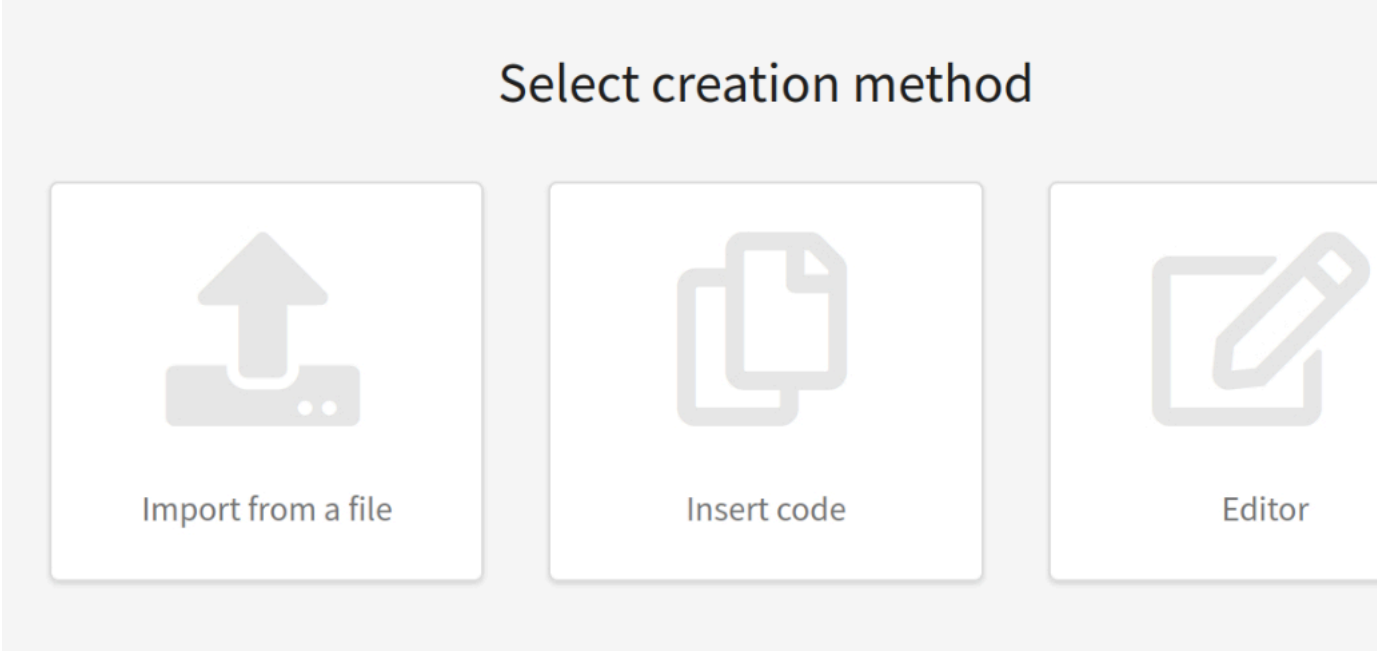
If you have ready-made message templates in other services, you can move them to Altcraft Platform. Yo templates.

TIP

Import is available for the following services:

- **Unisender**
- **Mailchimp**
- **Sendsay**

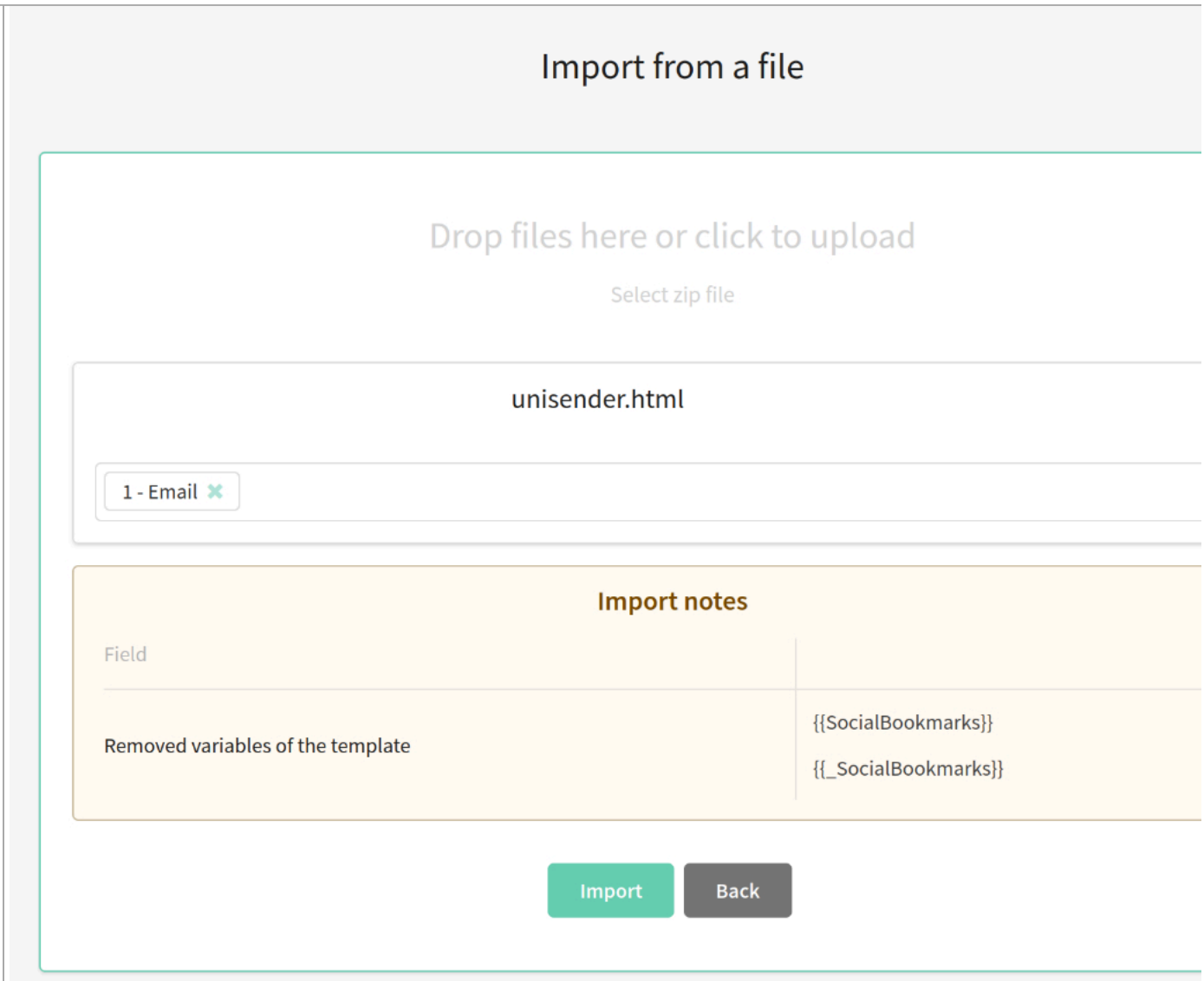
To get started, go to the "Templates section and click the Create button on the top panel. Next, select th template from a file or insert the code.



Import from file

If you want to move an Email template to the platform, you must first export the html file with the ready- service. To move an SMS template, you need a file in txt format.

In the platform, click on the upload area and select the file on your device. After uploading, you will see a variables removed from the template will be listed here.



TIP

Each service uses its own variables in message templates. In our platform, they look like this `{lead.emai`
When importing a template from a third party service, Altcraft will automatically replace some variable
the platform, it will be removed from the template.
See below for information on which variables will be saved.

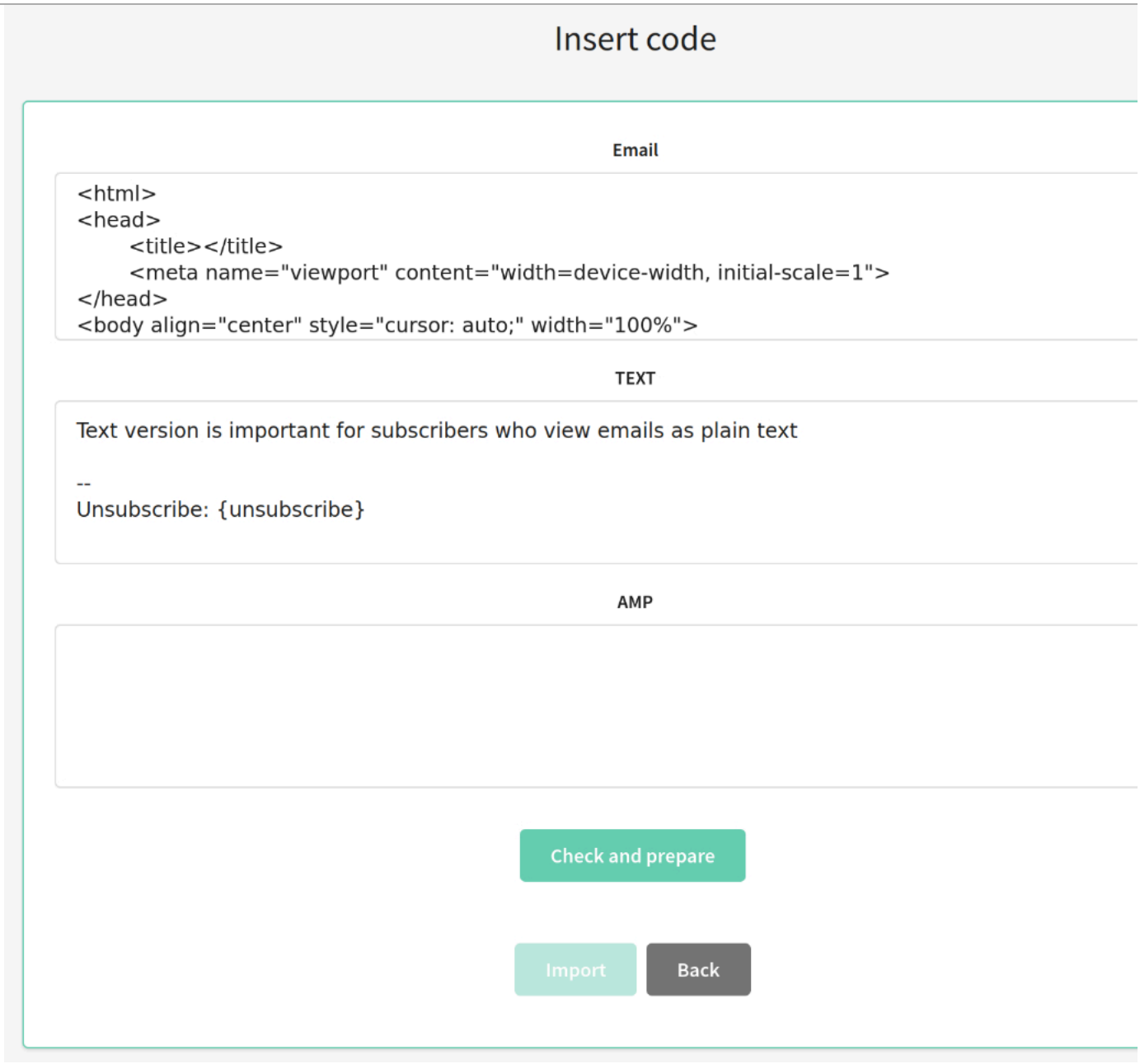
Insert code

When you create a template by inserting a code, the first thing you will be asked is to select a channel (E

If you choose Email, you will need to copy the template html code from the editor in a third party service
Email text box. The text version of the letter will be generated automatically.

If necessary, you can manually add code for each version of the email:

- **Email — html version of email.**
- **Text — text version of email.**
- **AMP — amp version of email.**



Importing a template for an SMS channel is easier. You only need to fill in one text box.

After pasting the code, run its verification by clicking the Check and prepare button. The import notes ta listing all the removed variables. Click Import".

Auto-replacement of variables in a template

Altcraft Platform automatically replaces third-party service variables with its own variables:

Unisender

Variable description	Altcraft
First name	`{lead._fname}`
Subscription email address	`{subscription.email}`
Subscription phone	`{subscription.phone}`
Email sender name	`{from_name}`
Custom profile field	`{lead.custom_field}`
Confirmation link	`<a href="#" title="Confirm your subscription" confirm-link name="link1">Confirm your subscription</a>` **When importing the text version of the email, the confirmation link is replaced like this:** `[Confirm your subscription](# <confirm-l` **URL.**

Variable description	Altcraft
Resource unsubscribe or global unsubscribe link	`{unsubscribe}`
Link to the email web version	`{webversion}`
Full date	`{datenow}`

Sendsay

Variable description	Altcraft	
First name	`{lead._fname}`	`[% anketa.base.firstName %]`
Last name	`{lead._lname}`	`[% anketa.base.lastName %]`
Subscription email address	`{subscription.email}`	`[% anketa.base.email %]`
Email subscription domain	`{subscription.email_domain}`	`[% anketa.member["confirm.host"] %]`
Gender	`{lead._sex}`	`[% anketa.base.gender %]`
City	`{lead._city}`	`[% anketa.base.city %]`
Date of birth	`{lead._bdate}`	`[% anketa.base.birthDate %]`
Custom profile field	`{lead.custom_field}`	`[% anketa.base.custom_field %]`
Condition	`{if lead.custom_field}{else}{end}`	`[% IF anketa.base.custom_field %][% ELSE %][% E`
Variable length	`{length(lead.custom_field)}`	`[% size(anketa.base.custom_field) %]`

Array iteration	`{for $index $i = items}`
	`  {$i}`
	`{end}`
		`[% FOREACH i IN items %]`
		`  [% i %]`
		`[% END %]`

Mailchimp

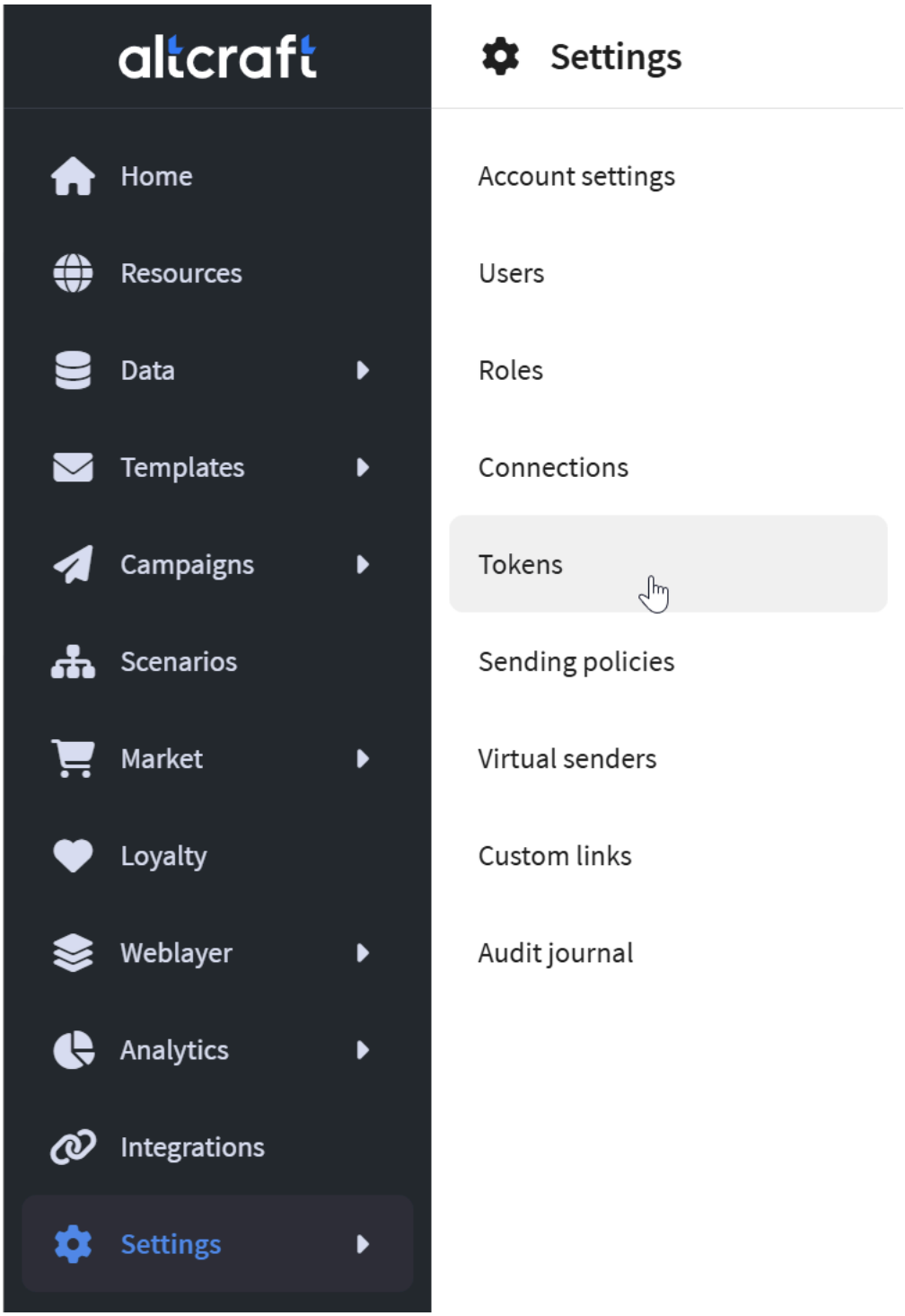
Variable description	Altcraft
First name	`{lead._fname}`
Last name	`{lead._lname}`
Subscription email address	`{subscription.email}`
Subscription phone	`{subscription.phone}`
Registration date	`{lead._regdate}`
Email sender name	`{from_name}`
Sender's email	`{from_email}`
Custom profile field	`{lead.custom_field}`
Confirmation link	`<a href="#" title="Confirm your subscriptiony" confirm-link name="link1">Confirm your subscription</a>` **When importing the text version of the email, the confirmation link is replaced like this:** `[Confirm your subscription](# <con` **to replace # with the URL.**
Resource unsubscribe or global unsubscribe link	`{unsubscribe}`
Link to the email web version	`{webversion}`
Full date	`{datenow}`
Full year	`{yearlong}`
Resource ID	`{resid}`
Mailing SUBID	`{subid}`
URL of a website, form, or application from resource settings	`{resurl}`
Uppercase	`{uppercase("your text")}`
Upper case for first character	`{uppercasefirst("your text is neat")}`
Condition	`{if lead.custom_field}{else}{end}`

TIP

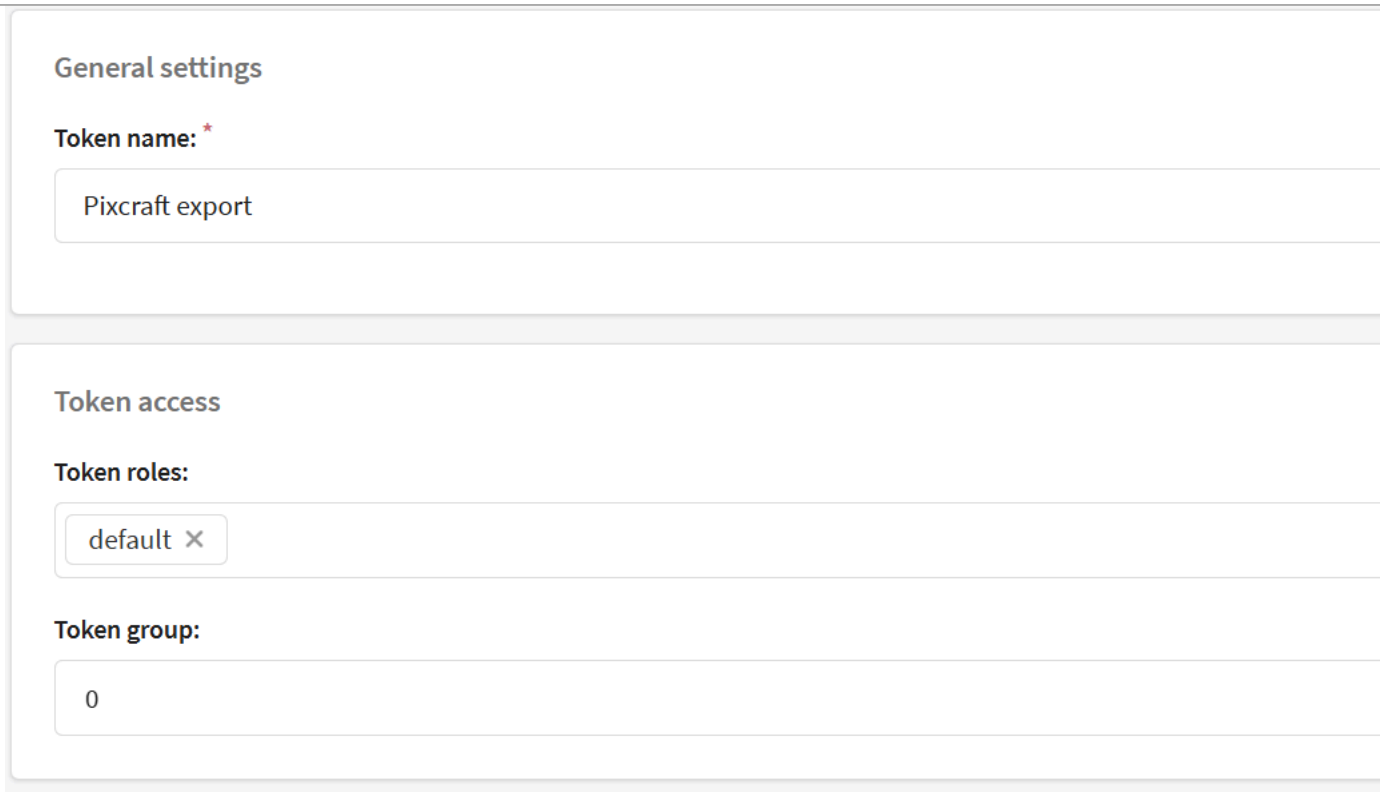
If you would like to extend the variable substitution syntax, please send a request to support@altcraft.c

Exporting a template from Pixcraft

To export from Pixcraft, you need to create an API token.
Go to Settings/Tokens:

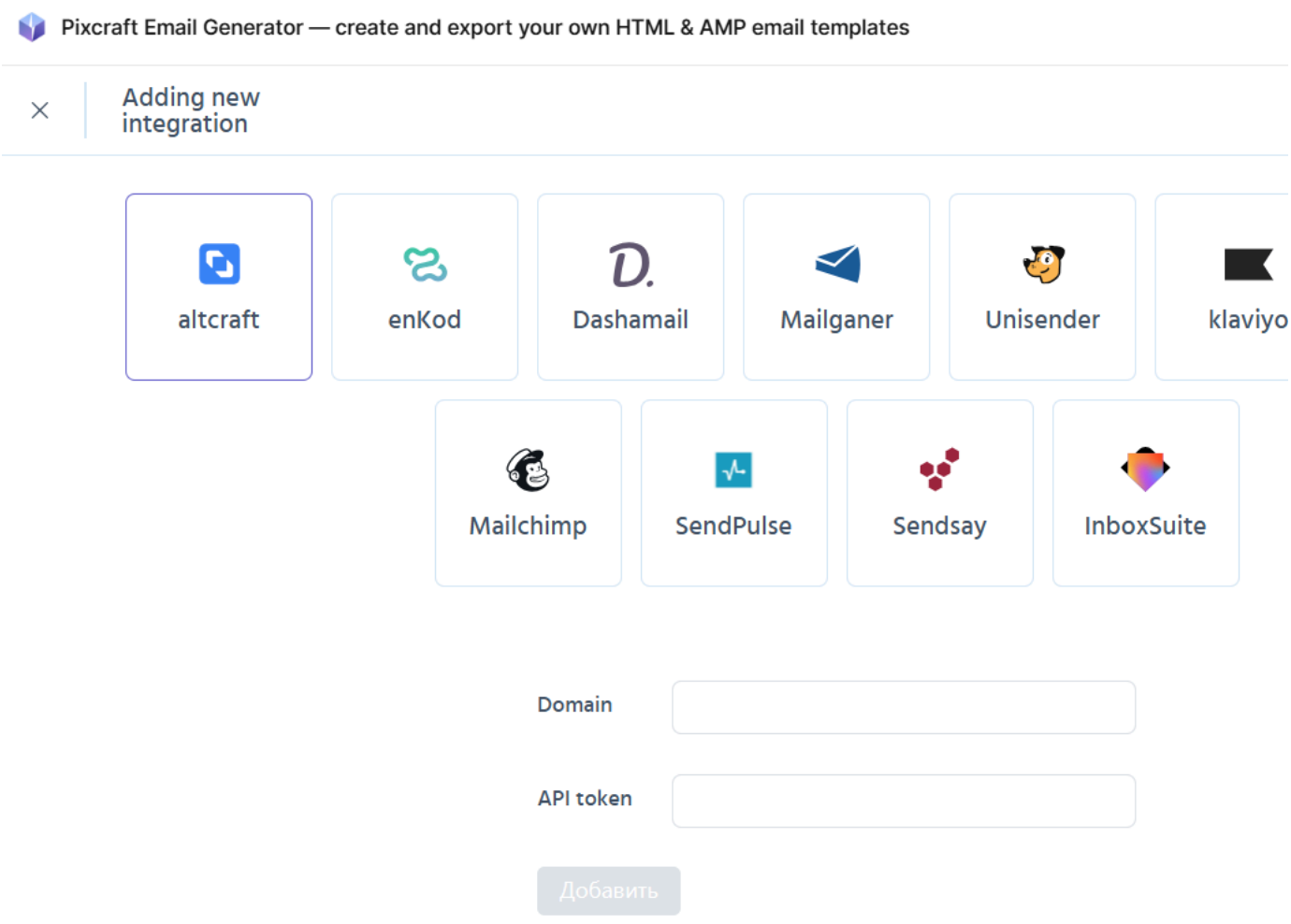


Click Create, specify a token name, for example, Pixcraft export.
Add a role that has permissions to create and edit templates.



Click Save.

Copy the API token and specify it when adding the integration to Pixcraft. To do this, in Pixcraft (Figma p and select Altcraft:

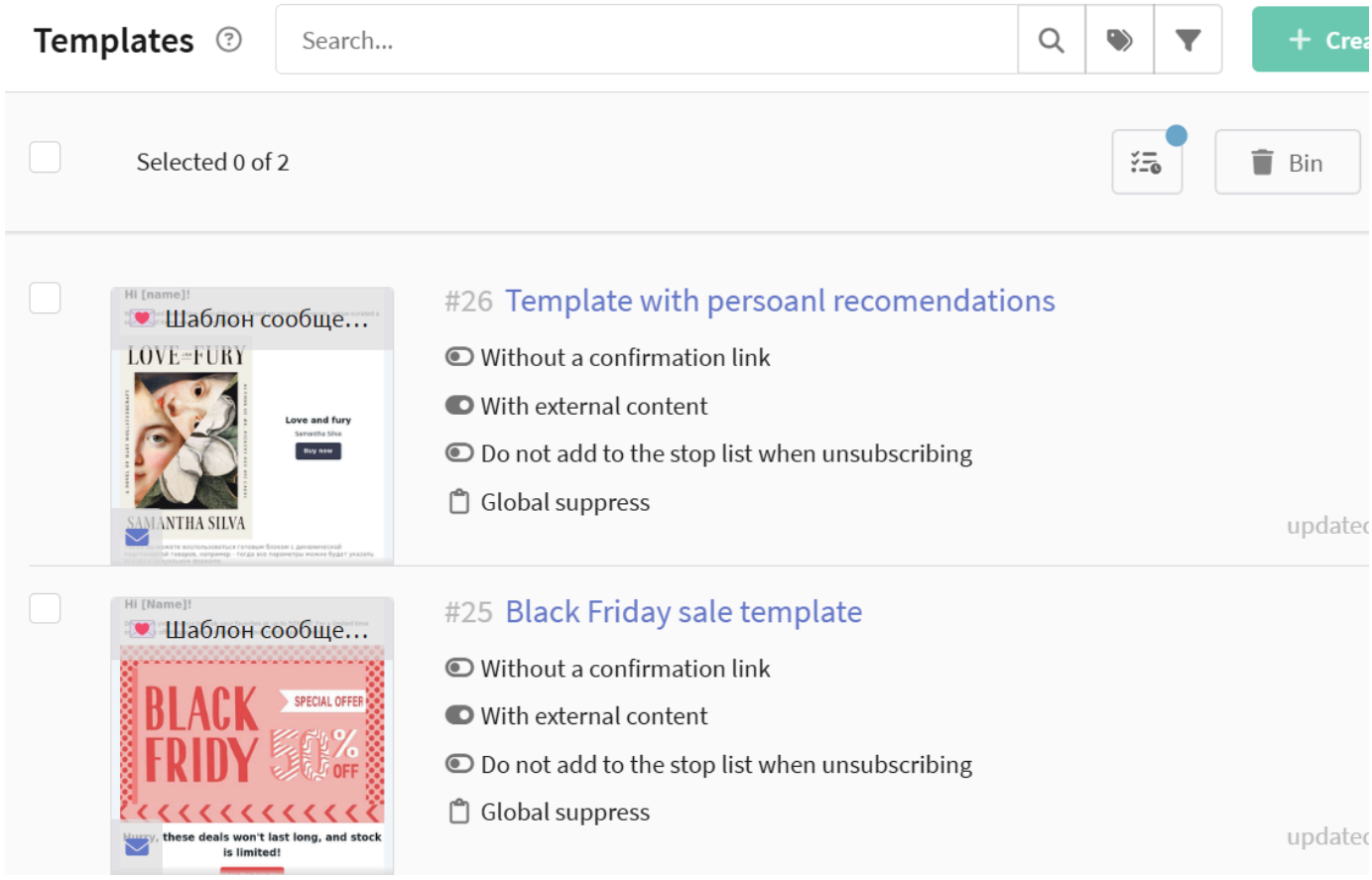


After adding the integration, select it from the list and click Export. The browser will open the exported

Working with message templates

![img](/img/attachments/66476/template-menu.png)	To view existing templates and add new ones, open the **Templates** section

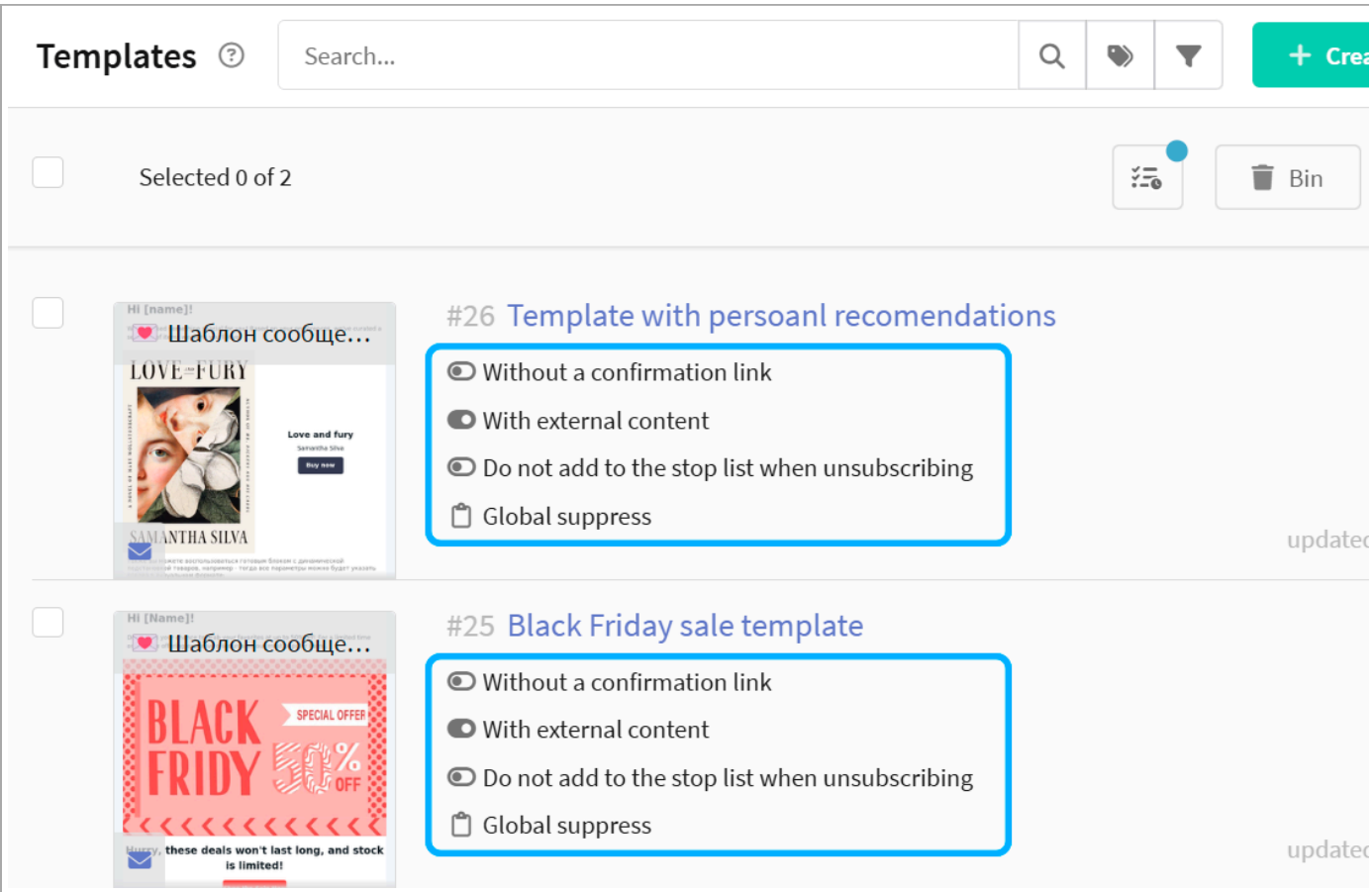
The list shows all of your message templates available in the account or in the access group. To find a t template name or its ID in the search bar. You can also filter templates by tags, sending (communication drafts or ready-made templates.



Click on the template and it will open the message preview. Here you can check how the message will lo different devices.

General information for each template is:

- **The presence of a confirmation link — a template with such a link is required when using Double opt**
- **The presence of external content — the use of links to dynamic JSON content in the template.**
- **Adding to the stop list when unsubscribing — if the subscriber clicks the unsubscribe link in the mes the selected stop list.**
- **Suppression list - a mailing using a template will block sending an email to a subscriber whose conta list.**



A quick action menu is available:

— draft mode.

— export the template.

— go to the template report.

— clone the template.

— delete the template.

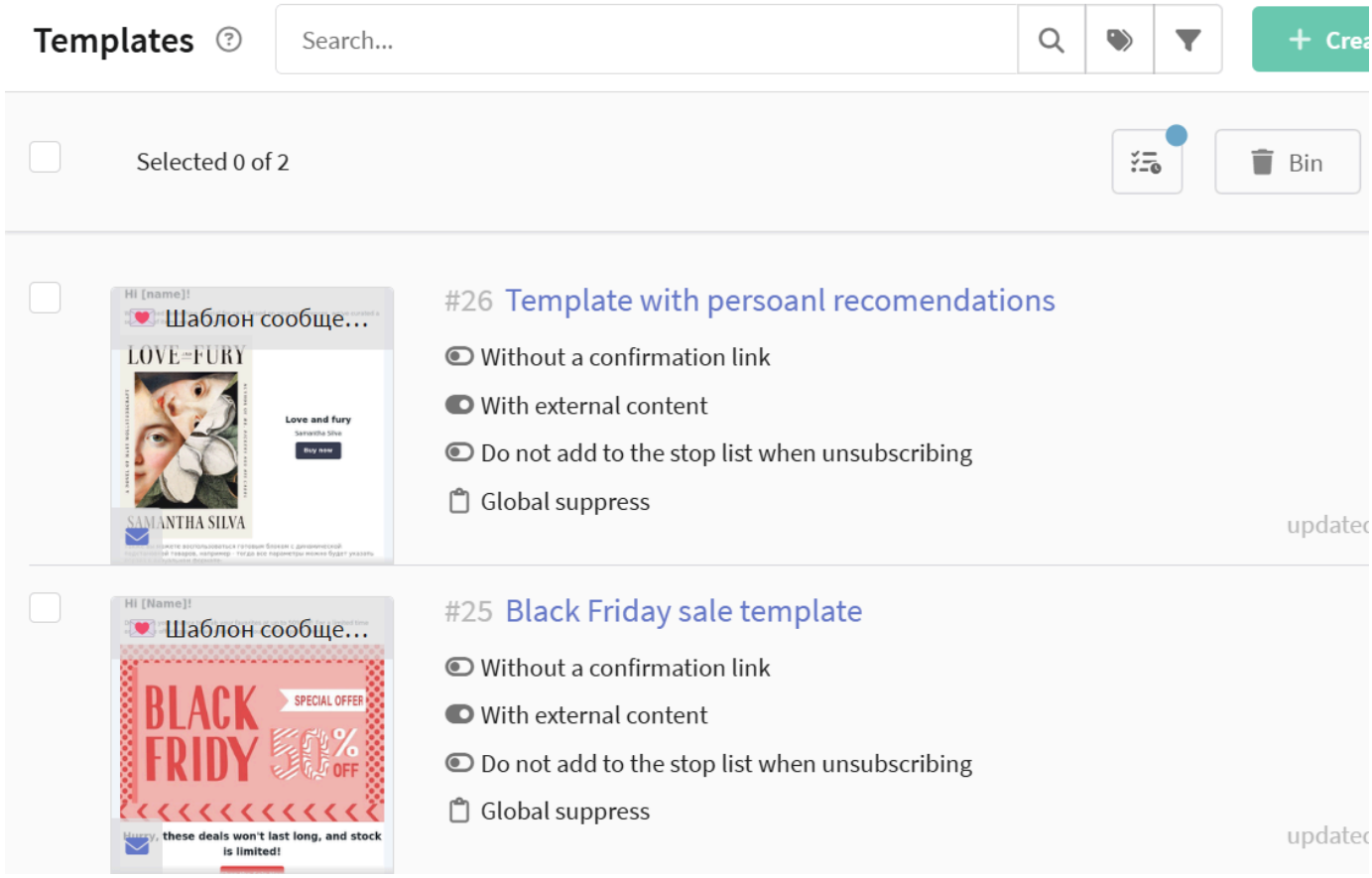
To add a new template, click the + Create button at the top of the page. If you need to edit an existing te

Read more about working with the template editor and about templates for each communication channel

Working in the editor

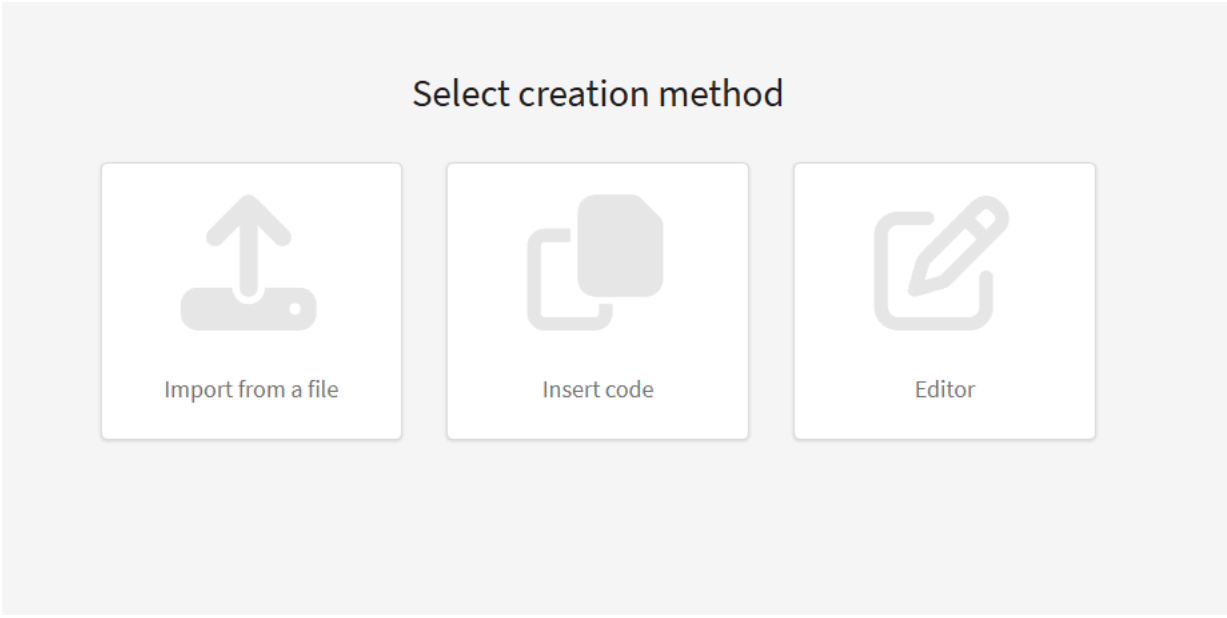
Creating a template

To add a new template, click on the + Create button at the top of the page:



Next you have to select the way to create a template:

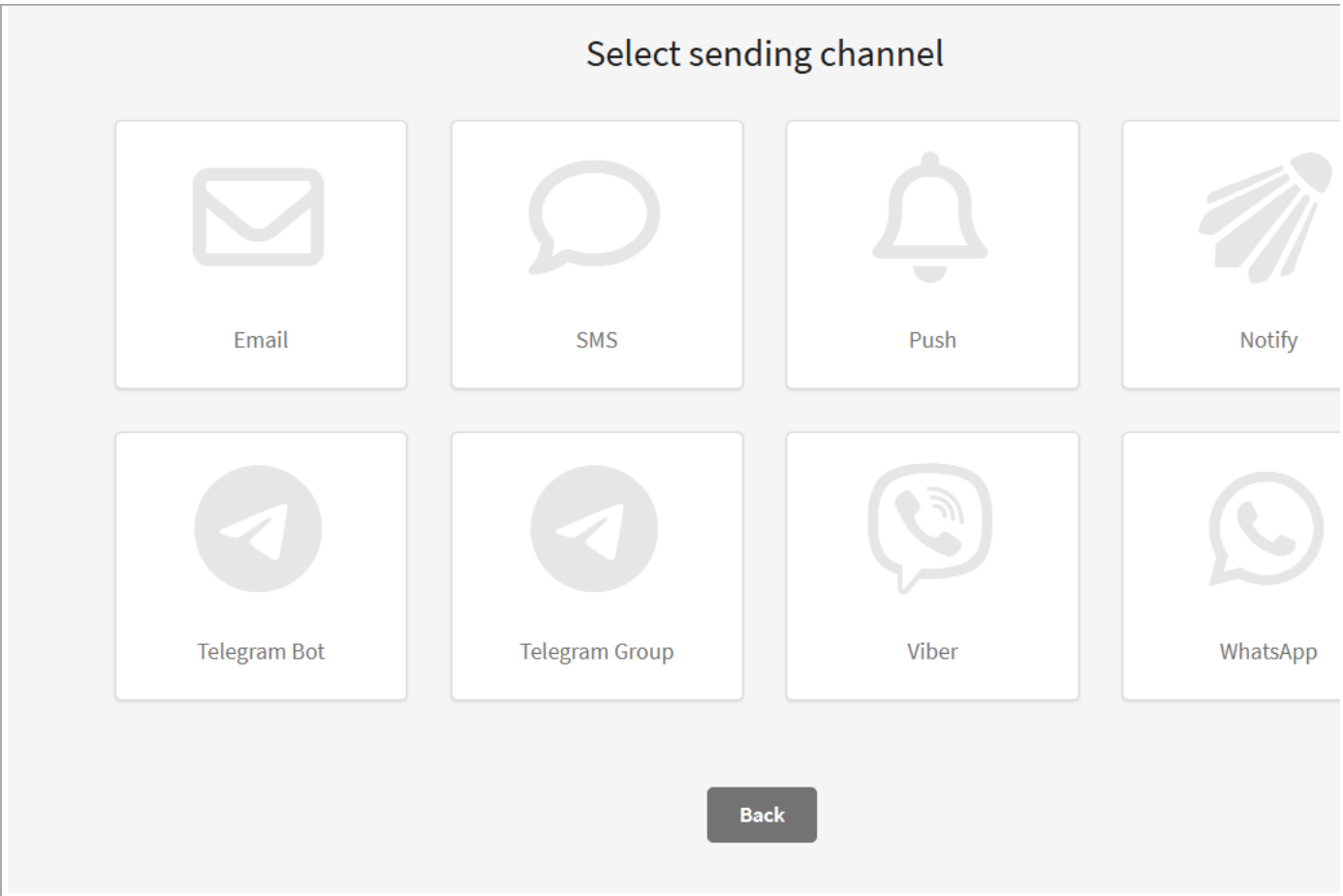
- **Import from a file — download the finished template as a zip archive or html file.**
- **Insert code — add the copied HTML code to the text window.**
- **Editor — create a template in the editor.**



TIP

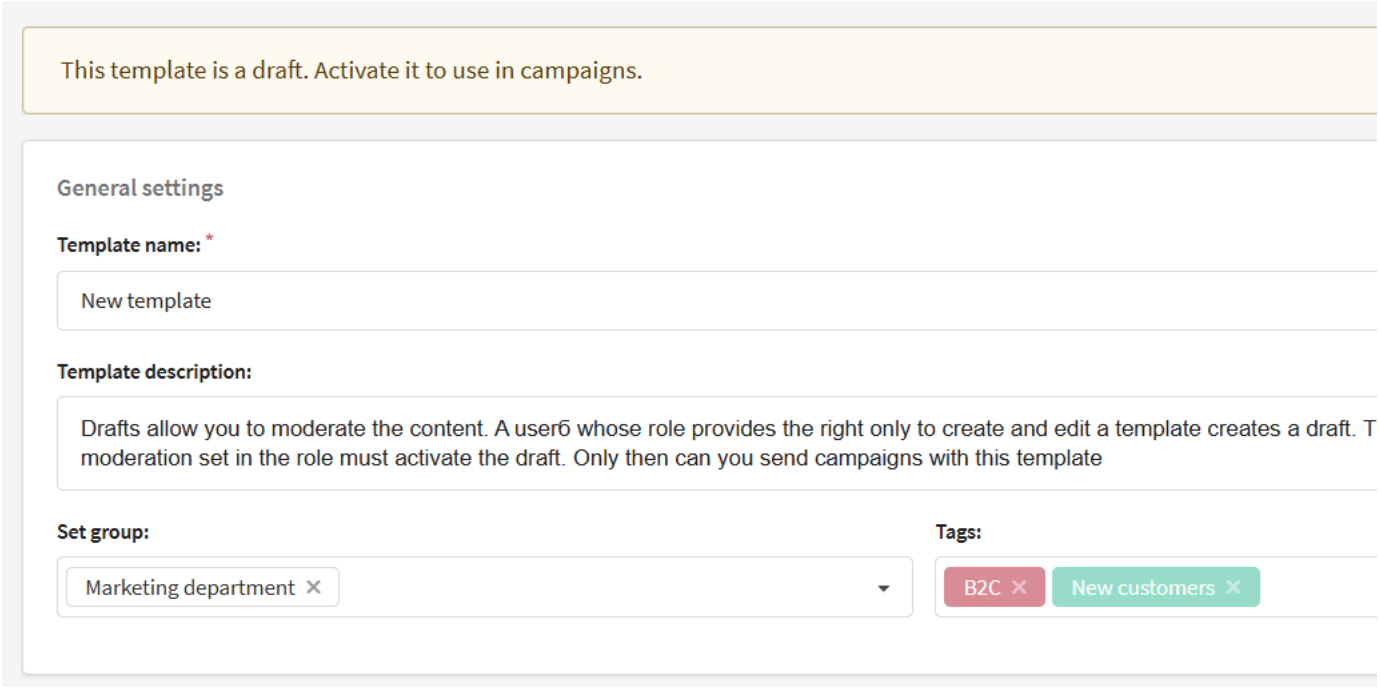
If you need to transfer a template from a third-party service, use file import or code insertion.

Select sending channel, or communication channel. In the future, the selected channel can be changed c



General settings

In the general settings of the template, specify its name and description, add tags and if necessary set a
If it is necessary to organize content moderation, the templates provide the function of creating Drafts. T
with a specific role. The role have to allow only create messages, without moderation. User with that rol
content, but templates can not be used immediately in newsletters. The second user with moderation rig
deactivate the Draft option in the General settings.



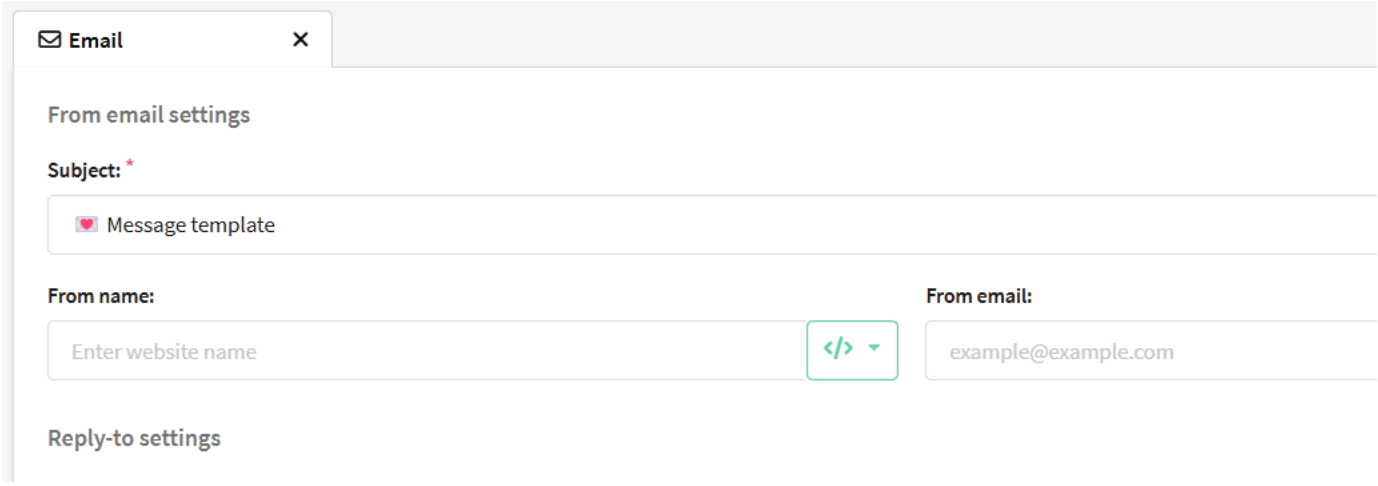
Content editor

The editor helps to create messages, quickly edit templates and also simplifies the personalization of co
A single template can combine content for sending over all available communication channels. You defin
creation's stage, but you can change it or add a new one. To add content to the template for a specific c

button

+

in the upper right corner:



Working with each type of content is described in the articles:

- **Email;**
- **SMS;**
- **Push;**
- **Telegram;**
- **WhatsApp;**
- **Viber™;**
- **MAX;**
- **Notify.**

Options

In the Options section, you can continue customize the template.

In the Dynamic content settings block, specify the data source for JSON content in messages. If there is preview of the first-level objects will open:

CAUTION

By default, the maximum size of dynamic JSON is 1 MB. In the On-premises version, this limit is set in t `MAX_REMOTE_CONTENT_BYTES` **parameter.**

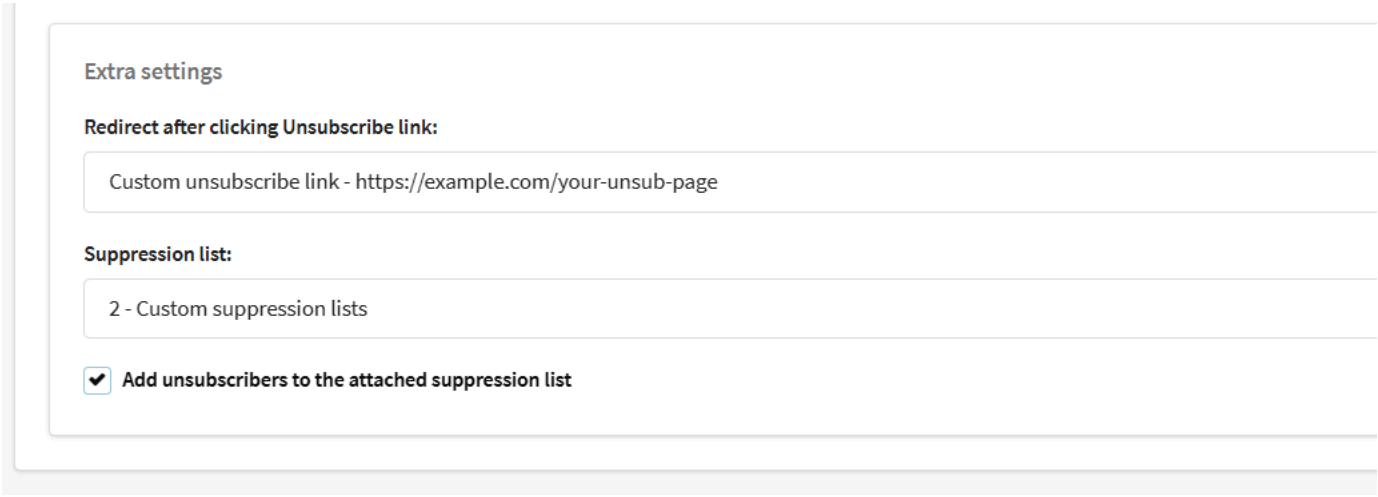
In the extra settings you can select custom links for the message unsubscribes and subscription confirm direct customers to the personal account of the site or to the page for subscriptions management.

TIP

User links unsubscribes and confirmations can be created by master account users. In the template op pre-created links. For more information, see the Settings section.

Also in the extra settings you can select a custom suspression list to use with the template. With this fe customers who are not suited to the type of content being sent. Additionally, you can enable the option A

suspression list:

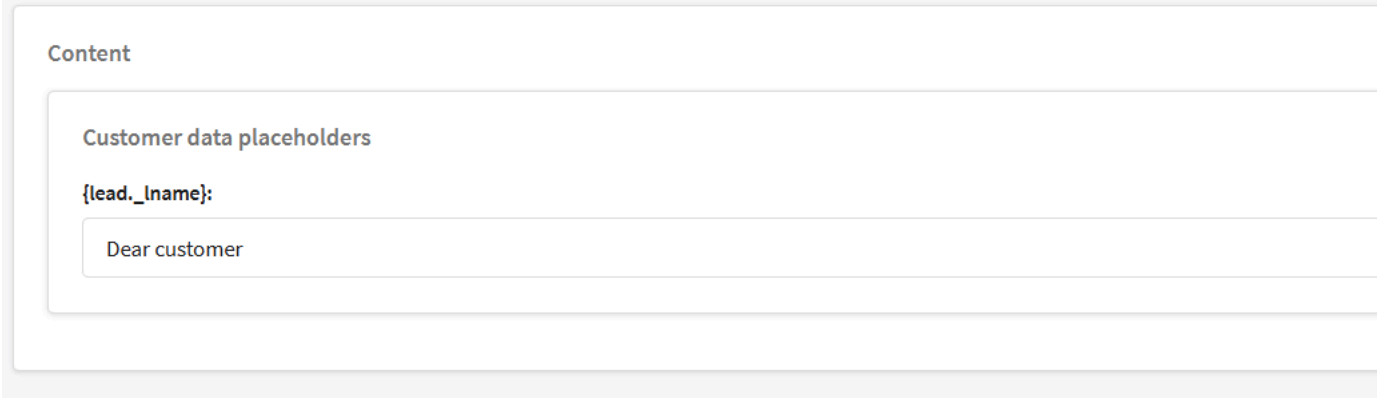


Default field values and parts

TIP

The Content tab appears in the editor when you use variables with profile data or parts of messages in

In the tab Content you can define default values for variables that you use in the template. So, if you use some profiles the name field is not filled in, the default value will be used, for example, “Dear customer”.

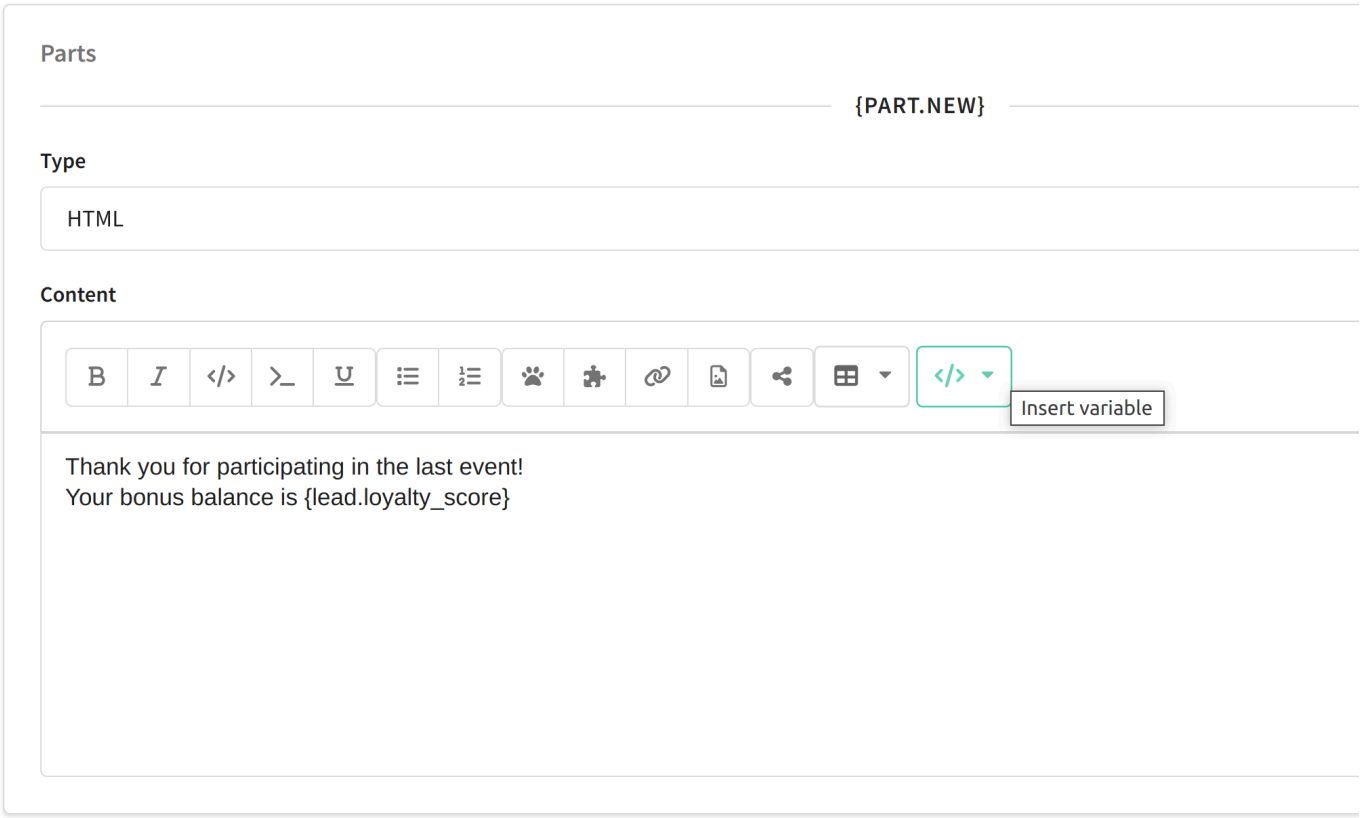


Parts of the message are similar to fragments. However, they can only be used in the template where th used in several templates.

TIP

To start working with the parts, add the variable `{part.example_part}` to the template code. After that, the bottom of the page, where you can customize parts of the template.

For each part, you can select the content type: HTML for email messages or text for the text version of e content. Depending on the type of content, the corresponding content editor will be opened with all the



TIP

It is convenient to use parts together with logical expressions for a more compact display of the messa the number of subscriber points (custom numeric field), one or the other part will be substituted:

```
{if lead.score gte 9000}
{part.part_1}
{else}
{part.part_2}
{end}
```

Message preview

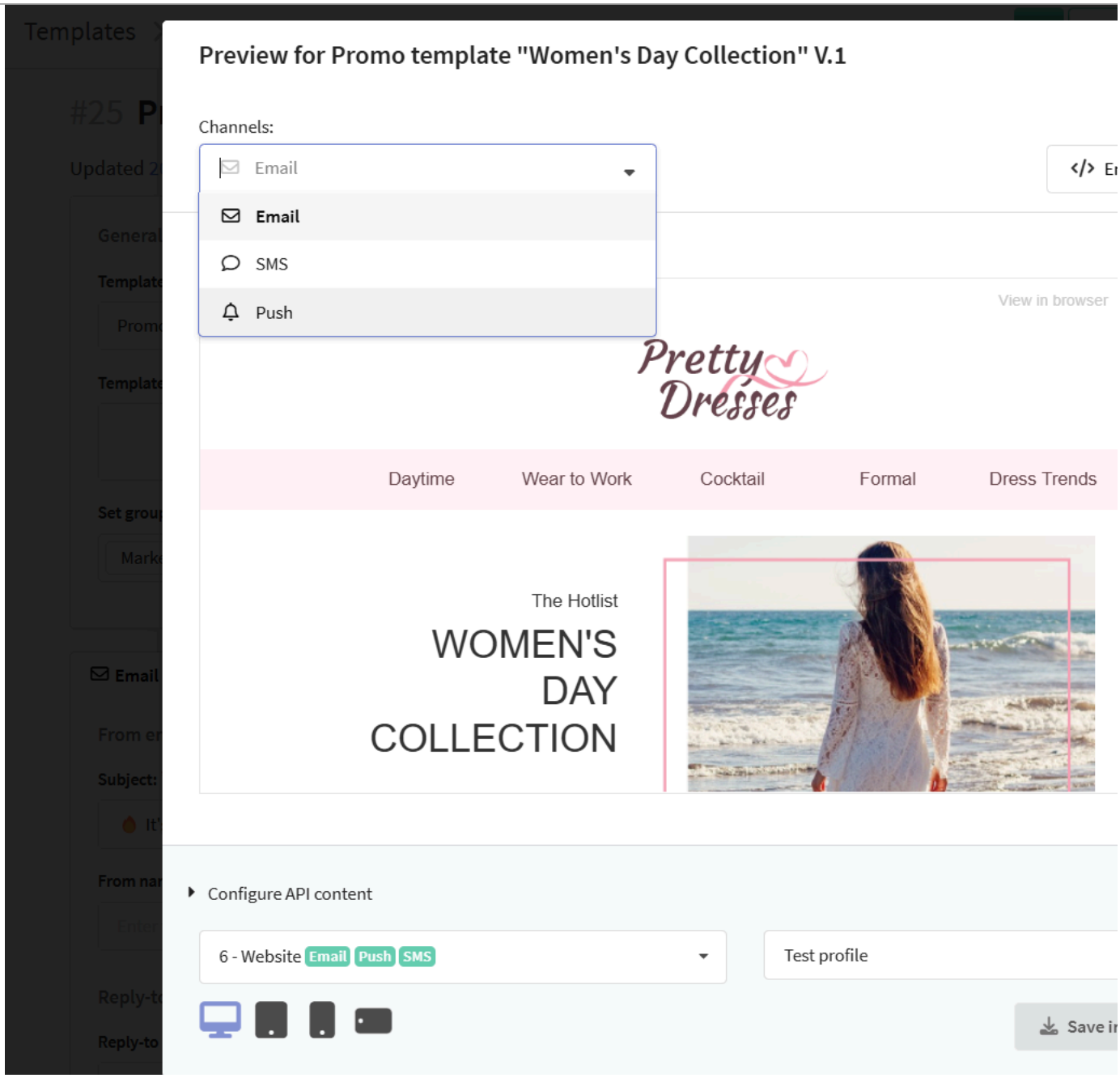
Previewing the message allows you not only to check the HTML layout, but also to see the results of con on subscriber data. Click on this button at the top of the page to activate the preview:

In the preview window, you can select the desired channel for which you want to view the content. A pre versions is available for email.

At the bottom of the window there are buttons for switching desktop/mobile display modes, as well as pe

TIP

Select the resource from which you plan to send the template, and add a new profile in the drop-down m database search. The segmentation rules are used there, so you can immediately select a profile with t



A preview of the template in `png` image format is available for any channel. To download an image to you

Follow the rules to correctly upload the preview of the email template:

1. **CSS styles `text-decoration`, `text-decoration-line` and `text-decoration-color` are installed directly into the p example:**

```
<div>
  <span style="text-decoration: line-through;">Text</span>span>
</div>
```

2. **If you use fonts imported from an external source (for example, Google Fonts), it is recommended to installed in the system in the `font-family` property. For example:**

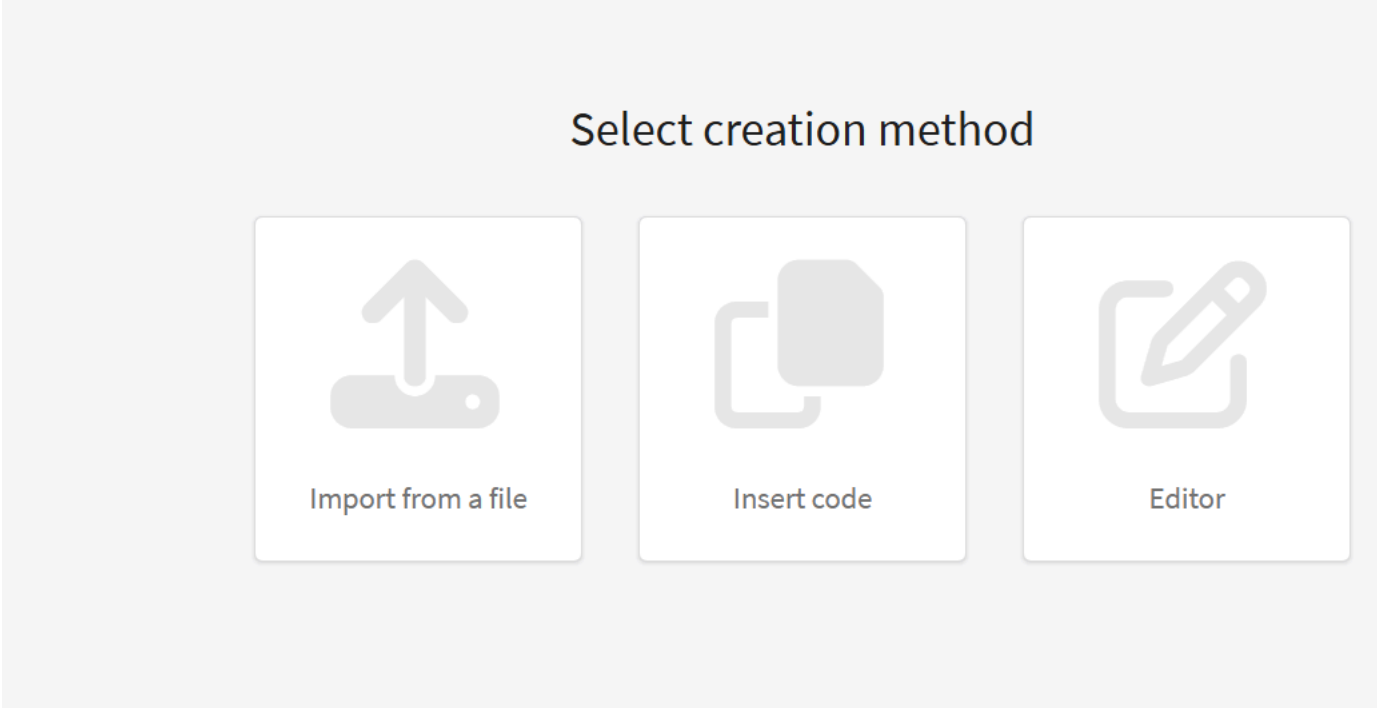
```
<!-- Correct display in preview -->
<span style="font-family: Roboto, Arial, Tahoma, sans-serif">Текст</span>

<!-- Incorrect display in preview -->
<link
  href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&dis
  rel="stylesheet"
/>
```

Email template

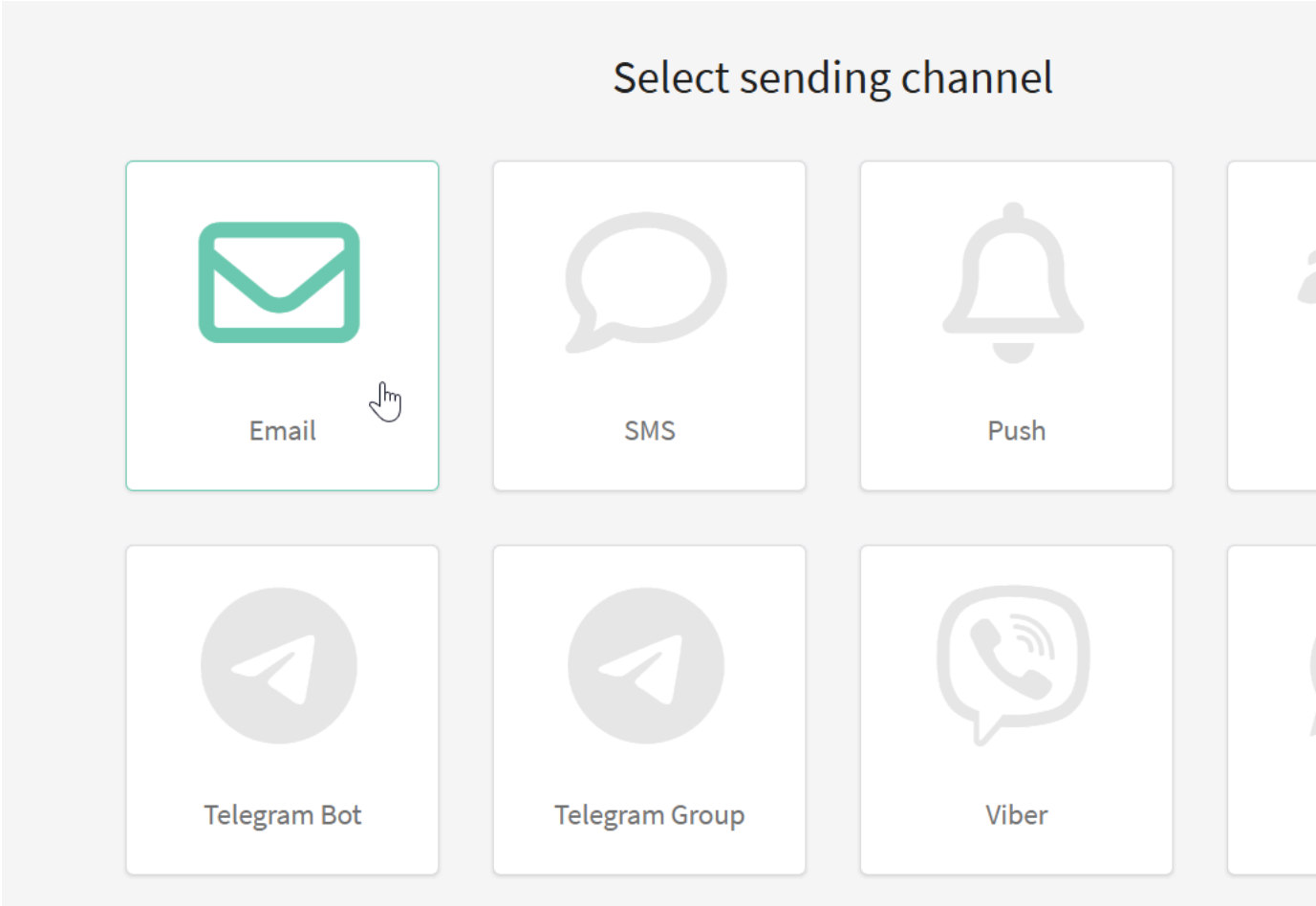
Getting started

In the main menu, select Templates and create a new message template. Next, select creation method:

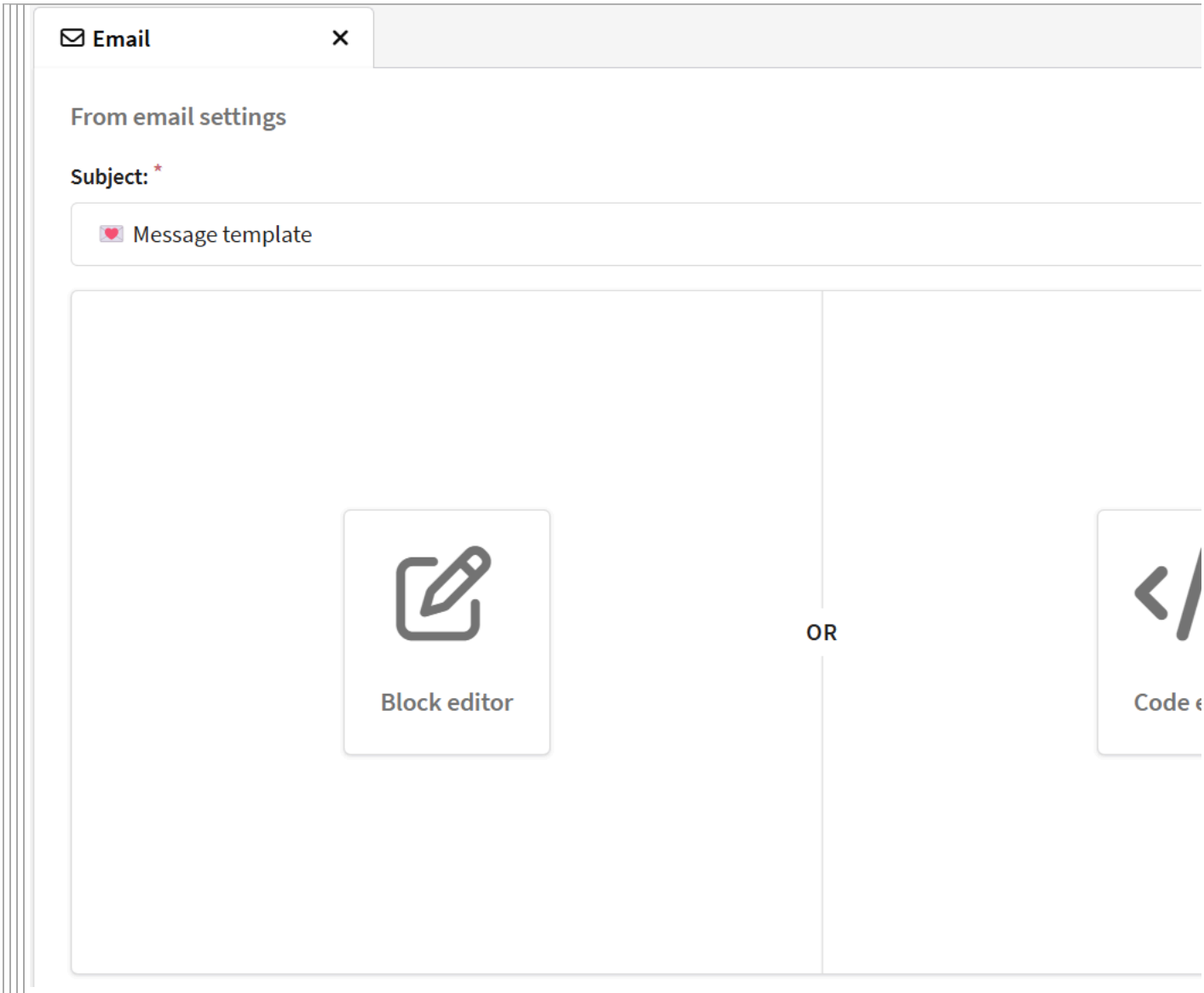


If you already have a ready template, you can import it into the platform from a file or insert HTML code. is described in [this article](). After loading the template you will automatically move to the editor.

To start working in the editor immediately, select Editor —> Email:

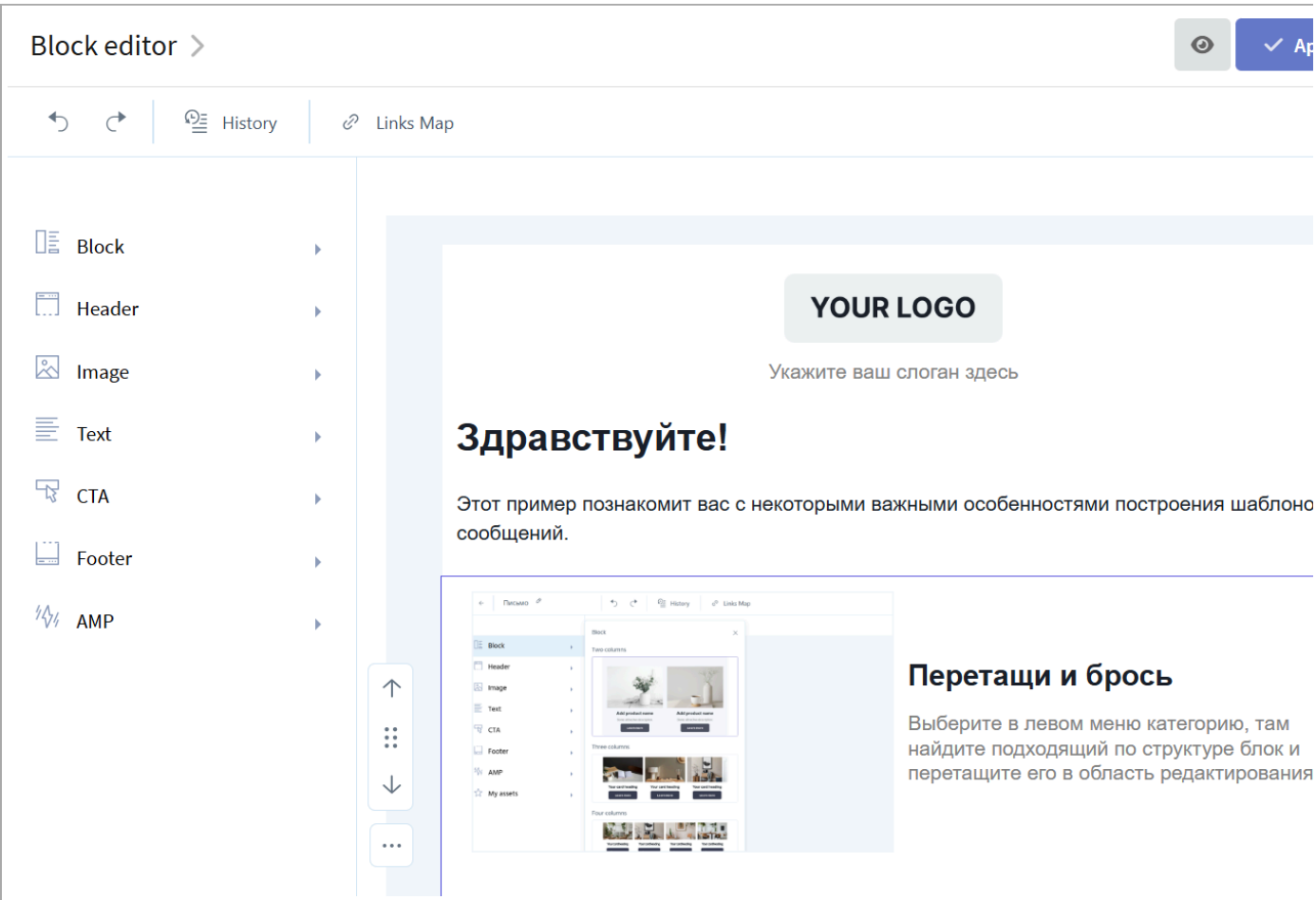


In the new email template customization window, you need to select the editor to continue working in.

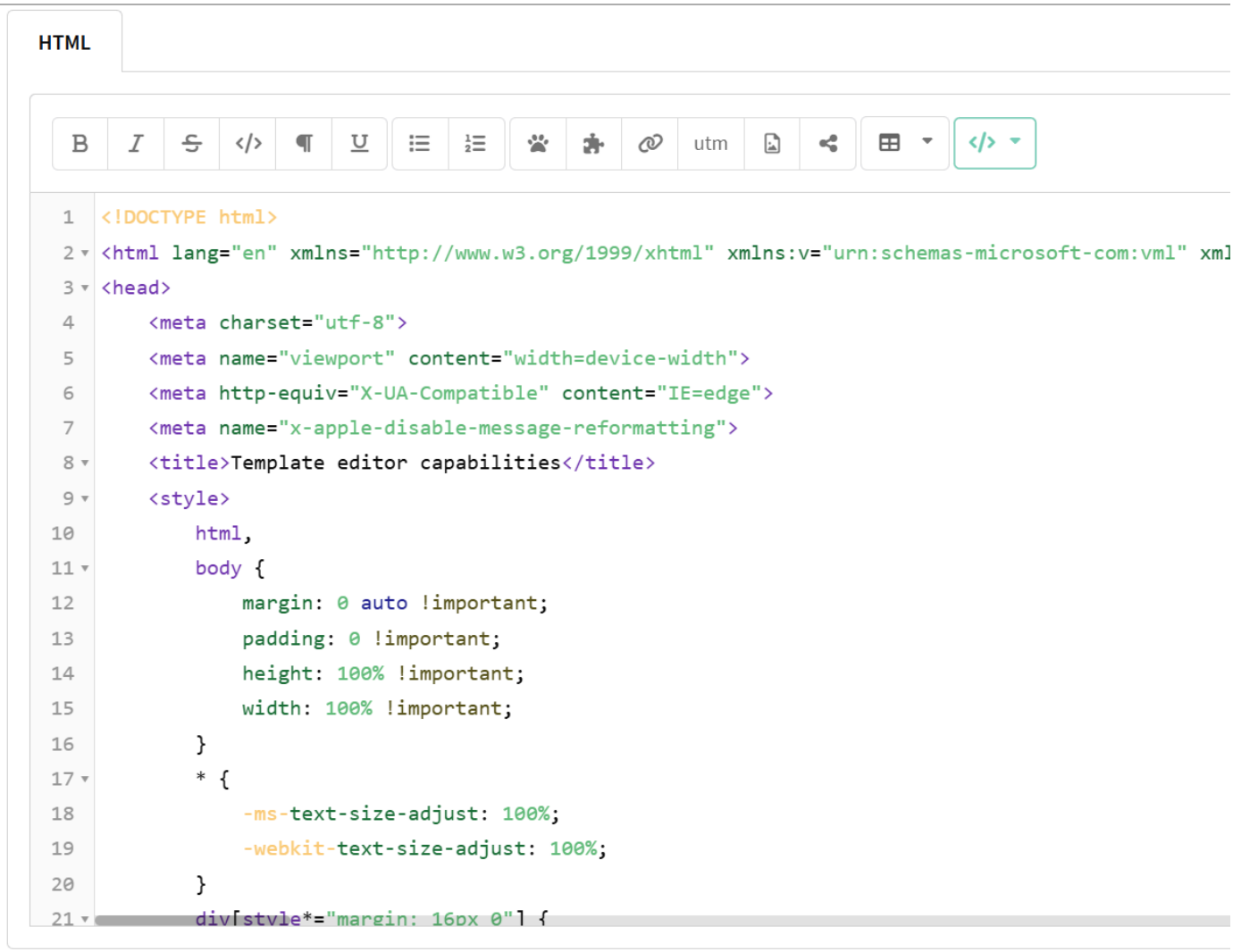


There are three editors built into the platform:

1. Block editor allows you to quickly create a template from scratch using ready—made blocks. It is suit with HTML and CSS. AMP elements for creating interactive emails are also available in this editor. Read this article.



2. Code Editor is designed for working with HTML and CSS. Use it to make changes to an existing templa about working in the code editor.



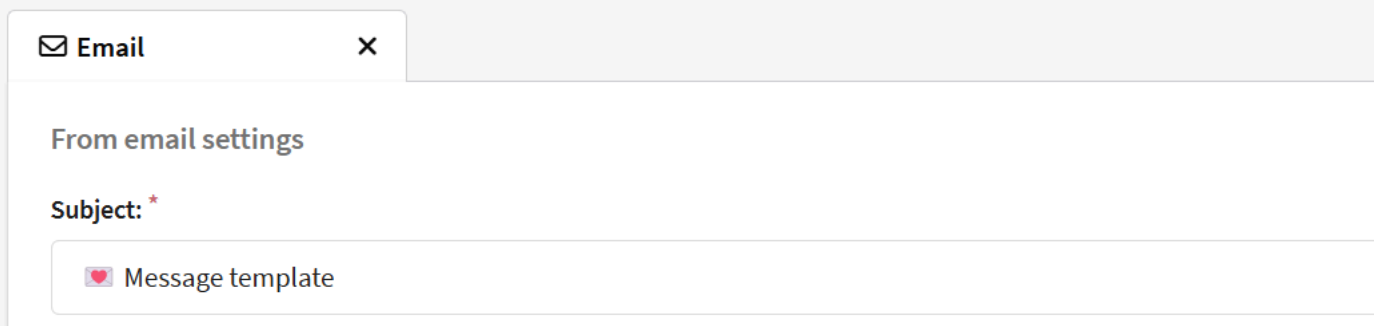
3. Visual Editor is suitable for editing a ready—made email template. With its help you can quickly chang image without any knowledge of HTML and CSS. To access the visual editor, first open the code editor, t the upper right corner. Working in the visual editor is described in [this section](#).

CAUTION

Notice, you can switch from the block editor to the code editor, but you cannot switch back.

Email-template settings

In the main content settings for the email—channel, specify the subject that will be displayed in the recip personalize the subject using the `</>` variable menu or manually using system variables. You can also ad



Fill in the following fields below:

- **From name — the name from which the email is sent to the subscriber. You can add personalization a**
- **From email — the address from which the email is sent to the customer.**

Fill in the following fields to Customize Feedback:

- **Reply-to name — name to reply to the message. Personalization is available: for example, you can sp personal manager.**
- **Reply-to email — the e-mail address to reply to the message.**

From name:

Enter website name

From email:

example@example.com

Reply-to settings

Reply-to name:

E.g. your company name

Reply-to email:

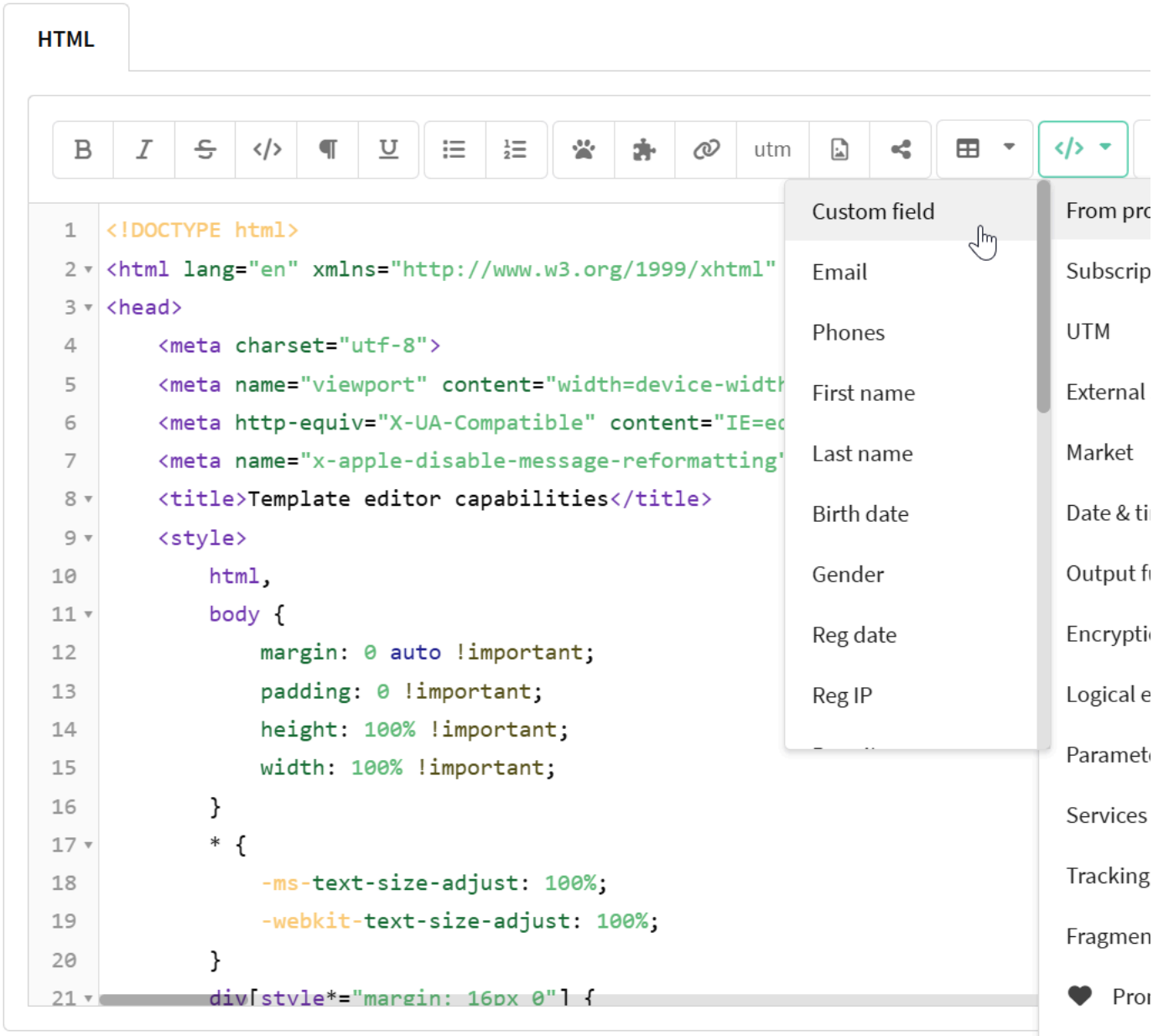
example@example.com

CAUTION

The From name and From email fields override the values configured for the Email channel in resource values of similar fields in profile database.

HTML-version of the email

Most subscribers see an HTML version of the message in their mailboxes. The code editor supports HTM variables.



The following functions are available to facilitate formatting and working with objects:

Formatting

B

— bold text

I

— italics

~~S~~

— ~~strikethrough~~ text

¶

— formatted text

U

— underlined text

— bulleted list

— numbered list

— insert table

Service functions

— adding pixels to track the opening and reading of the email. If you don’t insert pixels while editing a m about it when saving.

— adding external HTML content. You can also specify the source address of the variable manually `{[htt`

— add or edit links.

utm

— setting UTM.

— add or edit images.

— adding buttons Social share links

— menu editor variables and functions. Using this button, you can add customer profile data, promo code well as dynamic content and system links to the message.

> TIP
> **Using the row of buttons on the right side of the window, you can enable toggle word wrapping**
>
> **or full screen mode**
>
> .
> **You can also use the visual editor to edit the finished template.**

Text version of email

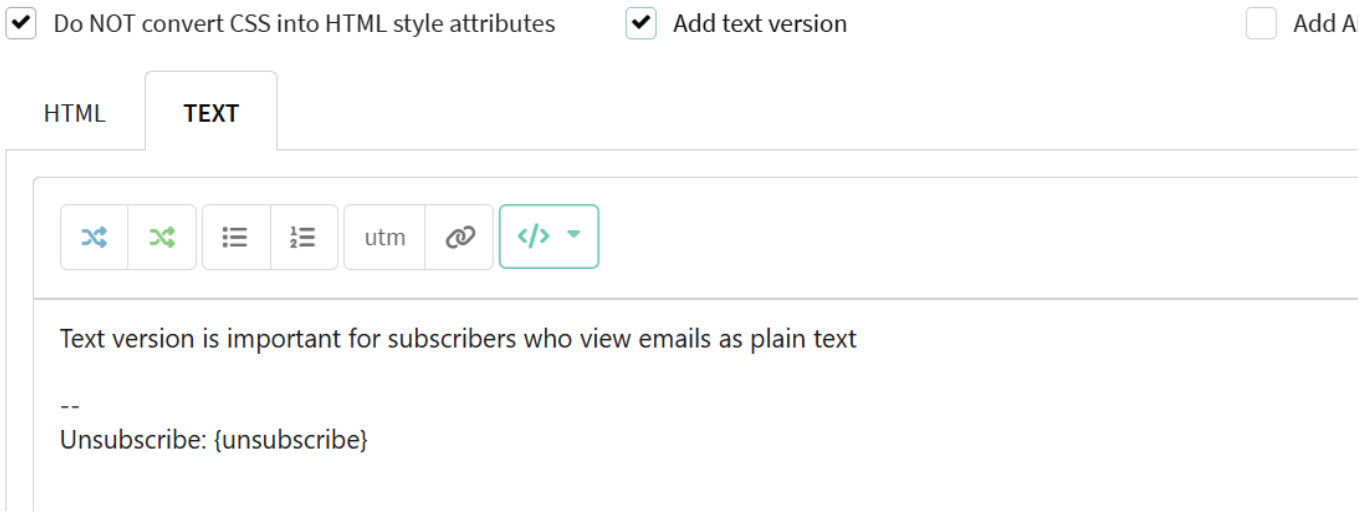
The text version of the email is text without formatting. It is displayed in the mail instead of HTML if the client itself blocks the HTML version. This version is disabled by default. To include it in an email, check email.

You can generate a text version based on HTML using the buttons:

— generates a text version with hyperlinks.

— generates a text version using Markdown links, which can improve compatibility with various mail sys

The text version also supports lists, UTM tags, links and variables.



CAUTION

If the HTML version of the email contains an unsubscribe link or a confirm link, it must also be present Otherwise, you will not be able to save the template.

AMP-version

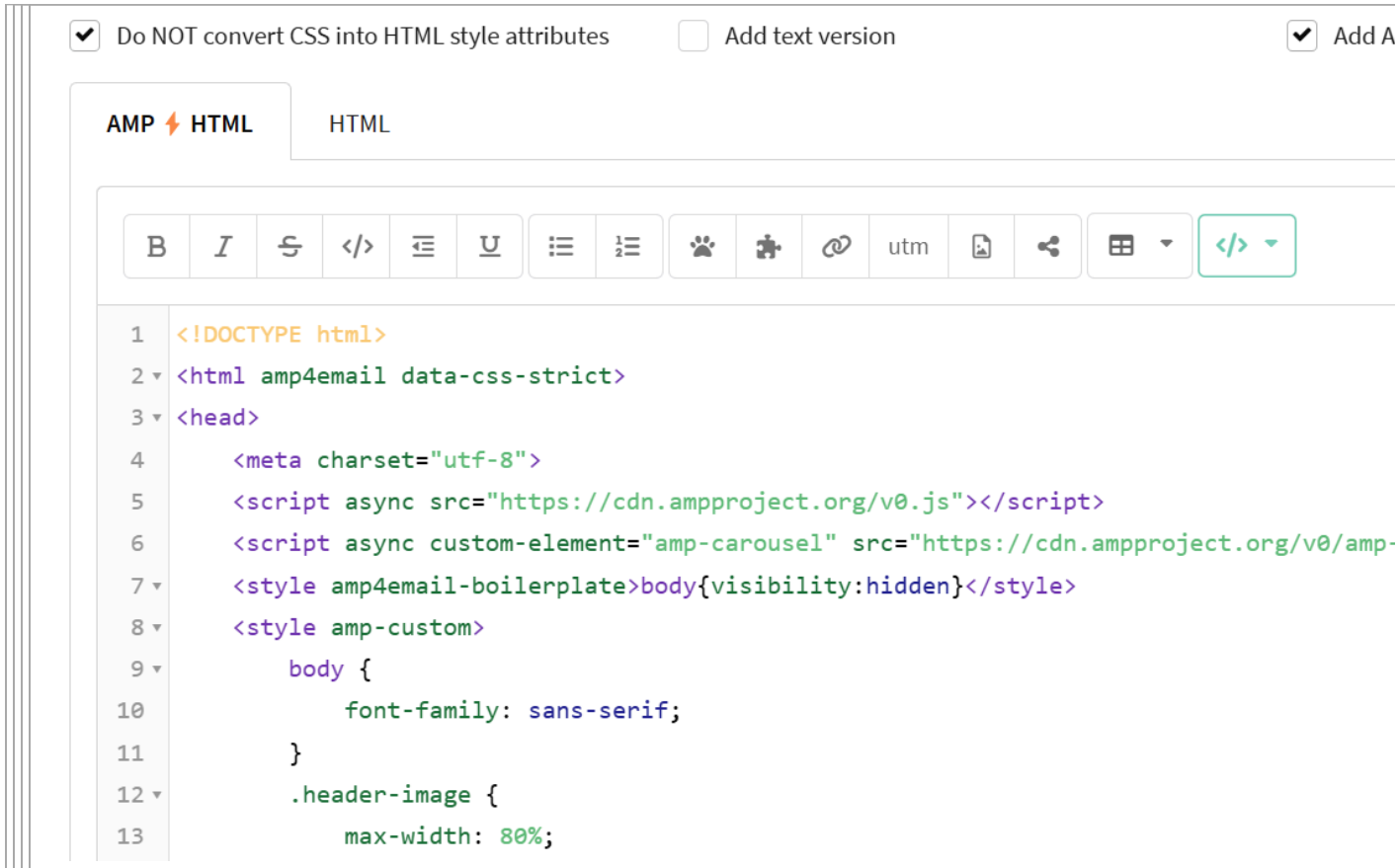
AMP allows you to add JavaScript components to emails and make them interactive. In an AMP email, th product, or confirm registration without going to the site.

Email clients require special registration from the senders of AMP emails. For instructions on adding you list, go to the official AMP documentation.

TIP

For those who are just starting to master AMP, this guide will be useful .

To send AMP emails, you need to check the box in the field Add AMP⚡HTML version. An additional tab AM you can use AMP components.



TIP

At the moment, AMP emails support the following email clients:

- **Gmail;**
- **Yahoo;**
- **Mail.ru.**

If an email arrives on an email client that does not support AMP technology, the recipient will see the H

Links in the AMP version

In the AMP version of the email, only the `data-name` attribute is supported for naming links. When saving t automatically insert the attribute into all links.

```
<a href="https://example.com " title="Link text" data-name="link1">Link text</a>
```

Also note that if you add an AMP parameter to a link, you do not need to wrap the link with tracking, oth tracking, add the `no-track` attribute. Clicks on such a link will not be tracked.

```
<a href="${{exampleUrl}}" title="Link text" data-name="link1" no-track>Link text</a>
```

Escaping template expressions

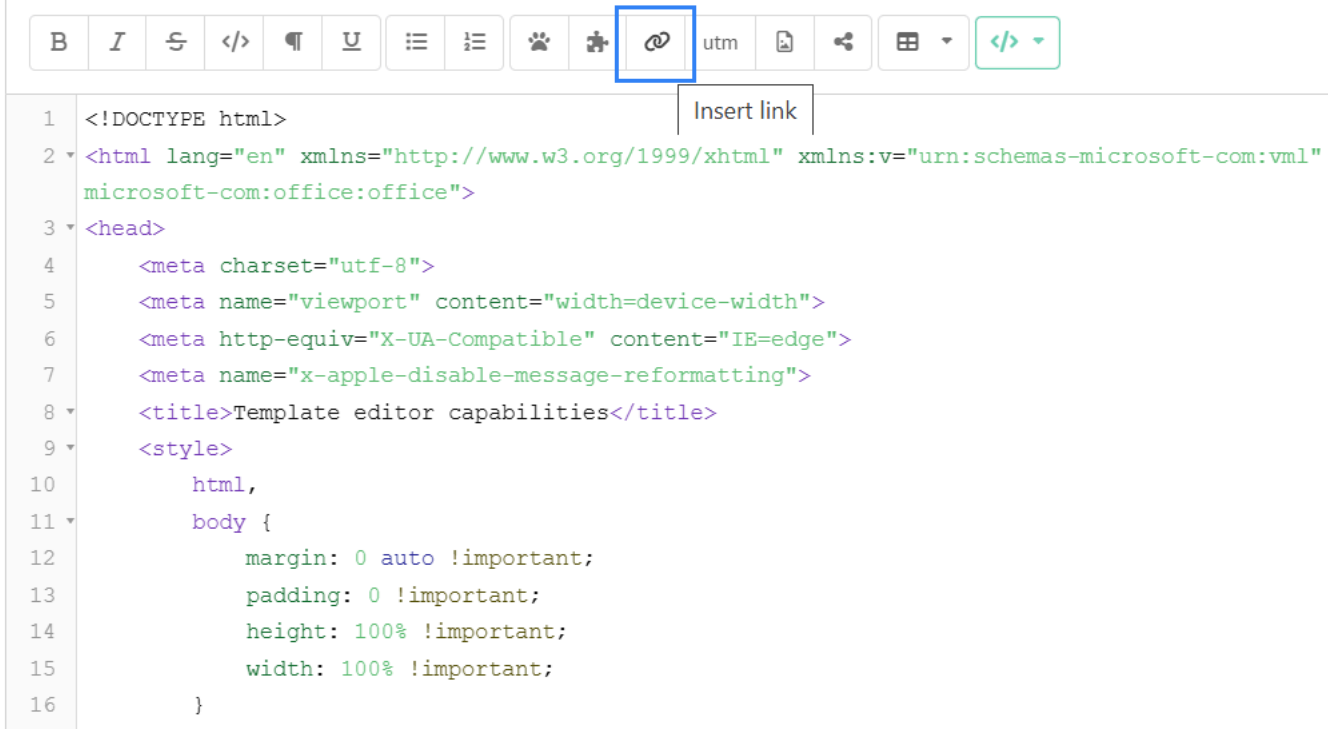
The AMP templating syntax is the same as the Altcraft templating syntax. To avoid uncorrected code pro template AMP expressions using the $ symbol.

For example, if you are using the `<amp-list>` component, then replace the variable for substituting the list

```
<amp-list id="myList" src="https://foo.com/list.json">
  <template type="amp-mustache">
    <div>${{title}}</div>
  </template>
</amp-list>
```

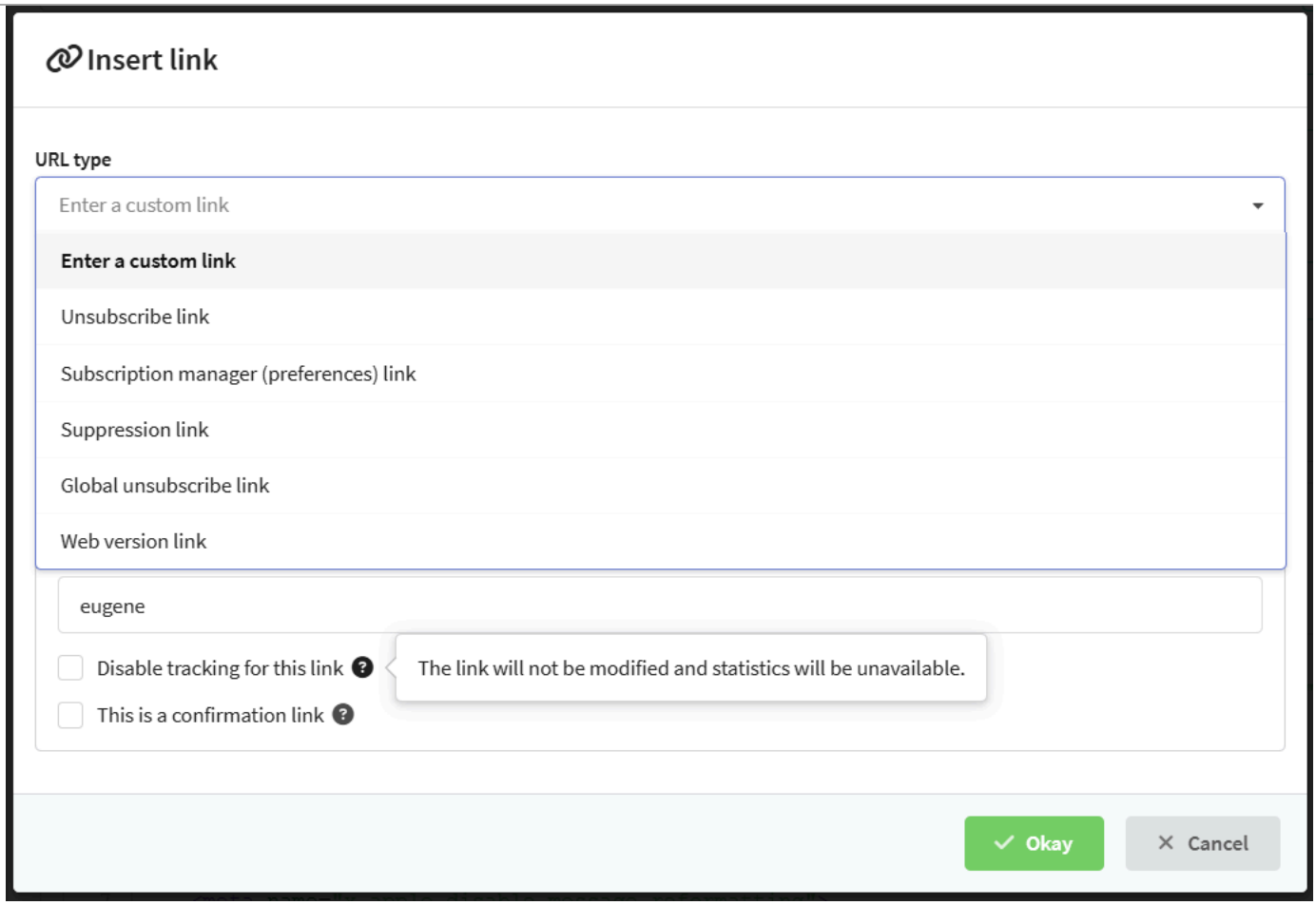
Link builder

A convenient link builder is built into the email editor. To add a link, place the cursor in the appropriate li link. If you want to change a link that you have already added to the template, select it in the editor and



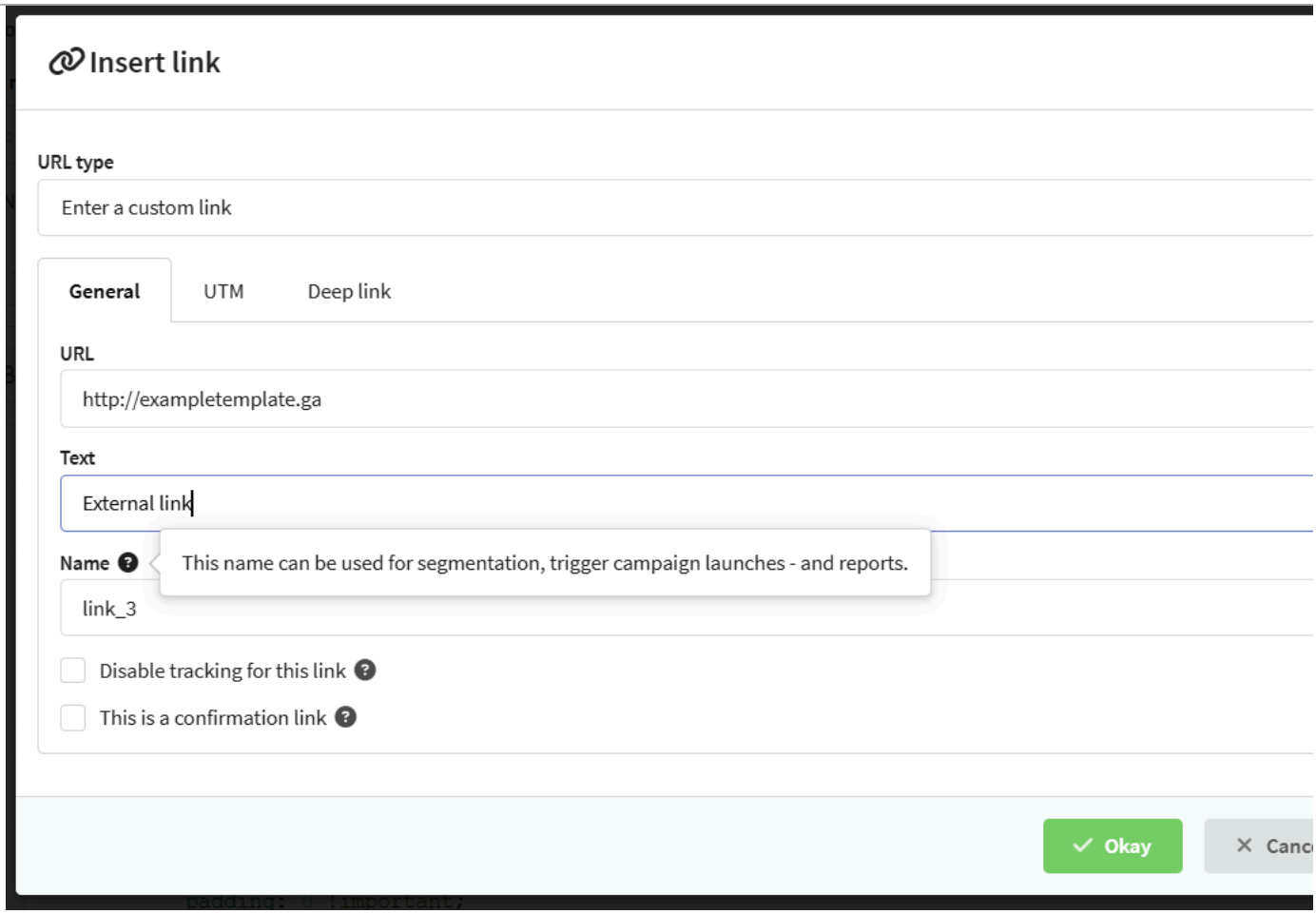
In the constructor, you can select the link type depending on the purpose:

- **Enter a custom link — link to any web page except user links unsubscribing and confirming.**
- **Unsubscribe link — a link to the system or user's subscription termination page for the resource. The Unsubscribed. Does not affect the global profile status and other subscriptions.**
- **Subscription manager (preferences) links — link to the subscription management system page. The subscriptions, resume or unsubscribe from newsemails.**
- **Suppression link - link to add to the stop list, by which the client will be able to block the receipt of**
- **Global unsubscription link — changes the global profile status of the client to Unsubscribed, comple communication.**
- **Web version link - link to an exact copy of the letter sent to the subscriber. If the client has problems available via a link from any browser.**



General

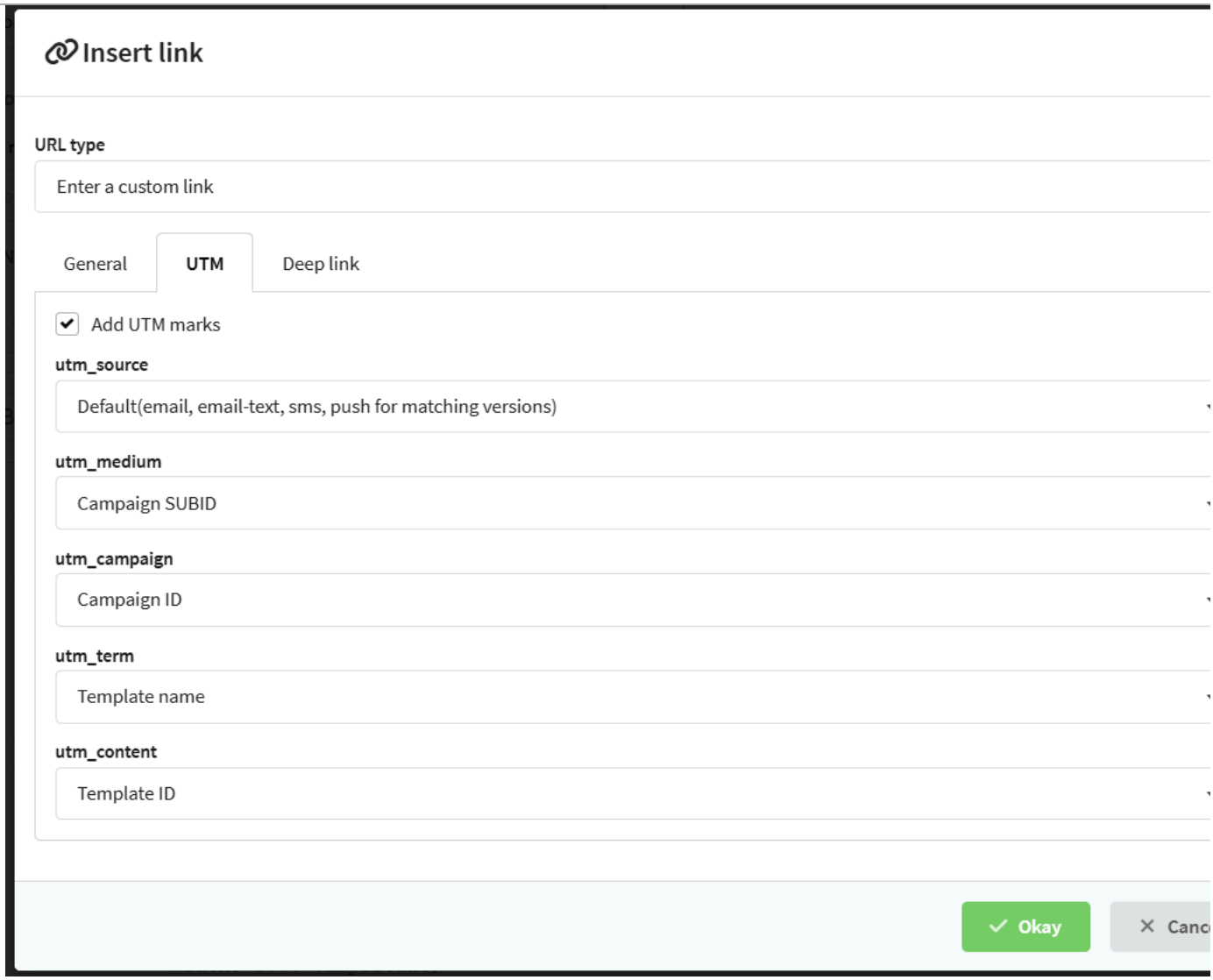
In the general settings enter the URL to go to if you are adding your link. For system links, the field will b in the email and name. Name is used in reports, segmentation and when launching triggers and scenario Additionally, you can disable redirection to tracking when clicking on the link, as well as make any link a



UTM marks

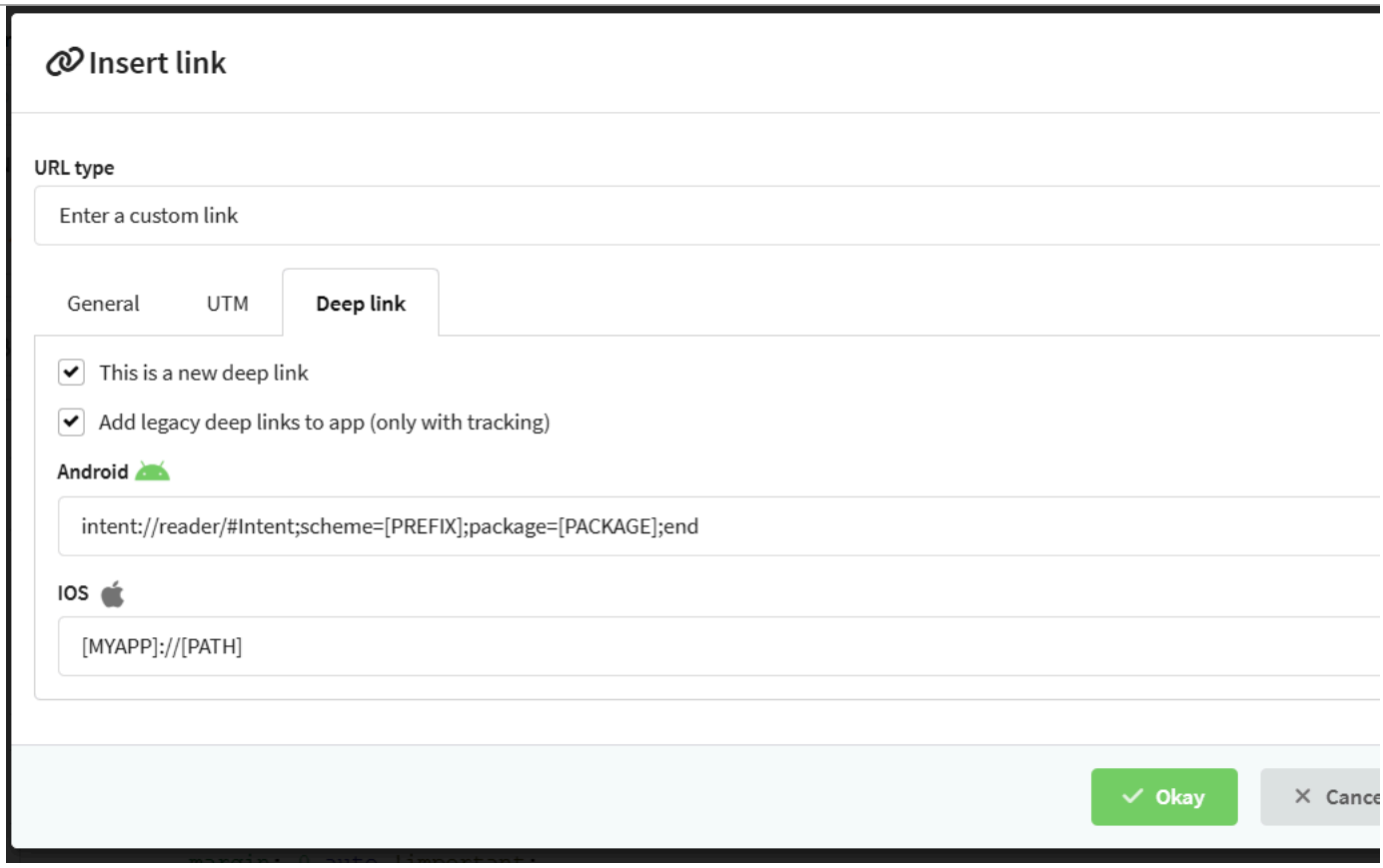
For each individual link in the email, you can override the UTM tags. To do this, go to the UTM tab and ac

Variables can be used as labels, such as the name of the mailing or databases.



Deep link

If you want to use a link in the email to go to the mobile application, activate the option This is a new de
To specify different links for Android and iOS platforms, add legacy deep links. Please note that such link



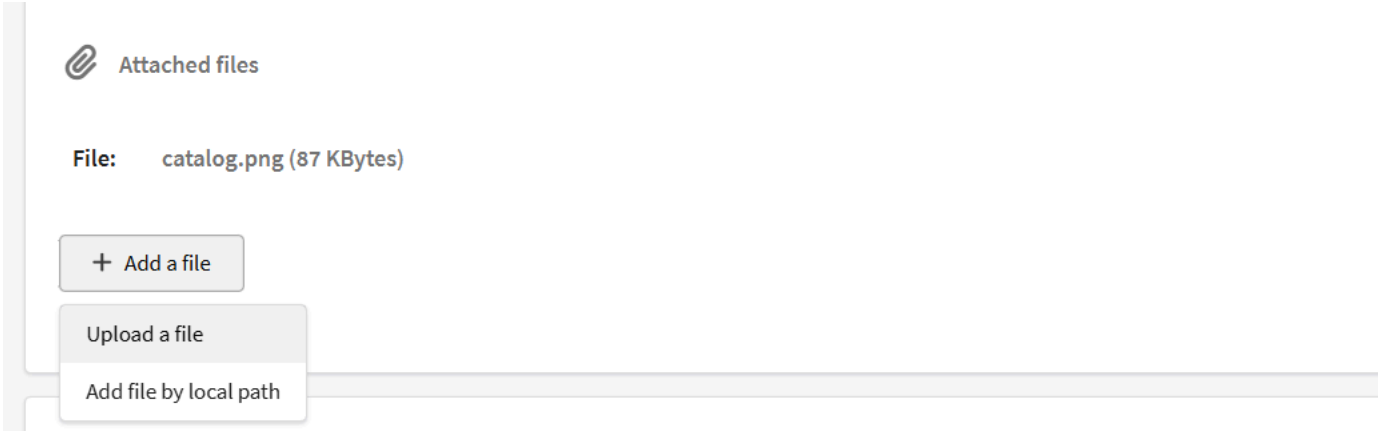
Email attachments

INFO

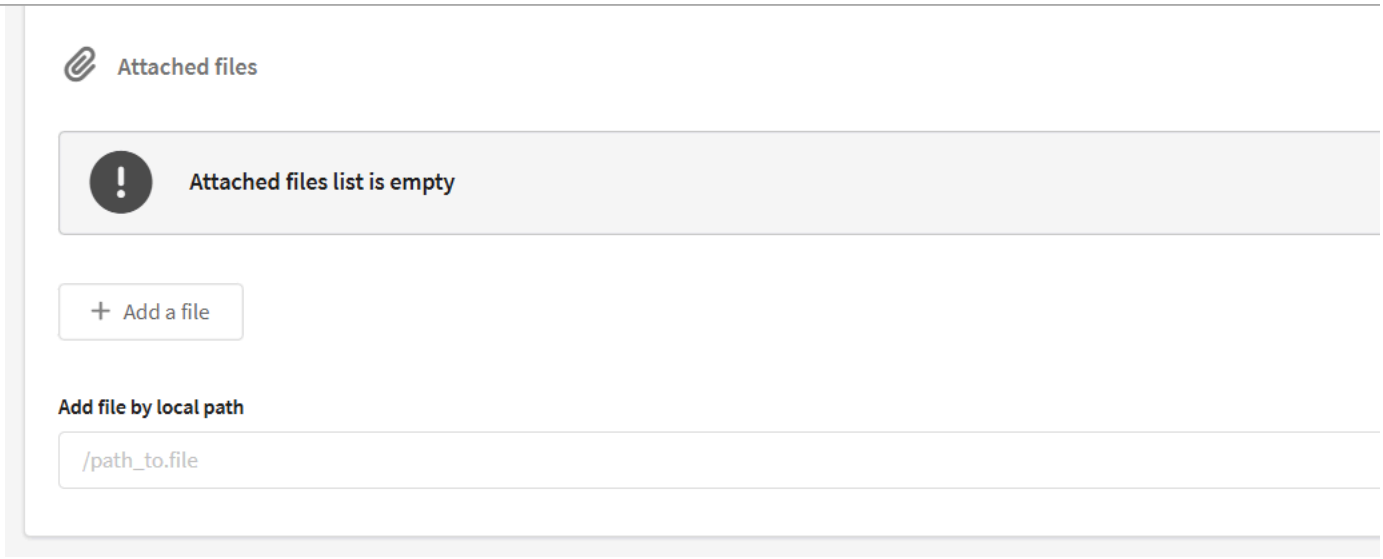
You can attach a file of any format to the email, but the maximum file size is limited to 12 MB. This valu
`MTA_ATTACH_MAX_SIZE` **configuration parameter.**

To attach a file to an email, use the form at the bottom of the editor. There are two options:

1. **Upload a file to the platform from your computer:**



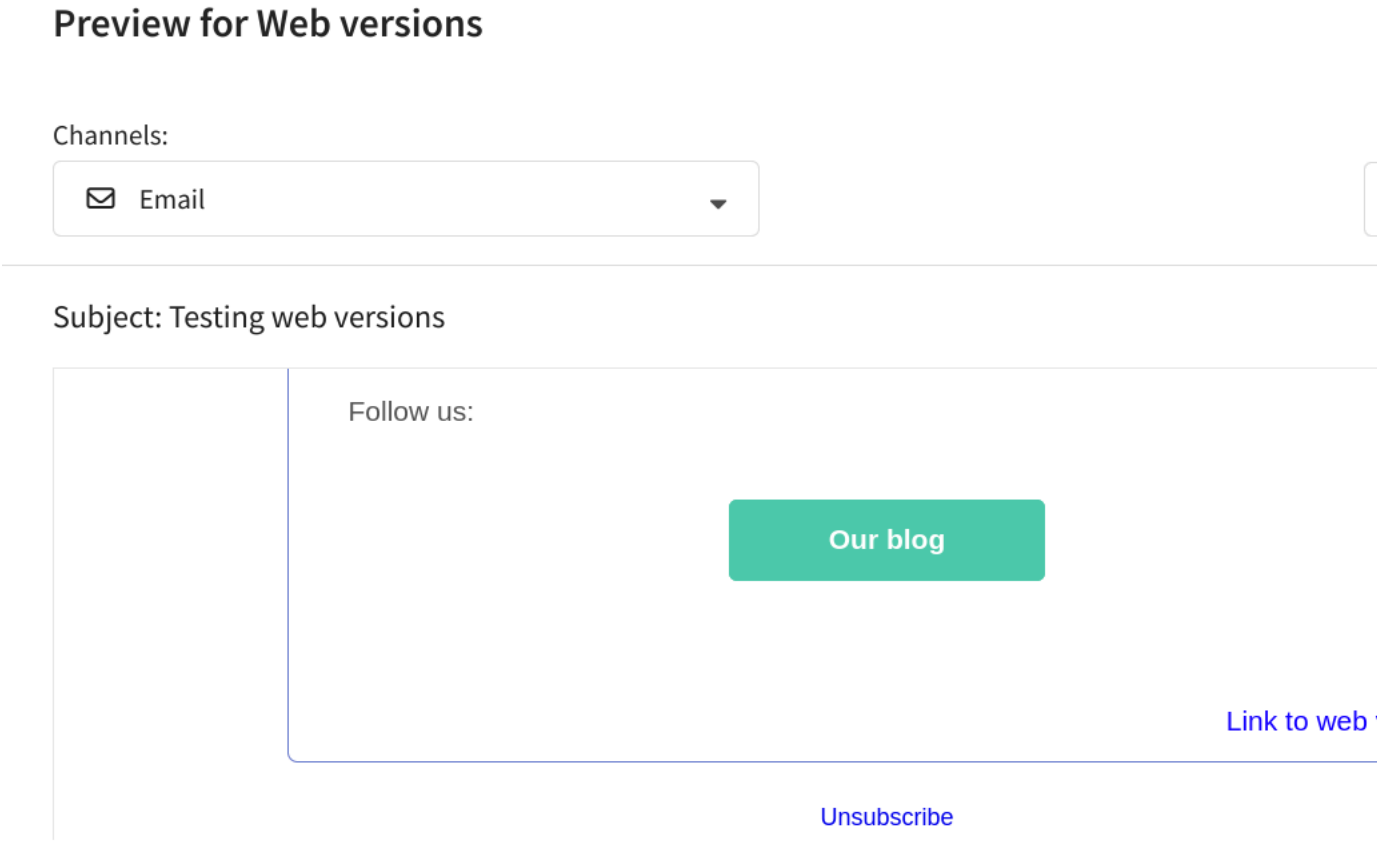
2. **Specify a file on the server on which the platform is installed. The file must be located in the attachr**
   `ACCOUNT_FILES_ATTACH_EXT_PATH` **configuration parameter). Enter the file path relative to this directory and**
   `image.png` **is located in the** `documents/` **subdirectory, specify** `documents/image.png`**. You can use profile vari**



You can delete downloaded files and saved file paths by clicking on the cross on the right side of the file

Web-version of email

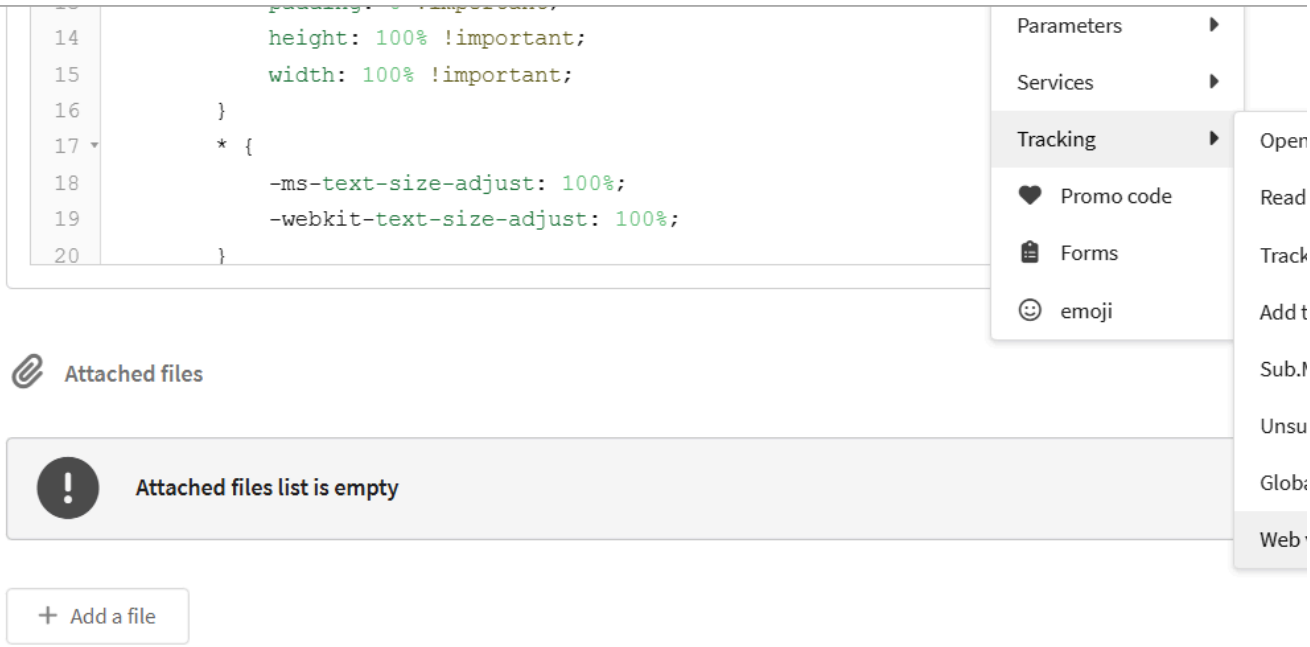
The web version of email is the version of the email message that the user can view as a web page in th display in mail services. Web versions increase the chances that your message will be viewed even with the presence of a link to the web version allows you to quickly and conveniently share a letter if necessa



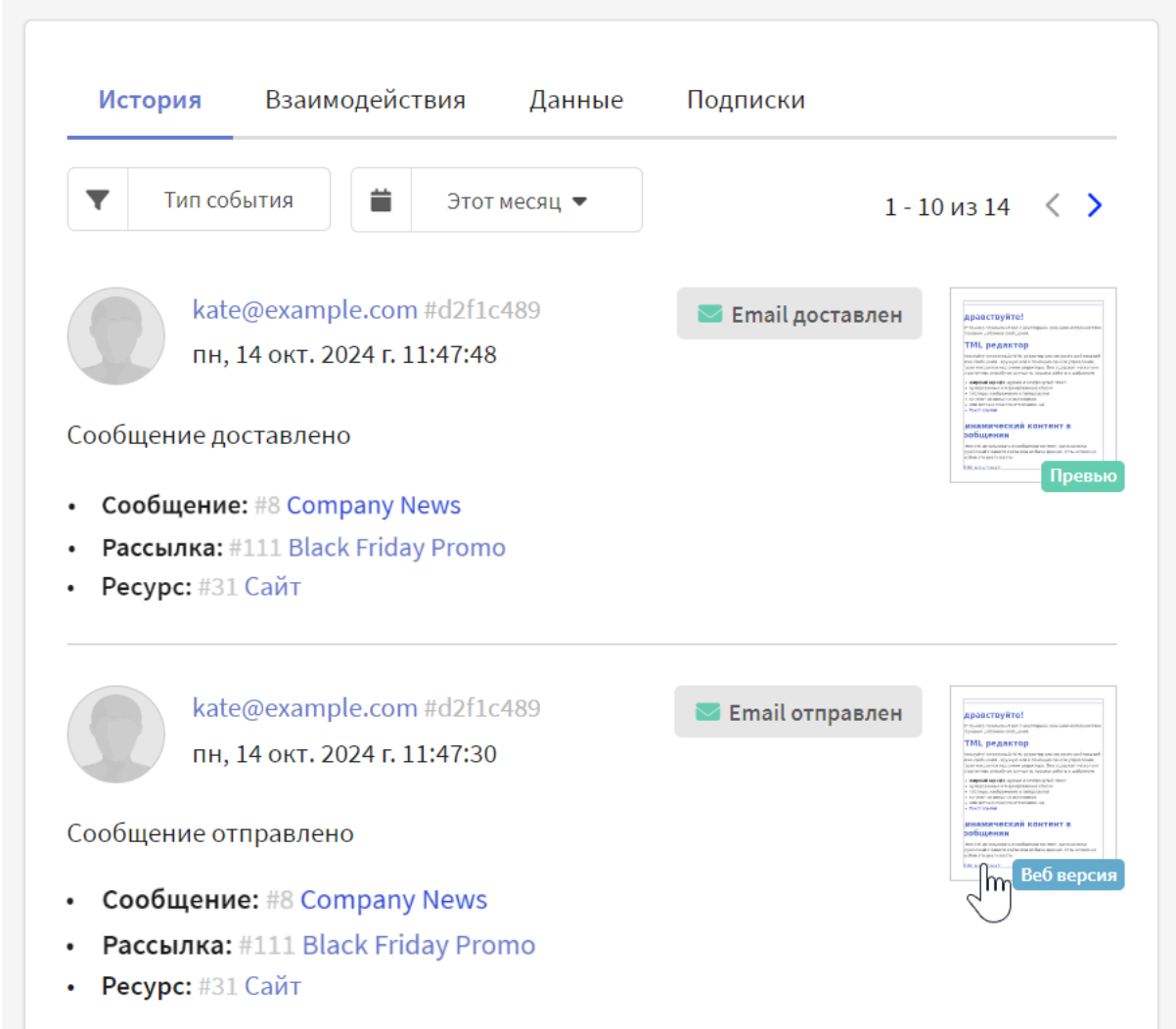
You can add a link to the web version through the link builder or manually:

```
<a href="{webversion}" title="Web version of the email" name="web_version"> Open an email in a browser</a>
```

It is also possible to add a link through the variables menu. Open it by clicking </> on the toolbar. Select



You can view the web version of the email sent to the client in [profile history](). By default, the web versio administrator can change the retention period in the account settings.



TIP

The link to the web version does not open in preview mode and when sending a test email. The web ver messages. To check the link, send an email to the test subscriber via the broadcast.

INFO

By default, saving web versions is disabled in the account. To activate this feature, the platform admini version policy. After that, links will appear in the profile history to view the web versions of the sent me

If a link to the web version is not included in the email template, the web version of the email will not b can modify this setting by enabling the Always save email web versions option in the account settings.

SMS template

This article describes how to create a message template to send via SMS channel.

Message body

Enter the text of the short message in the input field. Links, UTM tags and variables can be used in the m

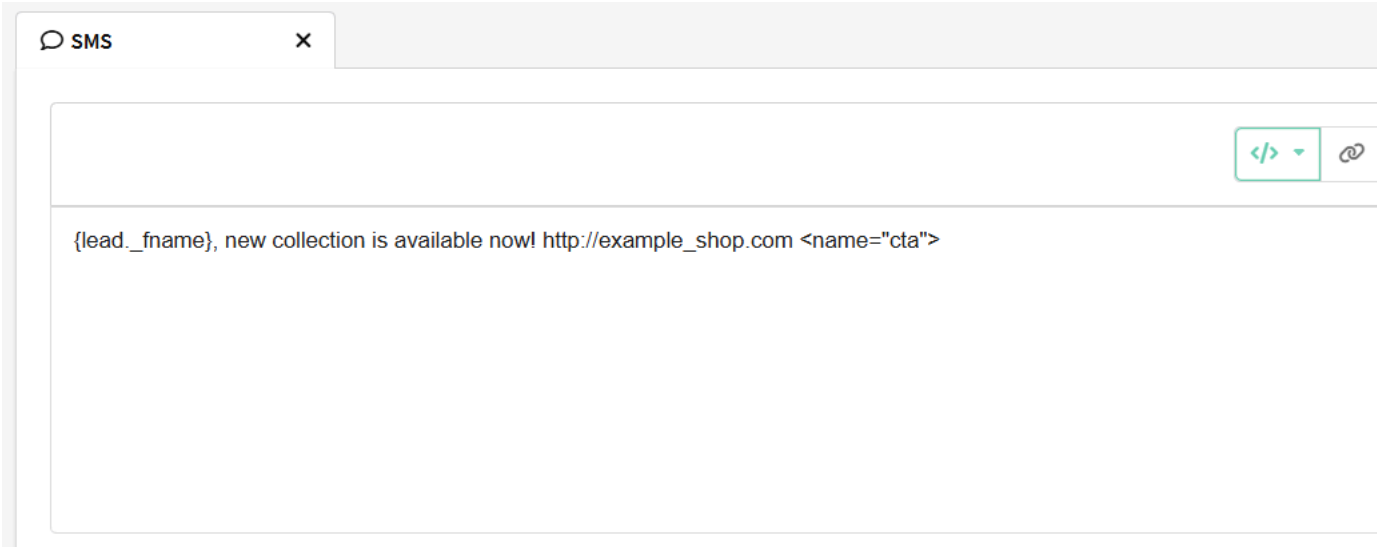
</>

— menu of variables and functions the editor. Using this button, you can add customer profile data, [prom](#) [boolean expressions](#), as well as dynamic content and system links.

— link editor.

utm

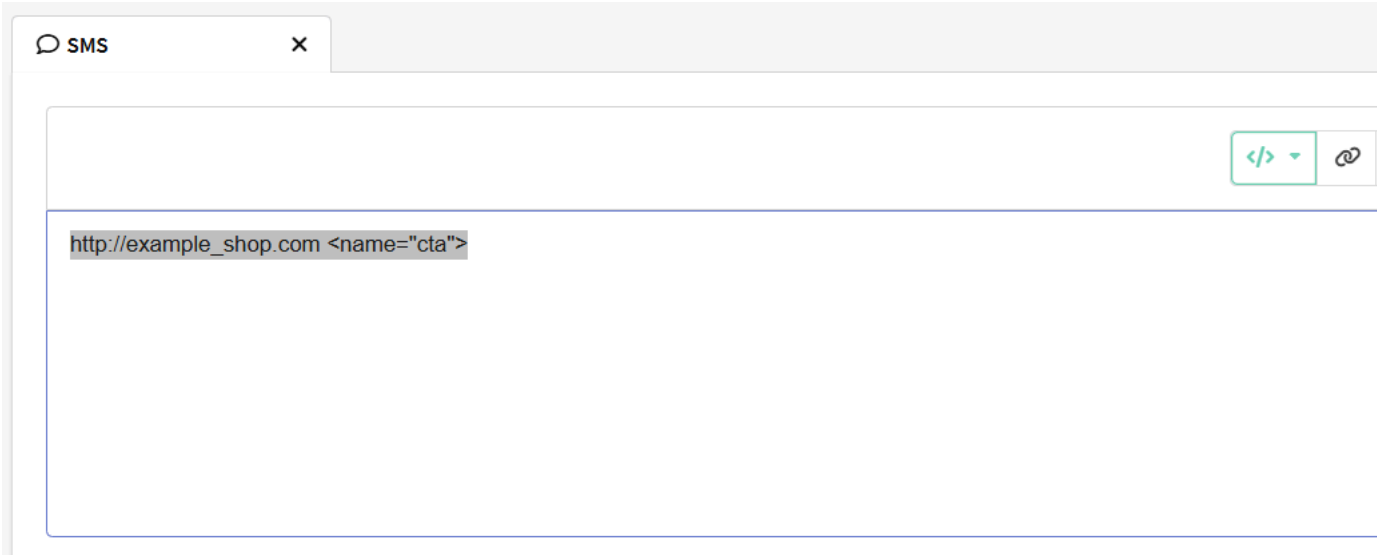
— setting UTM labels.



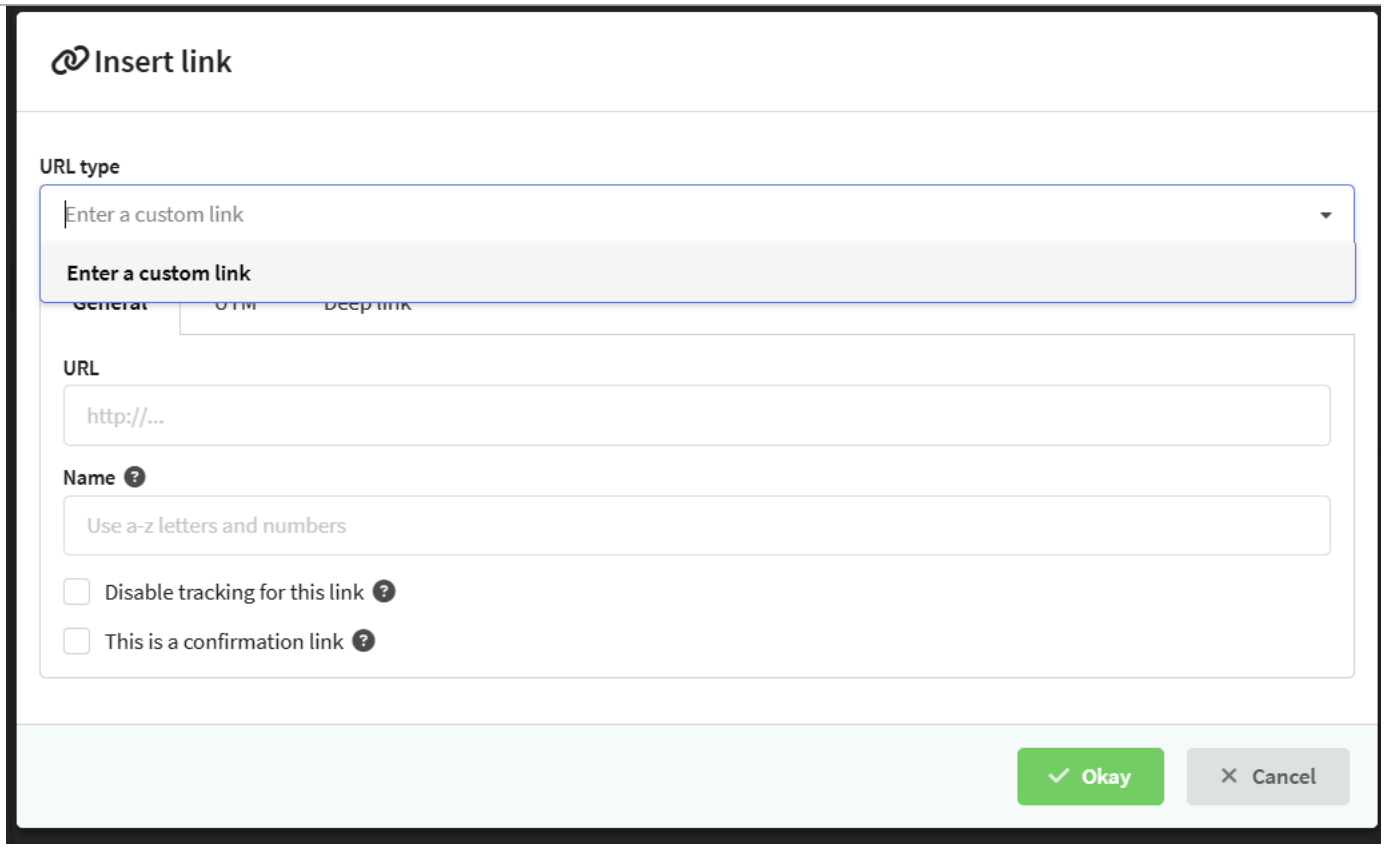
SMS link editor

> **TIP**
>
> **Links inserted through the platform are automatically shortened. The final link path (excluding tracking including the `/`.**

The link editor is available by clicking on the button Insert link. If you want to edit a link that has already it in the editor and then open the form:



One type of links is available for SMS messages. It is a custom link. It allows you to add a link to any wel

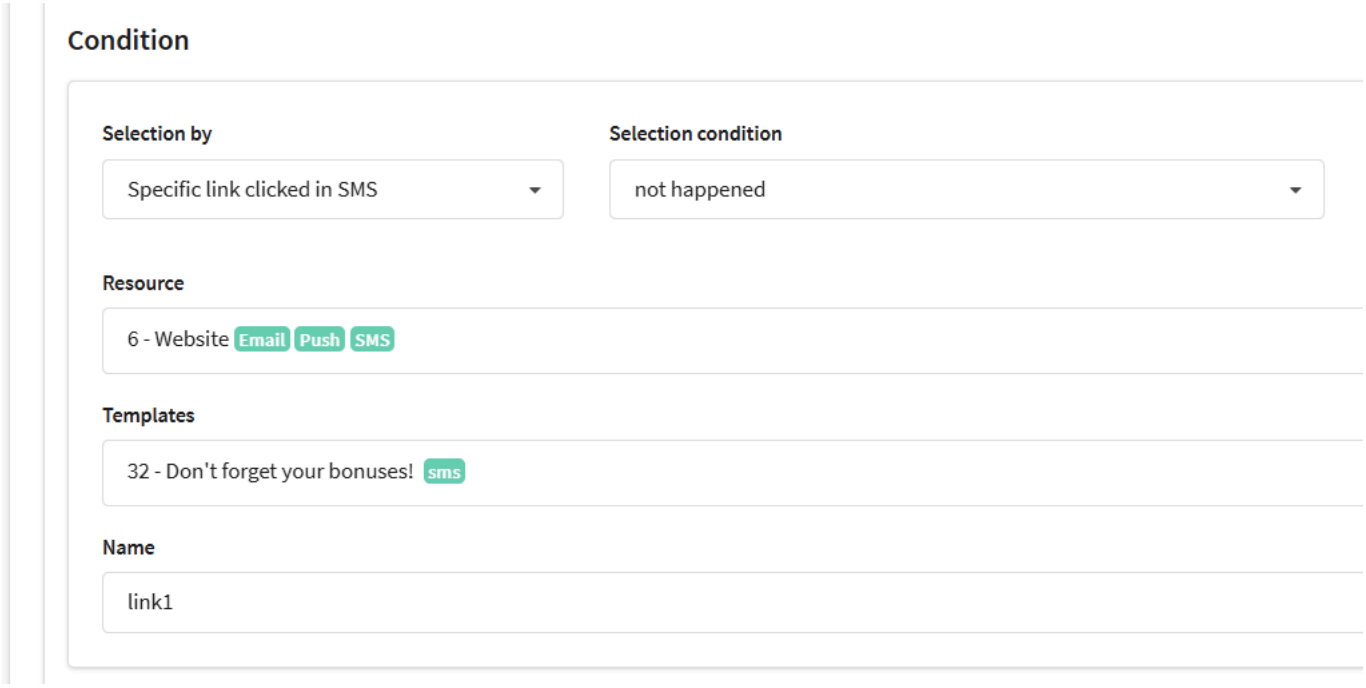


General

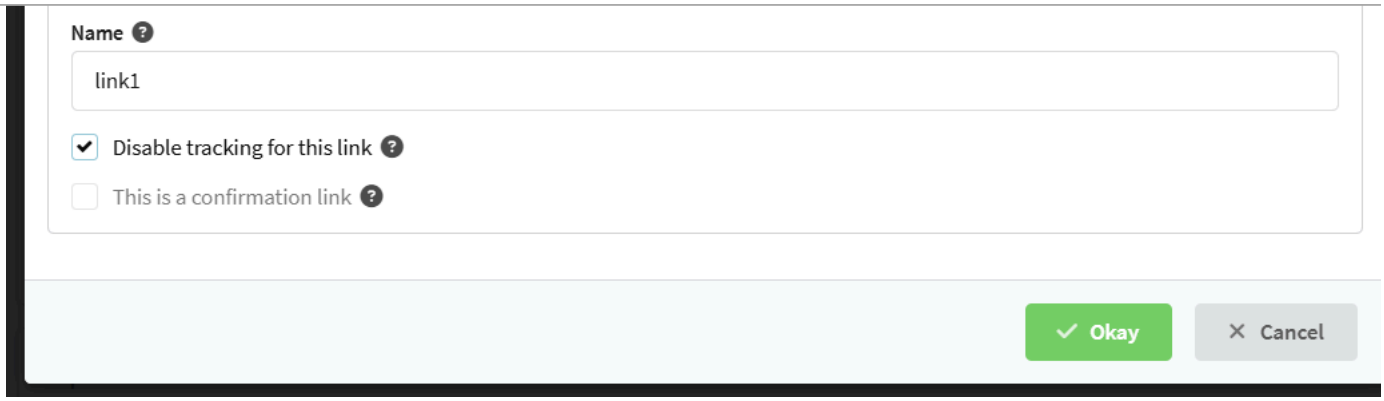
In the general settings enter the URL to go to. Also specify the name, which is used in reports, segmenta scenarios. The link name can only consist of numbers and Latin letters.

CAUTION

Please note that when you select profiles by clicking on a link in an SMS, you must specify the link nam conditions.



Additionally, you can disable redirection to tracking when clicking on the link, as well as make any link a

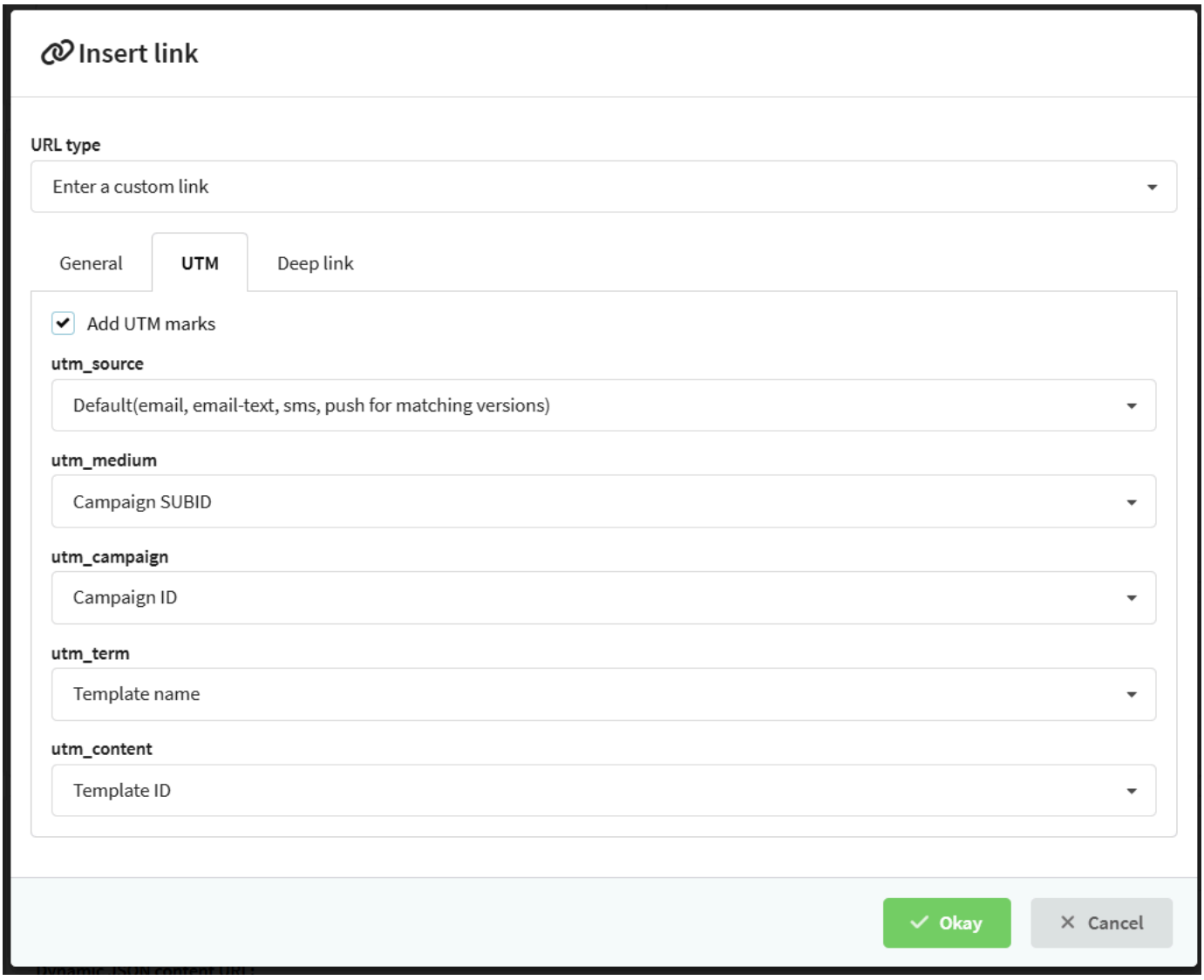


In SMS messages, angle brackets are used to indicate parameters (for example, `<name="">` will be used fo you specify in angle brackets will not be included in the final message. Usage examples:

- `<no-track>` **— disables redirection to tracking when clicking on a link.**
- `<confirm-link>` **— makes a confirmation link.**
- `<name="example_name">` **— sets the name of the link.**

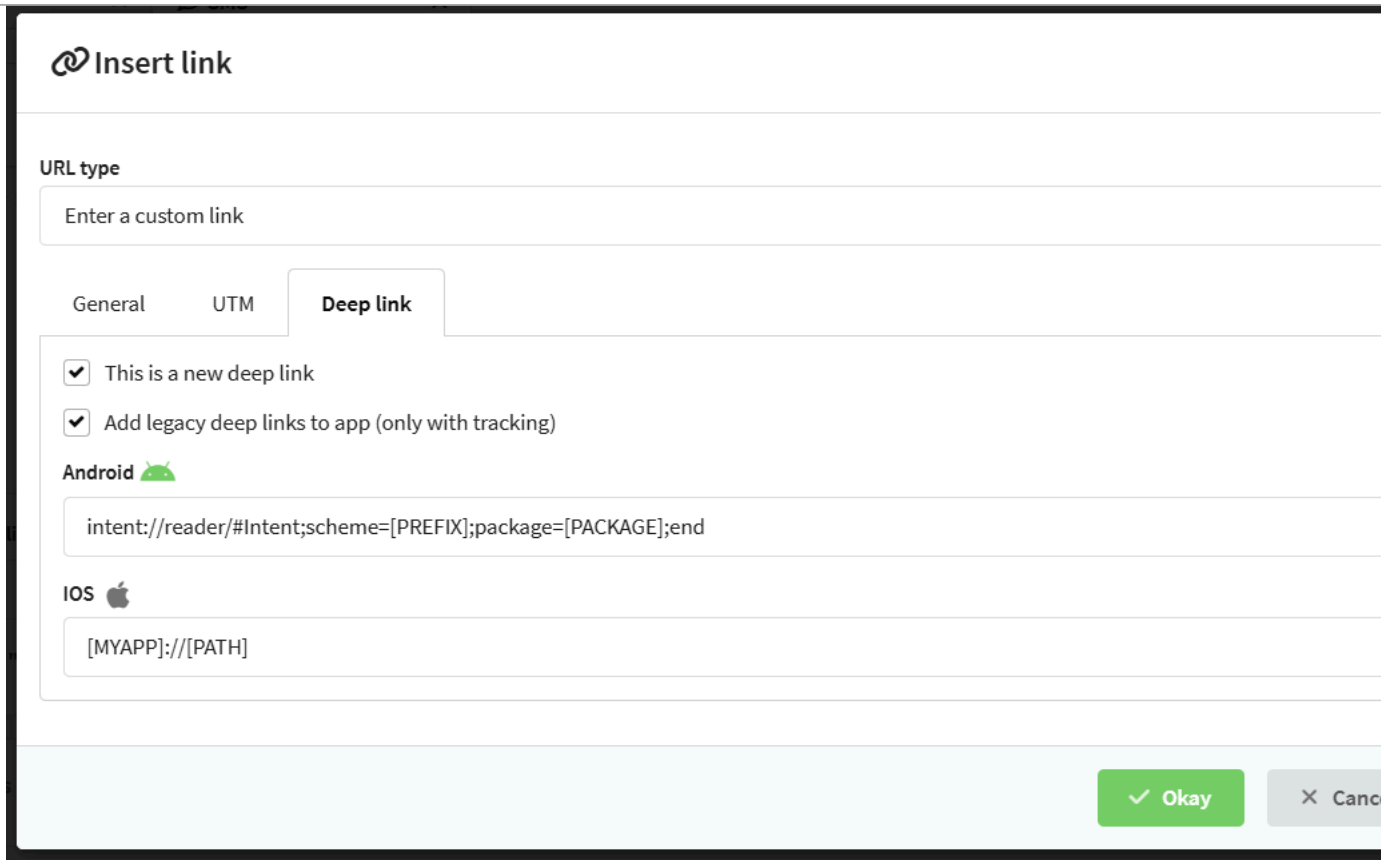
UTM marks

For each individual link in the message, you can override the UTM tags. To do this, go to the UTM tab and Variables can be used as labels, such as the name of the mailing or databases.



Deep link

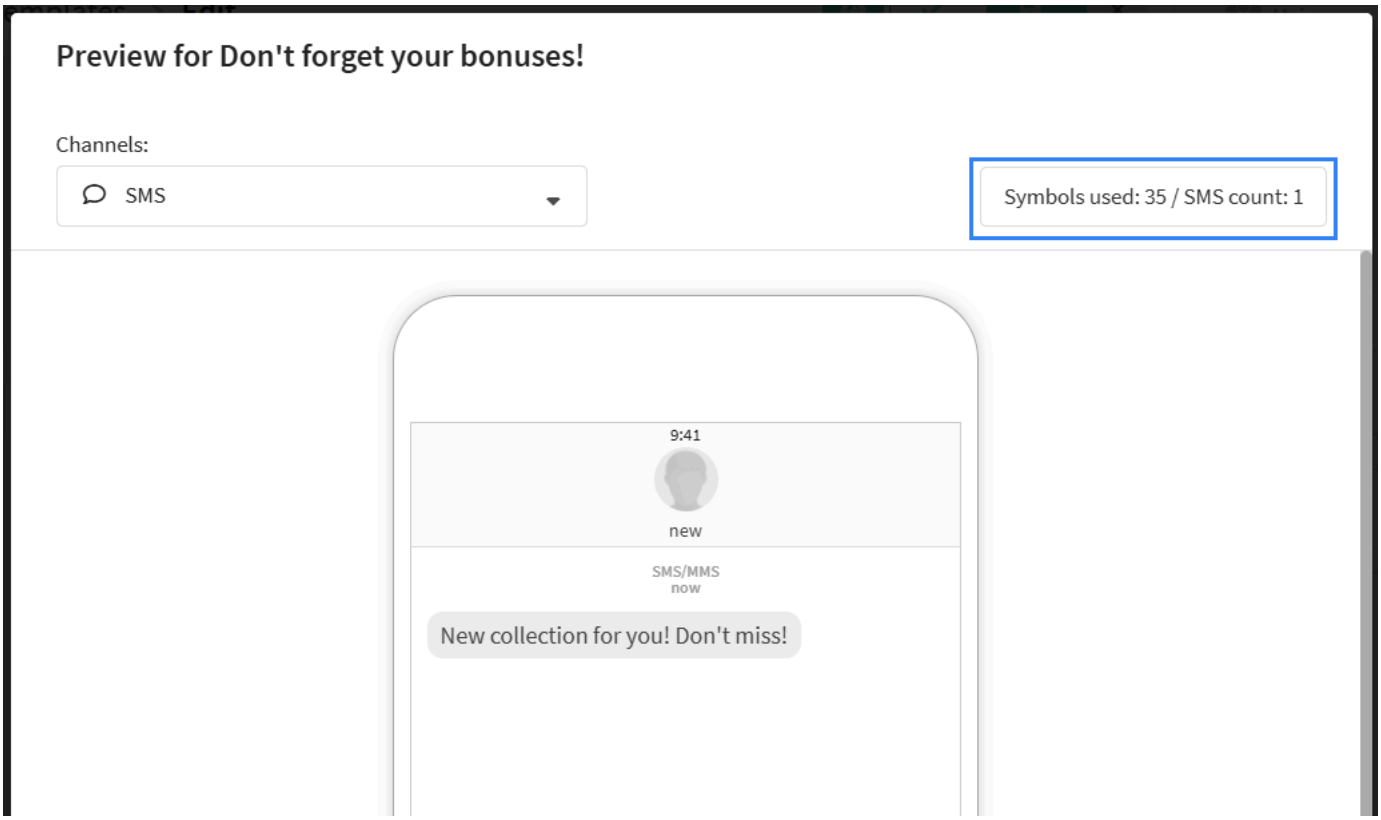
If you want to use the SMS link to go to the mobile application, activate the option This is a new deep lin To specify different links for Android and iOS platforms, add legacy deep links. Please note that such link



Limit on the number of characters

When sending SMS messages, it is important to take into account the length of the text, since the numb sending. One SMS message in Cyrillic can contain up to 70 characters and in Latin the are 160 character marks. If you exceed these limits, your message will be charged as two SMS. The presence of at least o message Cyrillic.

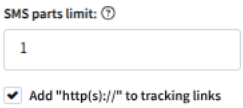
The editor automatically counts the number of characters and messages. To see the counter, save the te



TIP

Please note that the data that is pulled from dynamic variables may exceed the expected number of cha

The excessive sending can be restricted by specifying a value in the “SMS parts limit” field. If the messa parts limit after splitting, the message will not be sent at all. For example, if a user sets the limit to 3 SM characters, no part of this message will be sent during the mailing...



In addition, the `http(s)://` protocol is not added to the template to shorten links, but you can enable auto tracking links.

INFO: Notice

When a template includes logical expressions or loops, the platform counts every formatting character of the final message. Each space or line break adds one character to the SMS and may affect the cost. message length, keep logical constructions on one line.

For example, instead of:

```
{if lead._city equal "Moscow"}
We invite you to a conference in Moscow on June 30!
{else}
We invite you to a conference in your city!
{end}
```

![img](\img\attachments\37848968\sms-template.png)

Use:

```
{if lead._city equal "Moscow"} We invite you in Moscow on June 30!{else}We invite you in your city!{end}
```

![img](\img\attachments\37848968\sms-template_2.png)

This approach keeps the logic intact and avoids unexpected SMS length growth.

Push template

The platform offers two editors for push templates: a standard editor and a dedicated Rich Push editor. T banners and buttons, as well as configuring push settings tailored to specific browsers or operating syst

Standard editor

For push notifications, you can configure:

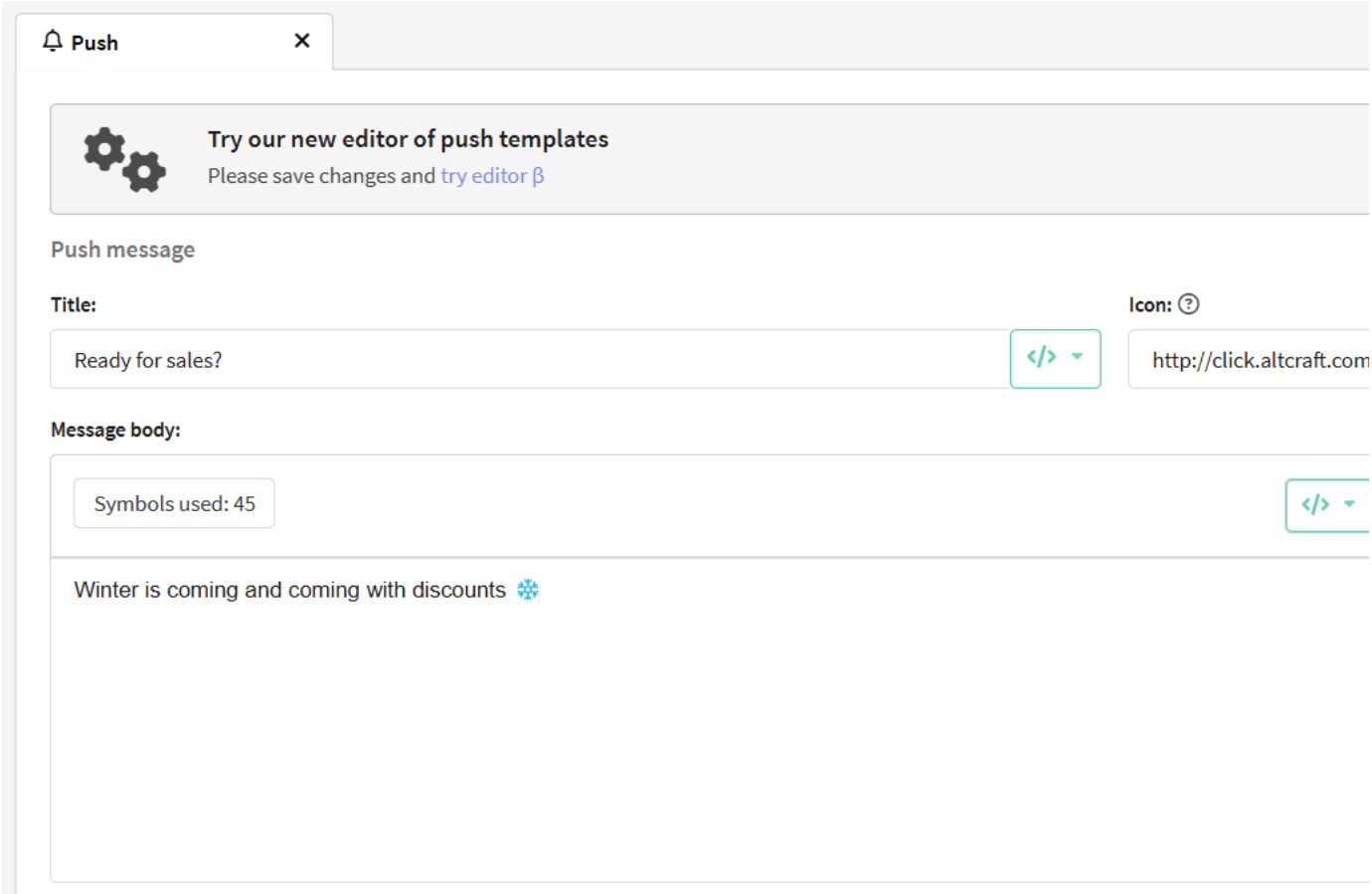
- **Title — It is recommended to use no more than 25 characters so that the text is not cut off on device can be personalized using the variable menu**

  </>

  .
- **Icon — an image to display in the notification. In the browser Apple Safari, only one image is support configuration is performed in setting up the push channel in resourse.**
- **Message body — the main text of the notification. Emojis are supported. The maximum text length is to use up to 150 characters. You can also use variables in the message body to personalize**

  </>

  . The editor automatically counts the number of characters:

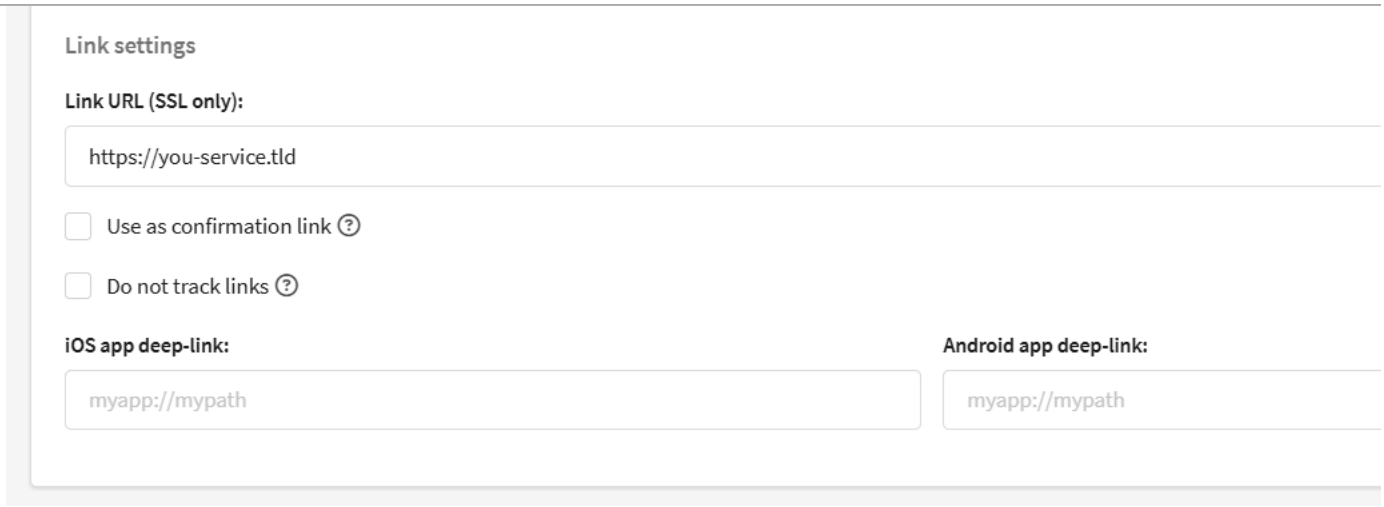


TIP

Links inserted through the platform are automatically shortened. The final link path (excluding tracking including the `/`.

In the block Link settings are available:

- **Link URL (SSL only) — The URL that opens when you click on the notification. Only the secure protoc is available for inserting, for example, promo codes, so that the user immediately goes to the page w discount.**
- **iOS app deep-link — link to go to iOS apps.**
- **Android app deep-link — link to go to Android apps.**



To add utm tags to the link click on the

utm

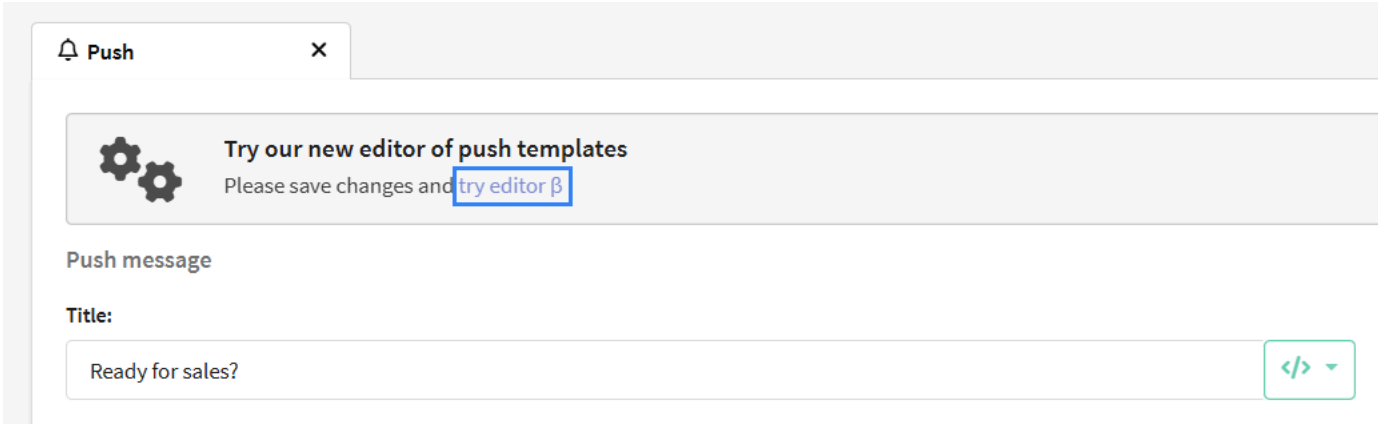
button in the upper right corner of the editor or set the labels manually.

Additionally, the link in the push can be a confirmation link. You can create a double opt-in subscription r tracking. If you decide not to wrap links for tracking, keep in mind that in this case the platform will not and deep-link links will become unavailable.

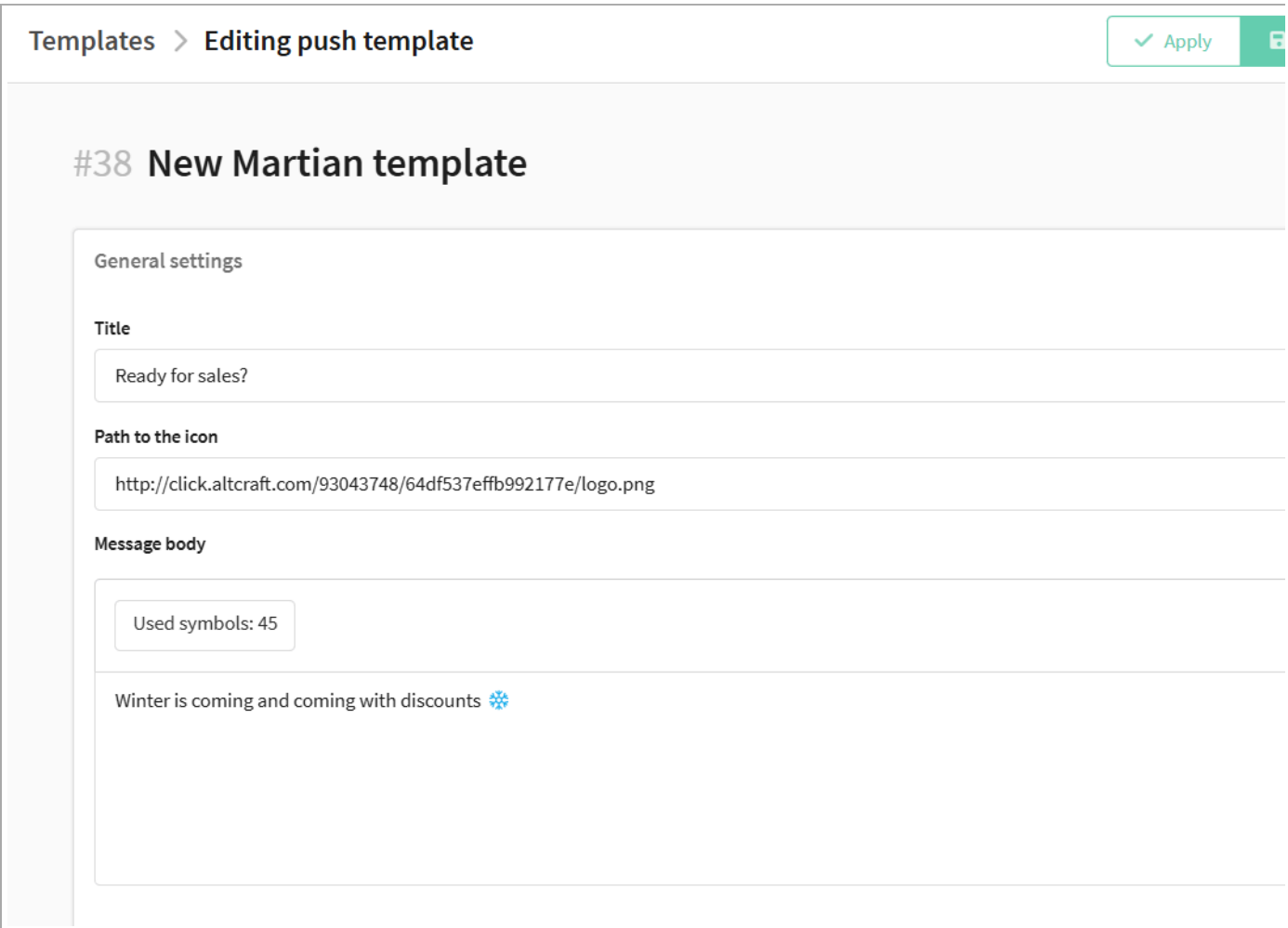
Rich Push editor

Get started

To switch to the editor for Rich push notifications, you need to prepare the initial version of the message without a link to go to) and save the template using the “Apply” button. After that, the try the β editor but



The title, the path to the icon and the body of the message will be filled with the data that you specified edit these fields if necessary.

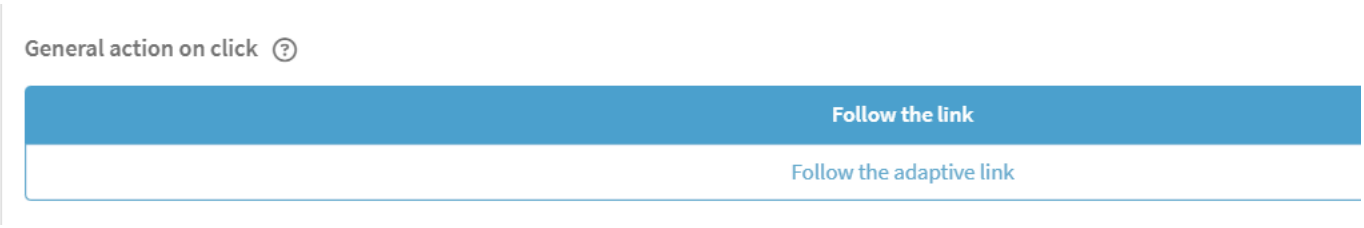


Click-through link

The general action on click is the link that the user will click when they click on the push notification. In will click on this link unless a unique link is set for different browsers and OS. The unique link is set belo

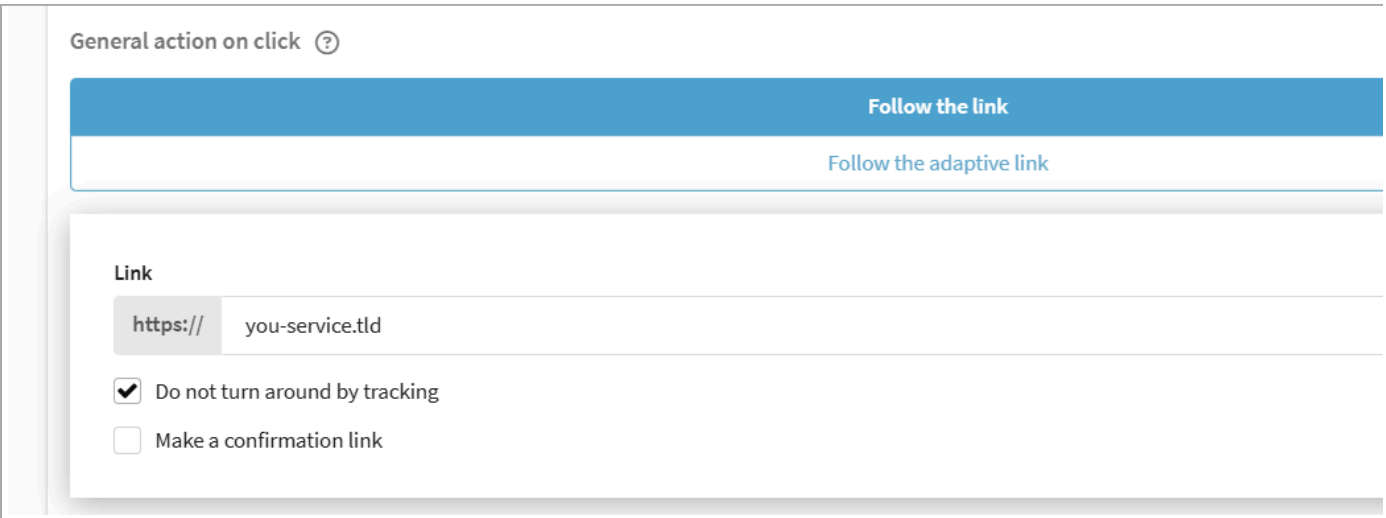
There are 2 types of links that can be used in the Rich push template editor:

- **regular link;**
- **adaptive link.**



Regular link

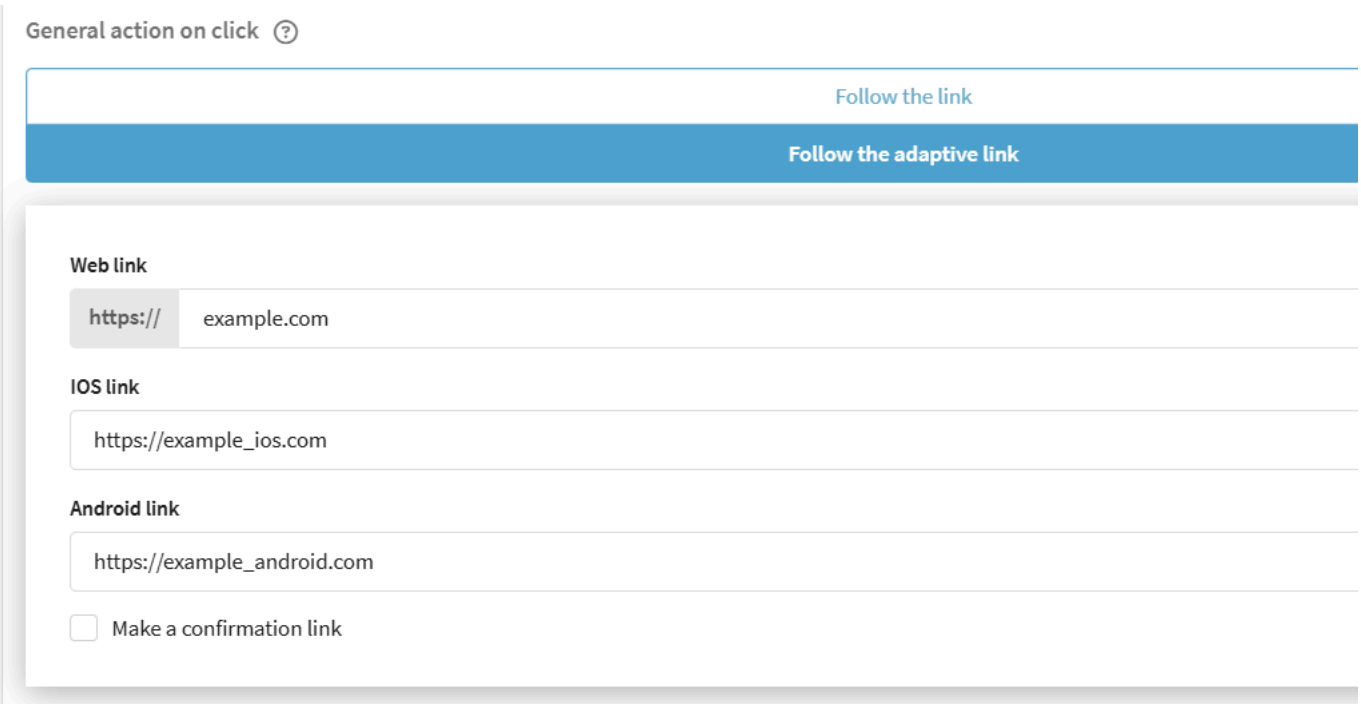
- **Specify the URL for following. You can only use https link, that means that a link must lead to a site t connection.**
- **Insert variables using the button `</>`.**
- **Add UTM tags to track subscriber traffic. Click utm and in the window that opens, configure the labe tags.**
- **Activate the field if necessary Do not turn around by tracking — this way you will disable the redirec link, and click statistics will not be collected.**
- **If you activate the field Make a confirmation link, then a template with such a link can be used to co on the link, the profile status will be updated: `Unconfirmed` —> `Subscribed`.**



Adaptive link

Adaptive link is a link to redirect the user to different URLs, depending on which device he opens the pus or iOS).

- **Web link — the user will click on this link if he opens a push notification from a desktop device**
- **IOS link — The user will click on this link if they open a push notification from an iOS device.**
- **Android link — The user will follow this link if he opens a push notification from an Android device.**



Additional settings:

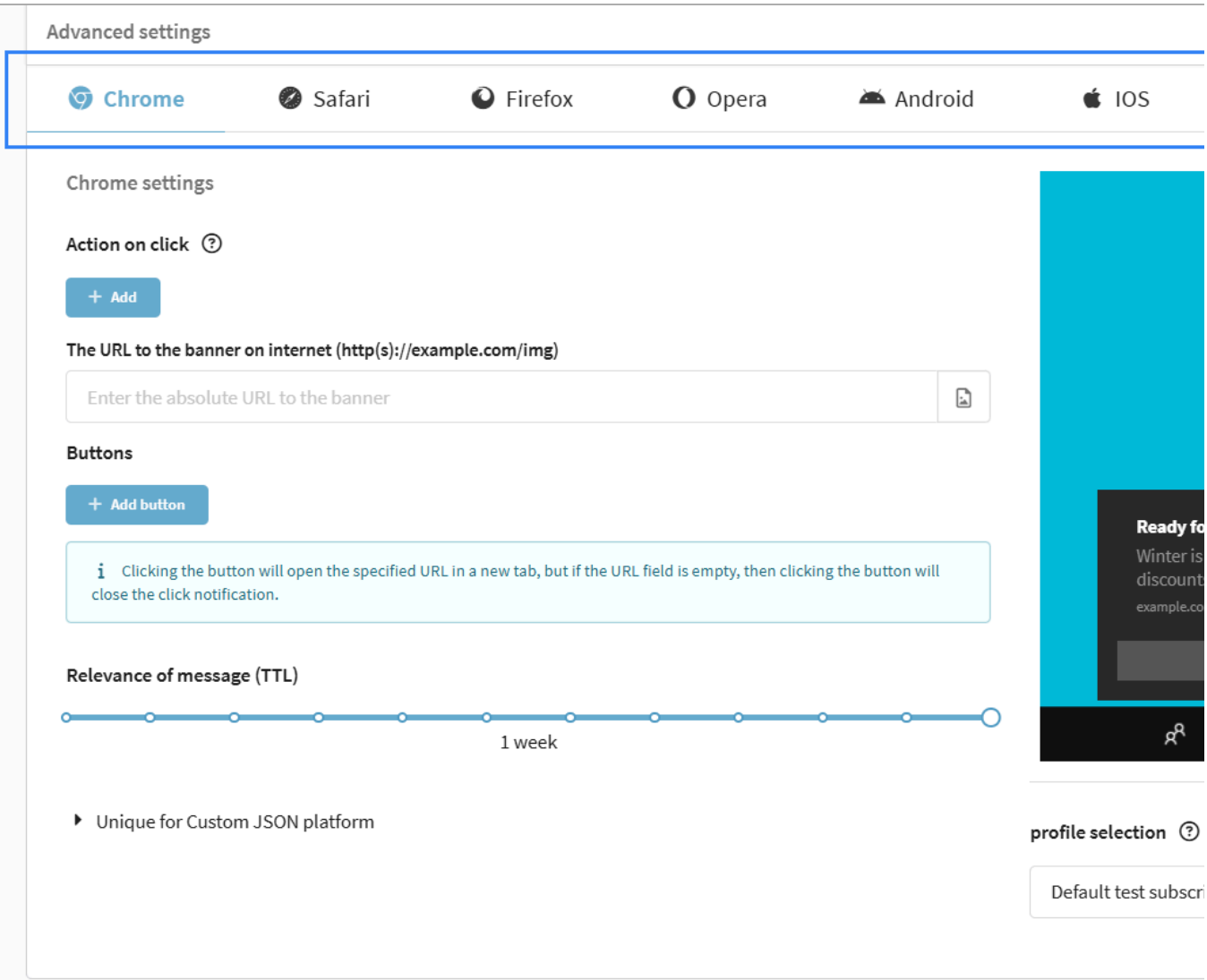
- **Make a confirmation link — a template with such a link is used to confirm the subscription. When cli status of the profile will change from `Unconfirmed` —> `Subscribed`.**
- **Add variables to the link using the button `</>`.**
- **Add UTM tags. To do this, click the "utm" button and in the window that opens, configure the labels,**

Advanced settings

The extended version of the push notification editor allows you to create push messages for different OS between OS and browsers using the panel and configure a push message for each of them.

In addition, a preview of push messages is available to you on the left.

You can also specify a test subscriber to see exactly how the created push message will be displayed or



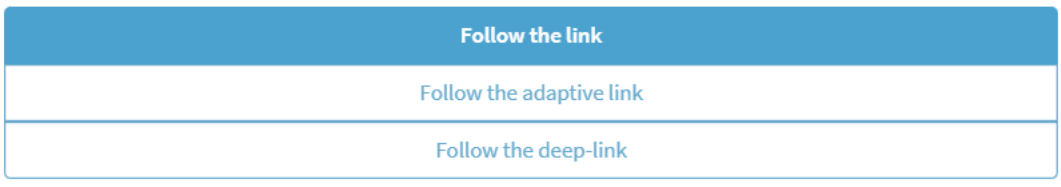
Individual push settings are available for 6 browsers and 2 operating systems. Most of the settings are tl

Action on click

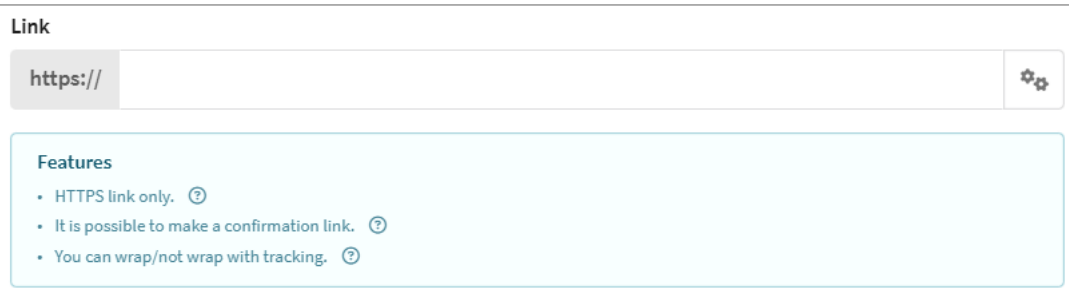
For each browser and OS, you can add [click-through link]. We remind you that it is a link for a user to follo
How do I set up a click-through link?
1. Click + Add:

2. Select the type of the click-through link:

- **Regular link;**
- **Adaptive link;**
- **Deep-link link.**



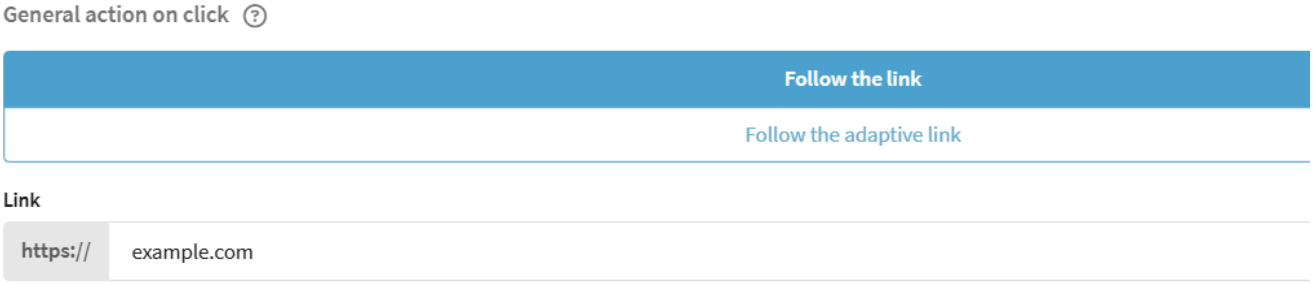
3. Enter the URL of the link and configure it:



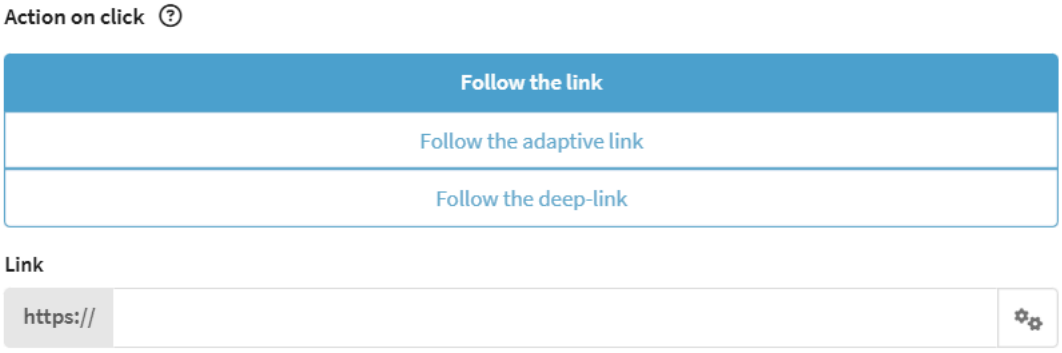
CAUTION

Please note that the link that you specify for each platform “overrides” general action on click.

General action on click



Action on click

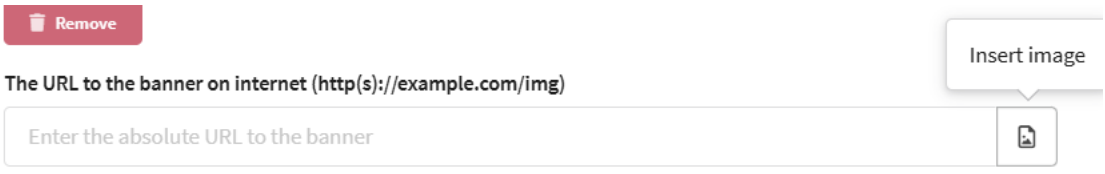


If you leave “action on click” to the platform empty, then even in this case it will “overwrite” general ac

When you leave the field with the click link empty, then when you click on it, the user will simply close t

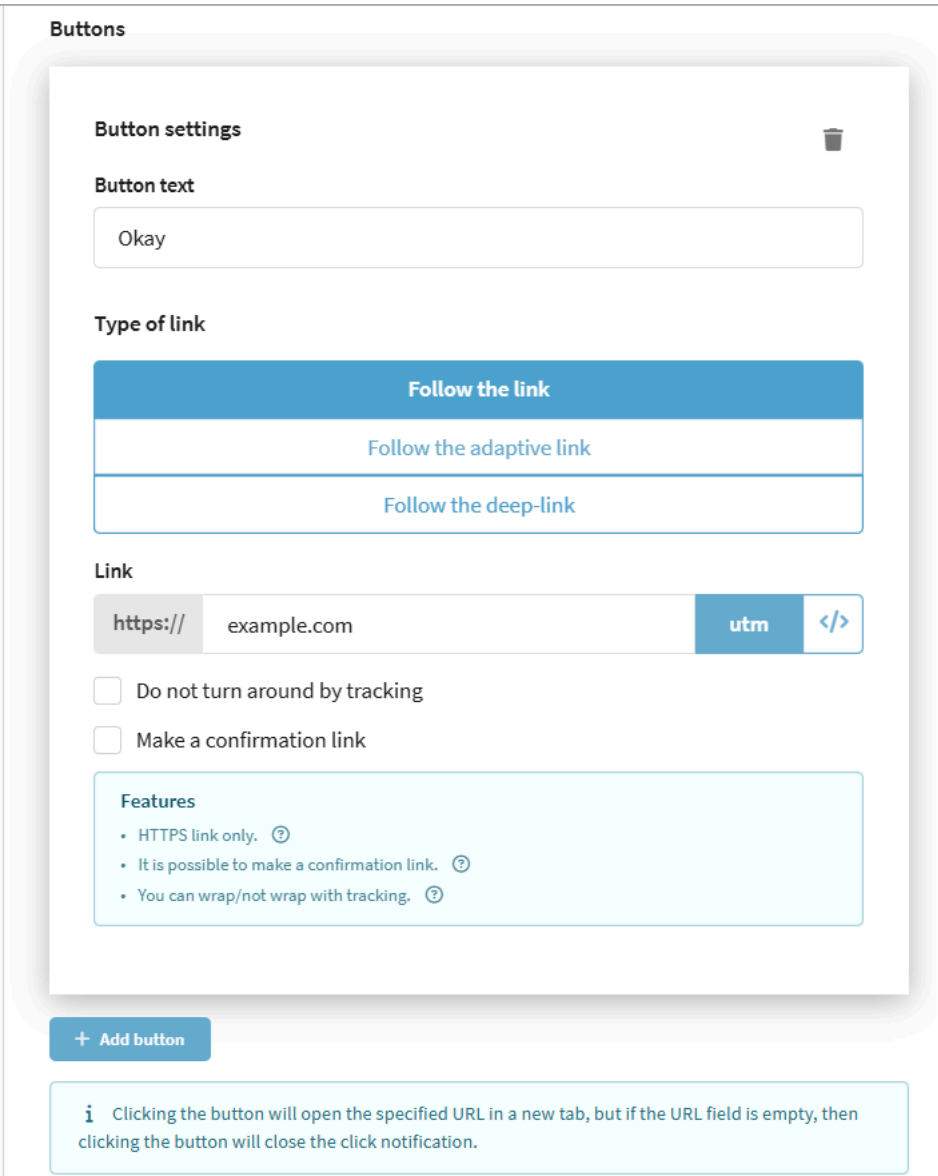
Banner

Specify the URL of the image. If the URL is correct, the banner image will be displayed in the preview. In the gallery.



Buttons

Click Add button. In the window that opens, enter the button text of the button and select the action tha on the button. Read about setting up a click-through link right there.

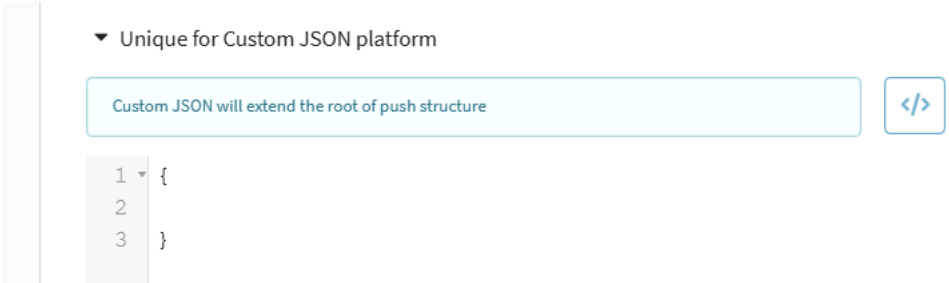


CAUTION

If the field Link is left empty, then when clicking on the button, the user will close the push.

Unique for Custom JSON platform

You can pass additional fields in JSON format to the message to expand the root structure of the push m



Unlike the General settings Custom JSON option, which is applied to all browsers and OSs, a unique Cus browser and OS separately.

Relevance of message (TTL)

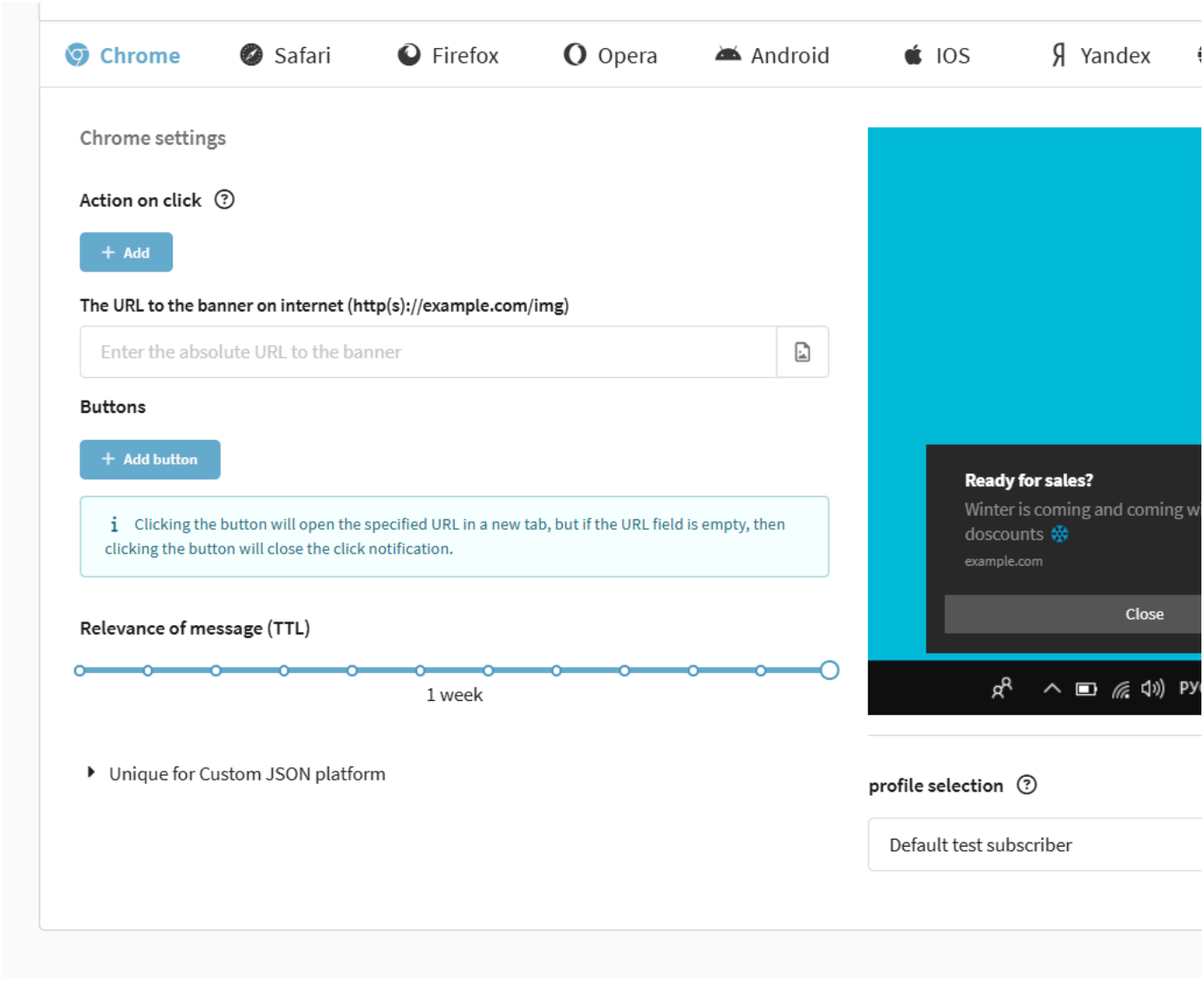
TTL (“time to live”) is the period of time during which the push service monitors the status of a recipient Internet. As soon as a recipient appears online within the specified time, a push message will be deliver a recipient has not appeared on the network, the system stops tracking its status, and, accordingly, the delivered.

Move the slider to set the “time to live” of the push message:

Chrome

The following settings are available for this browser:

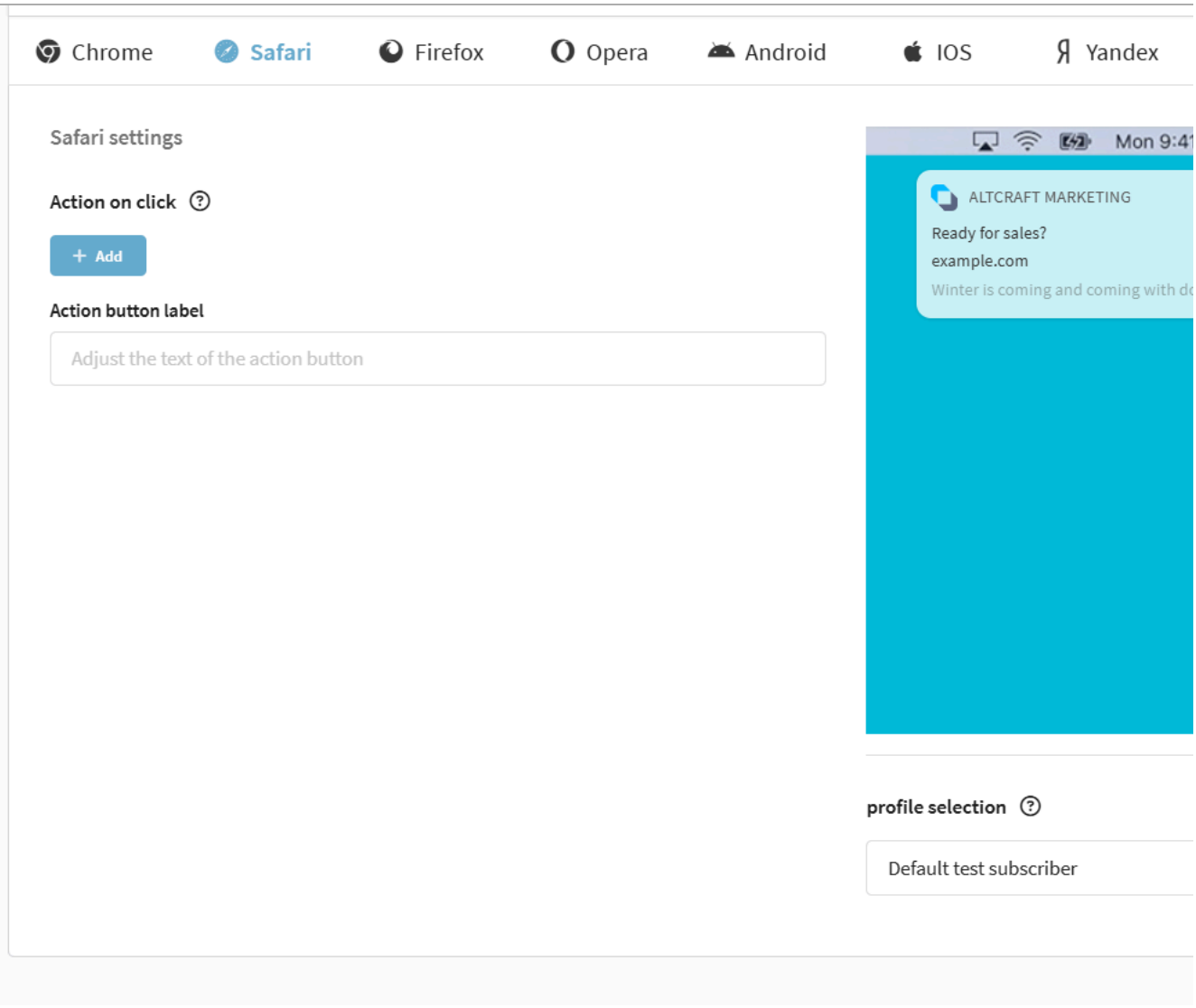
Field	Value
Action on click	**The link that a user will follow when they click on a push message**
Banner	**Link to a banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
TTL	**The time during which the service monitors a device’s connection for delivery of a message. Then an expiration happen**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Safari

The following settings are available for this browser:

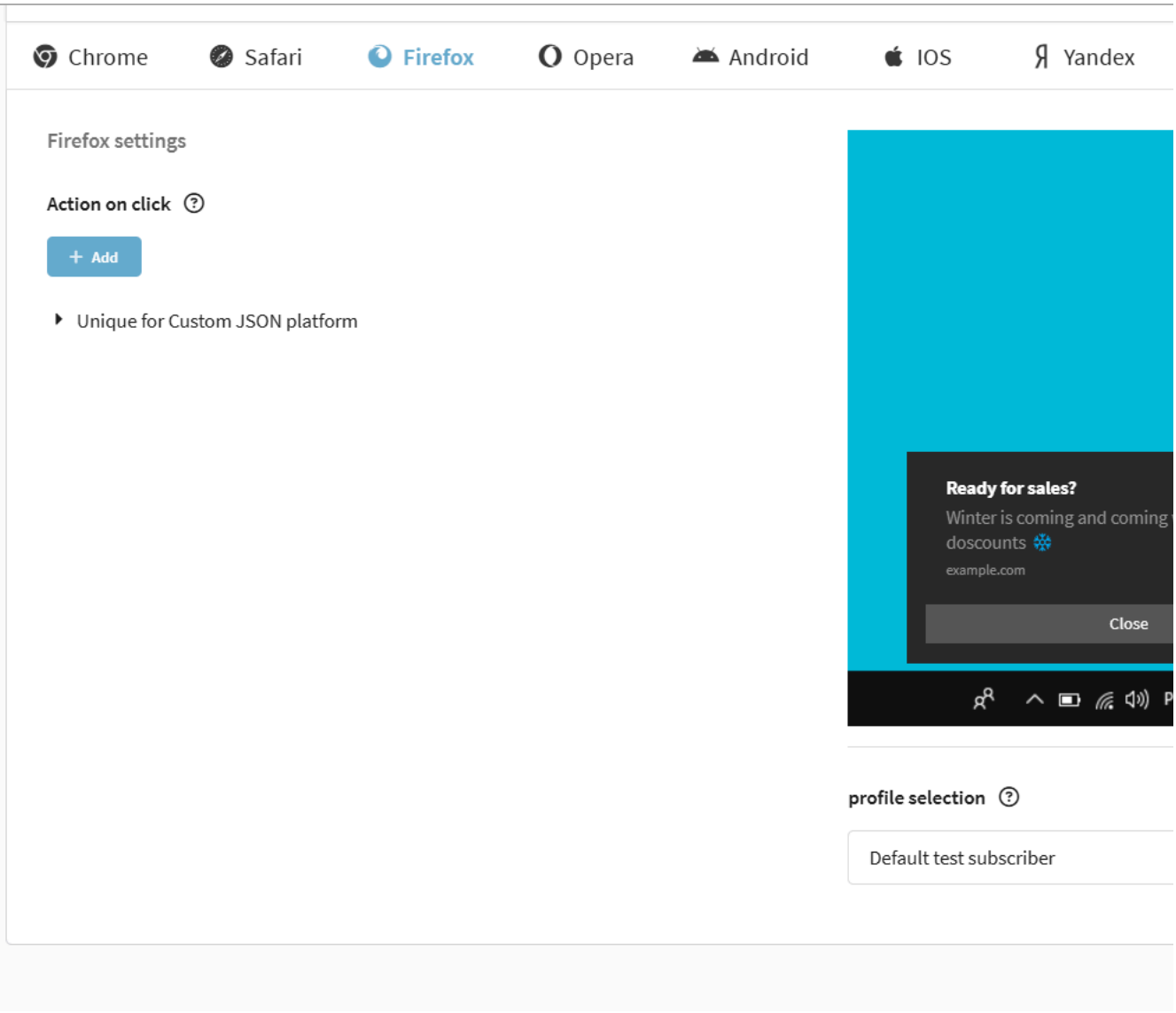
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Action button label	**When the button is clicked, a user will perform a certain action**



Firefox

The following settings are available for this browser:

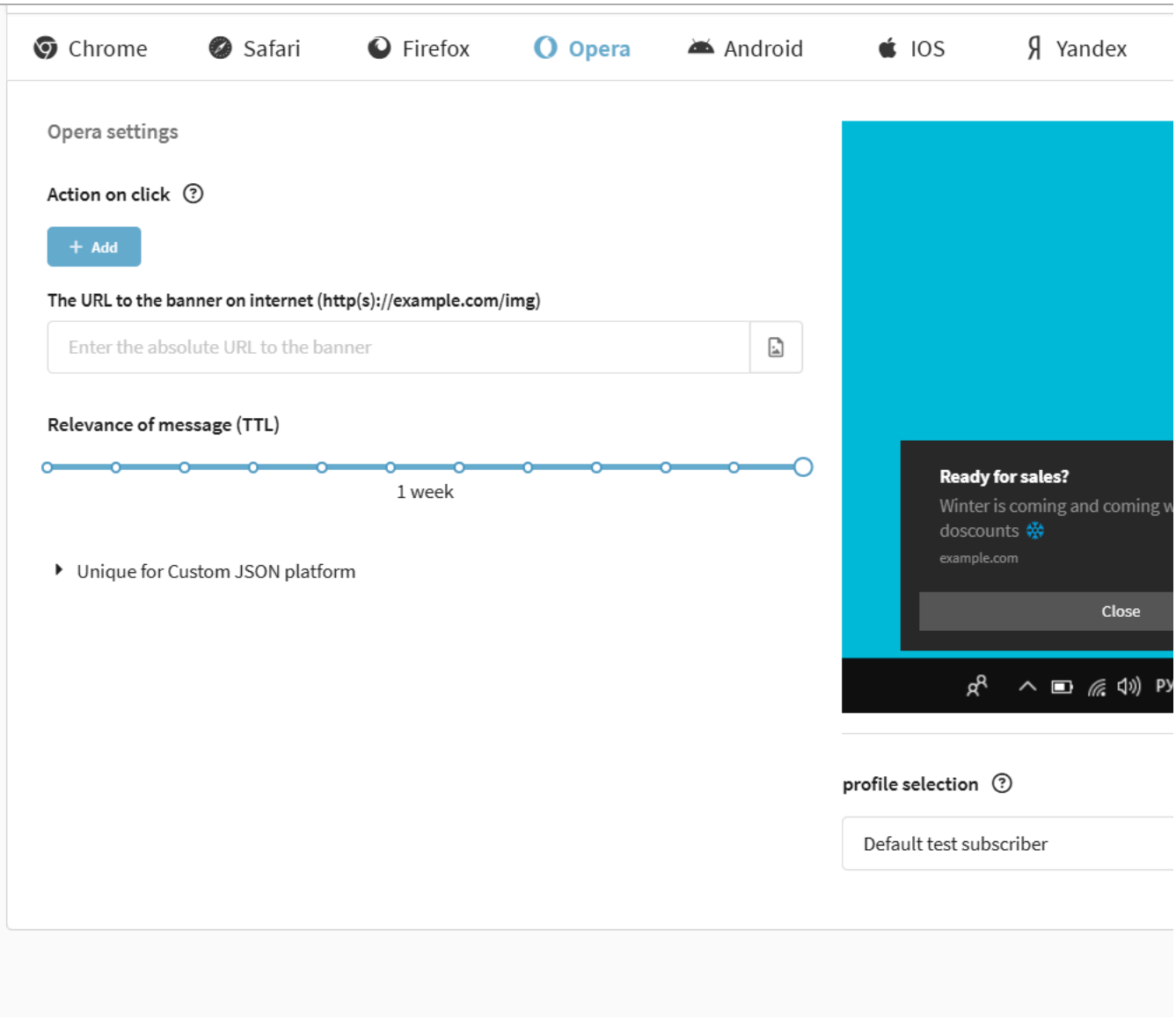
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Opera

The following settings are available for this browser:

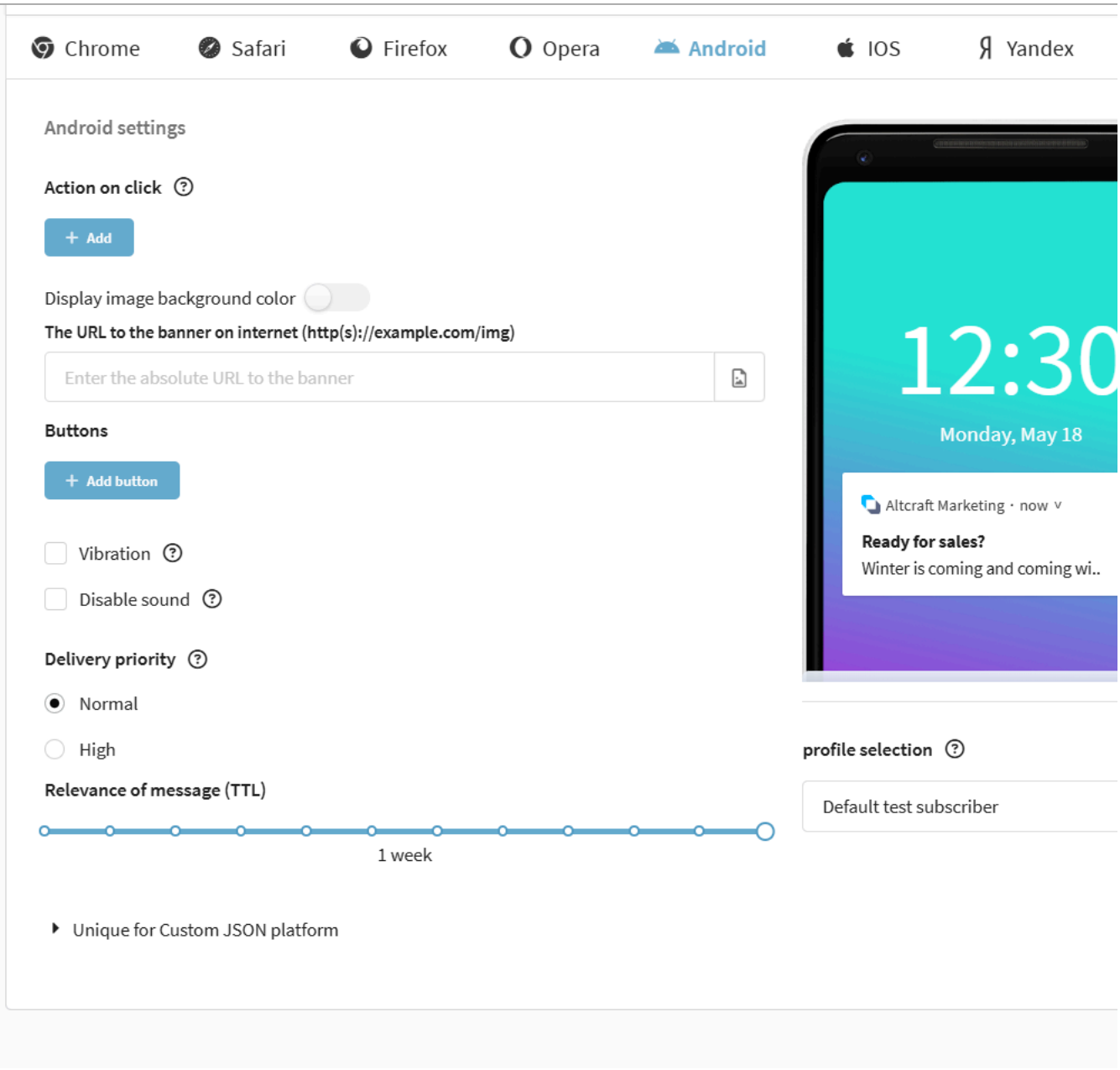
Field	Value
Action on click	**A link that the users will follow when they click on a push message**
Banner	**Link to the banner image**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happen**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Android

The following settings are available for this OS:

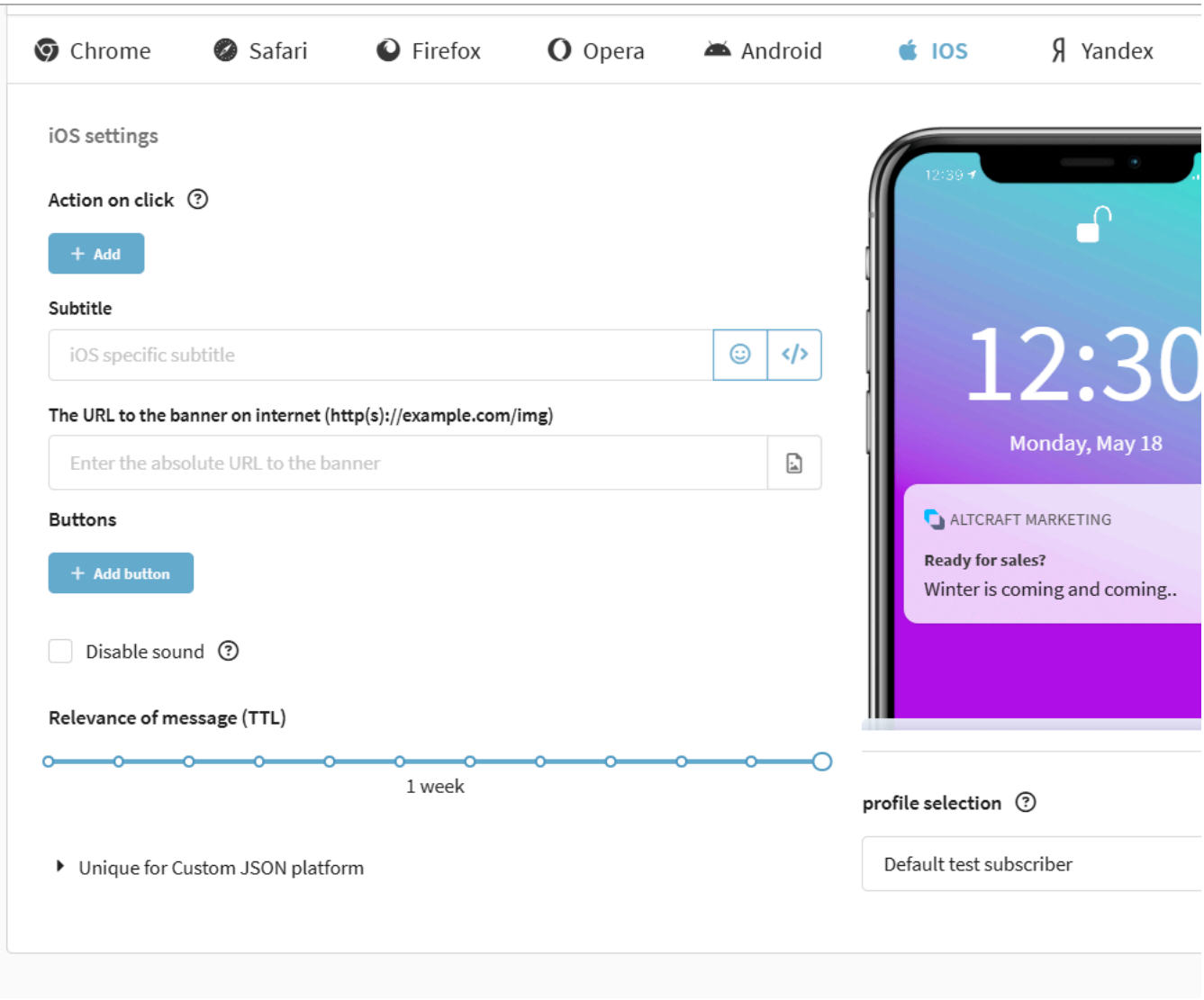
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Display image background color	**Background of the push notification**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
Vibration	**When receiving a push notification, user's device will reproduce vibration**
Disable sound	**When delivered to a device, a push notification will not play sound or vibration**
Delivery priority	**Normal: the delivery of messages with normal priority can be postponed indefinitely to save battery power.** **High: push notification launches the app in the background and immediately gets internet access.**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happen**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



iOS

The following settings are available for this OS:

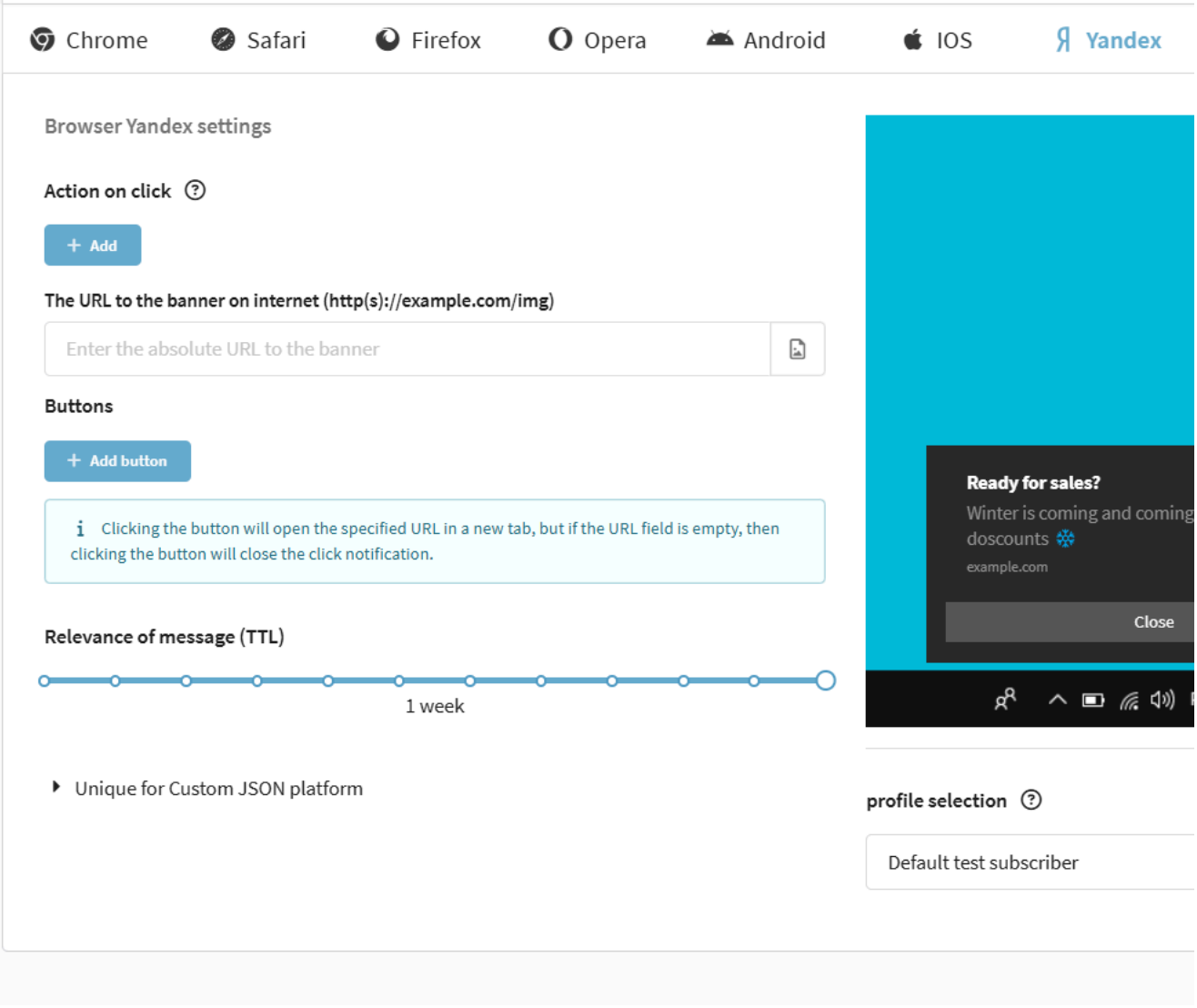
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Subtitle	**IOS-specific subtitle**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
Disable sound	**When delivered to the device, the push notification will not play sound and vibration**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happen**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Yandex

The following settings are available for this browser:

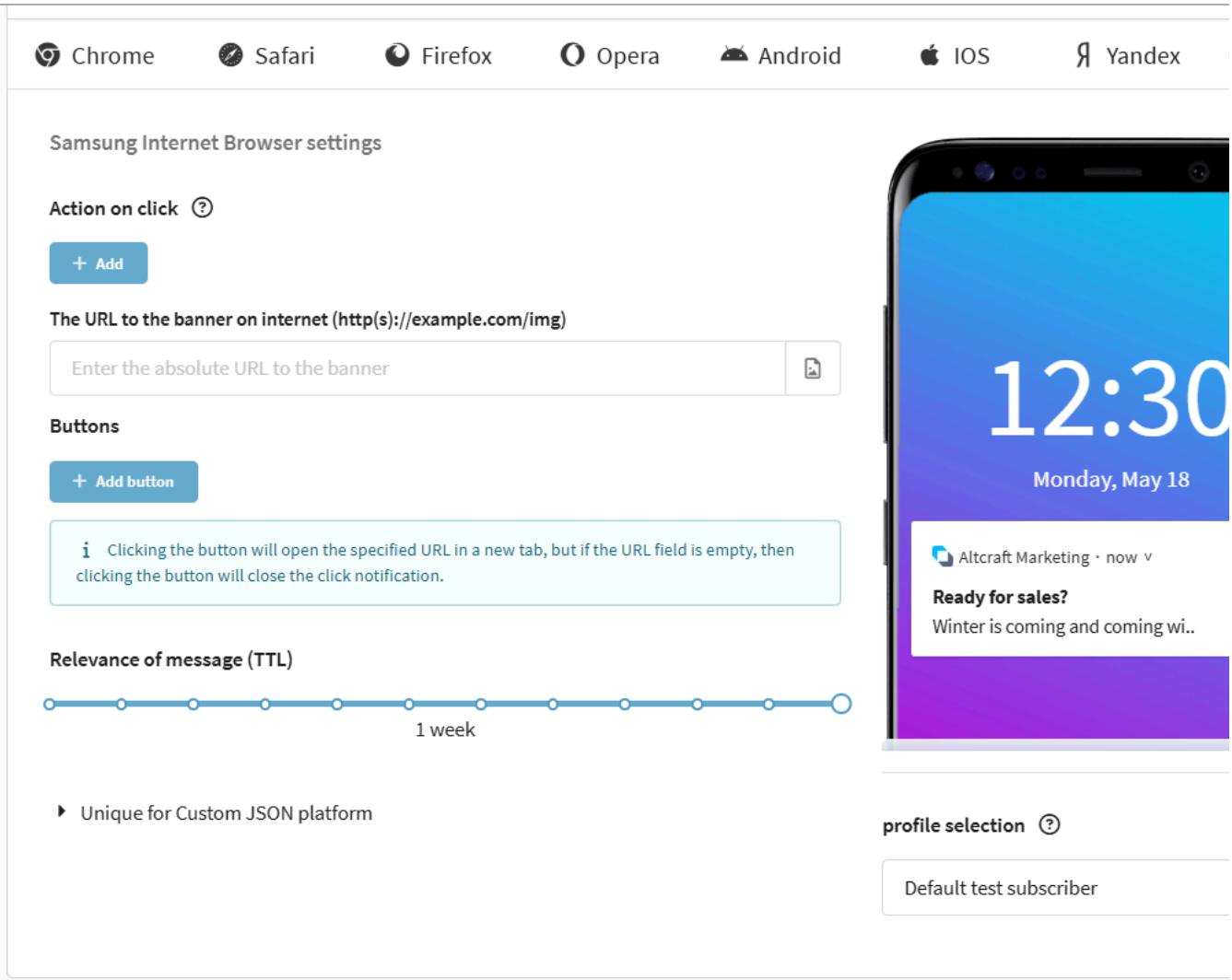
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, the user will perform a certain action**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happen**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Samsung

The following settings are available for this browser:

Field	Value
Action on click	**A link that users will follow when they click on a push message**
Banner	**Link to a banner image**
Buttons (no more than two at once)	**When the button is clicked, the user will perform a certain action**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happe**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Once you have set up all the necessary parameters for your template, save it by clicking on the “Save” b

If you made some mistake when filling template fields, it will be displayed at the top of the screen. Corre again.

Now you can use this template to send Rich Push notifications. Read more about how to do this here: We

MAX template

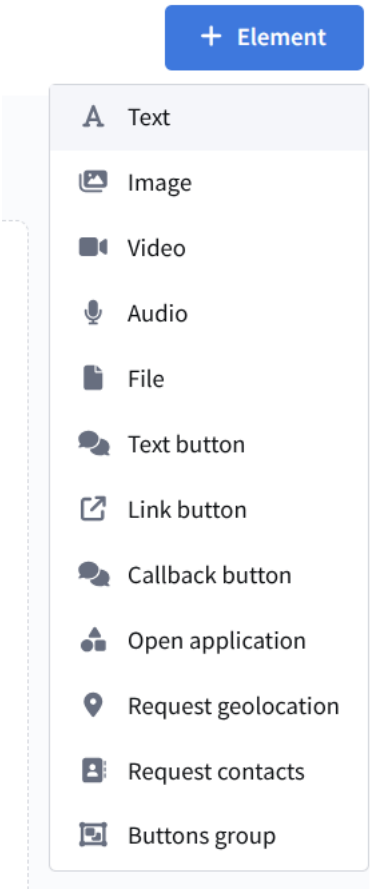
TIP: MAX Bot and MAX Group

Templates for sending via MAX Bot and MAX Group are configured in the same way, except that profile
MAX Group because it is an indirect communication channel.

This article describes how to create templates for sending messages via MAX Group and MAX Bot.

Message elements

A MAX template consists of elements. To start editing a template, click + Element:



The following elements are available:

- **Text — sends a text message**
- **Image — attaches an image to the message**
- **Video — attaches a video file to the message**
- **Audio — attaches an audio file to the message**
- **File — attaches a file to the message**

To interact with the recipient, you can add inline buttons to the message:

- **Text button — displays a button with text**
- **Link button — opens the specified link**
- **Callback button — after being clicked, sends the specified text or command back to the bot**
- **Open app — opens a mini app**
- **Request location — requests the user's location**
- **Request contact details — requests the user's contact and phone number**
- **Button group — a combination of buttons of several types**

These buttons cannot be used without other elements.

The following limits apply to MAX templates:

- **Up to 32 elements in one template — up to 30 buttons, text, and one media file**
- **You can attach only one media type to a message (images, video, audio, or file)**
- **Up to 12 images in one template, while all other media support only one attachment**
- **A single button group can contain up to 7 buttons if only text and callback buttons are used, or up to another type is used (link, open app, request location, or request contact details)**

TIP: Link previews

If the message text contains links to web pages, MAX displays a preview of that page by default:

If you do not want the preview to be displayed, enable the Disable link previews in MAX switch.

Text

A text message that will be sent by the bot. For easier formatting, the top panel includes the following fu

B

— bold text

I

— italic

~~S~~

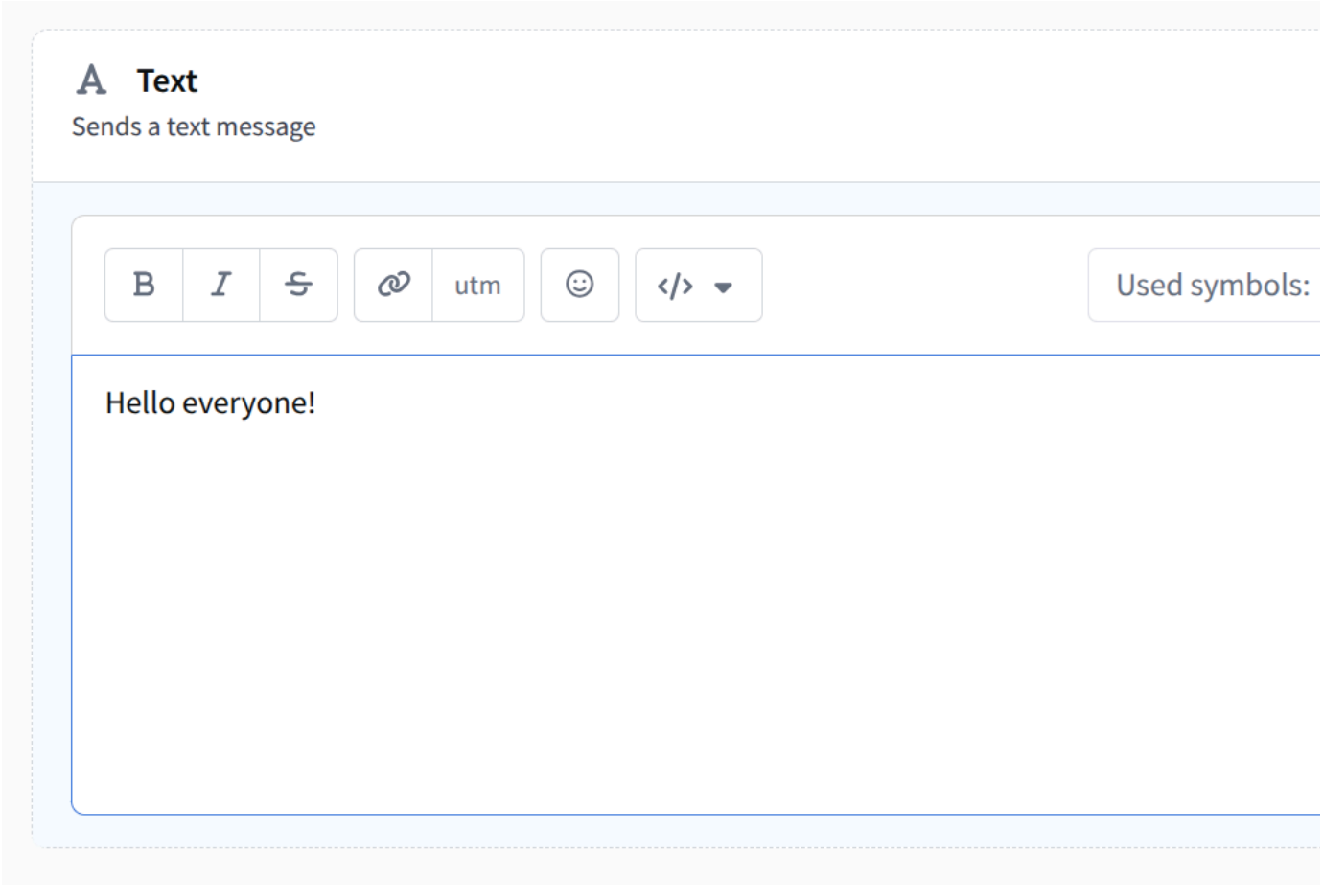
— strikethrough text

— link editor (see more here). Links inserted through the platform are shortened automatically. The final tracking domain is 12–14 characters, including `/`.

— add UTM tags

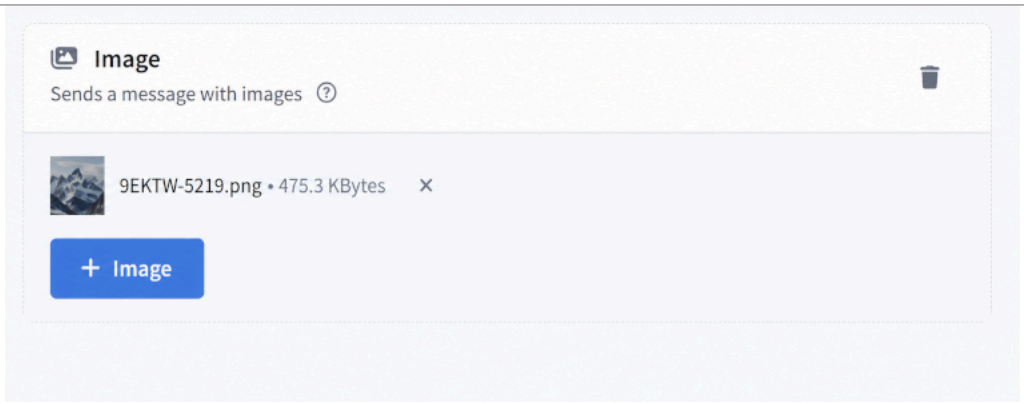
— list of editor variables and functions for message personalization

— insert emoji. Note that each emoji counts as two characters.
On the right, there is a character counter. The maximum length of a single message is 4000 characters. variables may exceed this limit. To ensure reliable delivery, we recommend monitoring the message leng necessary.

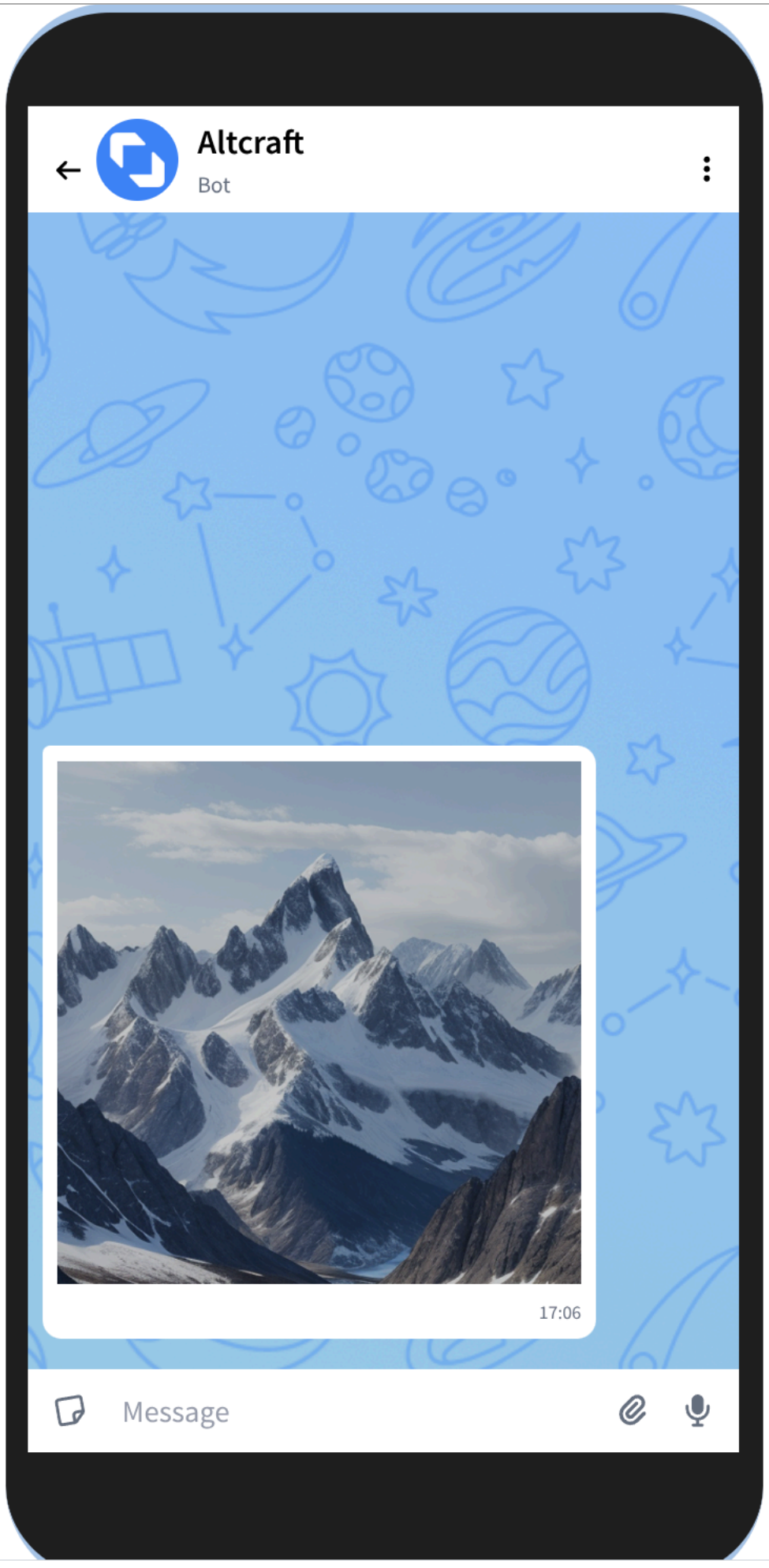


Image

Sends a message with images. You can attach up to 12 images. Files are selected from the gallery or up
The maximum image size for insertion into a template is 10Mb. Supported formats: .jpg, .png, .gif
If this element is used in the template, other media elements (video, audio, or file attachments) cannot b

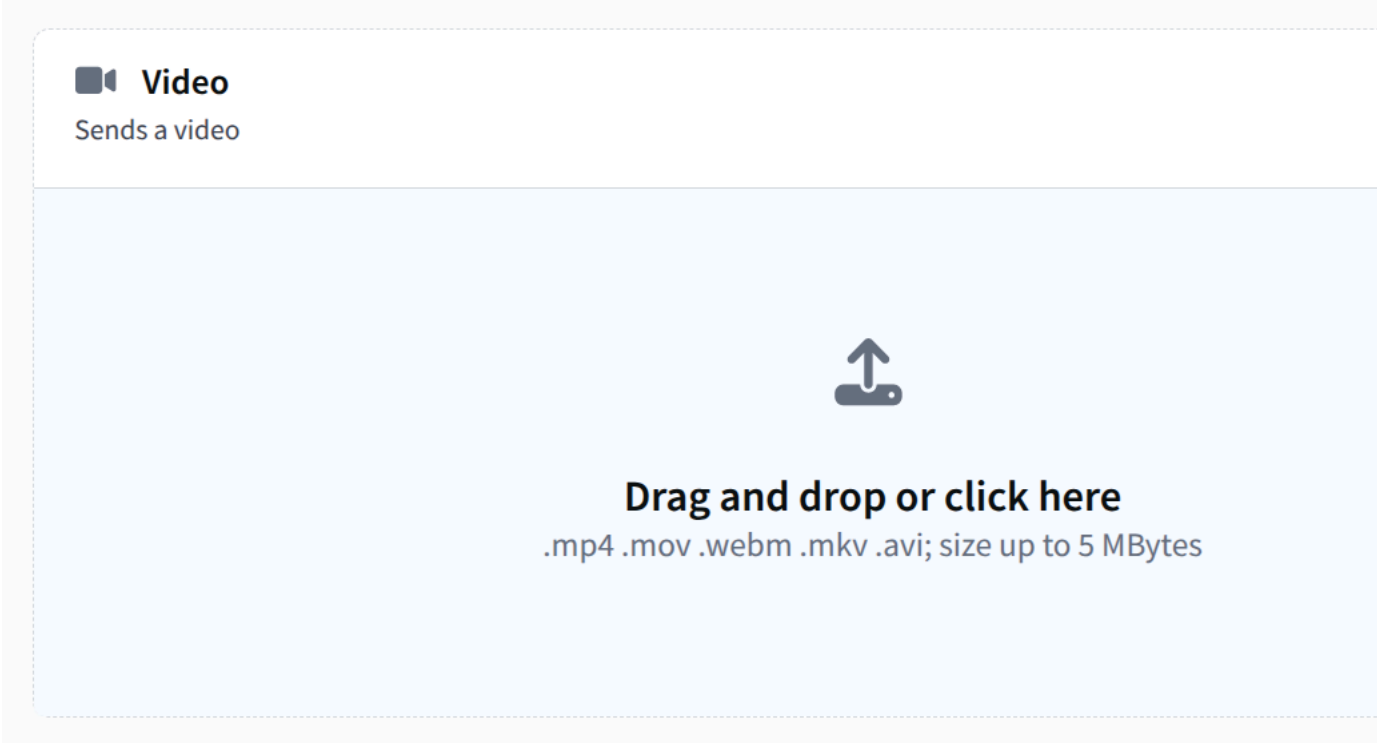


Preview in MAX

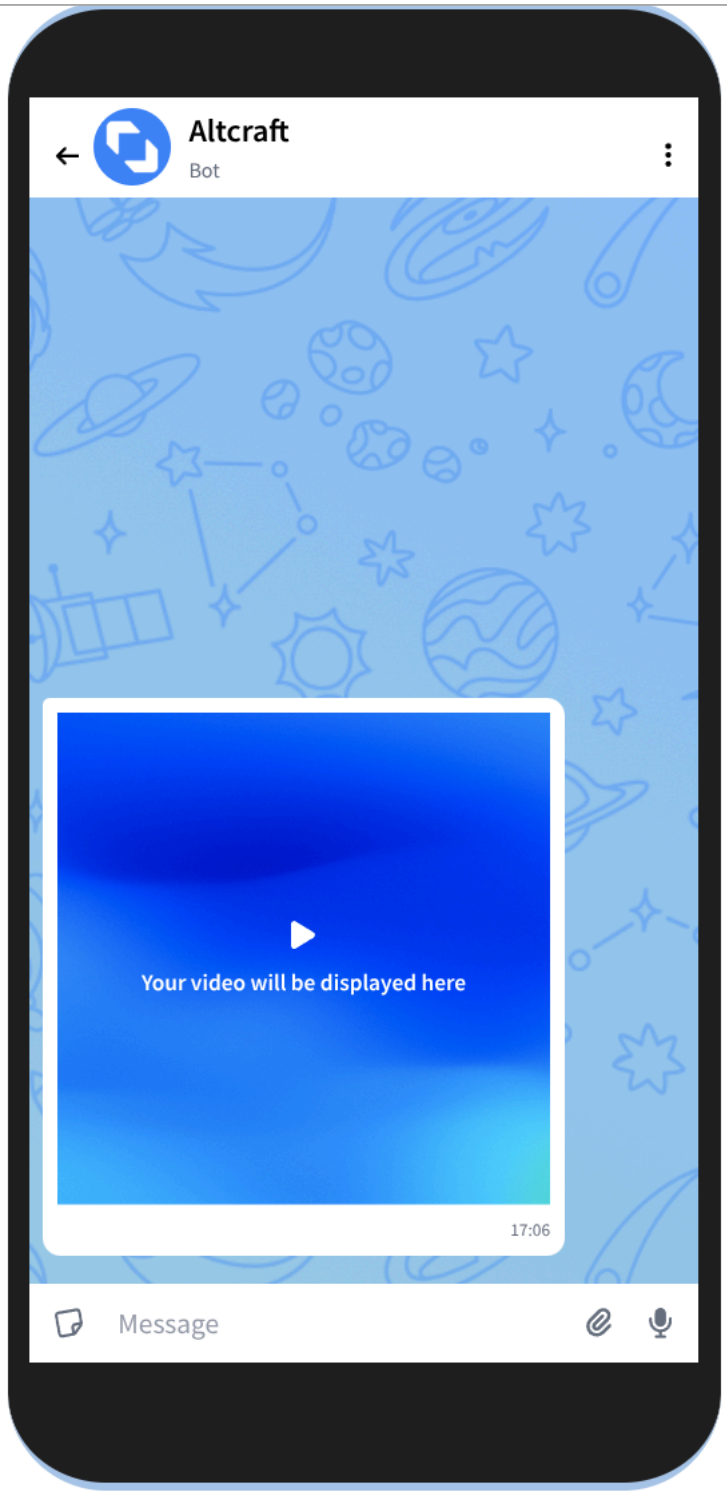


Video

Sends a video. You can attach only one video file. The maximum video size for insertion into a template i
.webm .mkv .avi
If this element is used in the template, other media elements (image, audio, or file attachments) cannot b

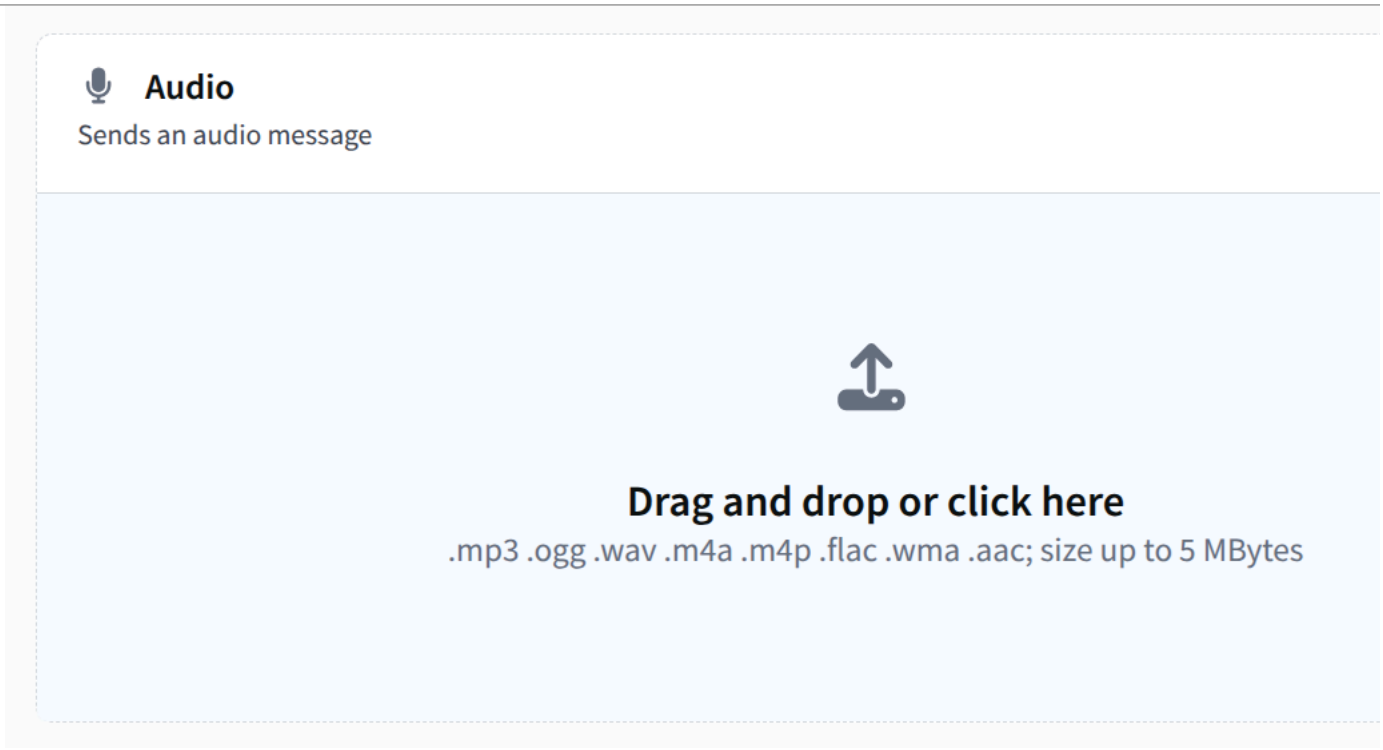


Preview in MAX



Audio

Sends an audio message. You can attach only one audio file. The maximum audio size for insertion into a
.mp3, .ogg, .wav, .m4a, .m4p, .flac, .wma, .aa
If this element is used in the template, other media elements (image, video, or file attachments) cannot l

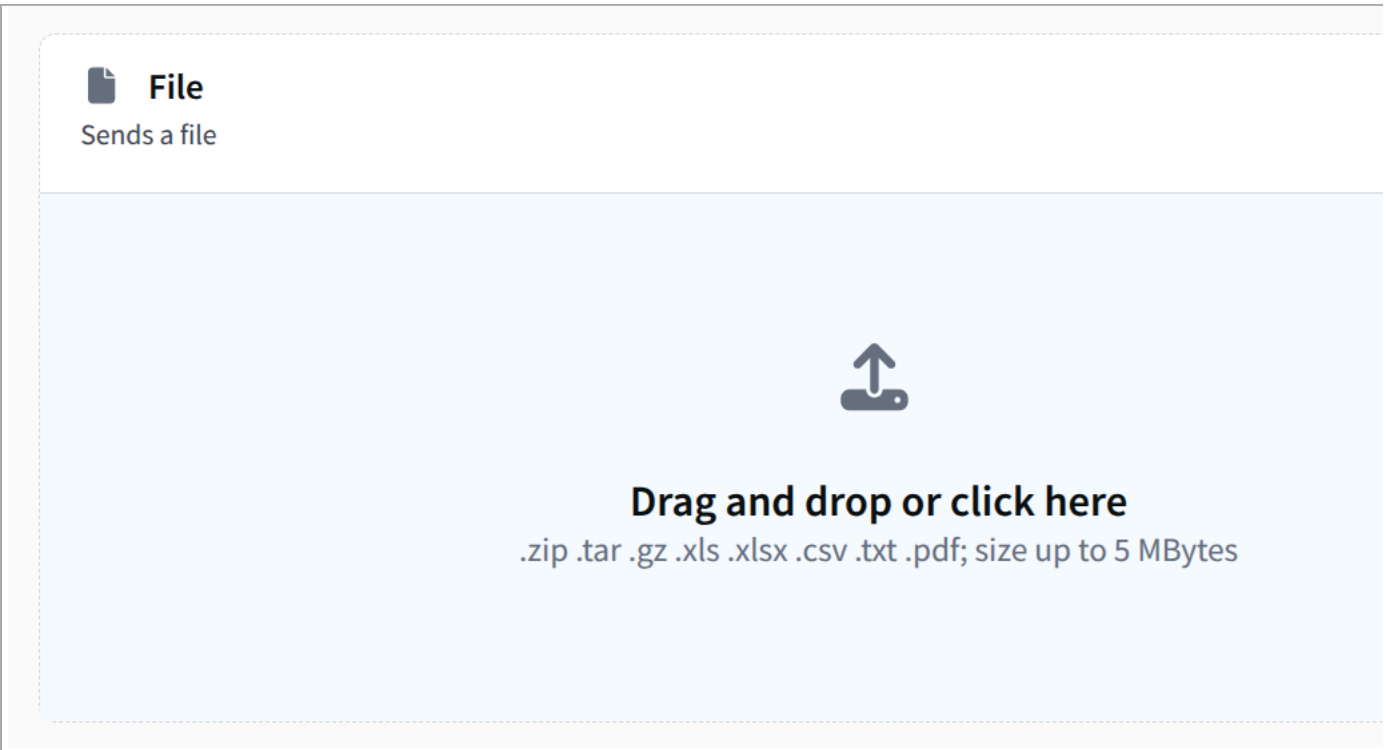


Preview in MAX

File

Sends a file up to 5Mb in .zip, .tar, .gz, .xls, .xlsx, .csv, .txt, or .pdf format

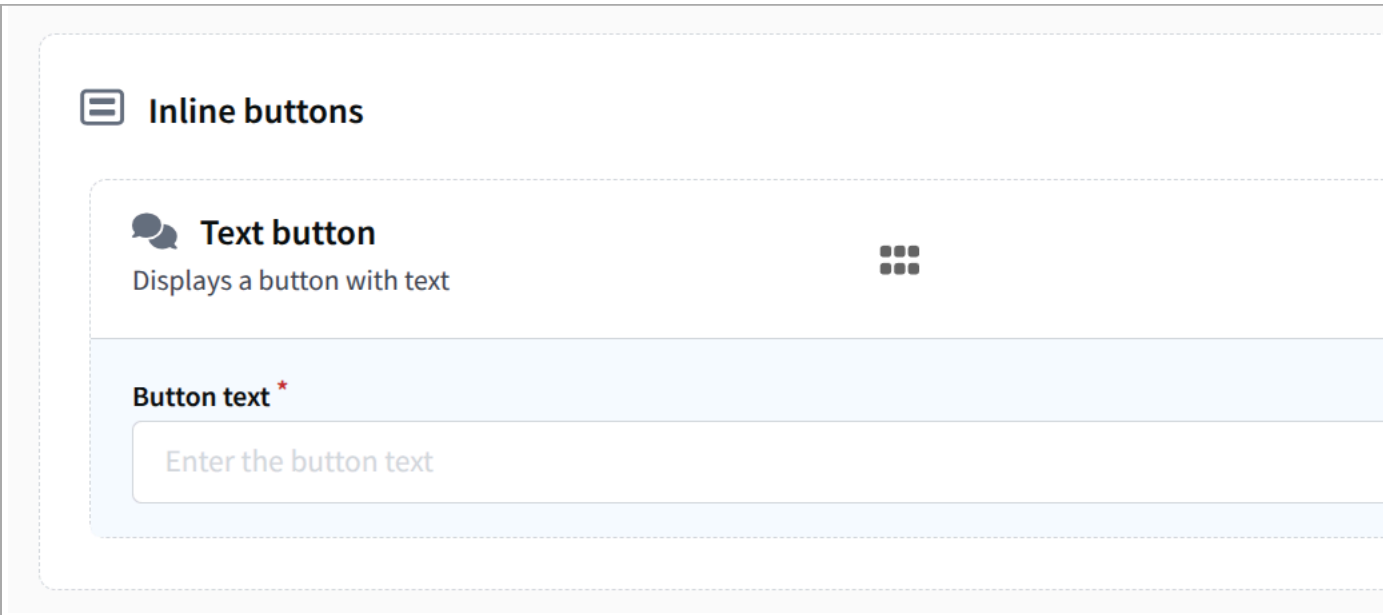
If this element is used in the template, other media elements (image, video, or audio) cannot be added.



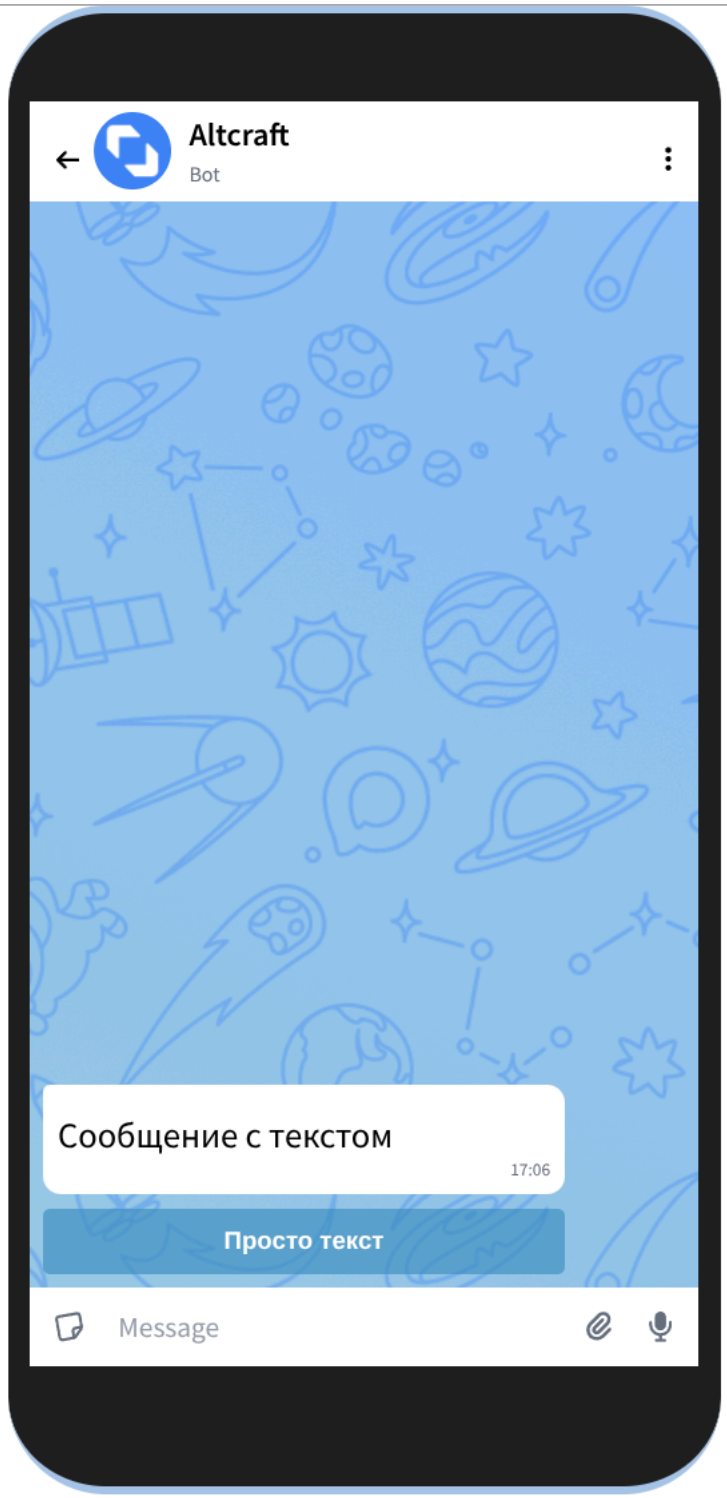
Preview in MAX

Text button

Displays a button with text. You can add emoji and platform variables to the button text. The maximum n 128.

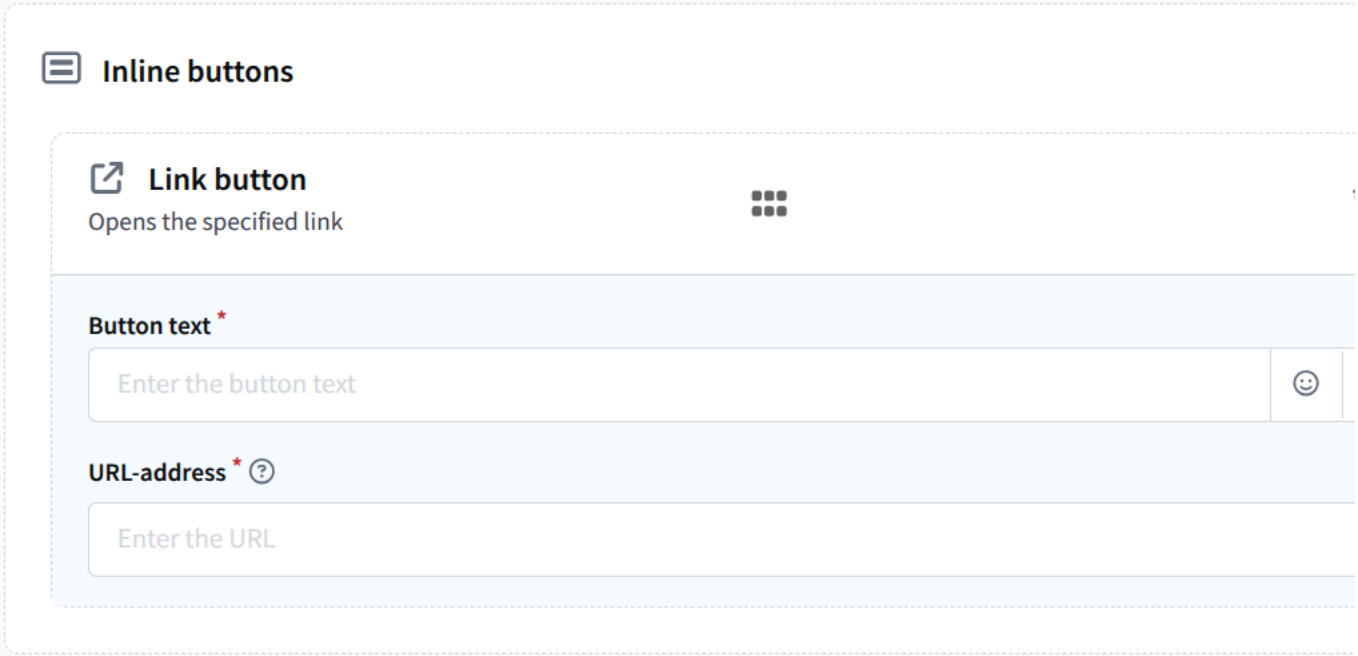


Preview in MAX

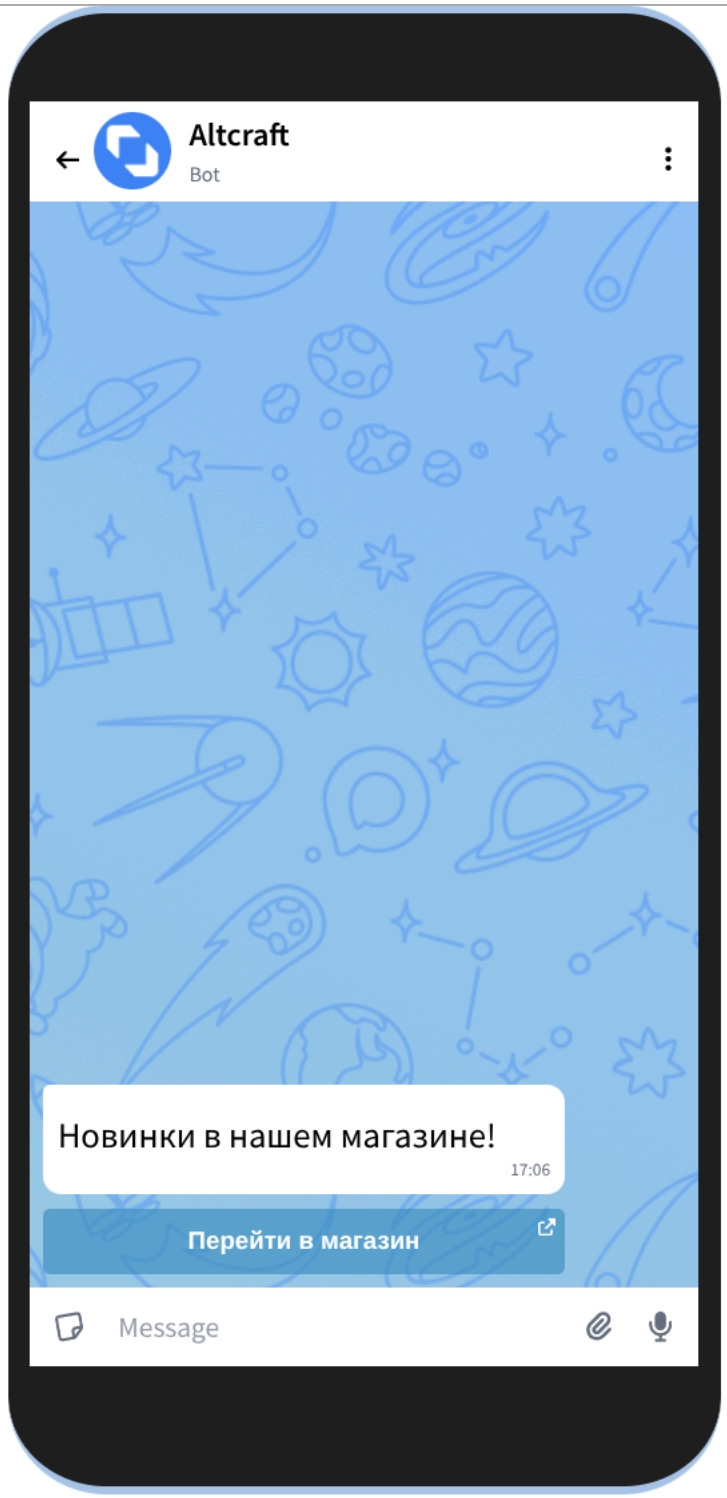


Link button

Opens the specified link. For this button, you need to specify the text displayed on it and the destination variables to the button text. The maximum number of characters in button text is 128.

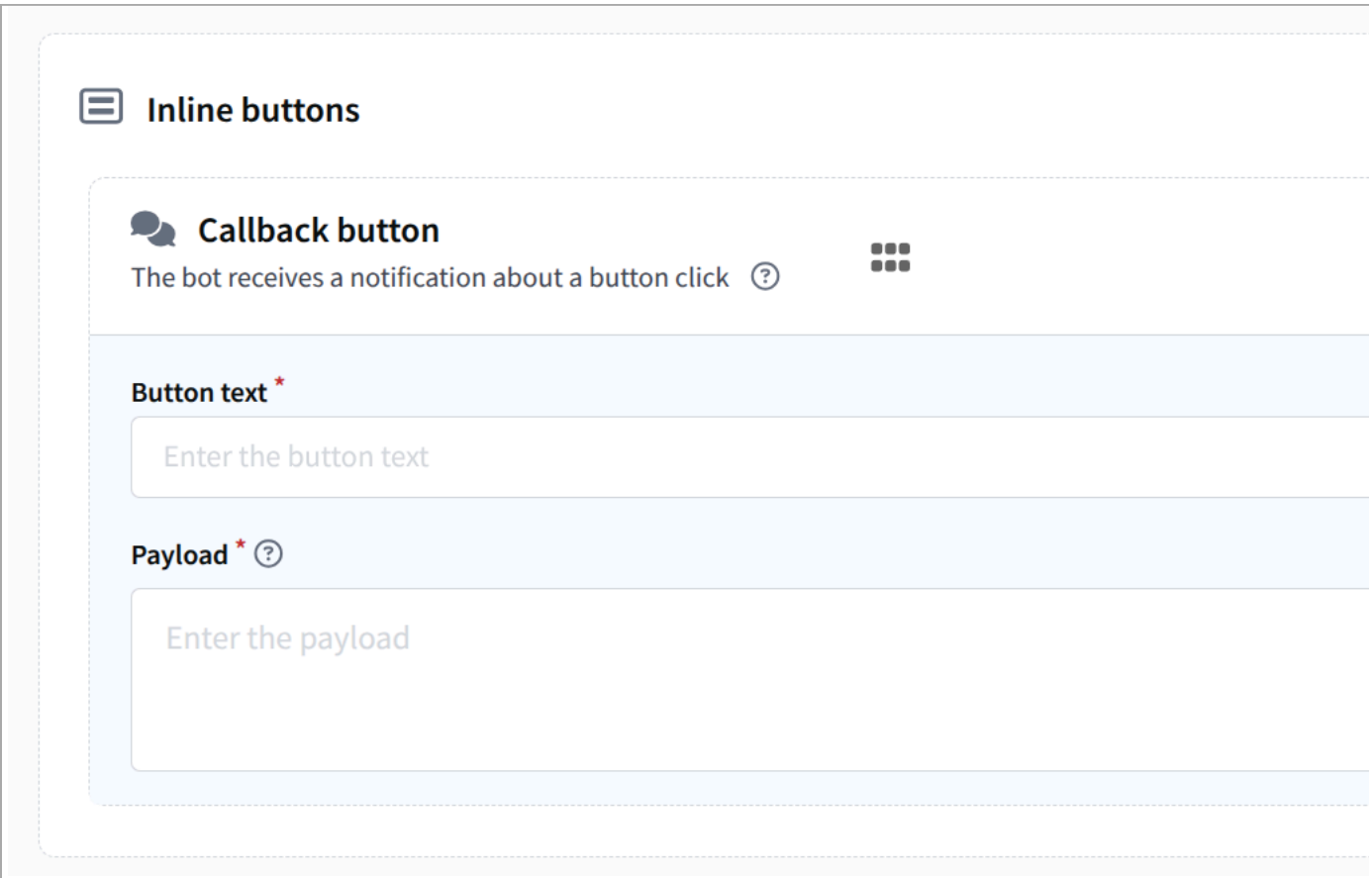


Preview in MAX

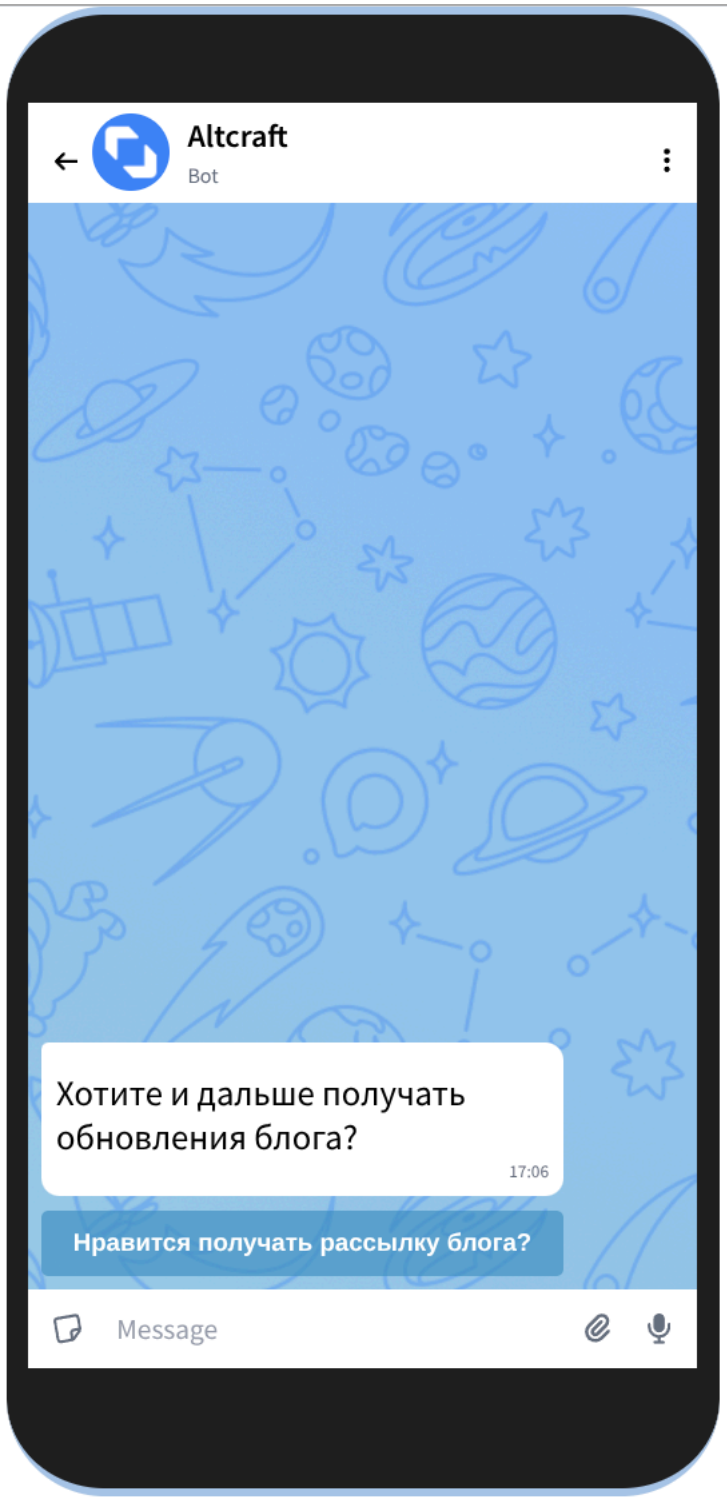


Callback button

When a callback button is clicked, the bot receives a notification and accepts the message payload. You to the button text. The maximum number of characters in button text is 128. The payload can contain pl should execute.



Preview in MAX

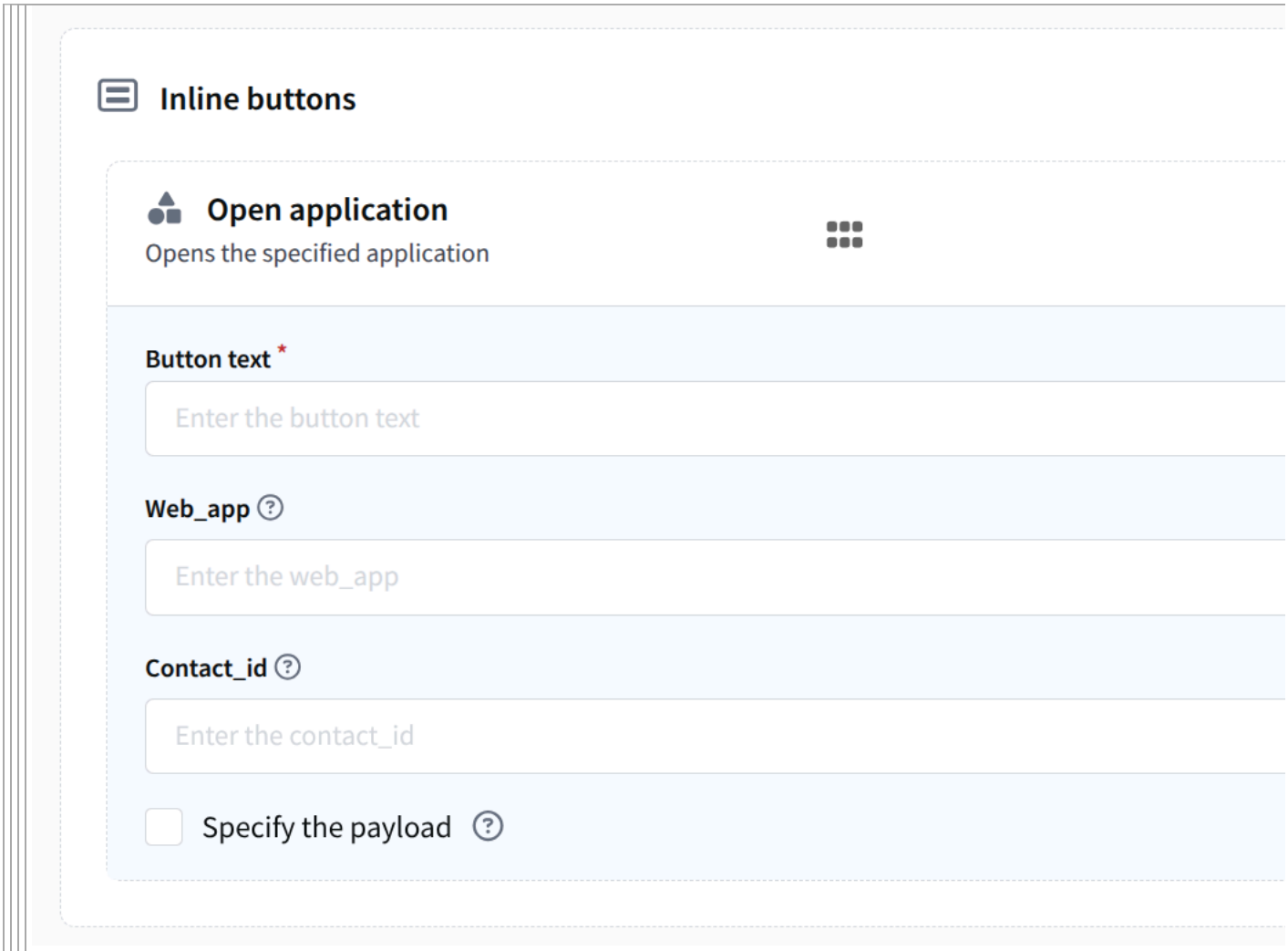


Open app

Opens the specified mini app when clicked. You need to specify the button text and the app identifier. It username of the bot whose mini app will be launched, or a link to it, passed as a string value) or as `Conta` You must pass only one parameter.

> INFO
>
> **We recommend passing `Web_app`, since using `Contact_id` may cause errors on the messenger side.**

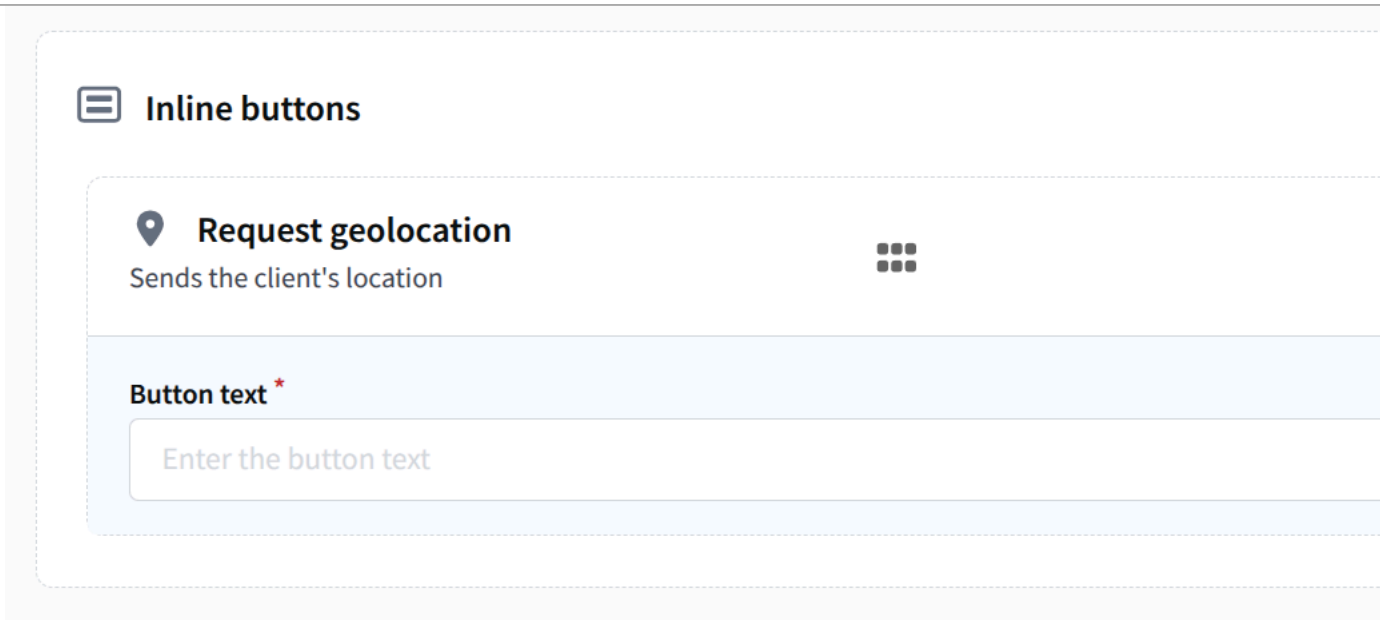
You can add emoji and platform variables to the button text. The maximum number of characters in butt contain plain text or a command that the bot should execute.



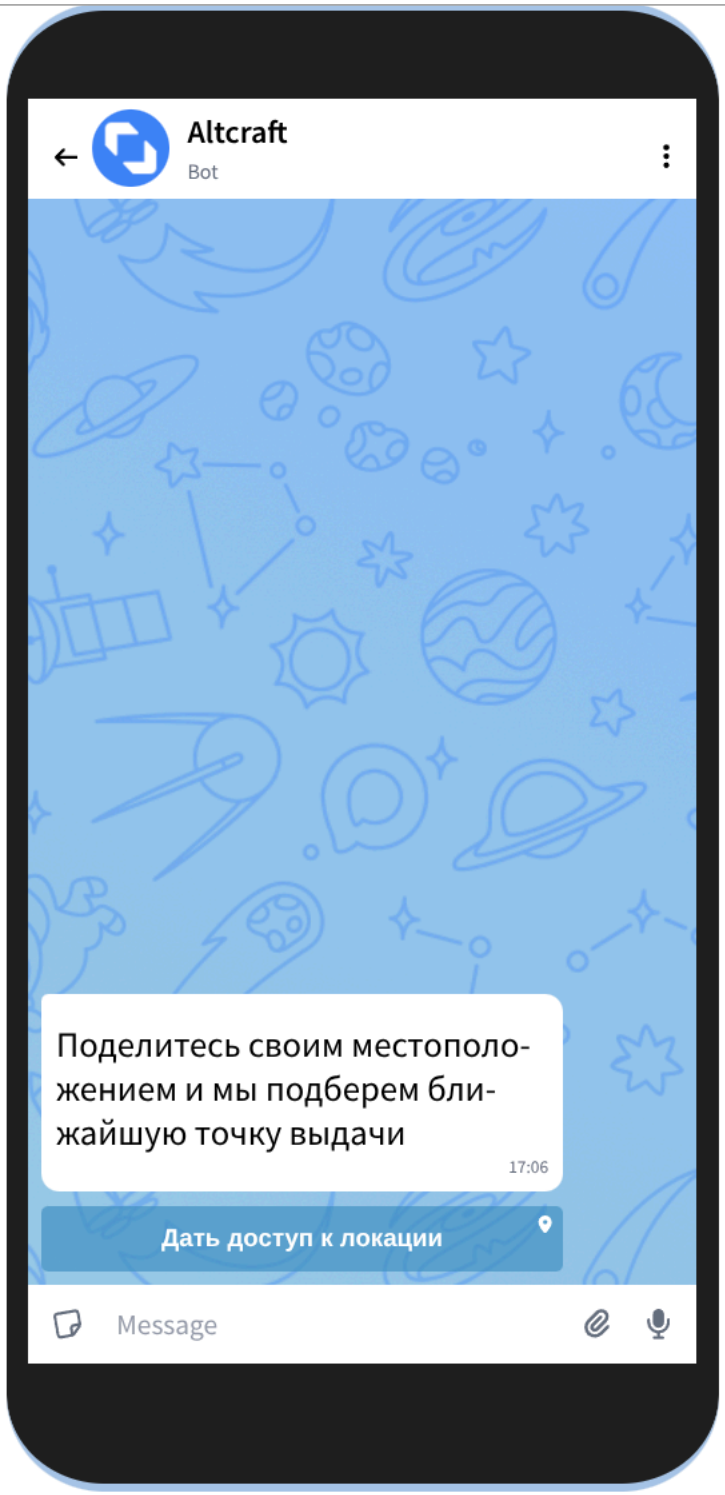
Preview in MAX

Request location

When clicked, sends the user's location. You can add emoji and platform variables to the button text. The button text is 128.

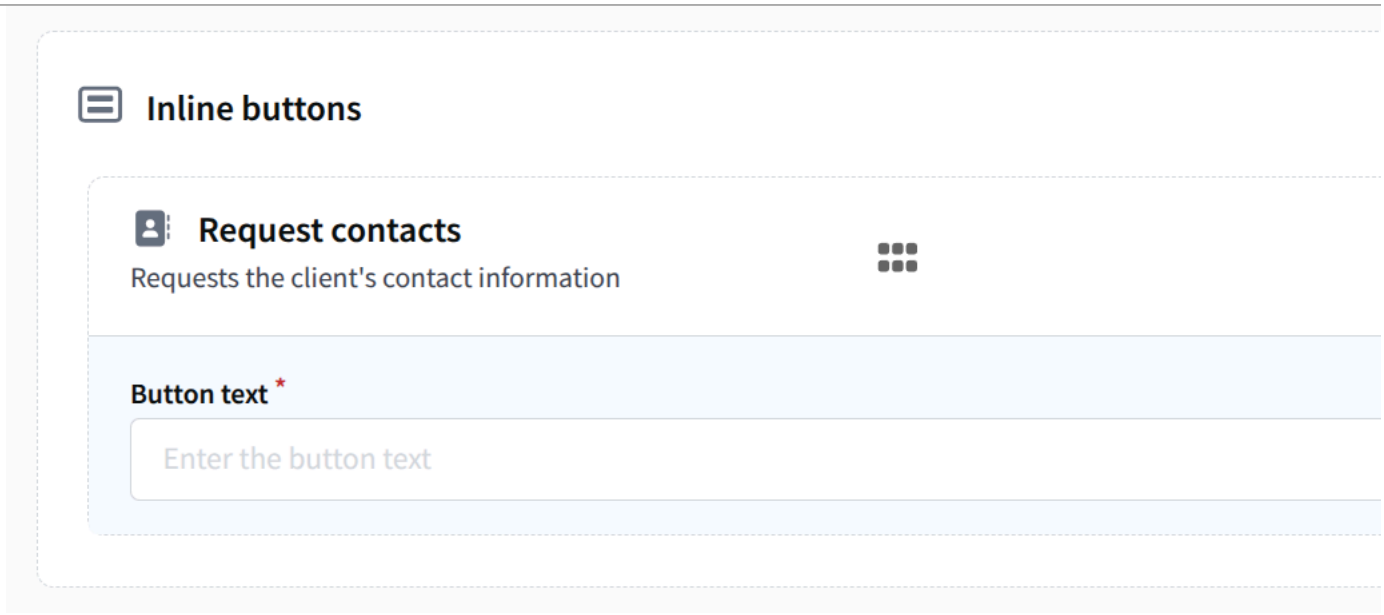


Preview in MAX

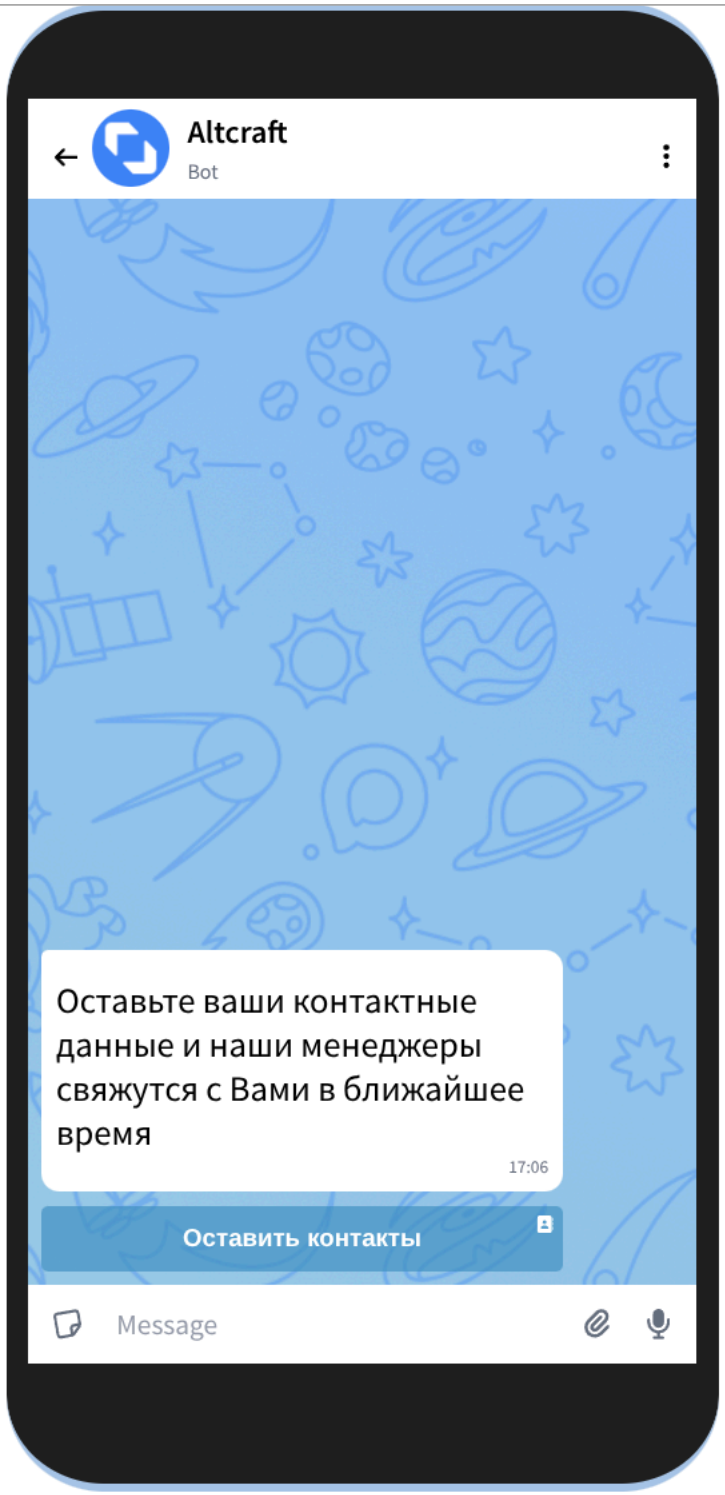


Request contact details

When clicked, sends the user's contact details. You can add emoji and platform variables to the button t characters in button text is 128.

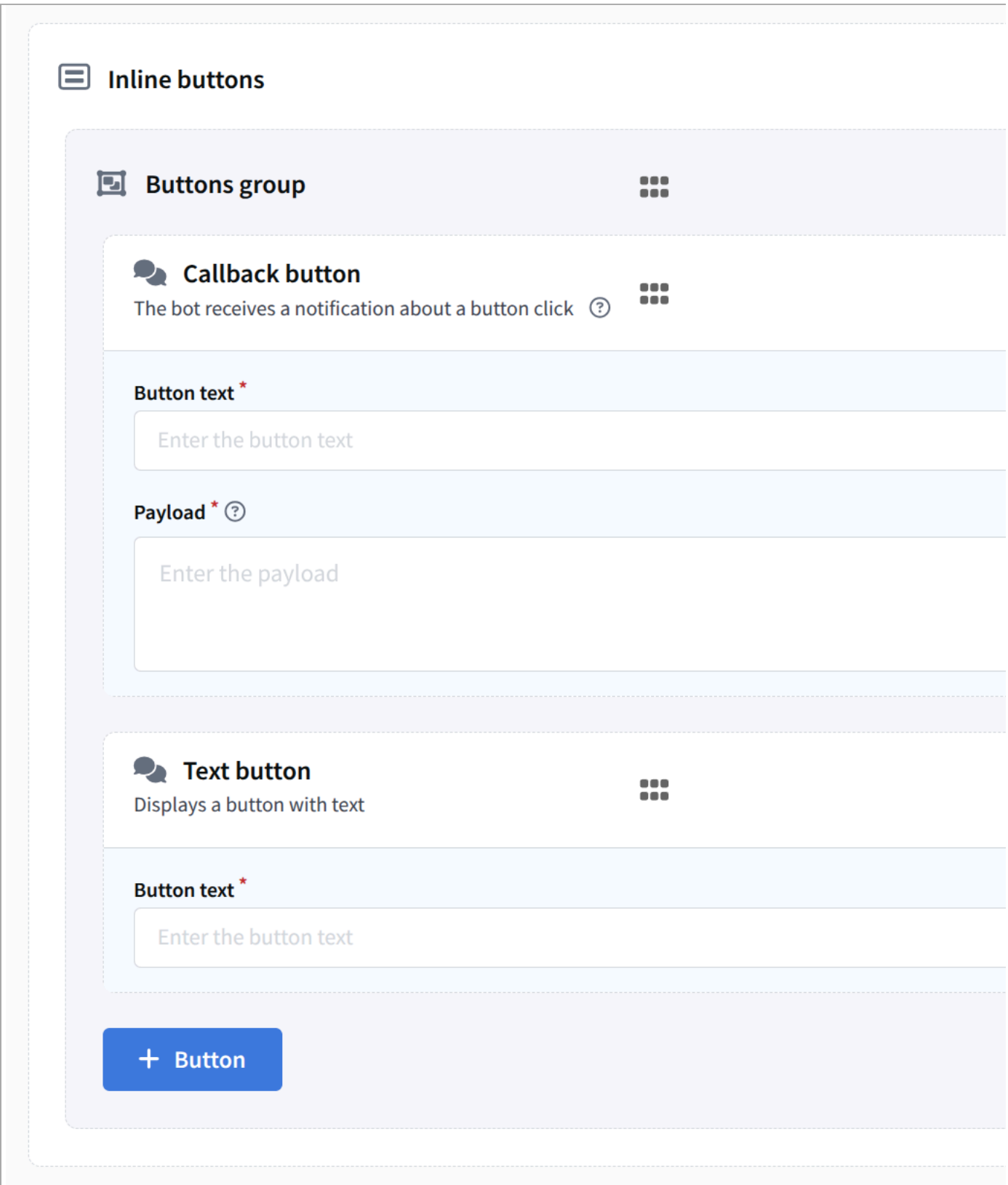


Preview in MAX

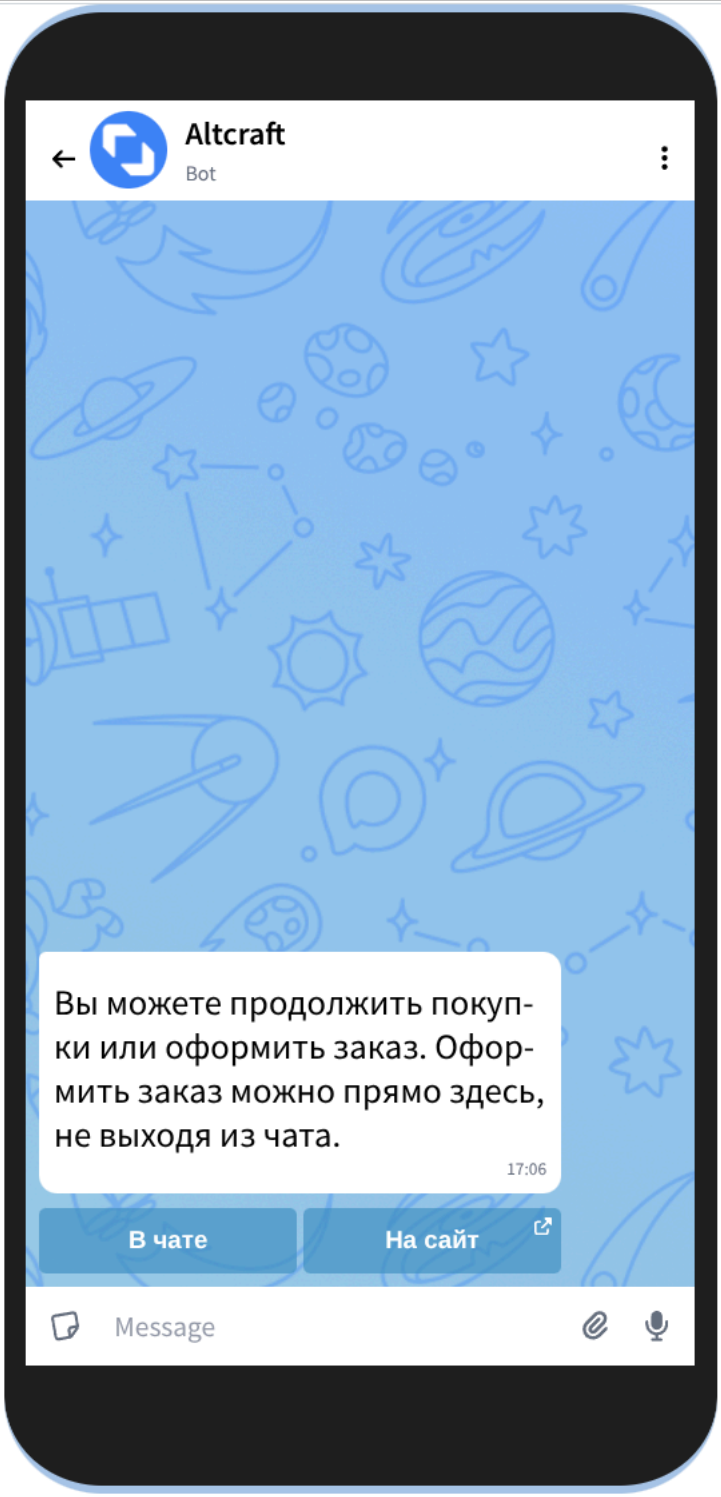


Button group

A combination of several buttons. A single button group can contain up to 7 buttons if only text and callb buttons if at least one button of another type is used (link, open app, request location, or request contac

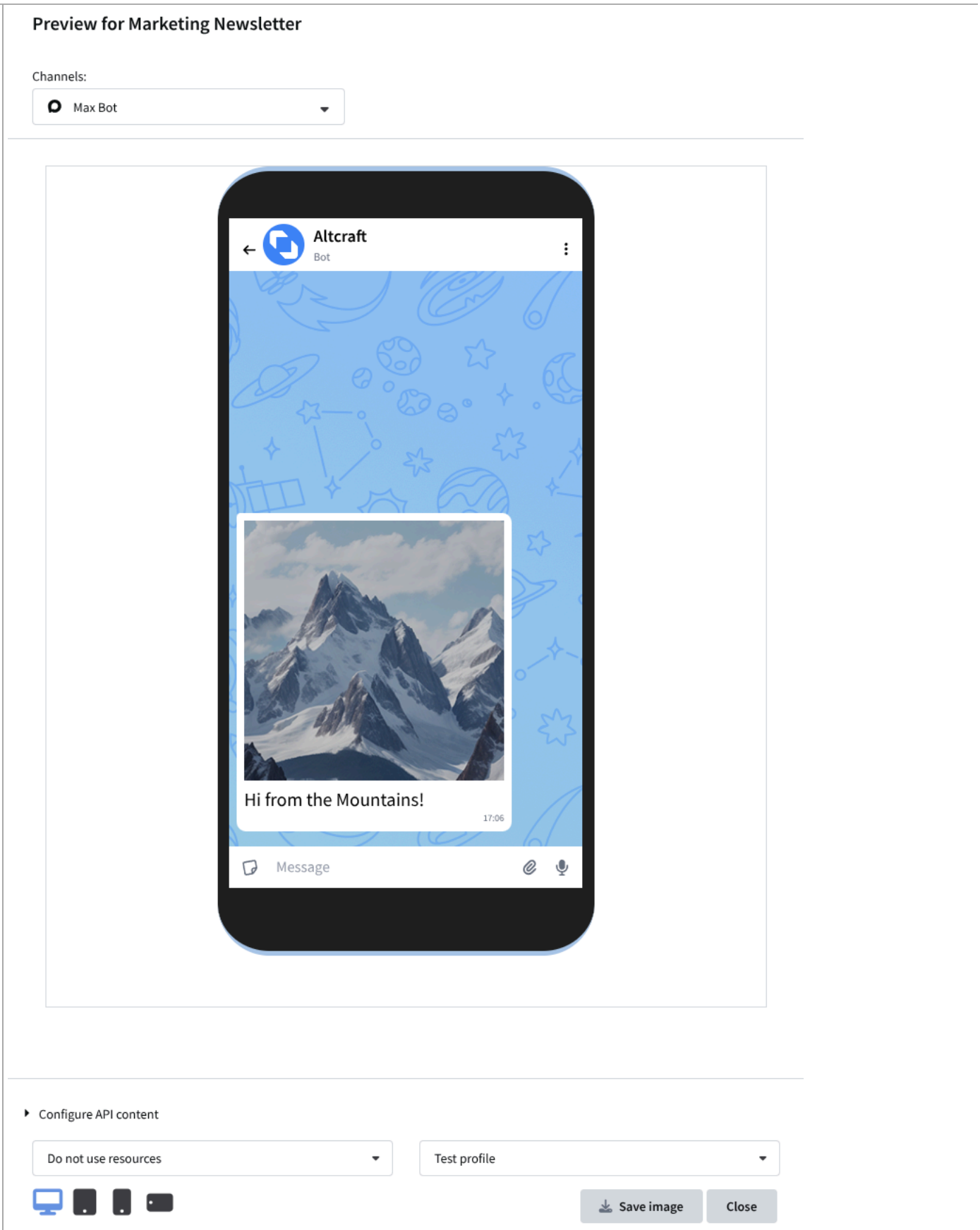


Preview in MAX



Preview and save

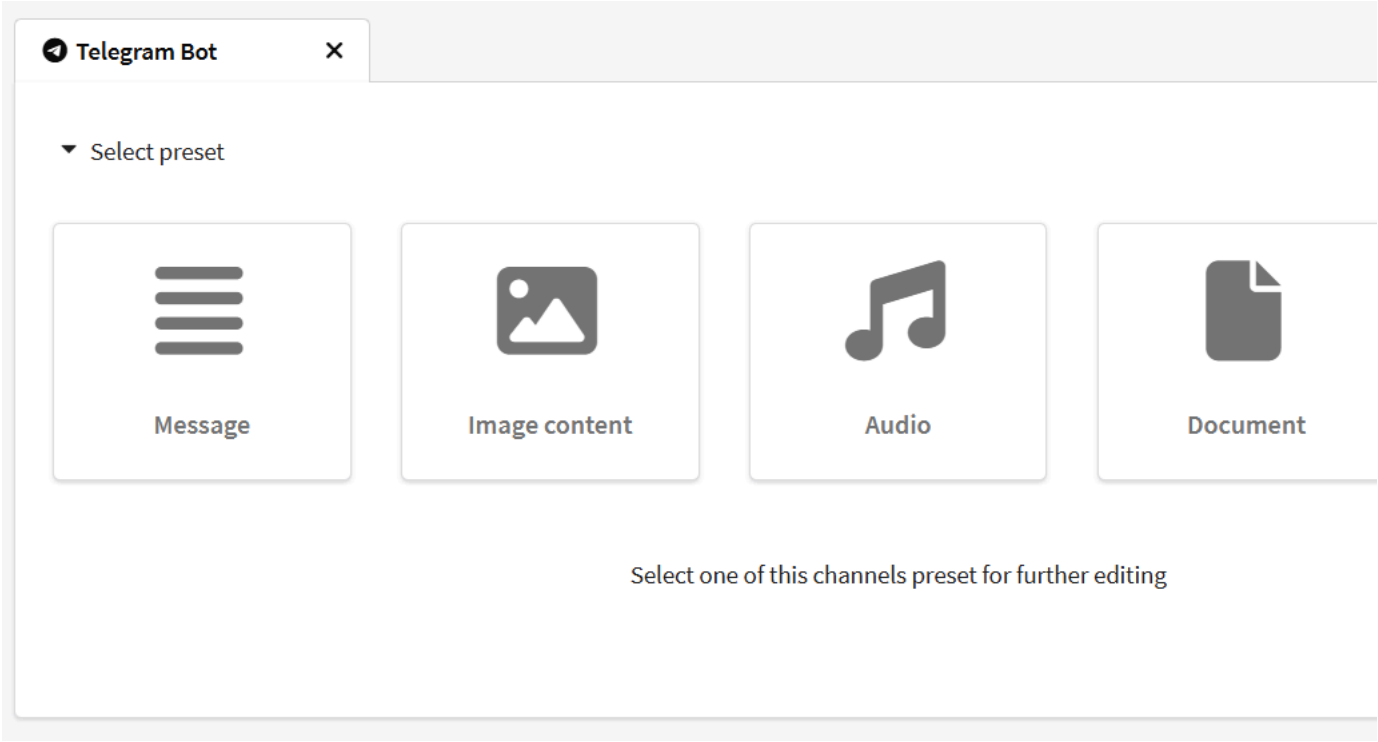
At the top of the template editor, there is a button for switching to preview mode. It can be used to chec recipient's messenger:



After all required elements have been added, save the template. You can then use it in Campaigns via MA

Telegram template

This article describes how to create templates for sending messages through channels such as Telegra
First, select the type of content you want to send to subscribers:

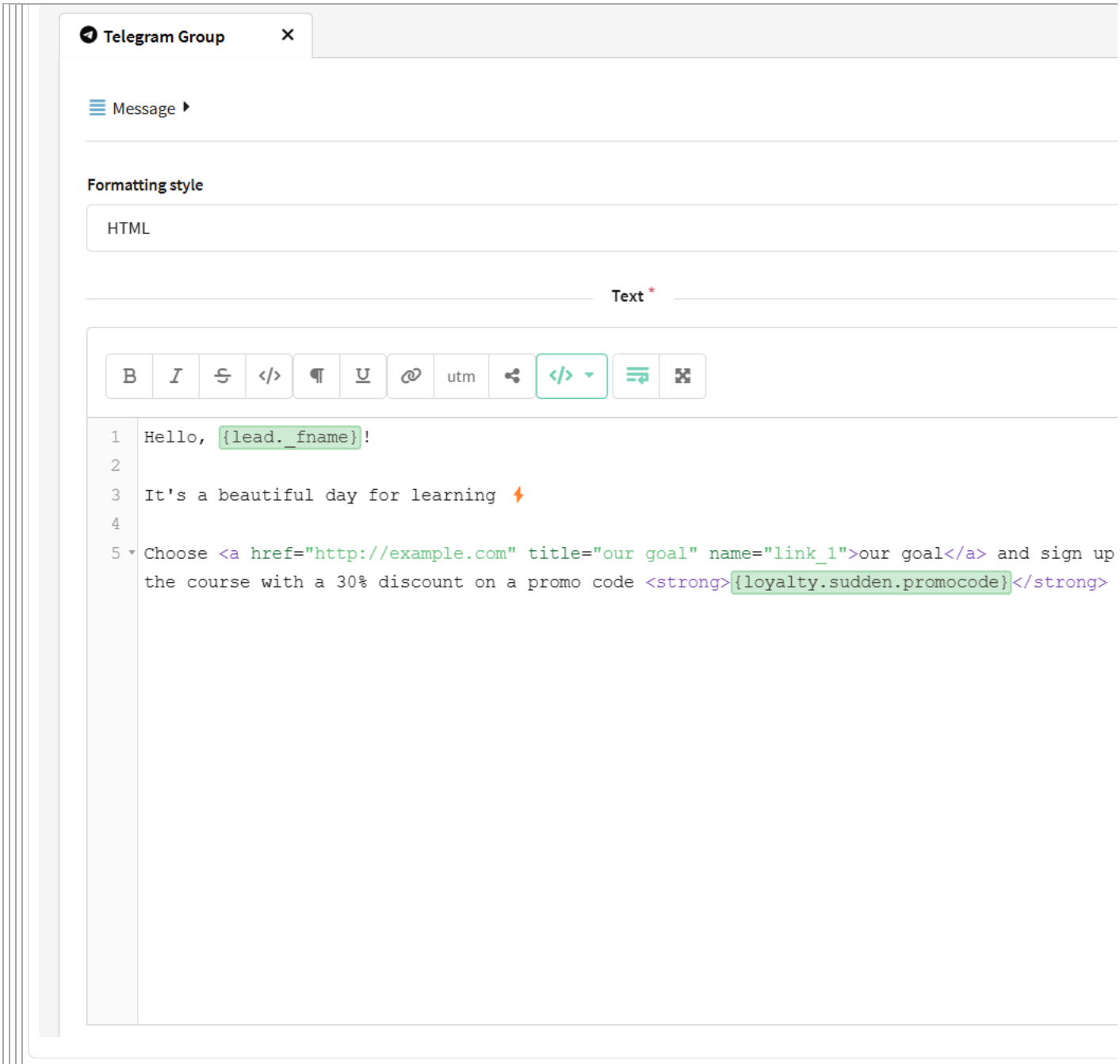


Message

Select the formatting style. We recommend choosing HTML, since the rest of the formatting styles are o message.

TIP

The maximum length of a text message in Telegram is 4096 characters. Please note that data that is pu exceed this limit.



The toolbar in the Telegram editor

The following functions are available to facilitate formatting and working with objects:

B

— bold text.

I

— italics.

~~S~~

— ~~strikethrough text~~.

</>

— code block.

¶

— preformatted text.

U

— underlined text.

— link editor, read more right there. Links inserted through the platform are automatically shortened. Th domain) length is 12-14 characters, including the `/`.

utm

— adding UTM tags.

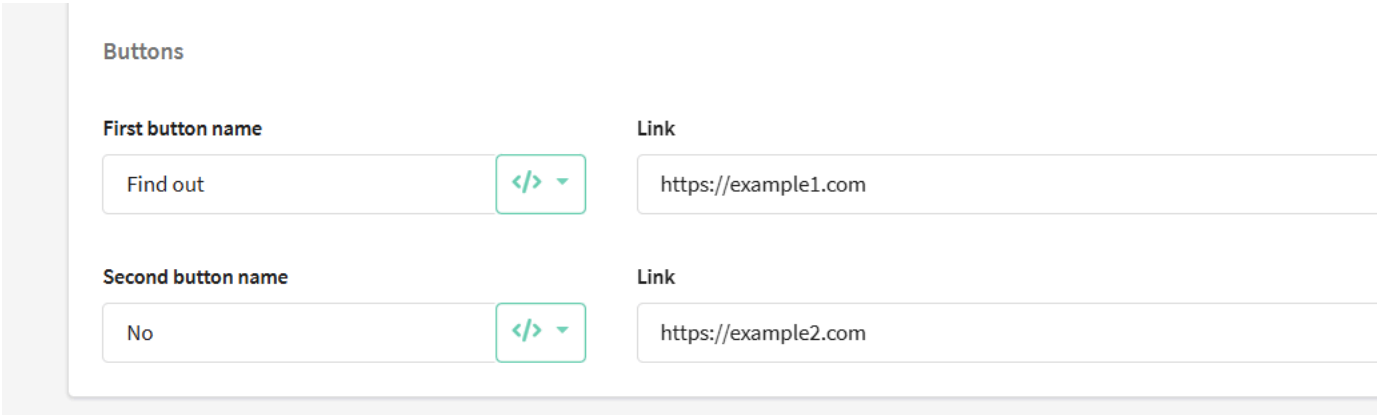
— adding buttons Social share links.

— menu of variables and functions of the editor. Using this button, you can add customer profile data, pr boolean expressions, as well as dynamic content and system links.

— toggle word wrapping.

— full-screen mode.

To add a button to a message, enter a name and a link that the user will follow by clicking on the button. buttons.

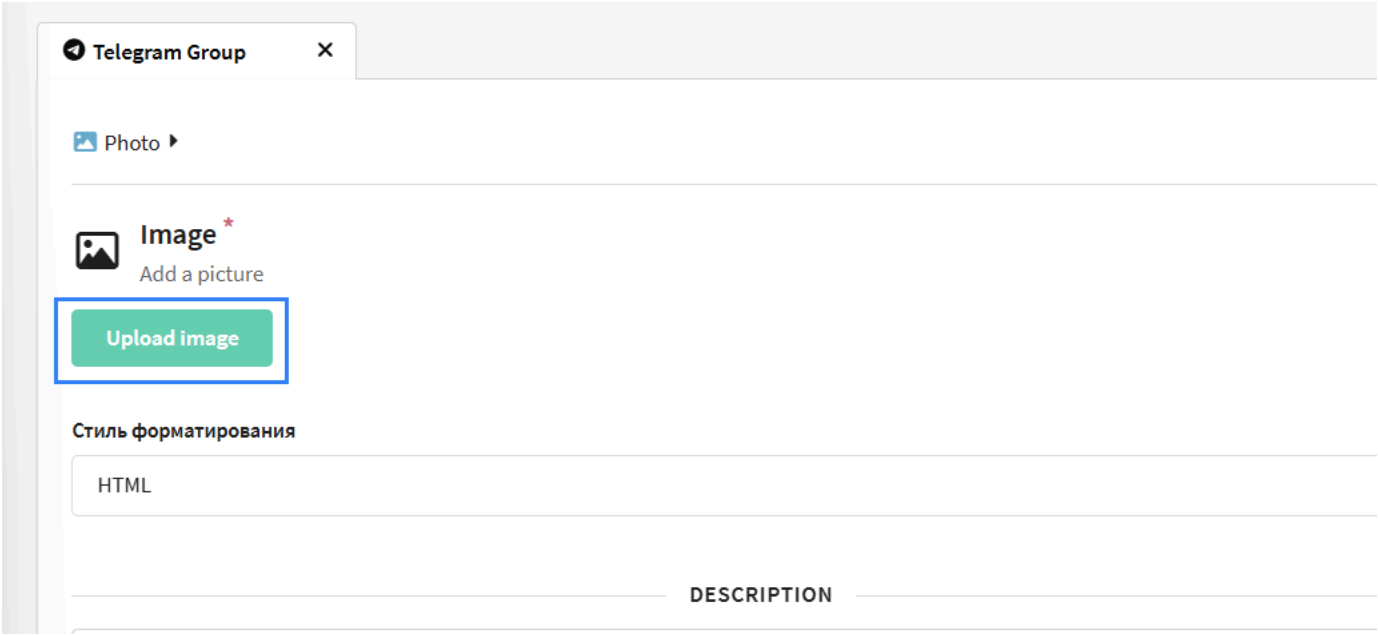


TIP

The links in the templates for Telegram Bot and Telegram Group are shortened.

Media files

In order to add am image, click on the button Upload image and select an image in the gallery.



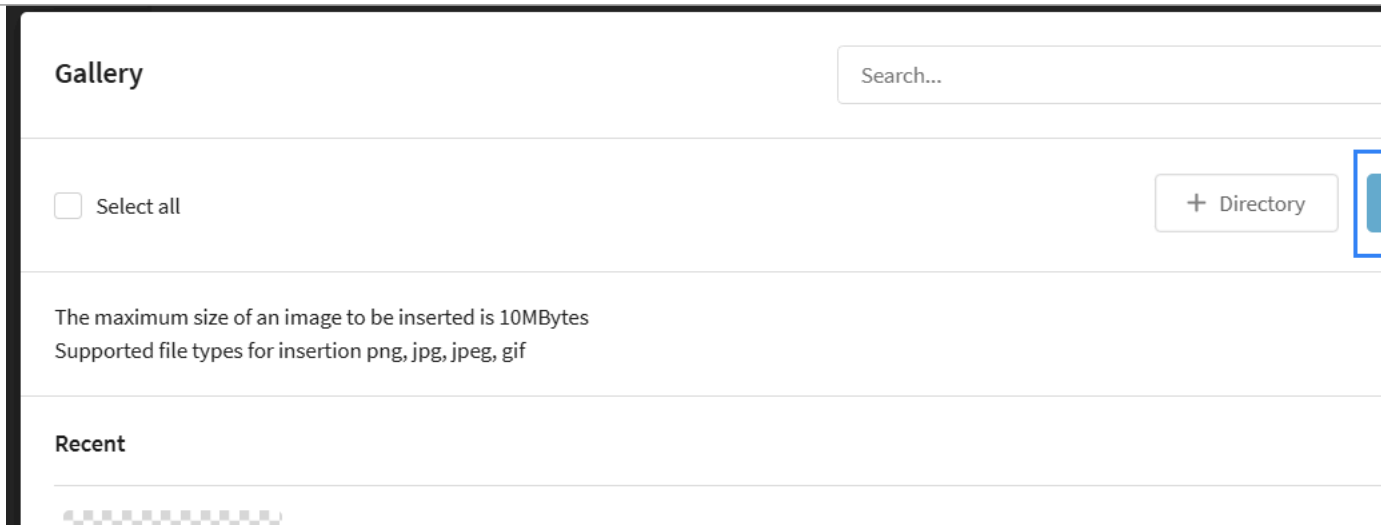


Image requirements:

- **Formats: PNG, JPG, JPEG, GIF.**
- **Maximum size: 10 MB.**
- **The sum of the width and height of the photo should not exceed 10000.**
- **The ratio of width and height should not exceed 20.**

In order to add an audio, video or document, use one of the options:

1. **Download the file from your device.**
2. **Add the file to the local path. Select the file storage directory on the server where the platform is ins variables in the directory path. Enter the location of the file and click “Add”.**

Audio file requirements:

- **Formats: MP3 and M4A.**
- **Maximum size: 5 MB.**

Document requirements:

- **Formats: any document format.**
- **Maximum size: 5 MB.**

Video requirements:

- **Formats: MP4.**
- **Maximum size: 5 MB.**

TIP

You can also add a description (up to 1024 characters) to files of any type.

WhatsApp template

Description

At the moment, only template messages (HSM) can be sent via Altcraft Platform.
A template message is an HSM message that must be pre-approved by Facebook. Use this type of templ with a user. As of July 1, 2025, each sent template is billed separately based on its category.
If a customer contacts you first on WhatsApp, a 24-hour window opens during which you can send servic charge.
Message templates for WhatsApp are created in the provider's personal account.

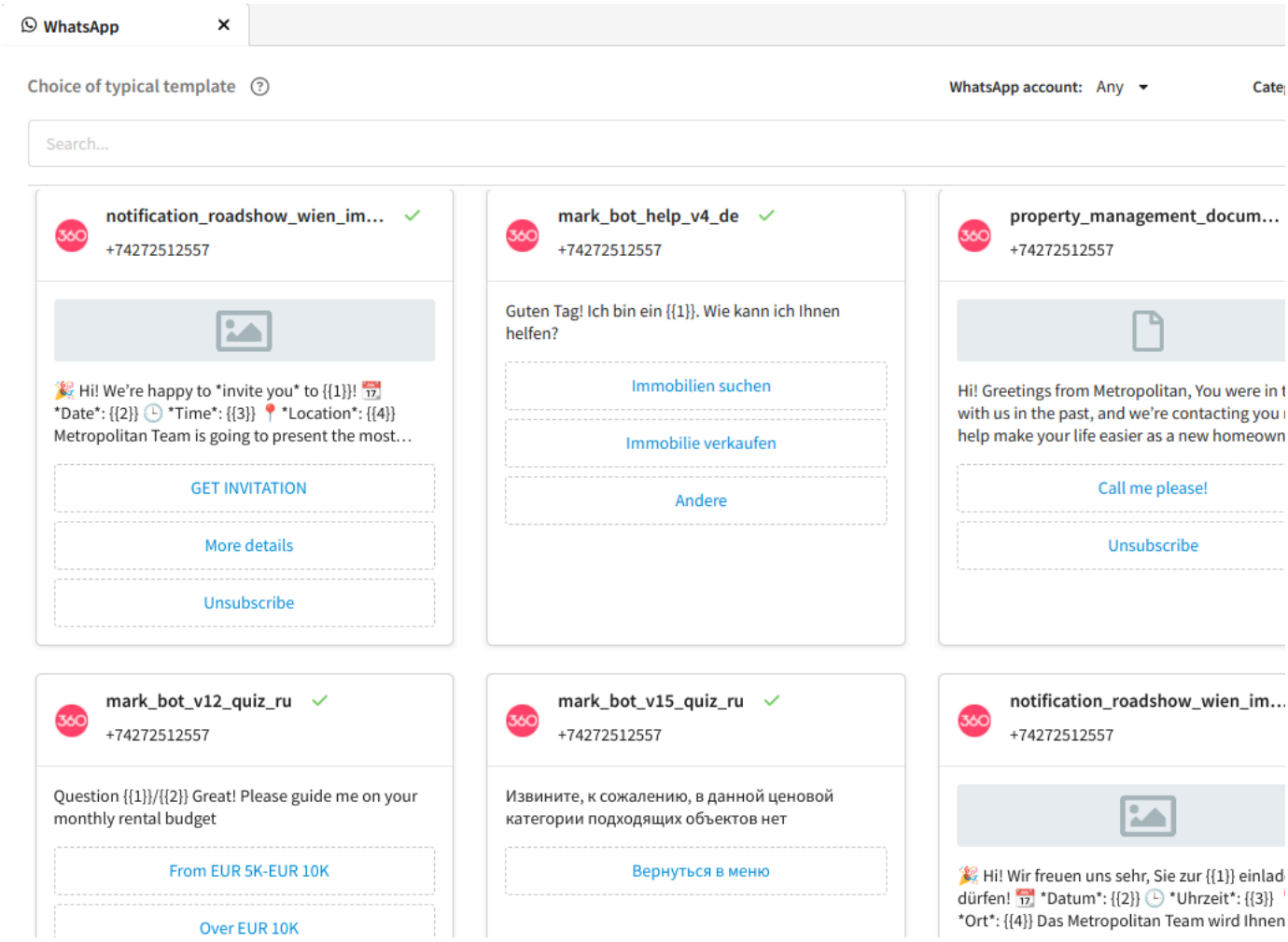
> CAUTION
>
> **Currently, only typical templates (HSM) can be sent via the Altcraft Platform.**

Message templates for sending to WhatsApp are created in the provider's personal account.

> TIP
>
> **You can find more information about creating templates in your provider's documentation.:**
>
> - **Edna: https://docs-pulse.edna.ru/docs/template/whatsapp/operator-template**
> - **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

Creating a message

All templates available in WhatsApp accounts connected via Edna or via 360dialog are automatically ava displayed in the editor. To search for the desired template in the platform, you can enter a query in the s account/category in the upper right corner.



> CAUTION
>
> **Mailings can only use templates agreed with Facebook.**

The status of the template is displayed to the right of its name:

— the template has successfully passed moderation;

— template rejected;

— the template is under moderation;

— the template has not been submitted for moderation.

Select the template you want to use in the mailing. To create a message based on a WhatsApp template, will be displayed in the editor:

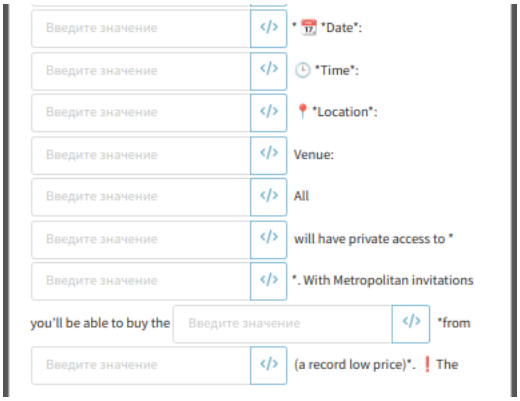
- **Header — template header with a welcome text or media file;**
- **Body — the main text of the message with variables (required block);**
- **Footer — additional text;**
- **Button — a block with buttons to add interactivity.**

You can change template elements such as variables, media files and buttons.

Variables

A text block in WhatsApp consists of fixed text and variables. Variables are the template text that chang address, event name, date and time, link, etc.).

You can manually set variable values or add Altcraft variables. To open the list of variables, click `</>`. W text is automatically inserted into the template.



TIP

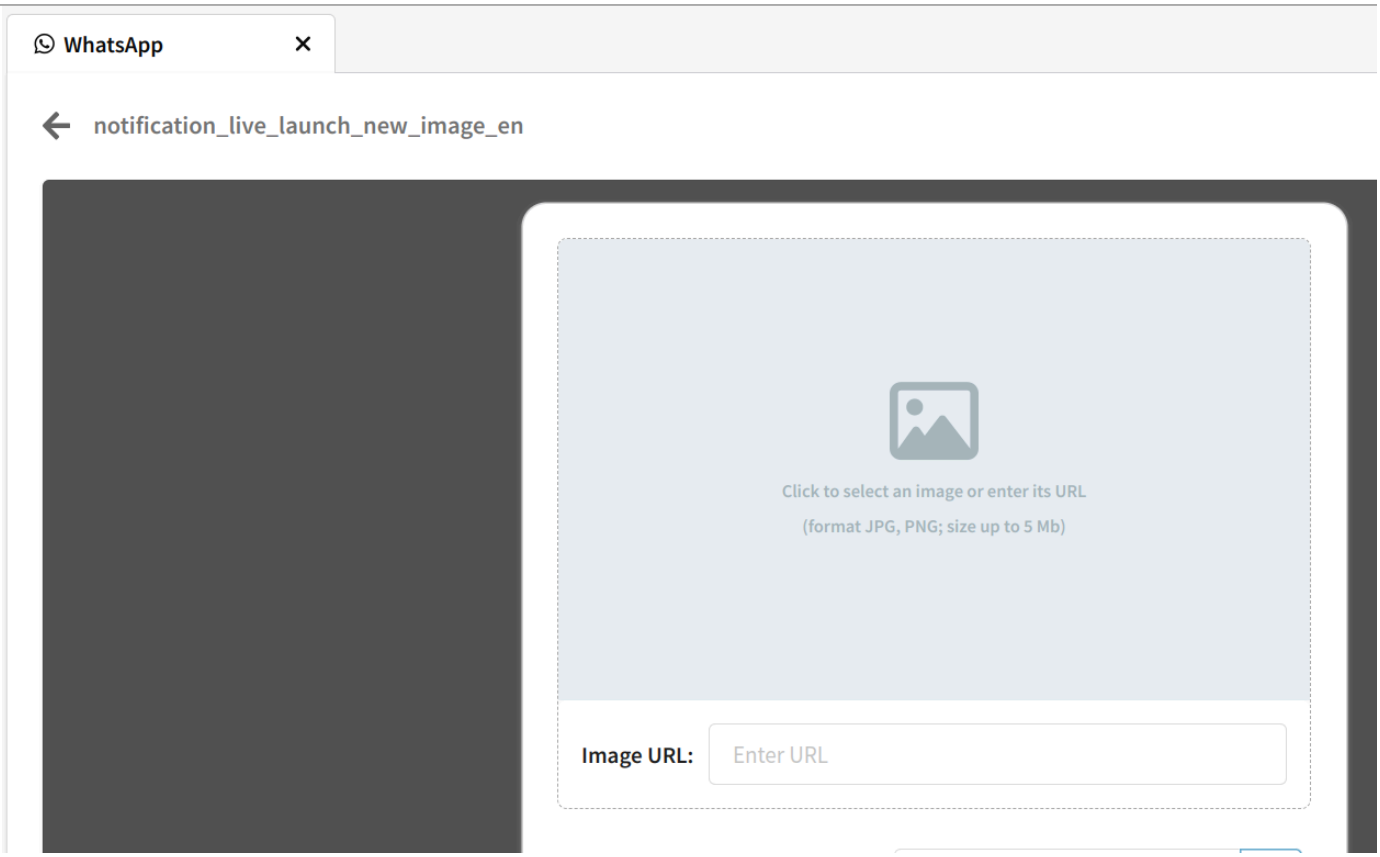
Links inserted through the platform are automatically shortened. The final link path (excluding tracking including the `/`.

Media Files

Available media files:

- **image (JPG/PNG, up to 5 MB, recommended image size is 640x640.)**
- **file (any correct MIME type, up to 16 MB)**
- **video (MP4, 3GP, 3GPP, up to 100 MB)**

To insert an image, click on the download area and select the image in the platform gallery or enter the



To add a file or video, enter the URL where the file is located.

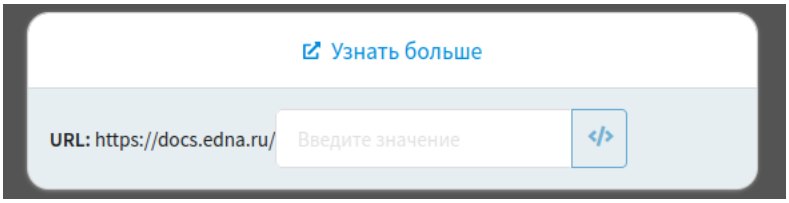
Buttons

WhatsApp offers two types of buttons for HSM templates:

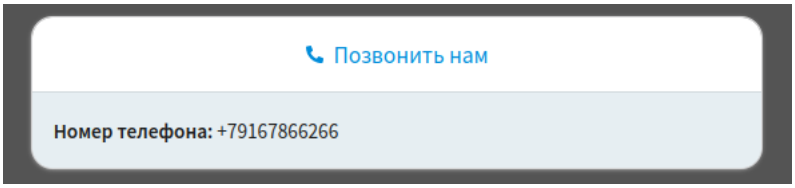
- **Call-to-action — call-to-action for users;**
- **Quick-reply — for user to reply quickly.**

Call-to-action buttons

The CTA is a link button (URL) and a call button (Phone). There can be only 2 buttons of this type in the t
When clicking on the link button, a user goes to the specified page. If a parameterizable URL is specified to fill it in.

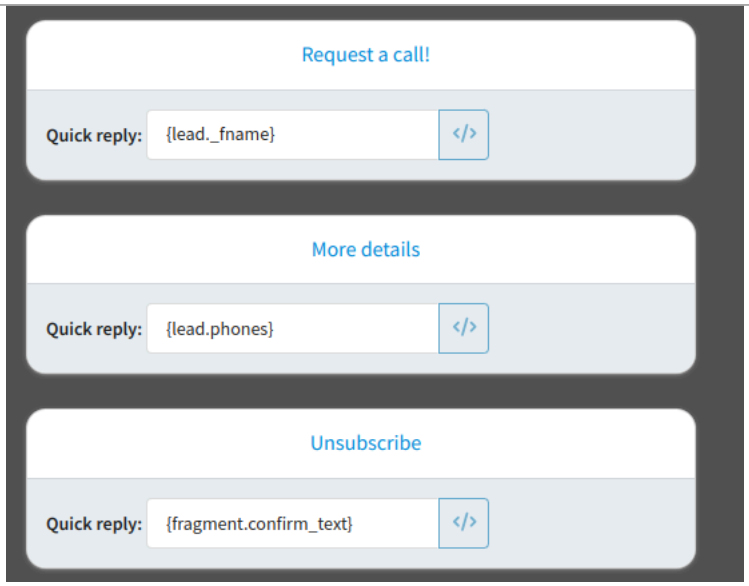


When you click on the call button, the number is dialed. The phone number cannot be edited.



Quick-reply buttons

When clicking on the button, a user receives a quick response written in the template. The template can this type.



TIP

You can save the template after filling in all the variables, buttons and adding a media file.

Preview the message

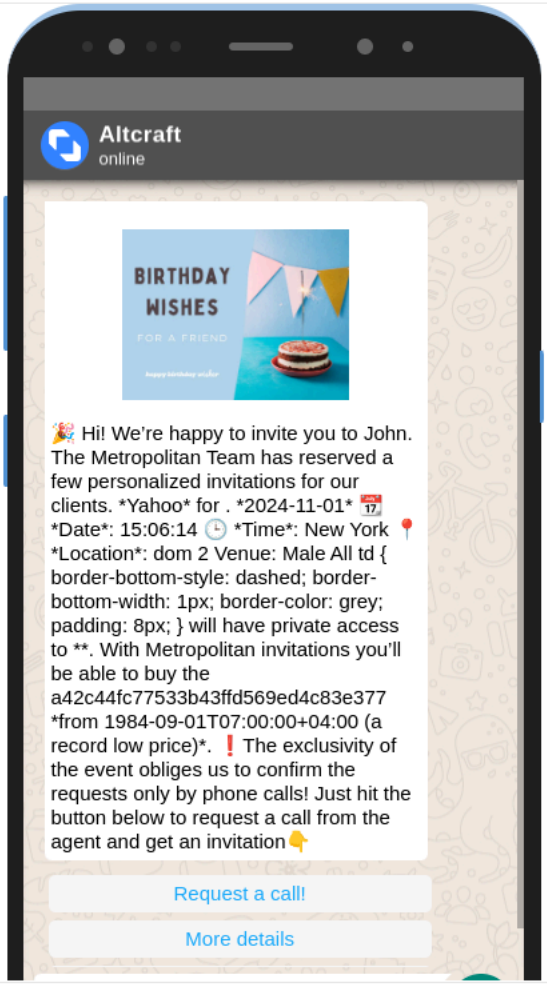
If you want to check how the message will look, go to the template preview mode by clicking on the butt

:

Preview for New planetary template

Channels:

WhatsApp



Configure API content

Do not use resources

Test profile

Save image

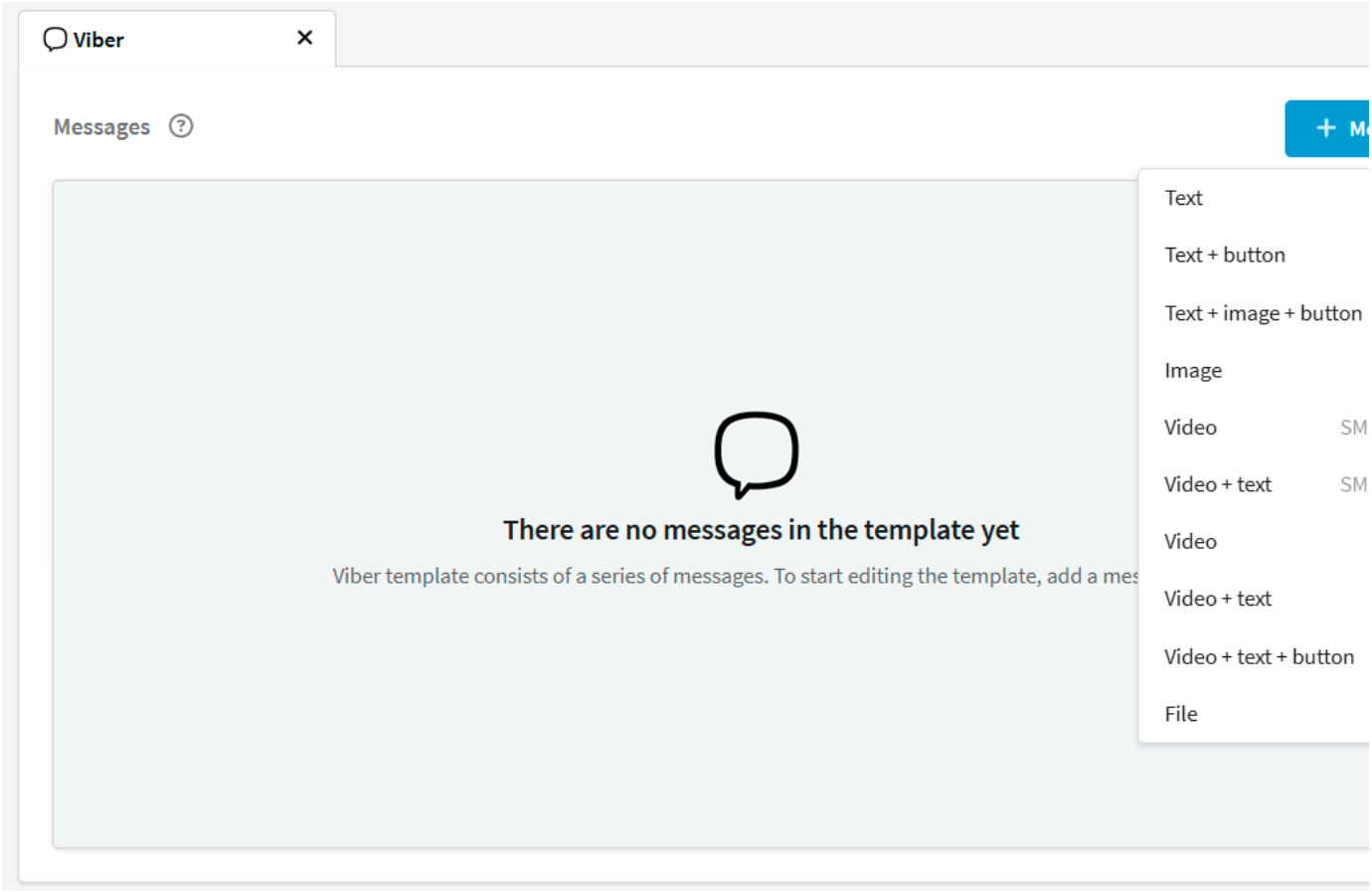
Close

Quick reply: {fragment.confirm_text}

Viber™ template

Creating a template

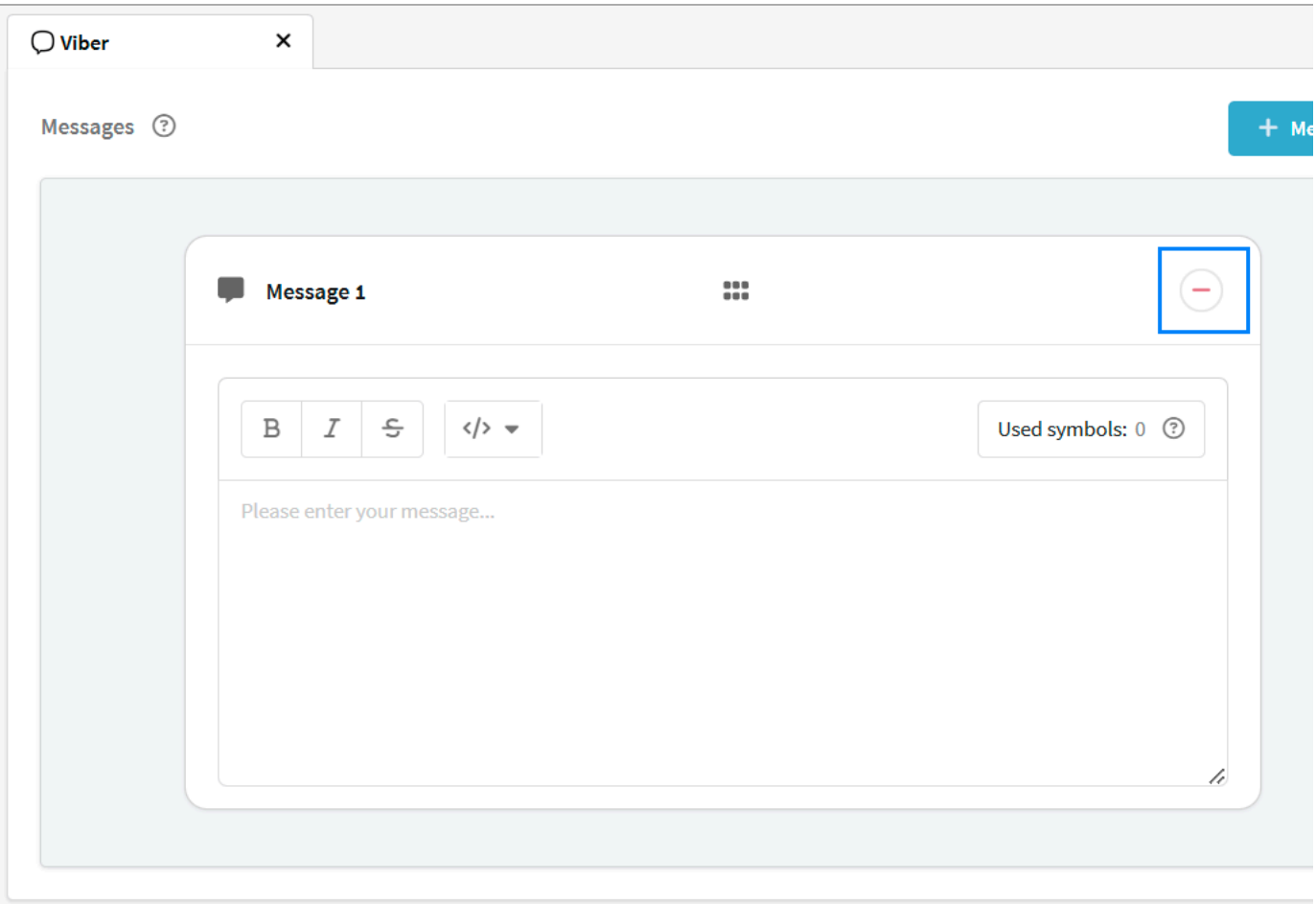
Viber*™ template consists of a series of messages. To start editing a template, add a message using the



The maximum number of messages in a series is 5. If multiple messages are used in the template, each of 2 seconds.

To remove an unnecessary message, click

.



Types of messages

The following types of messages are available:

Message type	Devino	SMSTraffic	MTS Omni
Text	+	+	+
Text + button	+	+	+
Text + image + button	+	+	+
Image	+	+	+
Video	+	+	-
Video + text	+	+	-
Video + text + button	+	-	-
File	+	-	-

Text

Enter the text of your message in the editor using the Markdown format. For ease of formatting, the follo top panel:

B

— bold text

I

— *italics*

~~S~~

— ~~strikethrough~~ text

</>

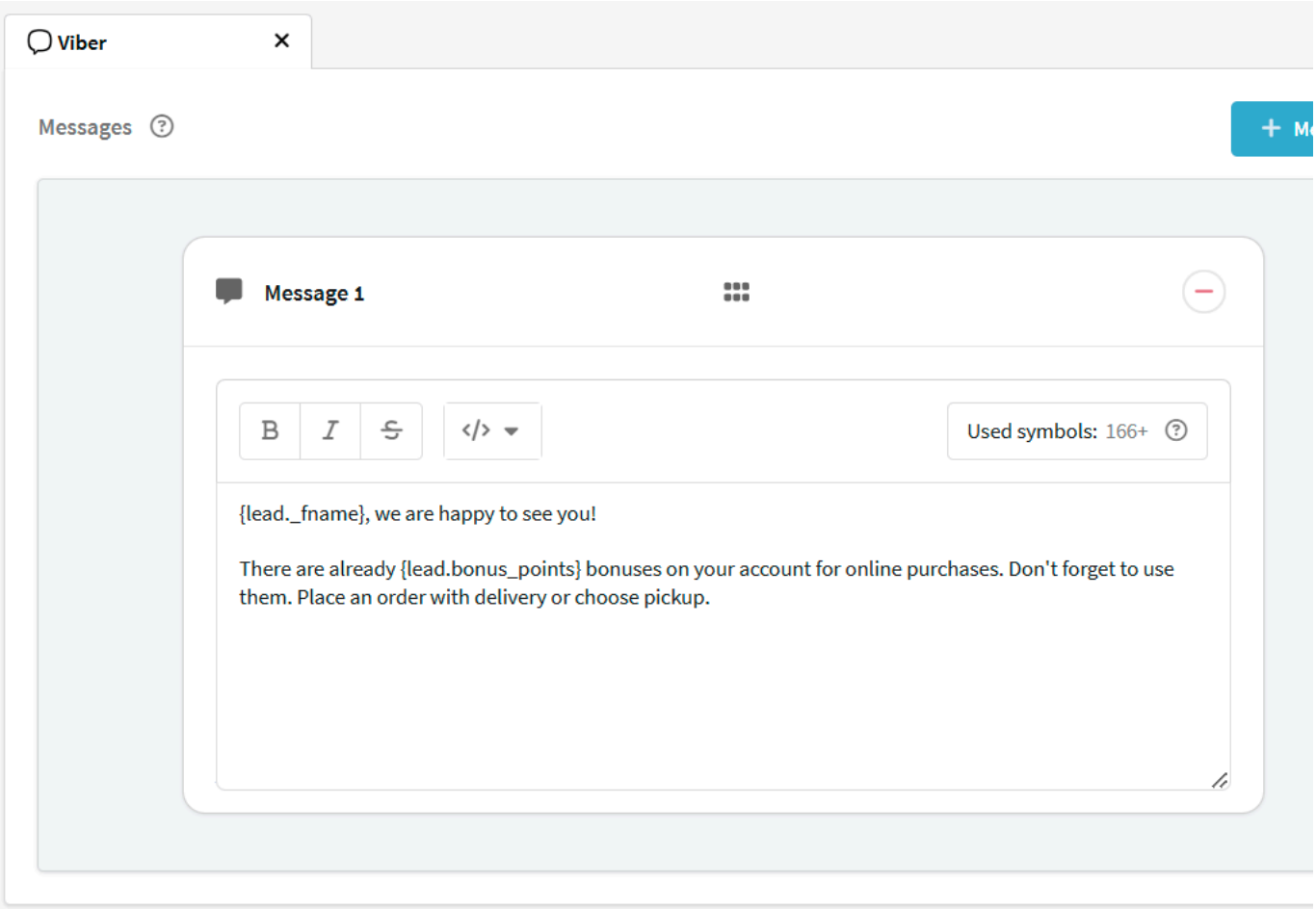
— list of editor variables and functions to personalize the message.

CAUTION

Please note that Markdown markup is displayed in preview mode (for example, `"text"`), but the recipien correct formatting (“text”).

Also, you cannot add links in Markdown format to the “Text” block.

Links inserted through the platform are automatically shortened. The final link path (excluding tracking including the `/`.



TIP

You can add emojis to Viber*™ messages. To insert an emoji, you need to use its code. For example, to available emojis can be found at [this link].

Please note that when calculating the number of characters in a message, each character in the emoji

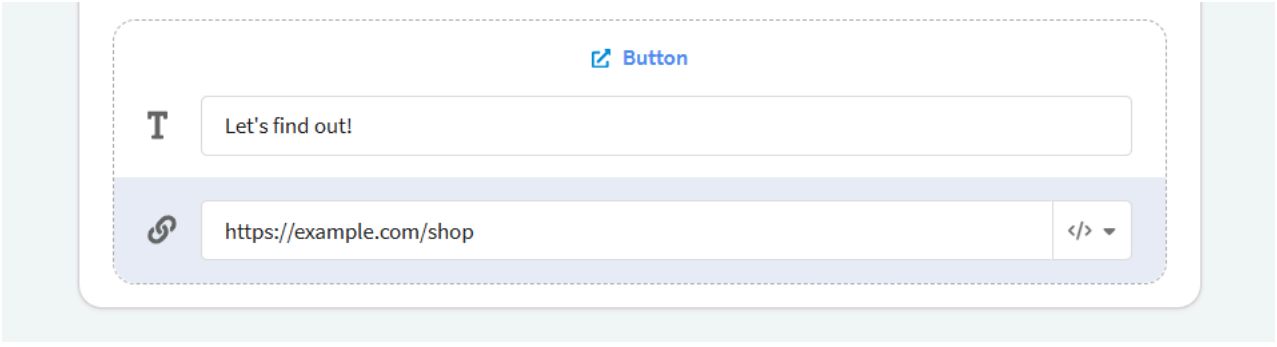
Emojis are not displayed in preview mode!

INFO

There is a character counter on the right. The maximum length of a single message is 1000 characters. from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend that you moni

Button

Specify the text of the button that will be displayed on the button and the link to click through. You can a

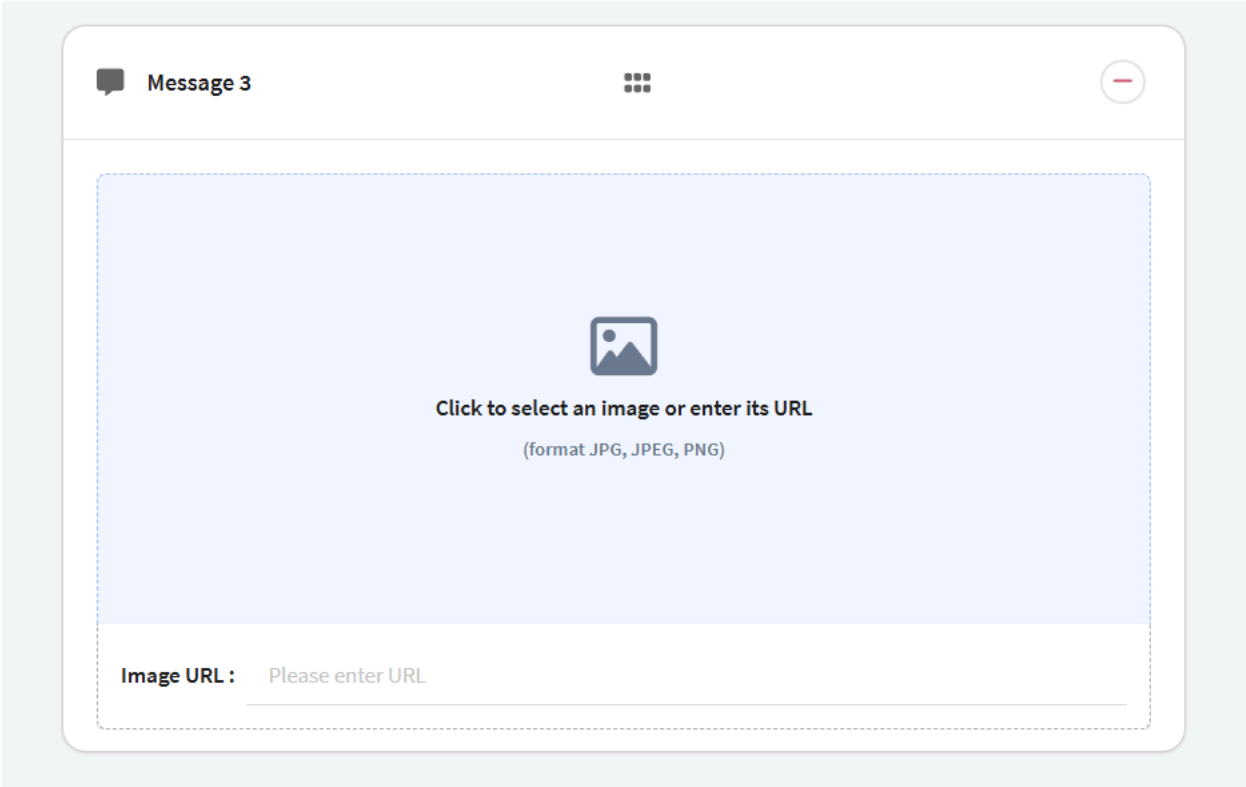


Image

Supported formats are jpg, jpeg, png. The maximum size is 5 MB.

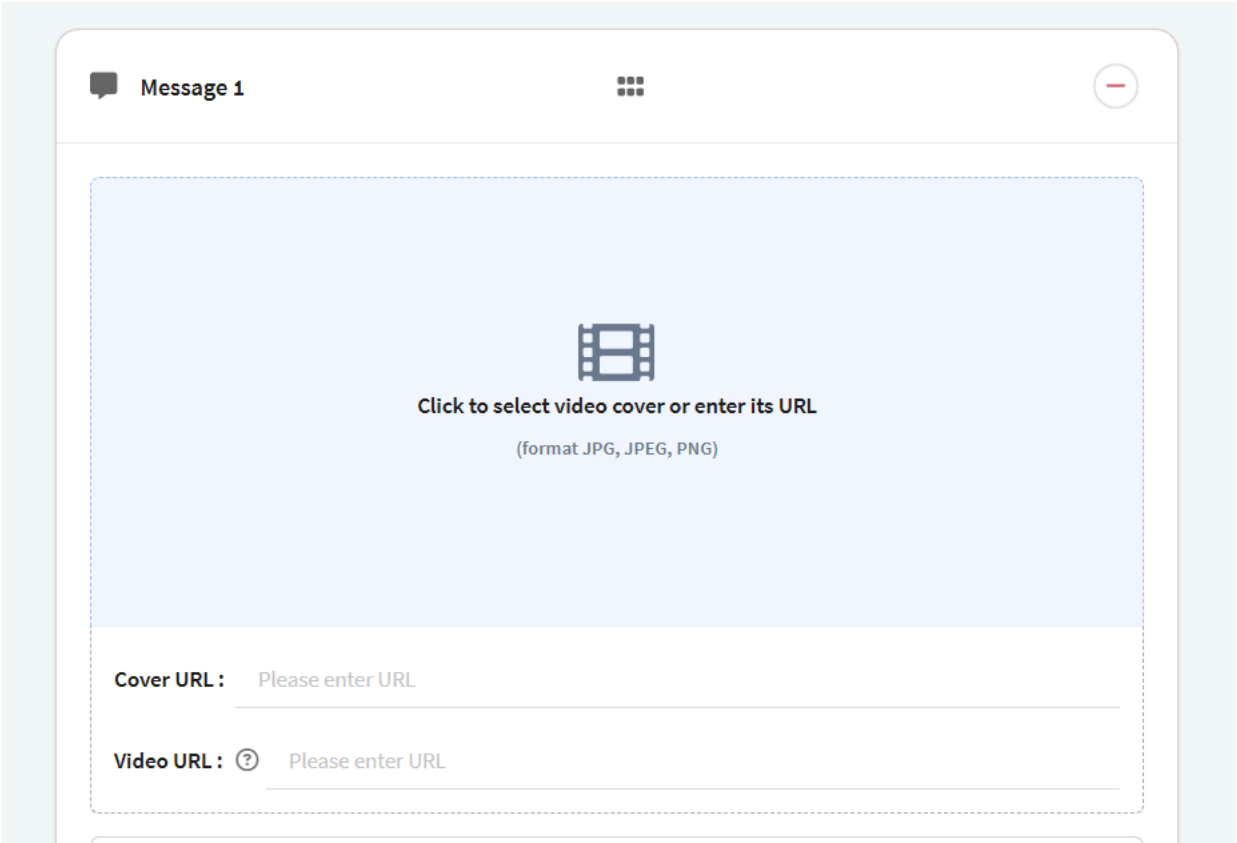
There are two ways to add images:

- **From the platform gallery. To open the gallery, click on the download area. Select the desired image device.**
- **From an external source. Specify the link to the image.**



Video

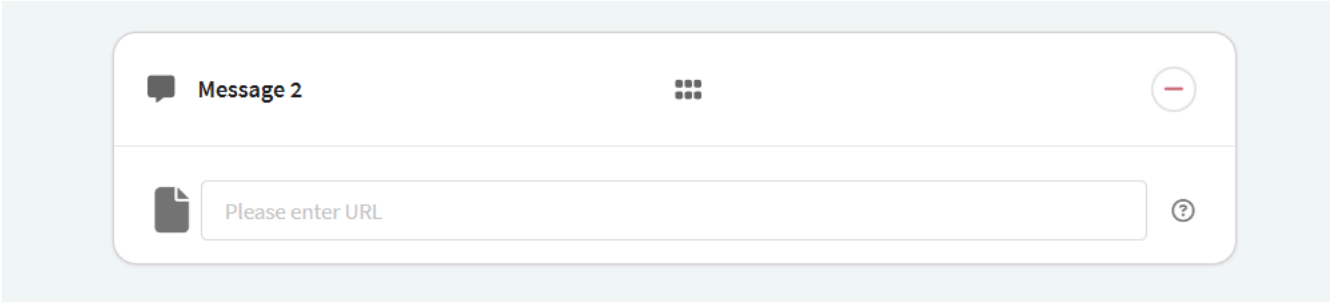
Supported formats are mp4, 3gpp. The codec is H.264. The maximum size is 5 MB.
To add a video, specify its URL. You also need to install a cover for the video. You can select an image fr link to the image.



File

Supported formats are doc, docs, rtf, odt, odf, txt, pdf, eps, xls, xlsx, ods, csv. The maximum file size is 2 characters in the file name is 25 (including the extension).

To send the file, specify the link to it.



**** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Notify template

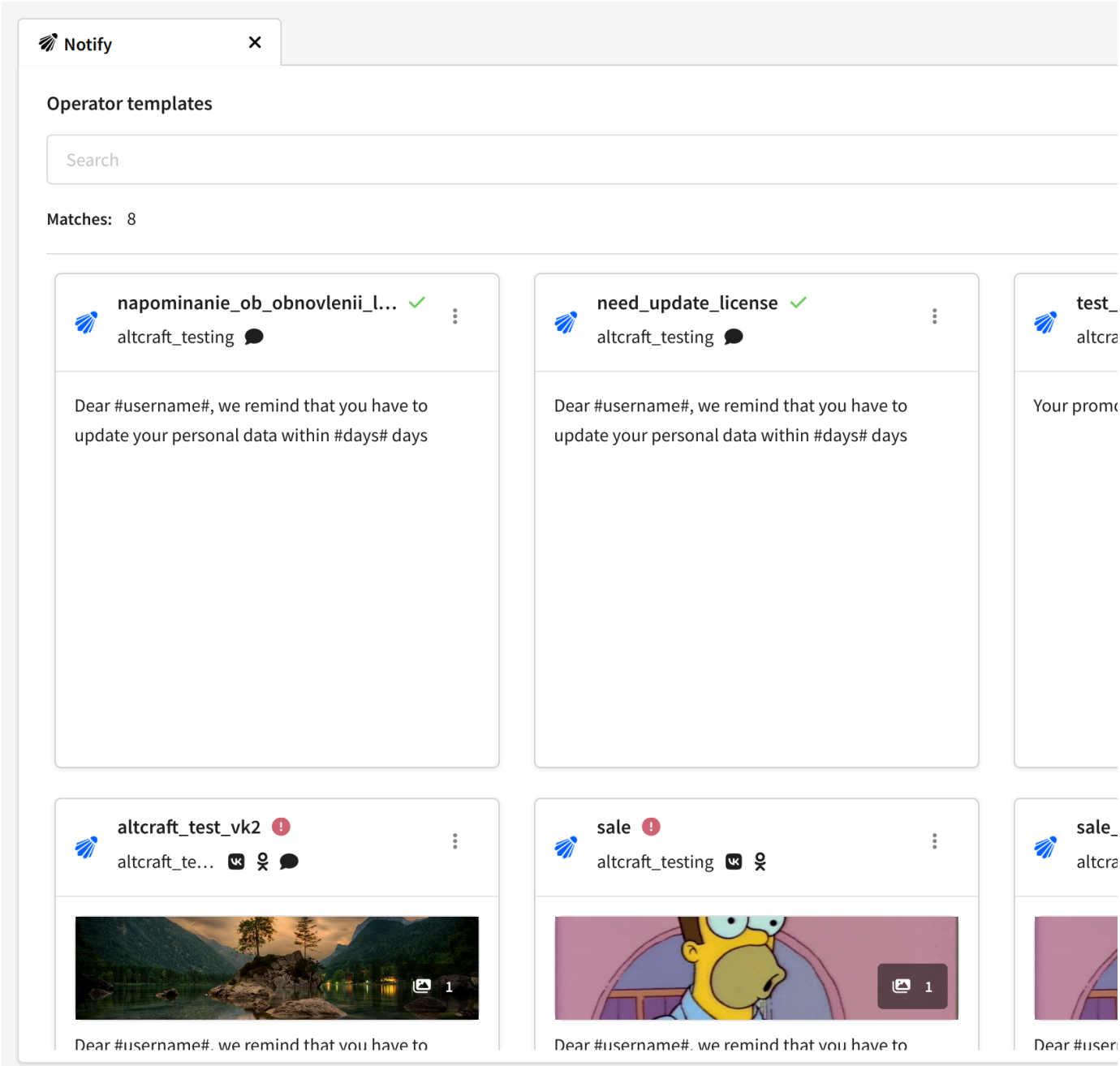
INFO

According to the service policy, new access to the Notify service personal account is currently not pro
Support for current users is provided by MAX Business tech support.

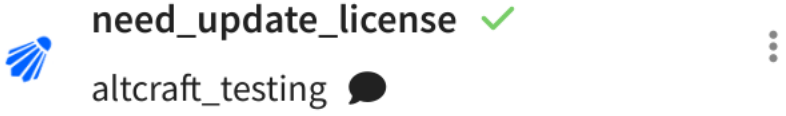
Using the Notify channel you can send business-messages via VK, OK and also send SMS notifications. S
messages are customized in two different sends that follow each other.

Existing operator templates

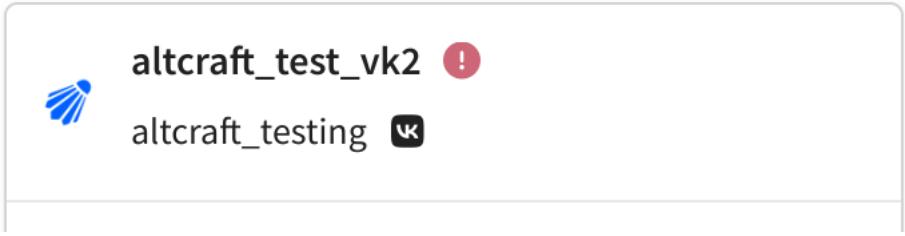
When you open the template editor for Notify, you will be shown the operator templates created in your p



All templates for sending are moderated by the service. Templates approved by the service are marked v

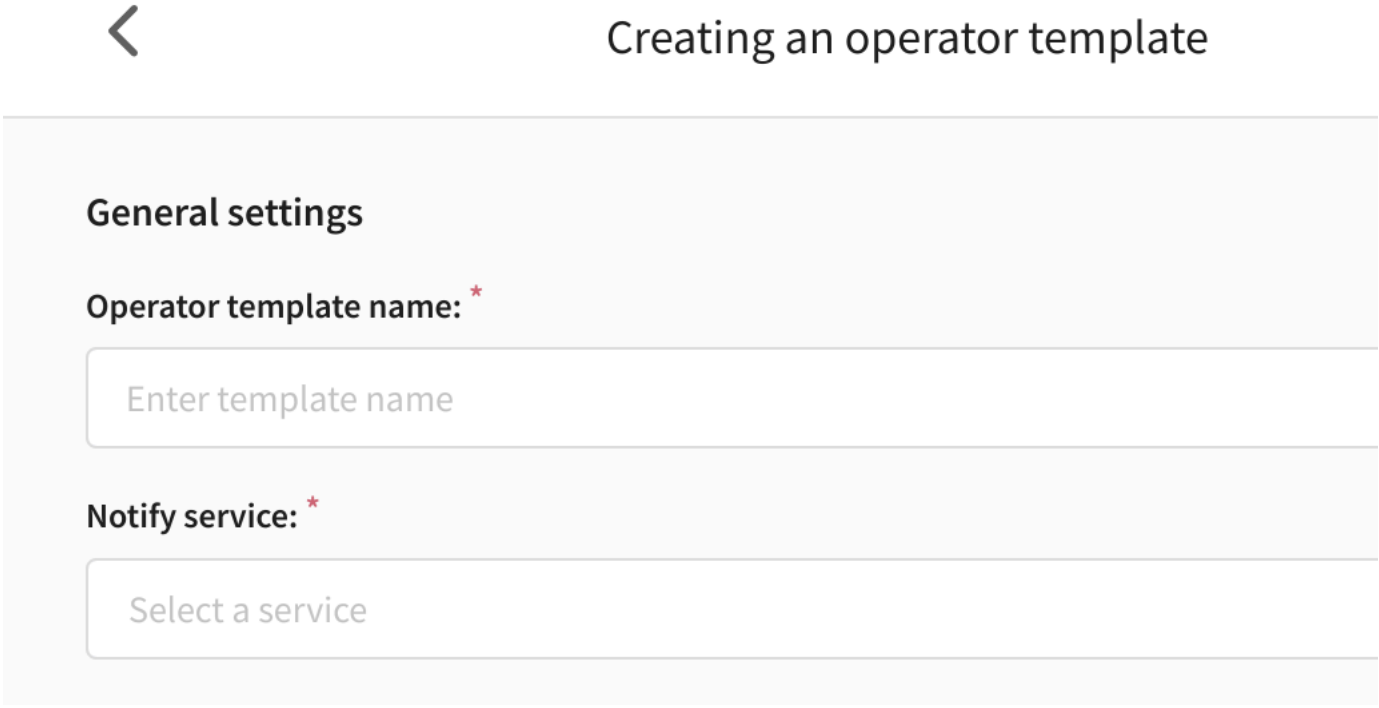


Templates that have been archived or have not been moderated will carry a warning. They cannot be use
approved:



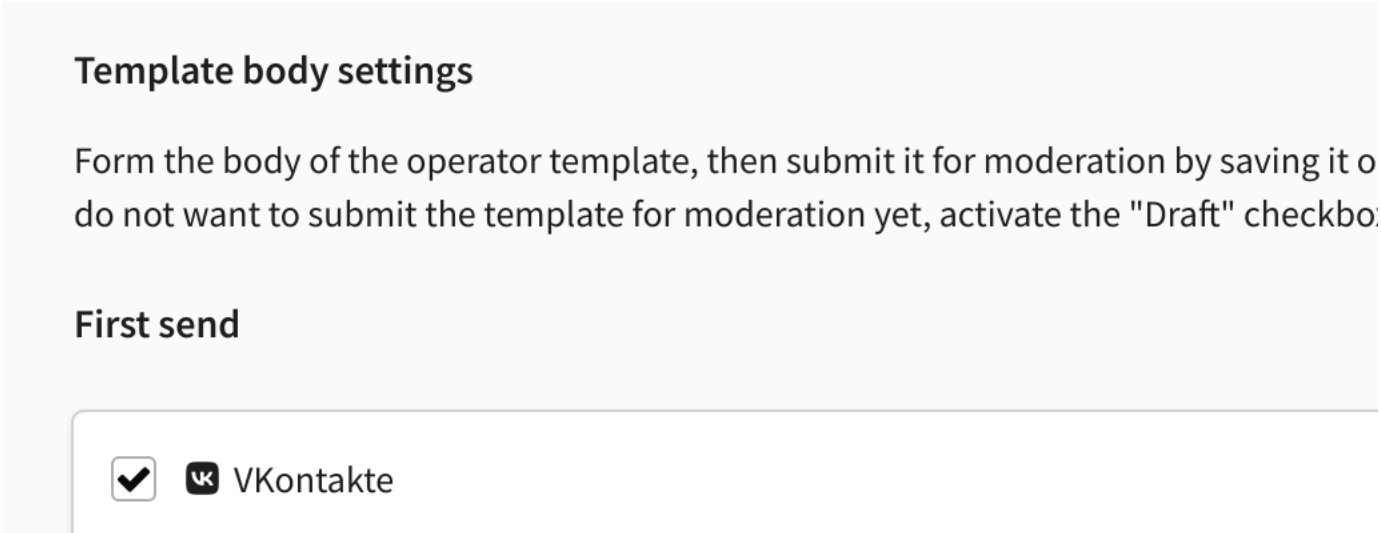
Creating an operator template

You can also create a new Notify template while in the platform. To create a template, click the corresp template creation will open, in this window you should specify name of the template and service configu

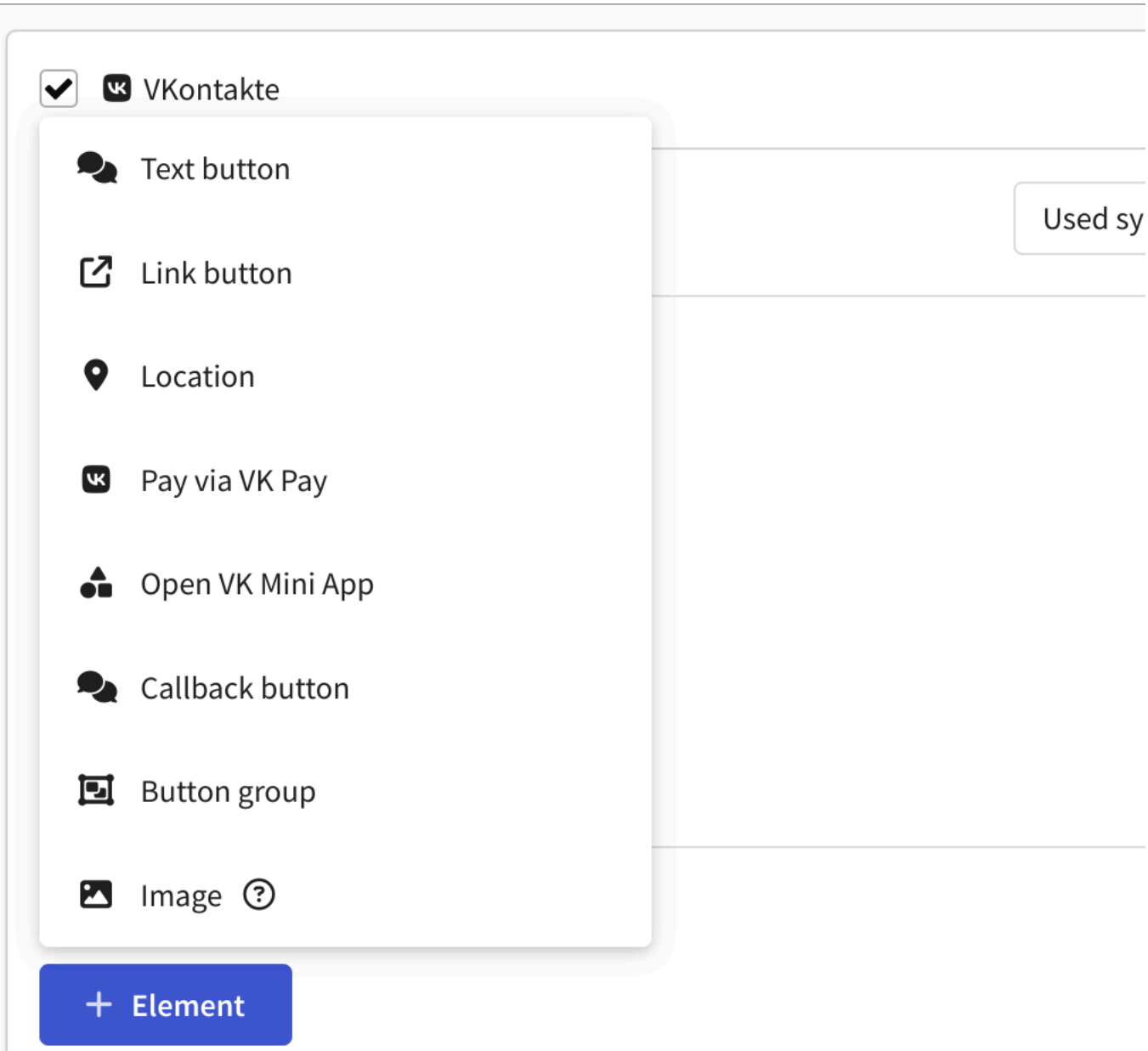


There is an option in the template to set time to wait for the message to be delivered and read. The deliv which Notify will attempt to resend the message if any error occurs. When this time expires, a second s waiting time is the amount of time Notify will wait for the message to be read on VK or OK. When this tir message will be triggered.

The Notify template consists of sends. One send is a message to VK or OK, the other sending is an SMS add or delete a sending, as well as change the order of their sending by clicking the corresponding butto



The message body defines the content of your notification. An OK message can be text-only. You can att



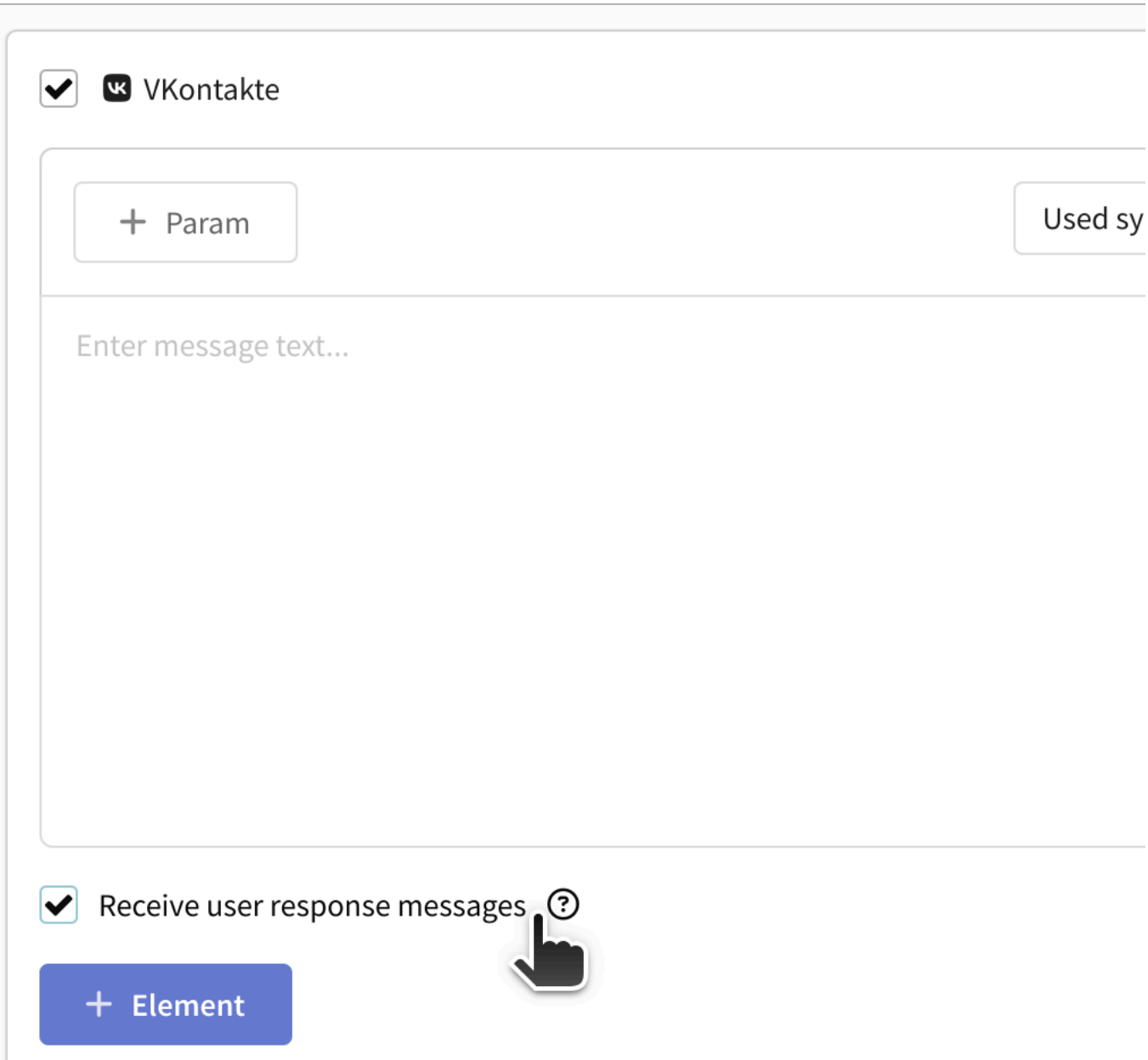
- **Text Button - sends a message with text. For a text button you can choose a color, as well as attach format.**
- **Link Button - opens the specified link. In the button you can specify its text, a link to go to, as well a**
- **Location - when clicked, opens a dialog box with location information. The user can select location i chat. Community administrators will receive the map image in chat.**
- **Pay via VK Pay - opens VK Pay payment window with predefined parameters. For the button it is nec containing payment parameters and application ID.**
- **Open VK Mini App - opens the specified VK Mini Apps application. To customize the button you need ID, as well as a hash for navigation in the application.**
- **Callback Button - allows you to receive a notification of a button press and perform the required acti the user. You can specify a caption and color for the button.**
- **Button group - cobmination of the above buttons.**
- **Image - up to 5 images with total weight not more than 50 MB. Supported formats - JPG, PNG, GIF.**

TIP

Read more about elements in Notify documentation.

You can insert parameters into the message body. Later you can substitute field values from the platform

You can also allow the client to send reply messages in the chat with the company. To do this, activate t box in the settings:



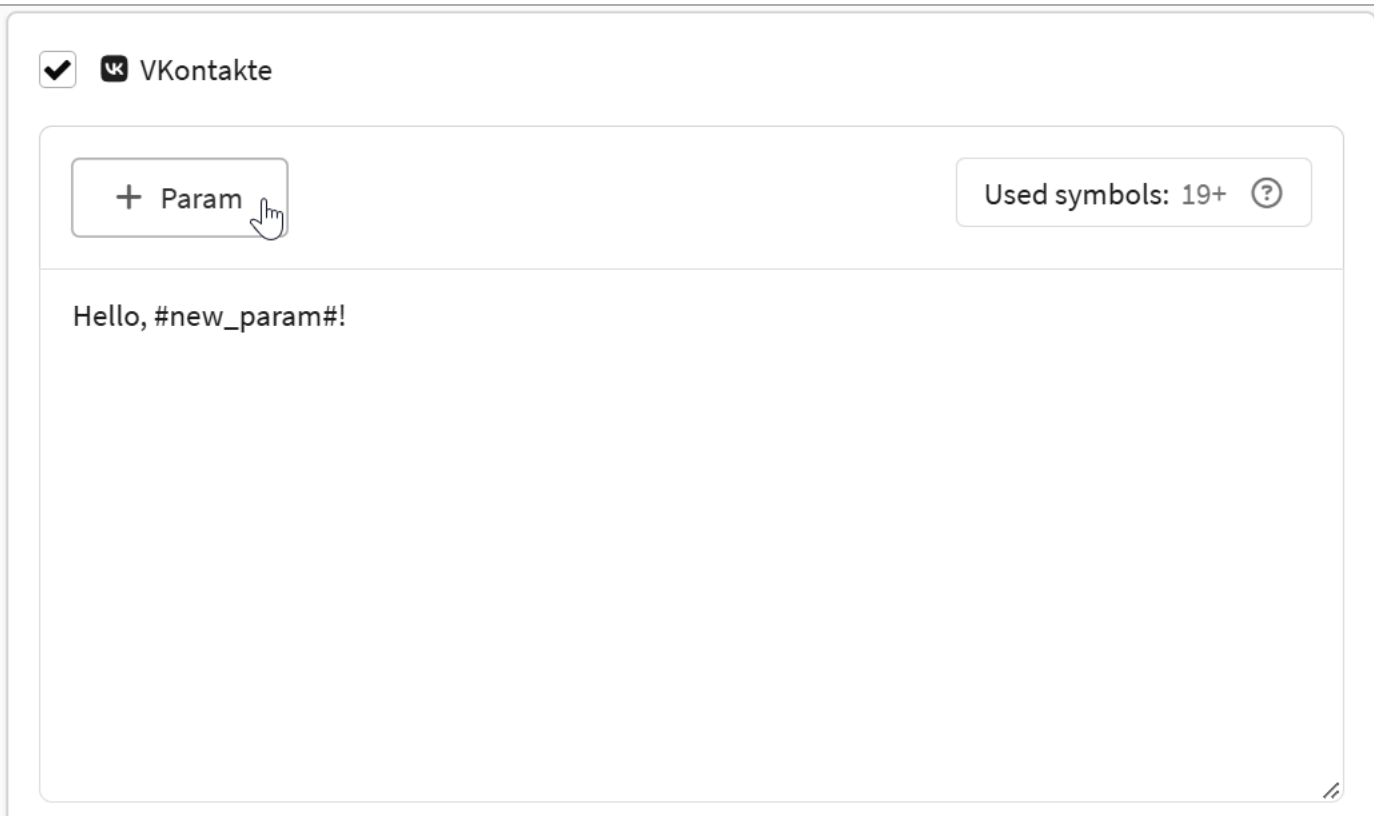
There are several types of SMS messages - advertising, identification, service and transactional. Messag
you can get detailed information about it from the service support.

Once you save the template, it will be sent to moderation in the service. If you don't want to send it for n

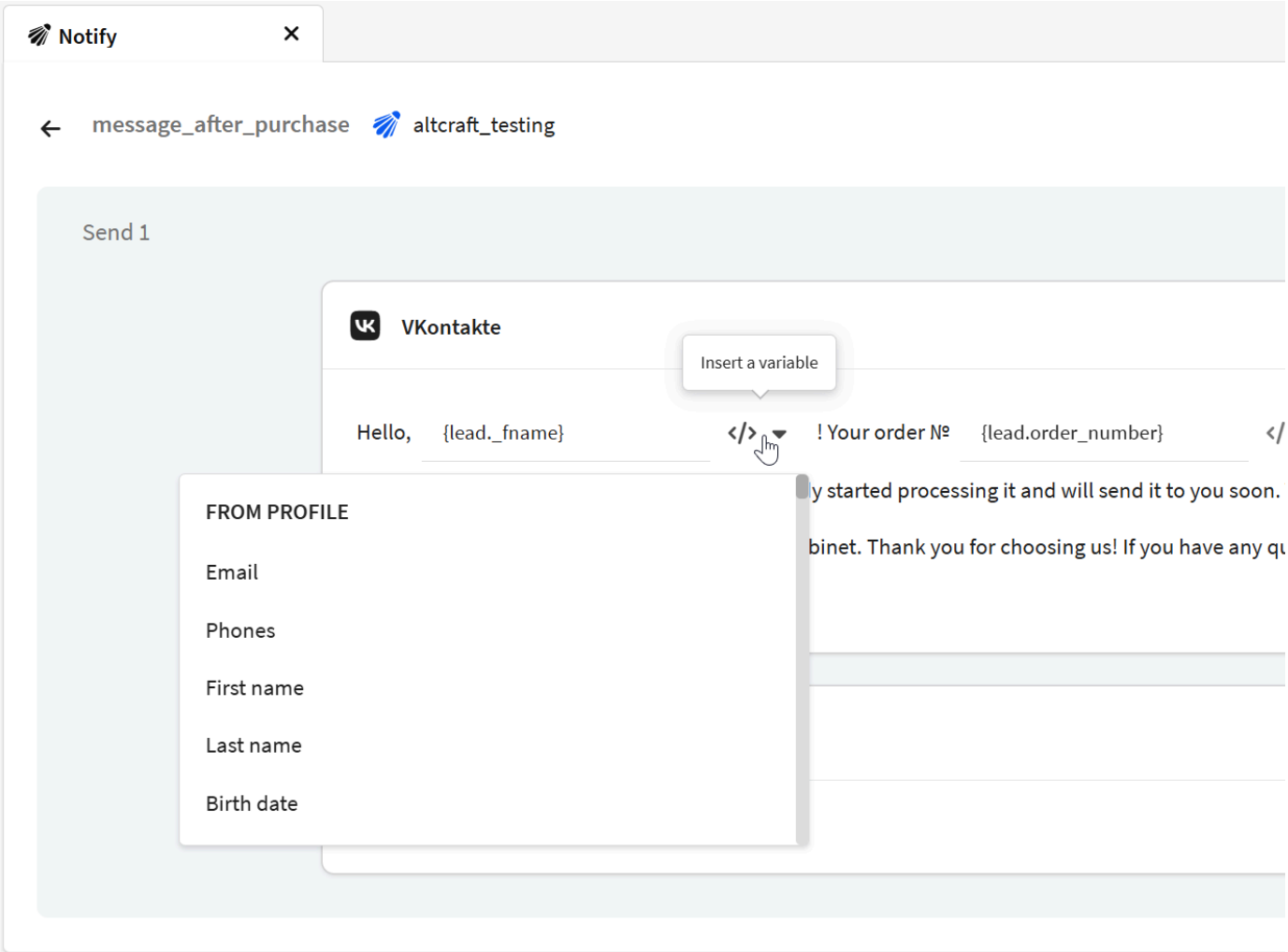
Managing parameters

The text block in Notify consists of fixed text and parameters. Parameters are template text that change
address, event name, date and time, link, etc.).

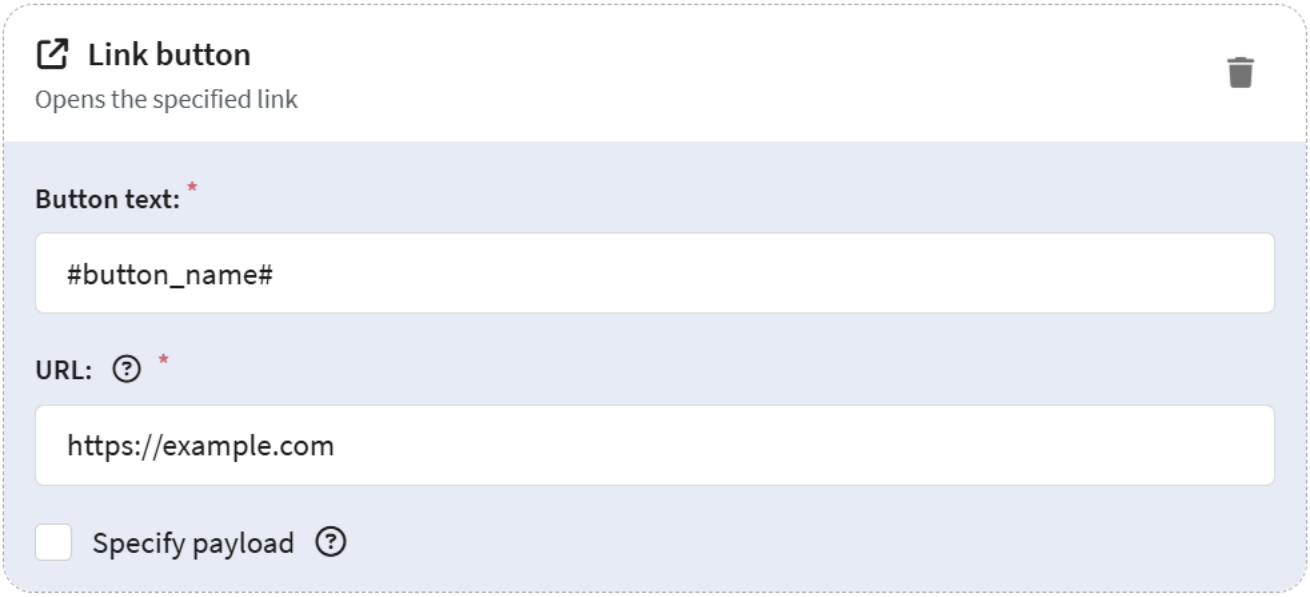
A parameter can be added to the text block in two ways: manually by typing `#parameter_name#`, where `param`
name, or by using the +Parameter button, which will automatically insert it into the text:



You can manually prescribe parameter values or add Altcraft variables. To open the list of variables, pres
the entered text will be automatically substituted into the template.

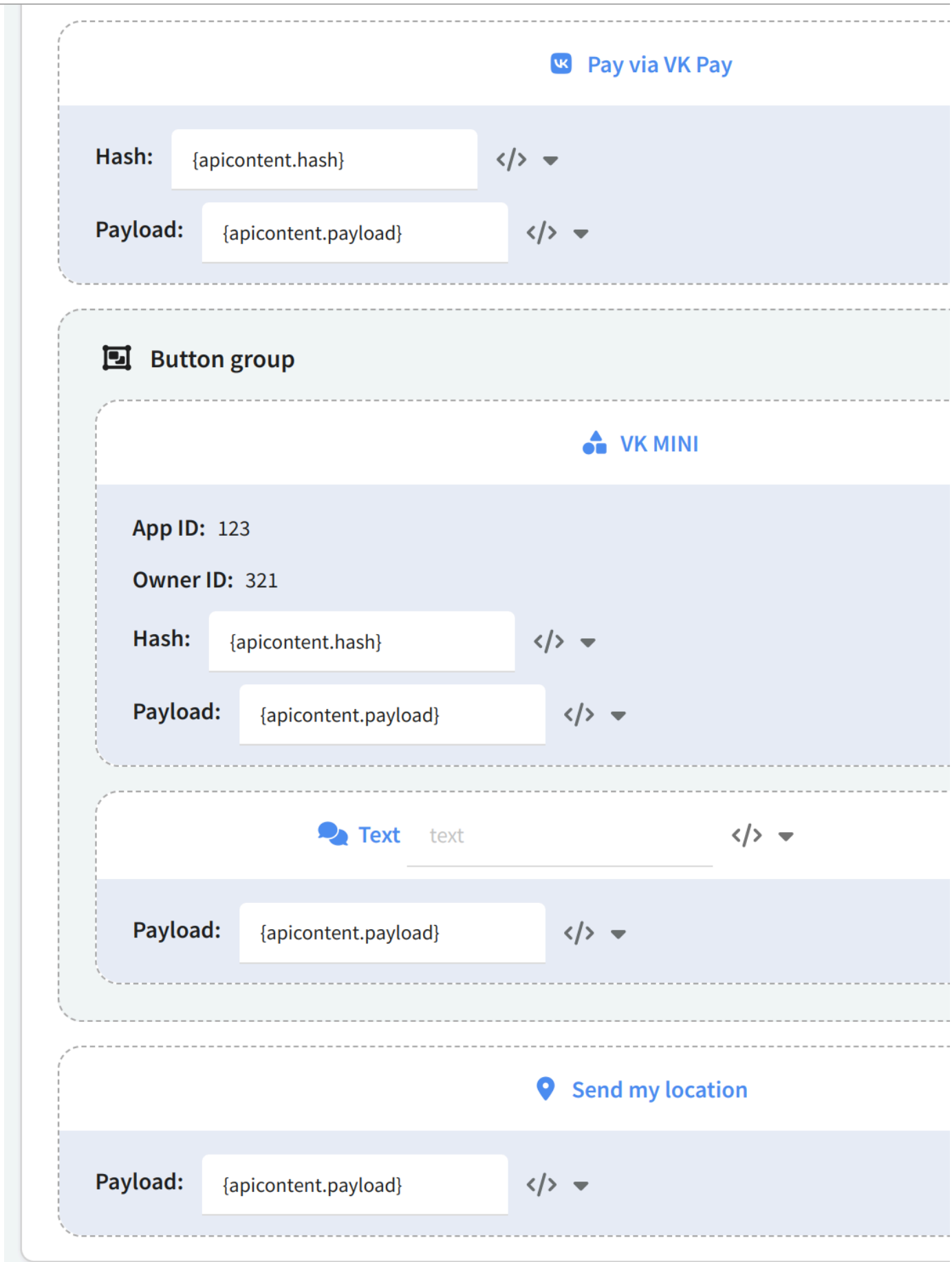


Parameters can be used not only in the Notify text block, but also in elements such as buttons:



Template Preview

After selecting an approved template, you can specify the required values or variables from the platform

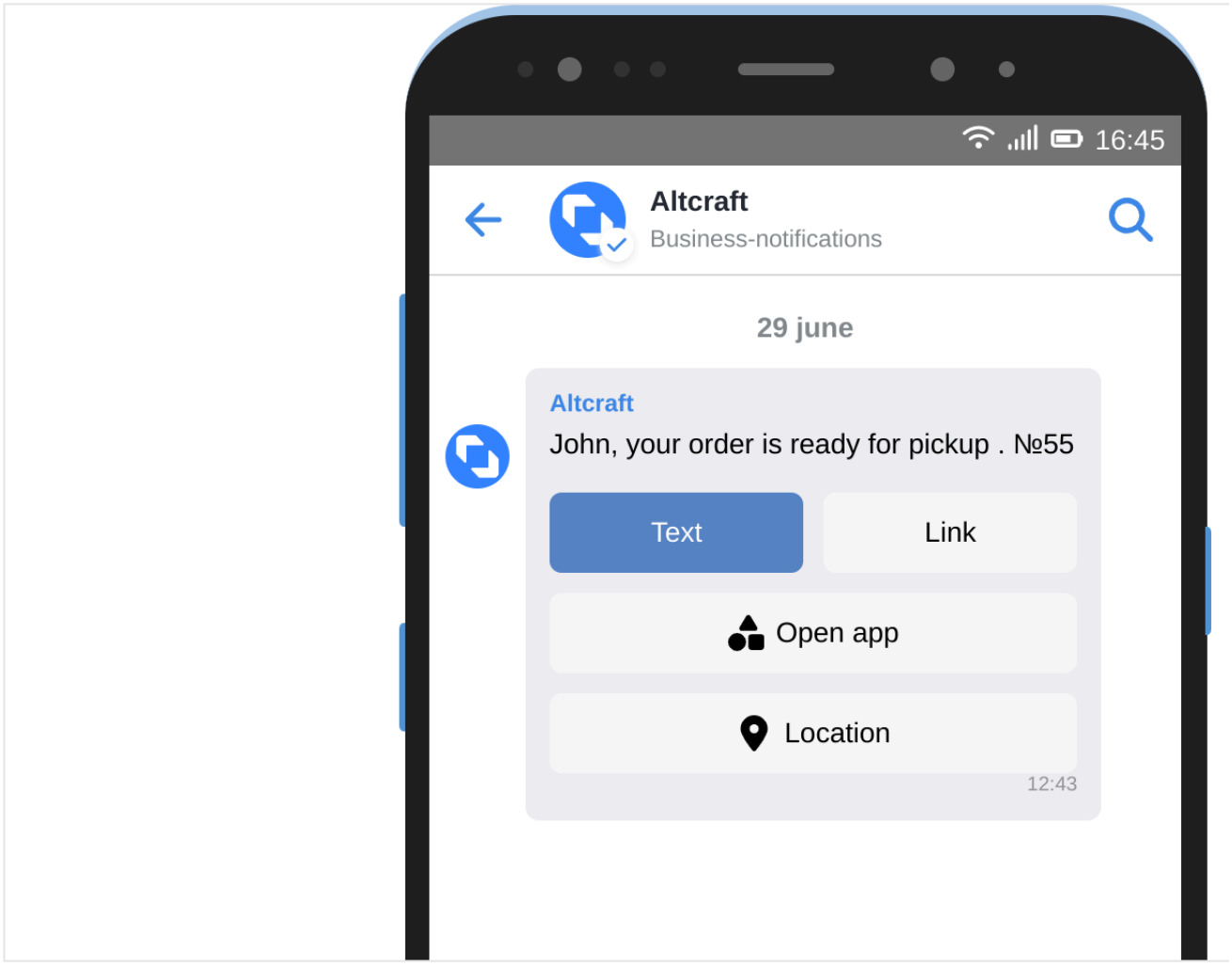


To see how the user will see the received message, click the preview button at the top of the screen:

Preview for Notify

Channels:

Notify



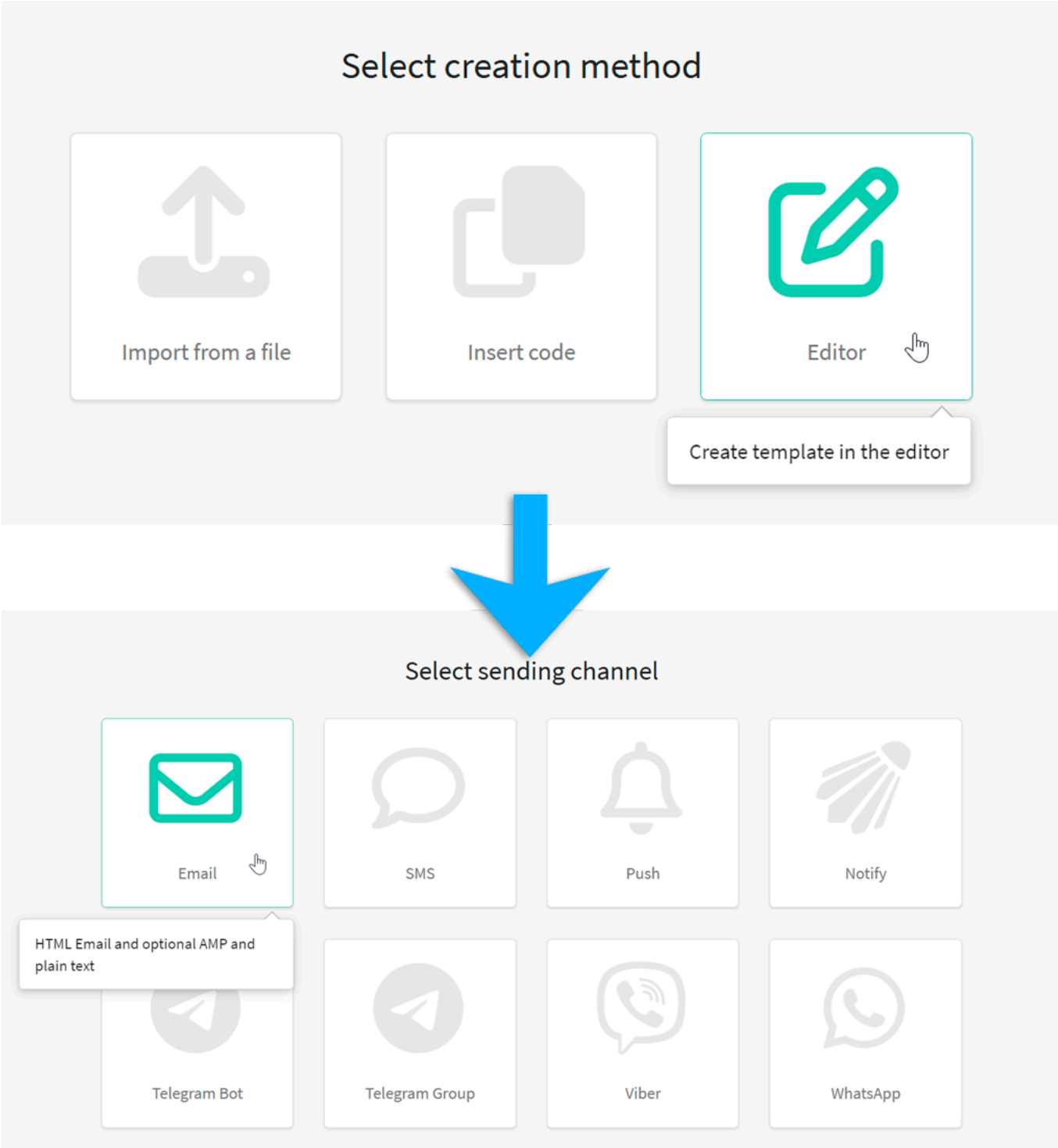
Don’t forget to save the template in the platform after making all the changes. You can now use this tem

Visual editor for email-template

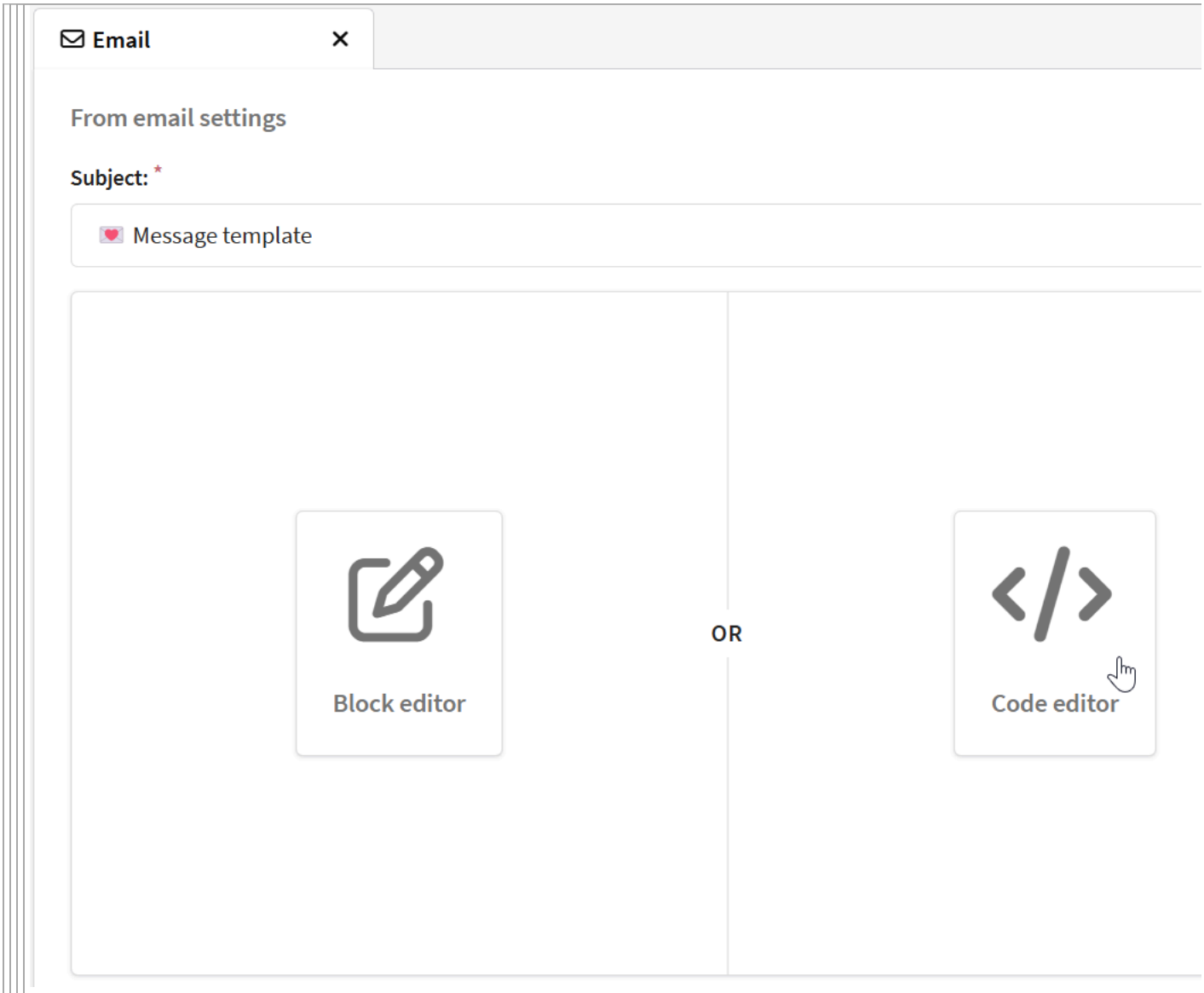
Visual editor for email-template is a tool for creating, editing and customizing marketing email templates of designing attractive and functional emails. Thanks to its user-friendly interface, users can easily cust their needs, even without programming knowledge. Among the key advantages of the editor are the abili design for different devices and a wide range of ready-made elements that can be easily combined.

How to get started in the editor?

In the main menu, open Templates and create a new message template. Select Editor —> Email.



You will see an email template customization window, select Code Editor.



You will find yourself in the HTML code editing window, in it click on the Visual Editor button.

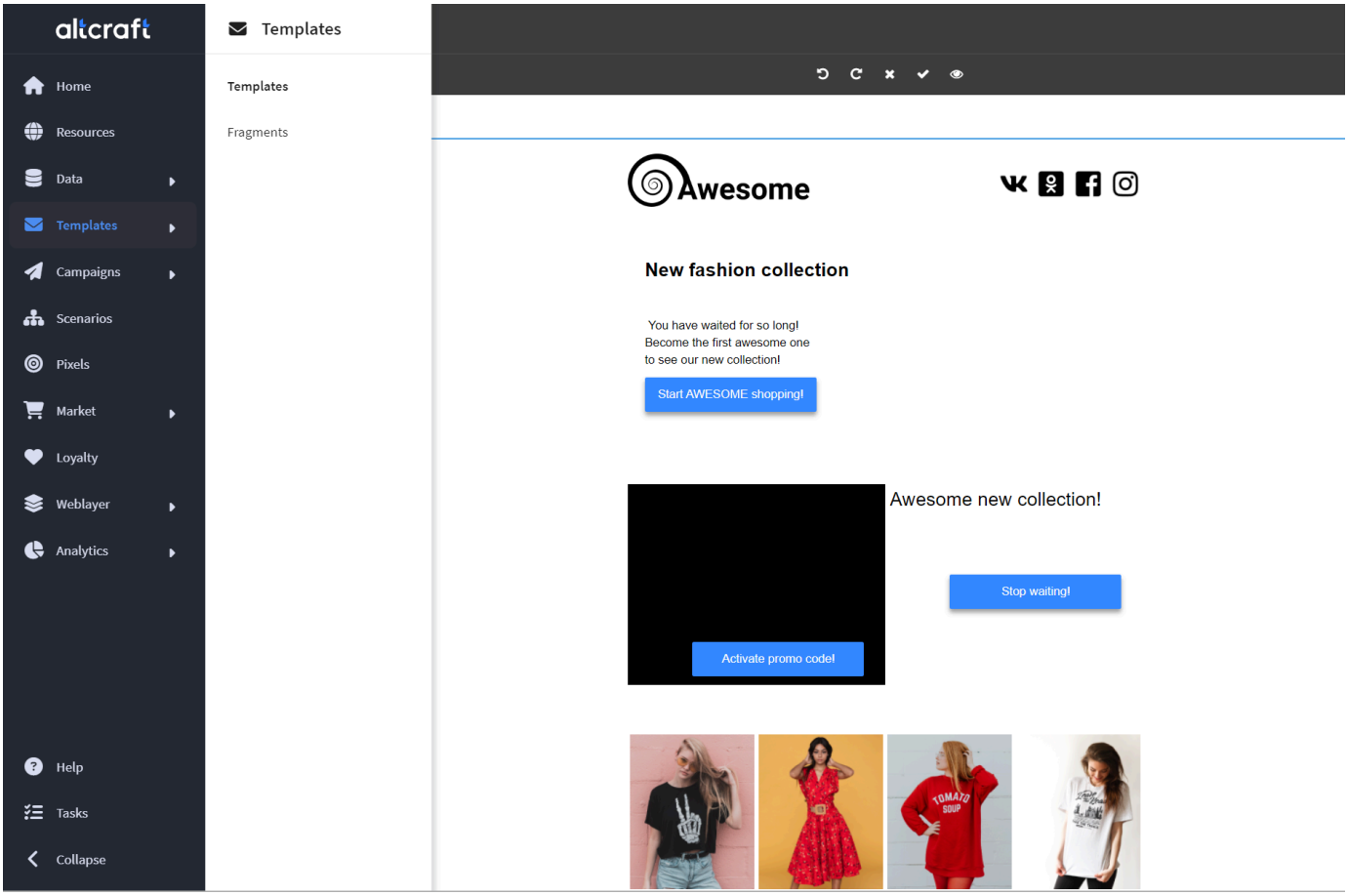
HTML

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microso
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="x-apple-disable-message-reformatting">
    <title>Template editor capabilities</title>
    <style>
        html,
        body {
            margin: 0 auto !important;
            padding: 0 !important;
            height: 100% !important;
            width: 100% !important;
        }
        * {
            -ms-text-size-adjust: 100%;
            -webkit-text-size-adjust: 100%;
        }
        div[style*="margin: 16px 0"] {
```

Read more about working with the visual template editor in the following articles:

Visual editor interface

The editor divides the page into two parts. In the central part, you see your template. Here you can selec
side, there is a panel where you can work with template elements: select the required elements and cha



Above the central part of the editor is the toolbar.

Using buttons in its left part, you can change the preview mode (desktop, tablet, mobile) to check your t
leaving the editor:

Buttons in the central part of the toolbar can be used to:

- **undo or redo the last action;**
- **discard or save changes in the template;**
- **open the message preview (you can select a test profile to know how exactly this profile will see you**

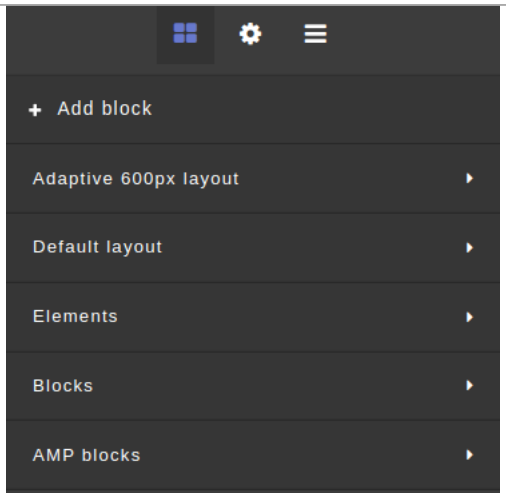
The buttons in the right part of the toolbar:

— view components: use it to see element borders;

— full screen.

The right panel of the editor contains three tabs:

1. **Blocks**
2. **Style manager**
3. **Layer manager**



💡

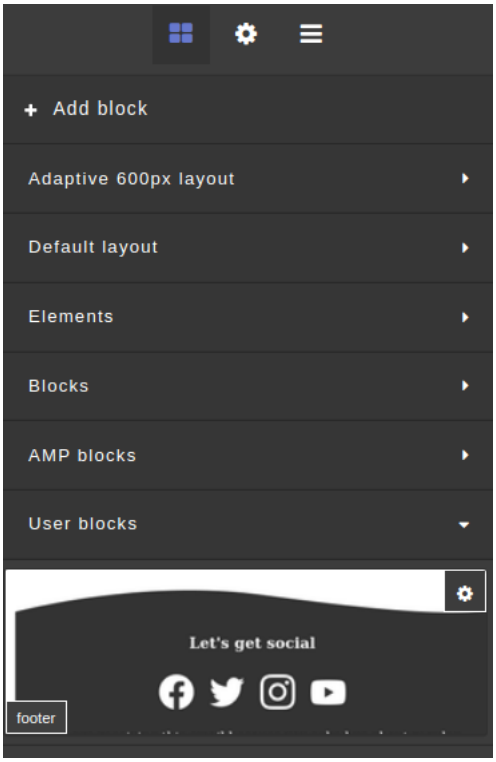
Our next step is to learn how to [add elements or blocks] to the template.

💡

Adding blocks

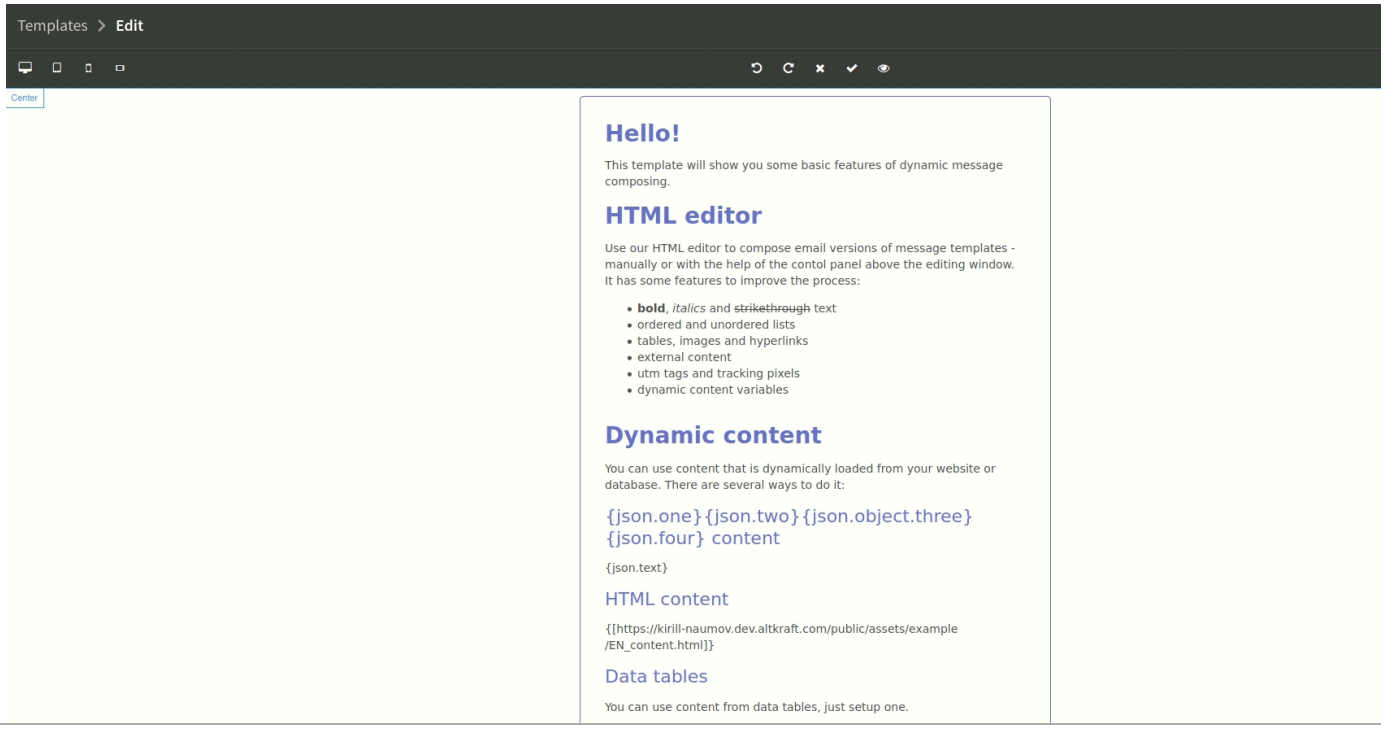
Available elements

Element	How to use
Adaptive 600px layout	**Several table rows, divided to cells with fixed maximum width. It can be used for main Email template sections with standard 600px width.**
Default layout	**Several table rows, divided to cells in percent ratio without width limit. It can be used in blocks limited by template width and sections for full screen.**
Elements	**Separate layout elements, like heading, divider, banner, link or button.**
Blocks	**Sets of elements for quickly creating template parts. These elements can be text with heading and image, menu, or social media links.**
AMP blocks	**Blocks for interactive AMP messages. “Carousel” and “Accordion” blocks are supported.**
User blocks	**You can save template elements and sections as custom blocks. They will be available in the visual editor for each template, like message fr To transfer message section to custom blocks, click “+ Add block” and copy the fragment content to the pop-up window.**



How to add blocks to templates

To add block to the template, just drag and drop it to the place you want. The parental block will be high element will be highlighted in green:



How to manage elements

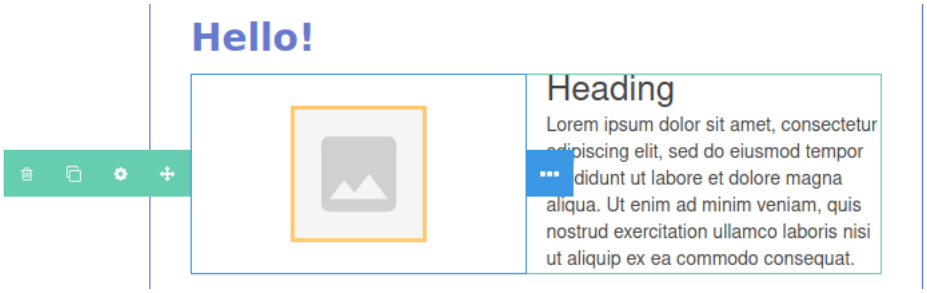
The content of the message in the visual editor consists of elements (text, headers, images, video) and t elements are placed, as in containers).

General operations are divided in 2 menus:

- **Container operations — can be displayed by clicking on the green icon at the left part of layout conta**

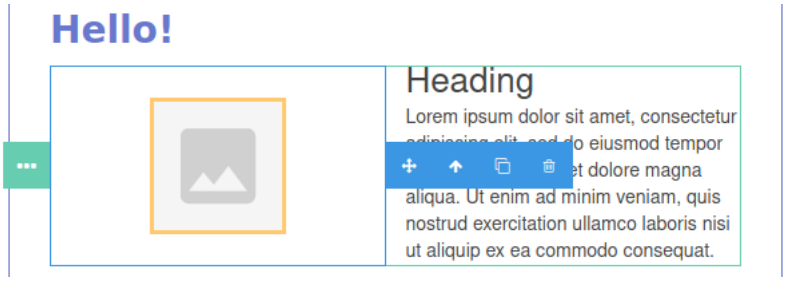
- **Elements operations menu — can be displayed by clicking on the blue icon at the left part of layout c**

When you add a new table cell or a new container, operations menu will be available. After tab is selecte or remove it using green buttons.

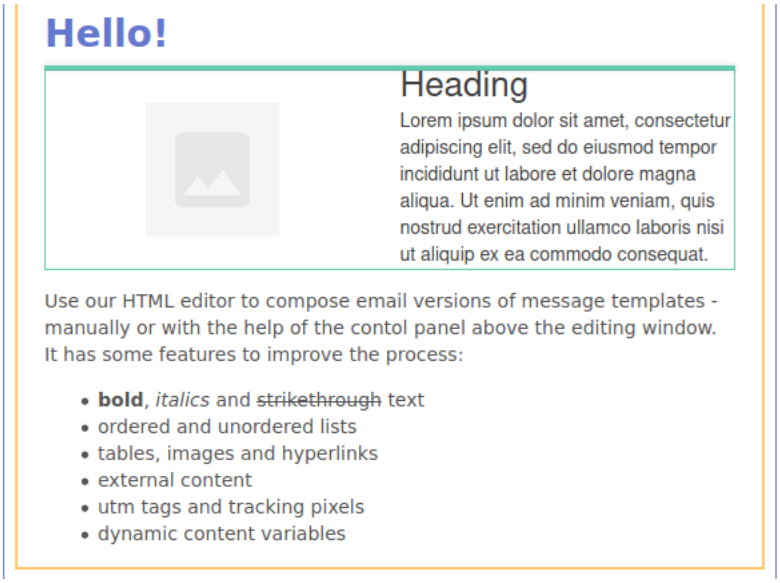


Operations menu is also available for each element inside the container. You can:

- **move an element inside a container,**
- **move it to the upper level container,**
- **duplicate,**
- **remove it, using blue buttons.**



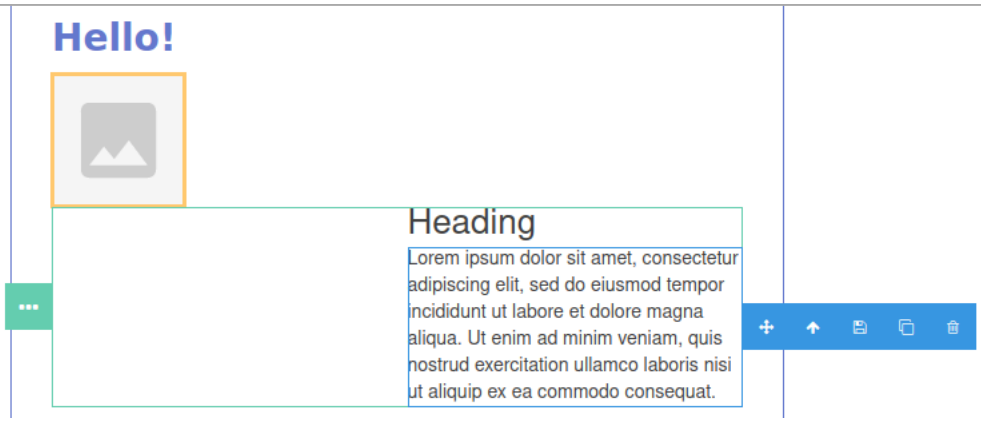
When you are moving the element, message sections is highlighted in green, parental container and othe highlighted yellow. Green line will mark a new place for an element or a container.



Elements and element groups can be saved as user blocks. To do this, click on diskette icon in the opera

TIP

To save entire sections content, select it by pressing container operations menu icon at the left part. A operations.



TIP

On the right side of the editor, for each selected element, a style menu is available. Here you can:

- **Configure the attributes and settings of the element. Attributes are described for each type of elem**
- **Configure display options for an element. CSS styles are added directly to the HTML tags to improv email clients. Styles are described in a separate article.**

Now let's take a closer look at each element.

Elements and their settings

Adaptive layout

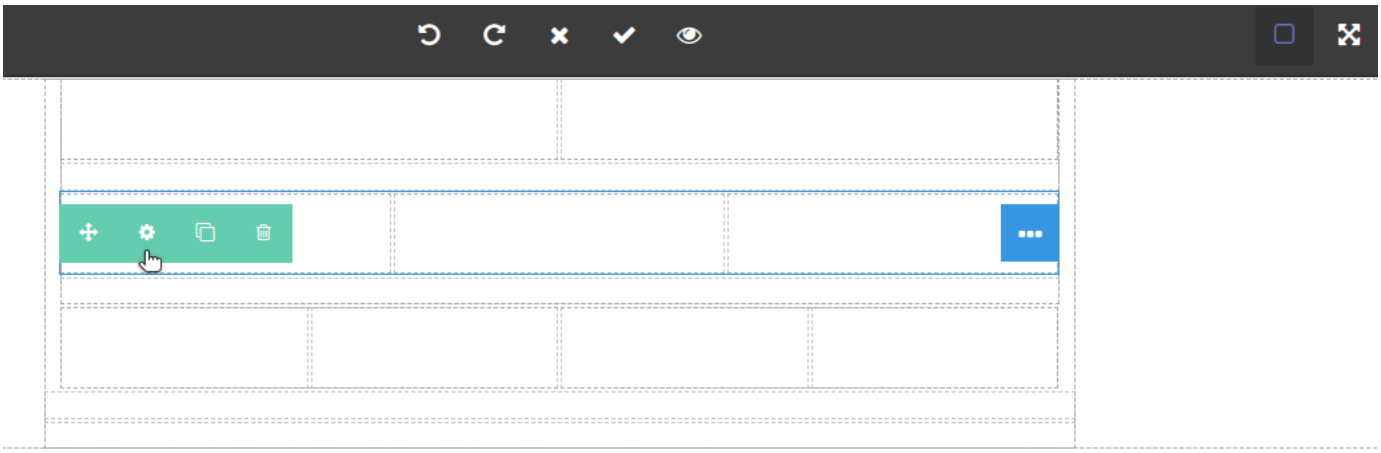
The following attributes and settings are available for adaptive layout strings:

- **HTML tag**
- **Element ID**
- **Title on hover**
- **Cellspacing**
- **Cellpadding**
- **Border**
- **Border color**
- **Background color**
- **Width of the section**

TIP

To see the Attributes and Settings tab, select the element of the adaptive layout, hover over the green i and click

in the pop-up window.



Adaptive layout cells are displayed as containers, for which you can set only title and identifier.

Default value

When it comes to the default layout, you can also add cells and change width percent ratio.

The following options are available fordefault layout cells:

- **ID**

- **Title**
- **Background color**
- **Width**
- **Horizontal and vertical align**

Elements

You can use WYSIWYG editor to edit the text in such elements as Heading, Text, Link and Button.

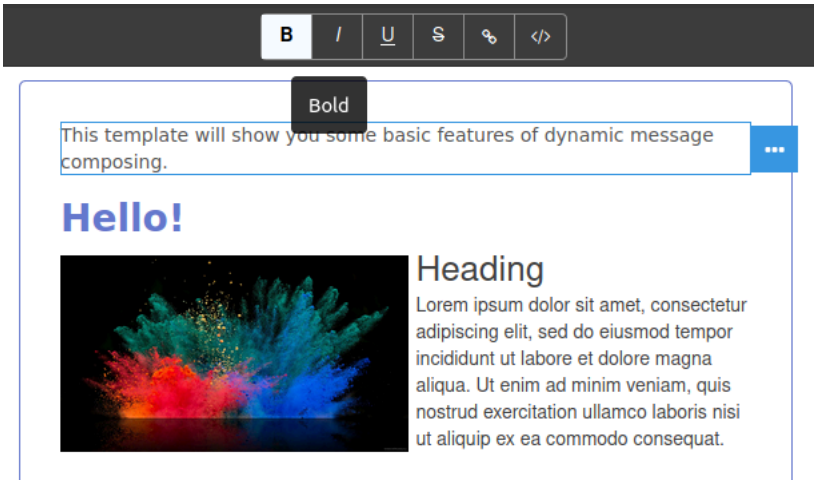
WYSIWYG editor for text elements

The WYSIWYG editor is available for text elements, like Header or Text. Click twice on the text element t of the editor.

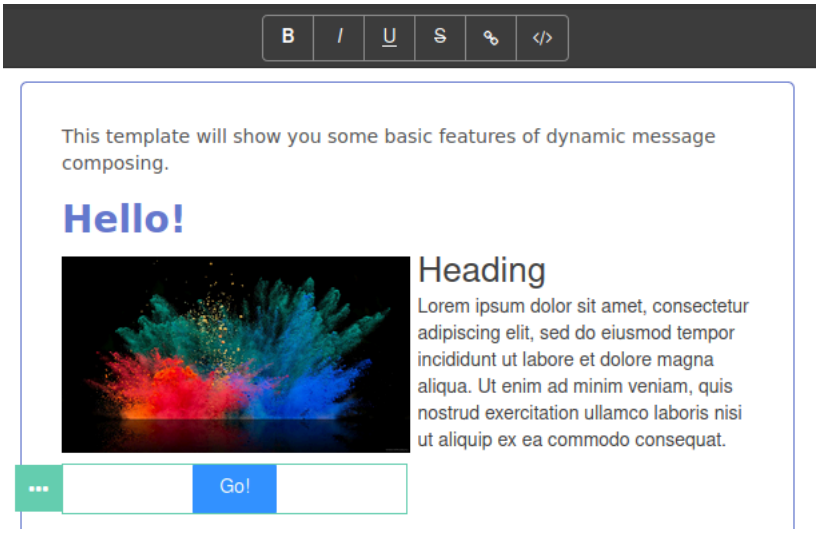
Double-click on the text element and the toolbar at the top of the editor will switch to text editor mode. element in the central part of the editor. By the way, you'll immediately see the result.

Toolbar let you:

- **format text (bold, *italic*, underline, ~~strikethrough~~),**
- **add links,**
- **add variables.**



You can edit text on the Button element in the same way:

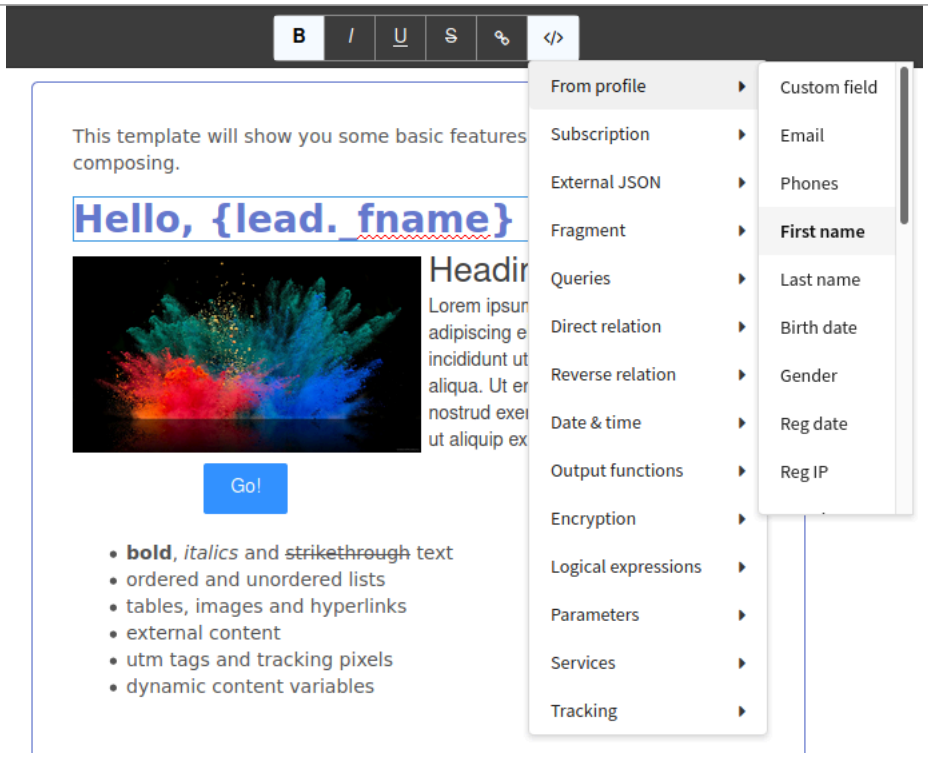


TIP

Text display options, as well as the behavior of links and buttons, are configured on the right side of the the next article.

You can use the variables menu to add profile data and logical expressions to the template.

To insert custom fields value into the template, use `{lead.custom_field}` expression (`custom_field` - full nam



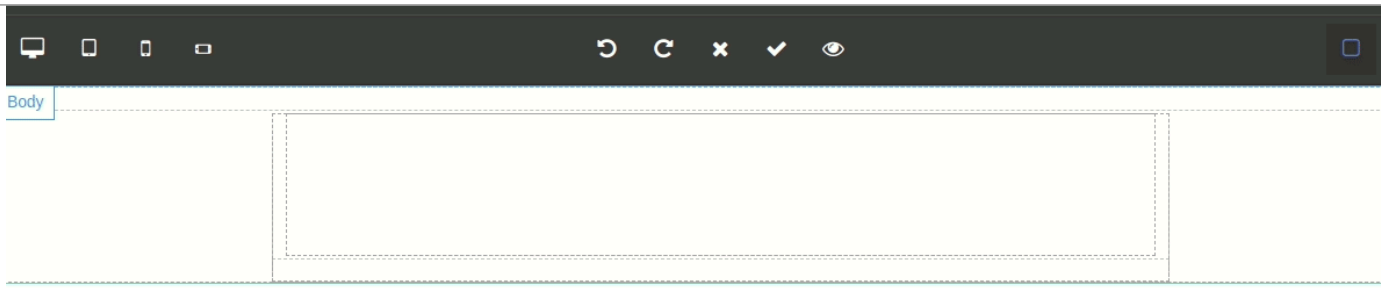
Let’s take a look at each element:

Heading

Heading attributes and settings: • **ID** • **Title**	**![img](/img/attachm**

Text

Text attributes and settings: • **ID** • **Title**	**![img](/img/attachm**



Link and Link block

Link attributes and settings: • **ID** • **Title** • **Link** • **Open link in (new window or current window)**	**![img](/img/attachm**

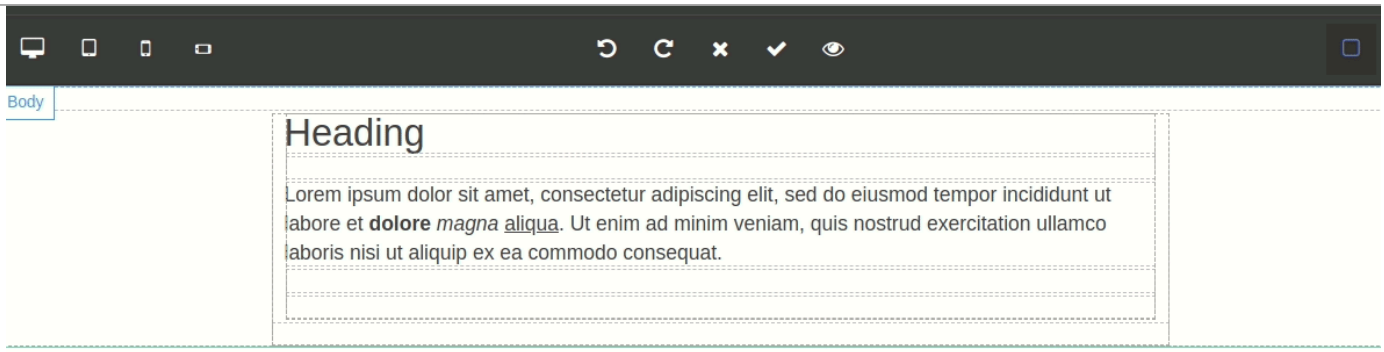


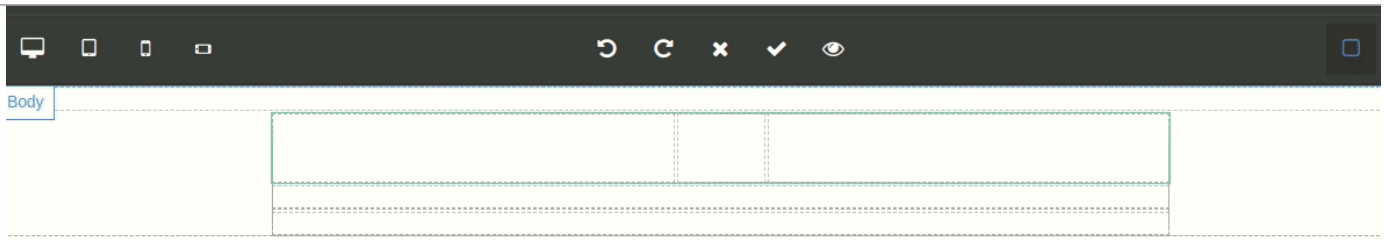
Image and Image with a link

Click twice on the element to open platform image gallery. You can use the search by file name or by tag

Using form at the top part of the gallery, you can upload your images. Just drag ad drop the file you want select image.

After you select image, open it and click Insert button. You can also add or remove tags for images.

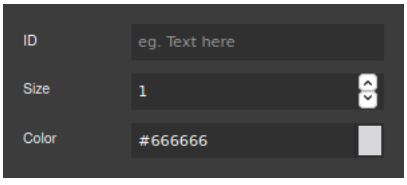
Image attributes and settings: • **ID** • **Title** • **Alternative text** • **Width and height (in pixels)** • **Image URL (in platform or network)**	**![img](/img/attachm**



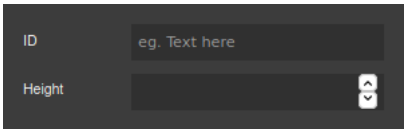
Divider and Hidden divider

Dividers allow you to visually separate items from each other.

For the divider, you can set the size in pixels and choose a color:



For the hidden divider, you can set height (in pixels):

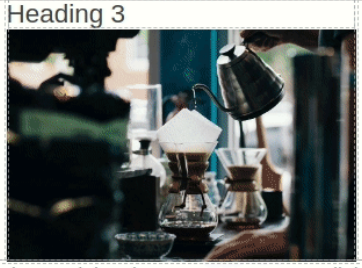


Body

Heading 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Heading 3



Heading 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Button

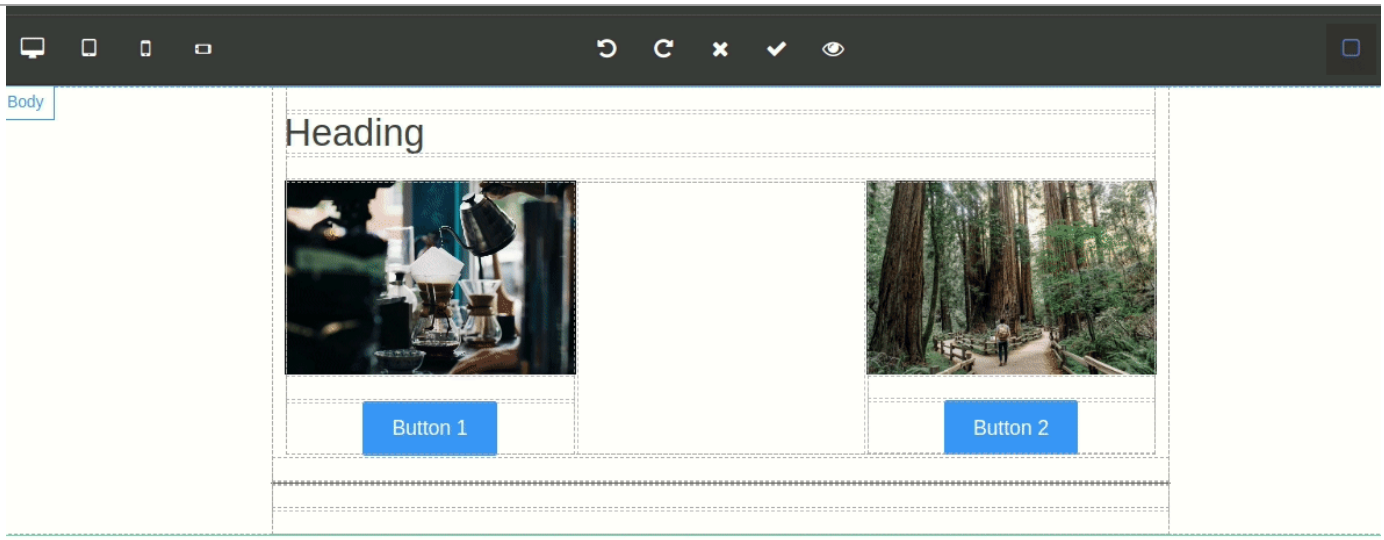
Button attributes and settings: • **ID** • **Title** • **Link** • **Open link in (new window or current window)**	**![img](/img/attachm**

Body

Heading

Video

Video attributes and settings: • **ID** • **Title** • **Video URL** • **Preview image** • **Link** • **Player width and height**	**![img](/img/attachm**



List

For an un ordered list, you can set ID and title, and select marks type (disk, circle or square):

- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitat ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

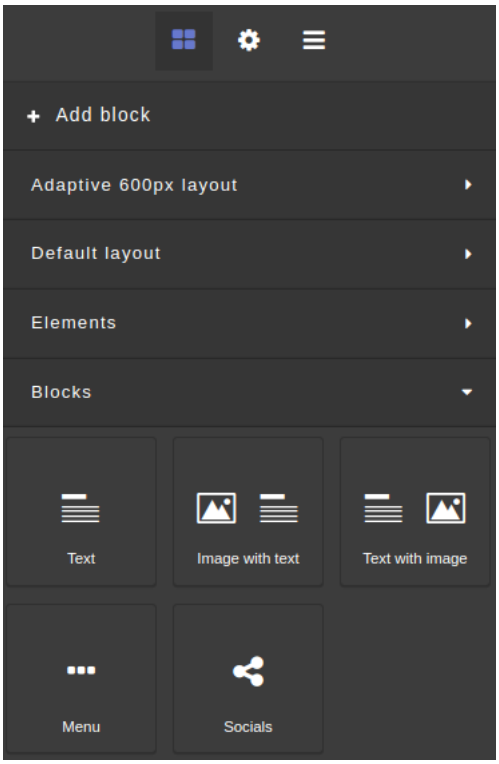
For an ordered list, you can select numeration type (numbers, Roman numbers or letters):

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ulla laboris nisi ut aliquip ex ea commodo consequat.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Blocks

Blocks are containers that already contain certain elements.

The blocks make it easier to work with the template, since they are some kind of templates for the platf degree of probability, uses in almost every email message. If you are tired of repeating the same monotc then it's time to use blocks.



Text

The Text block is a container that includes an h1 heading and a paragraph.

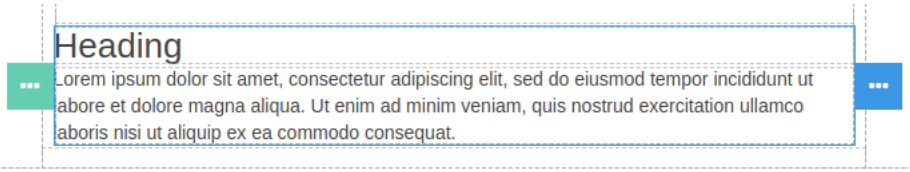
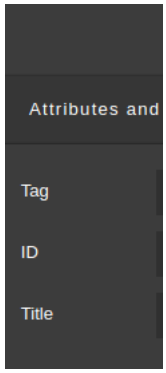
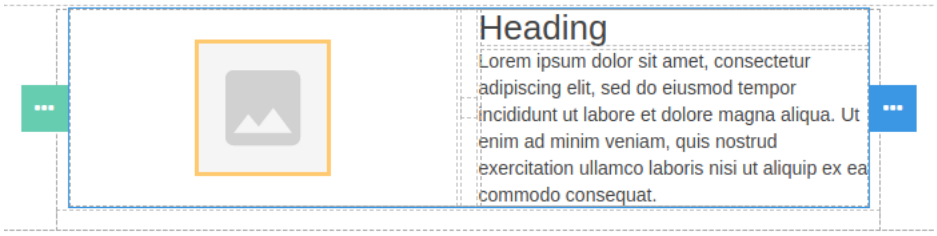
How it looks	
	

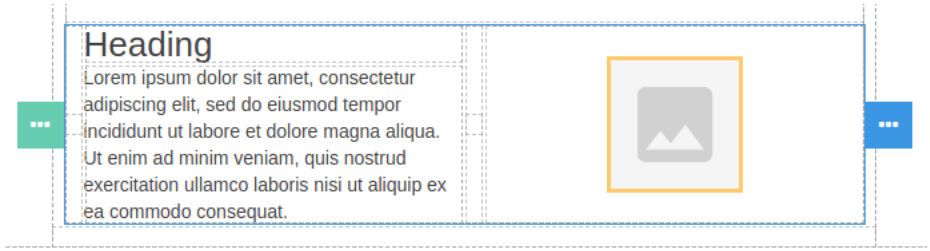
Image with text

The Image with text block is a table consisting of two cells. The cell on the left contains the image, and heading and paragraph.

How it looks	
	Attributes a Tag ID Title Cellspacing Cellpadding Border Bordercolor Background color Width

Text with image

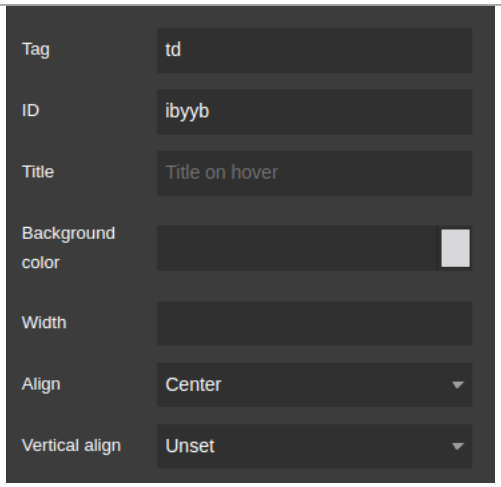
The Text with Image block is a table consisting of two cells. The cell on the left contains the h1 heading right contains the image.

How it looks	
	Attributes ar Tag ID Title Cellspacing Cellpadding Border Bordercolor Background color Width

Menu

The Menu block is a table consisting of 4 cells of the same width.

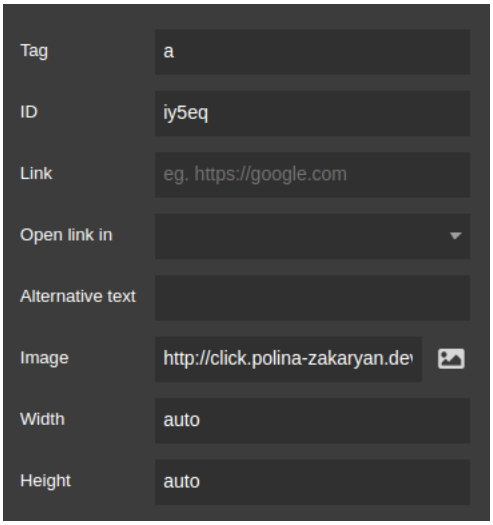
Each cell includes a link that you can customize in the Attributes & Settings panel. There you also have t inner margins between cells, adjust the border width and the total width of the element, set the backgro align.



Socials

The “Socials” block is a container that includes 8 images with links to social networks.

For each item, you can set your own link to click, add alternative text, customize the image, and change



AMP blocks

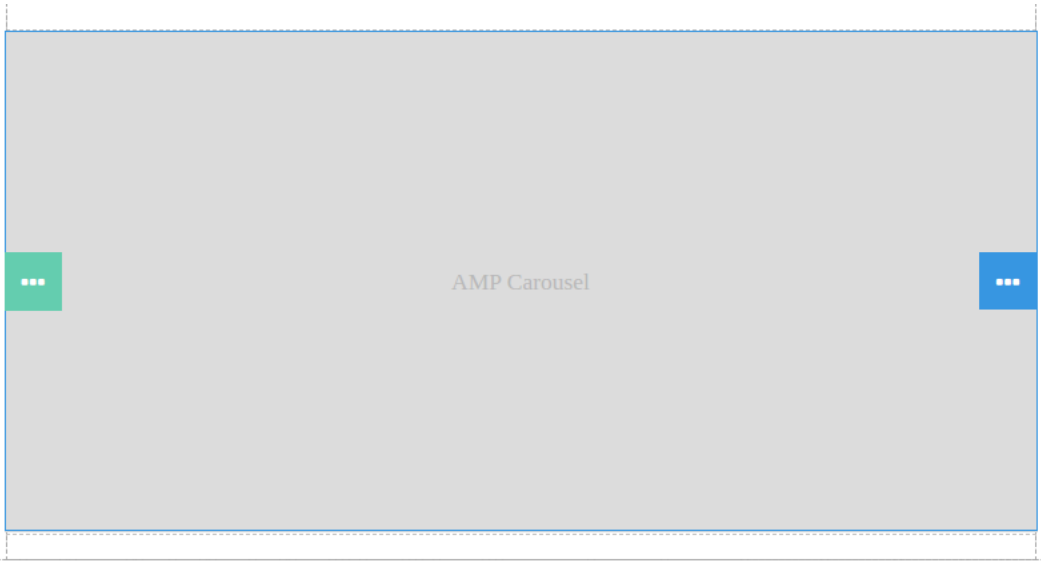
AMP emails is a technology that allows marketers to embed interactive elements such as carousel and a in turn, can view the data without even opening a new tab to visit the website.

How to use AMP blocks in the Altcraft Platform:

Carousel

AMP carousel is a block that includes several slides. A carousel allows multiple pieces of content to occ page or screen.

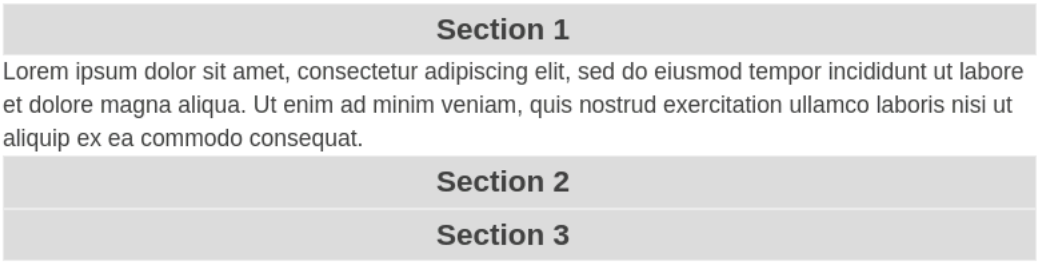
AMP carousel settings are set in the “Attributes and settings” section. You can set element ID and heigh set images and links. You can upload your image, select it from the gallery, or insert image URL:



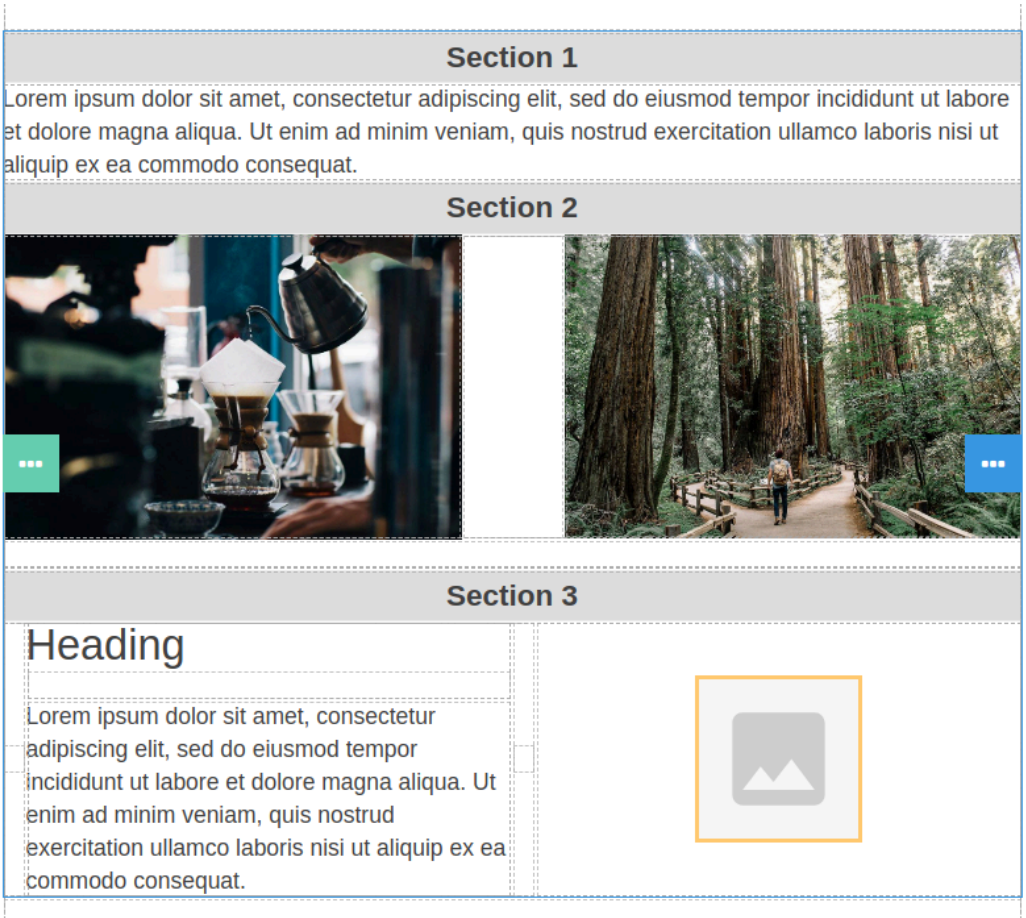
Accordion

AMP accordion is a block consisting of several tabs with elements. Each element can be “expanded” wit (click, mouseover). You can add any elements to the sections.

An interface element such as the accordion not only helps users navigate quickly through content, but a large amounts of information in a limited space.



AMP accordion settings are set in the “Attributes and settings” section. Select the block and add the des opened or closed by default.



💡 **If you’ve already figured out how to add and configure elements in the template, the last thing left is**

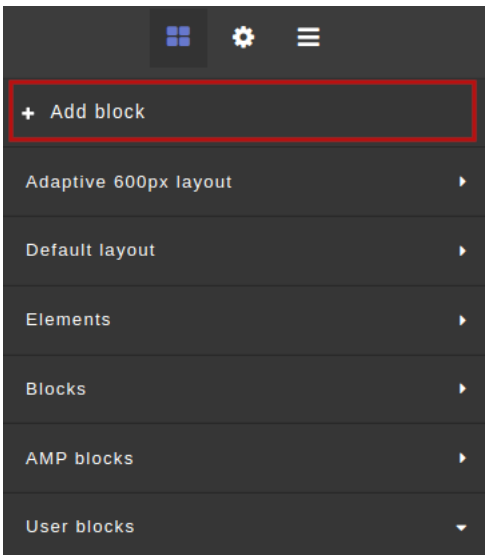
Custom blocks

Save an entire section of the template as a custom block and then use that block over and over again. B the template where it was created, but also in any other.
What can be custom blocks? Any content that repeats from template to template. For example, the head

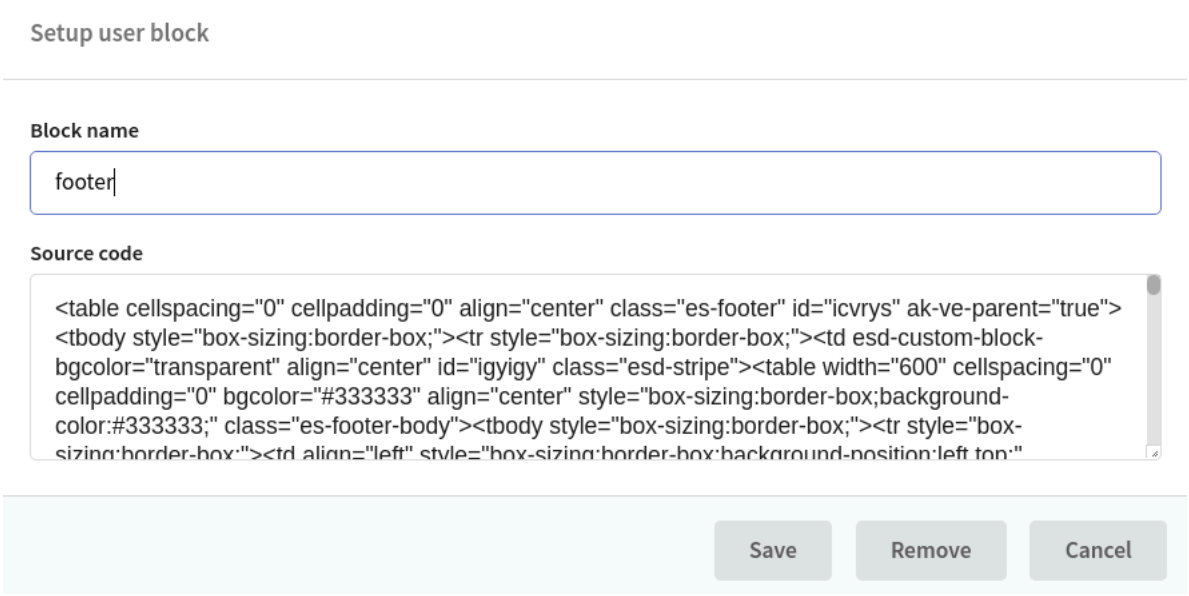
How to create a custom block

Option 1: Add source code

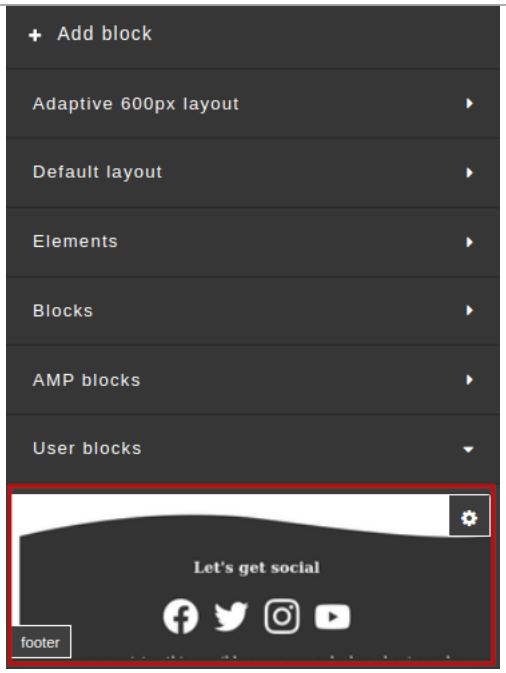
In the right panel of the visual editor, go to the “Add blocks” section. Click “+ Add Block”.



Enter source code of the fragment in the pop-up window and save the block.



The new custom block will be displayed after all available elements of the visual editor.



Option 2: Save the template container

Select the group of elements you want to save. You can simply hover the cursor over the desired contain section “Layer Manager”.

After you select the desired section, you’ll see the button for calling the menu of container operations —

. Click the floppy disk icon

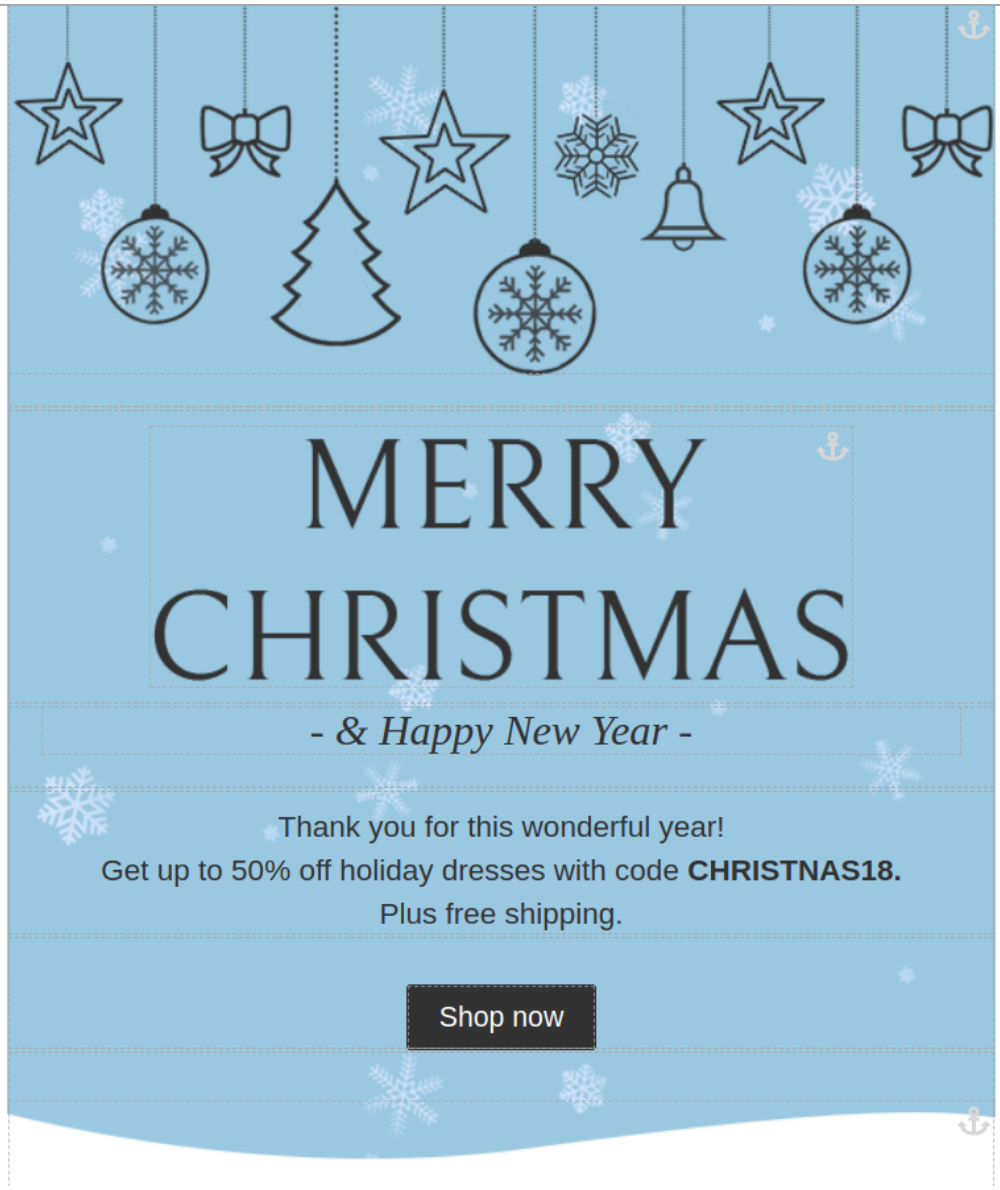
to save the block.

The new custom block will be displayed after all available elements of the visual editor.

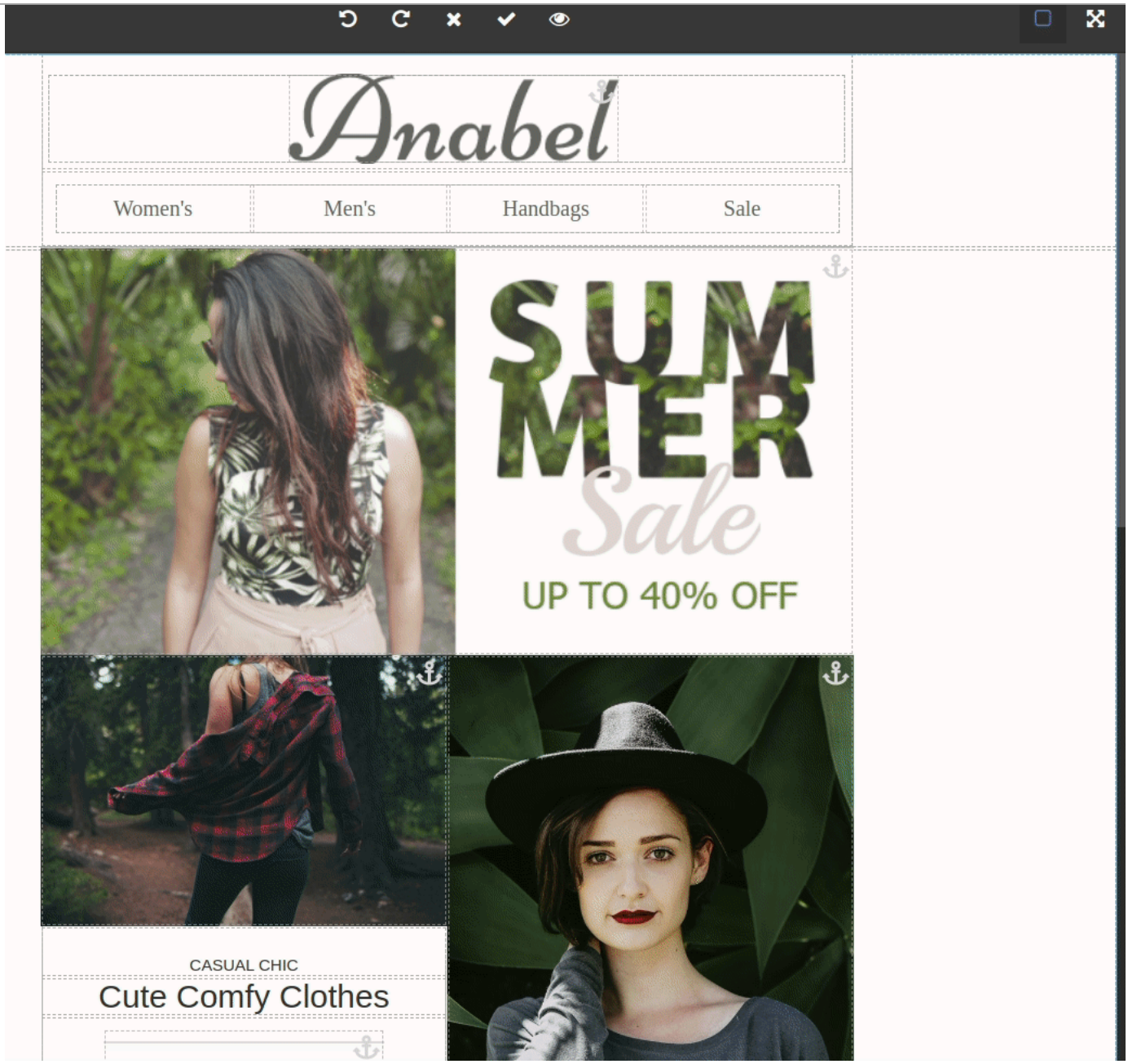
How to add a custom block

Select a custom block by clicking it, and then drag it to the part of the template you need. Let’s get into

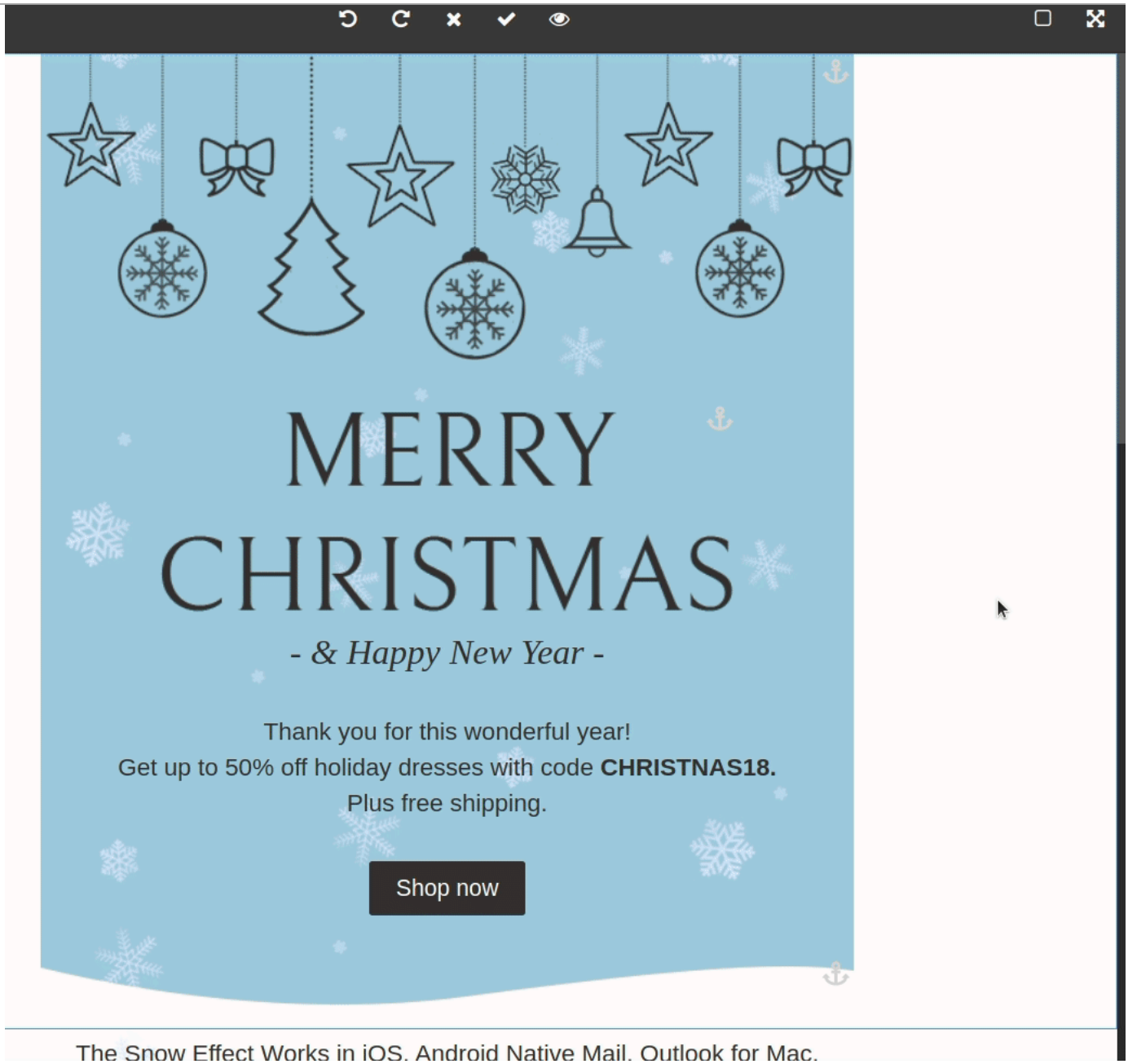
Let’s say we send emails to customers of the clothing store. Christmas is just around the corner, and we promotional code for a discount. This is how our template looks like:



Of course, the main part of the message has changed in compare with usual messages, but the header v
our contacts will remain the same as in the daily mailings. Fortunately, we have long since saved these e



Let’s add our custom blocks to the template.



Good job, the template is ready, and we saved a lot of time!

👍

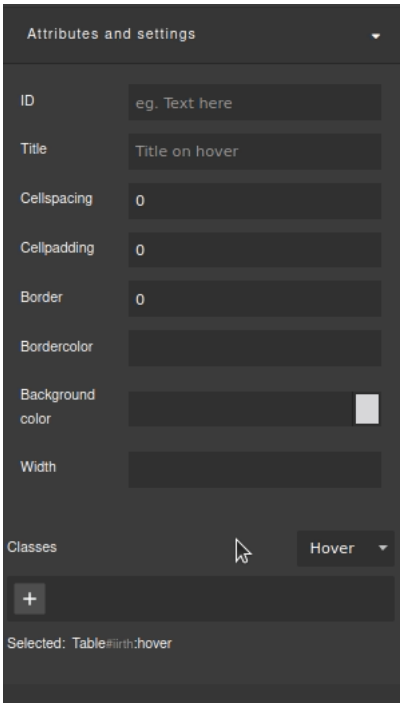
Style manager

Click on the central button in the right panel to open style manager for the selected element.

Each element has its own settings. Basic attributes are available for all elements. They are stored in the you can easily configure your template for perfect display in all mail clients.

Classes and state

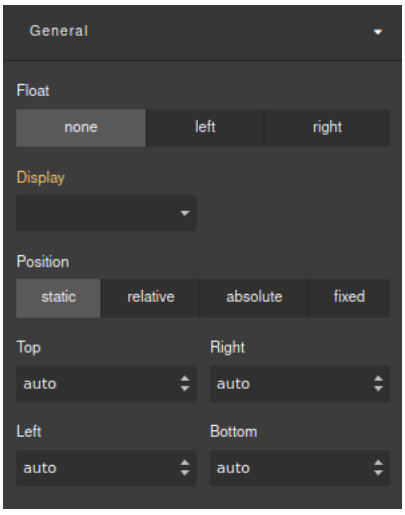
You can set class for element or configure state for different effect during click or moving cursor



General

You can set element position in the template:

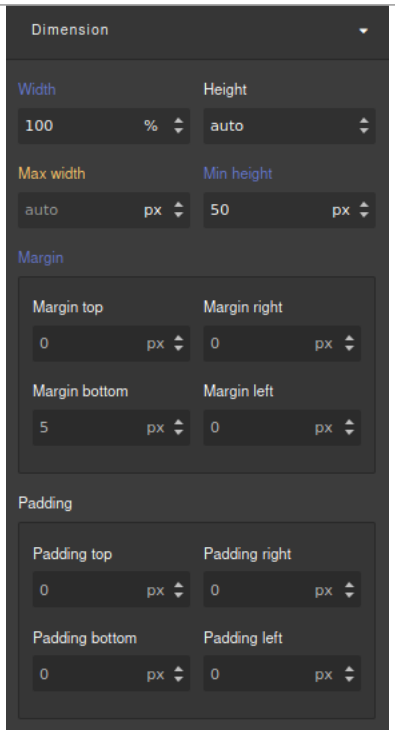
- **Float — element align by left or right side**
- **Display — inline, block, inline-block или flex — element display mode**
- **Position, Top, Left, Bottom — element position and spacing**



Dimension

In this section, you can set element display parameters:

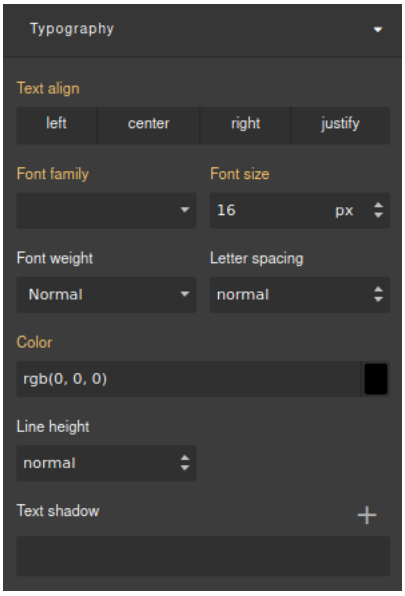
- **Width, Height**
- **Max width, Max height**
- **Margin**
- **Padding**



Typography

Text elements display parameters are situated in Typography section:

- **Text align**
- **Font family**
- **Font size**
- **Font weight**
- **Letter spacing**
- **Color (RGB or HEX)**
- **Line height**
- **Text shadow**



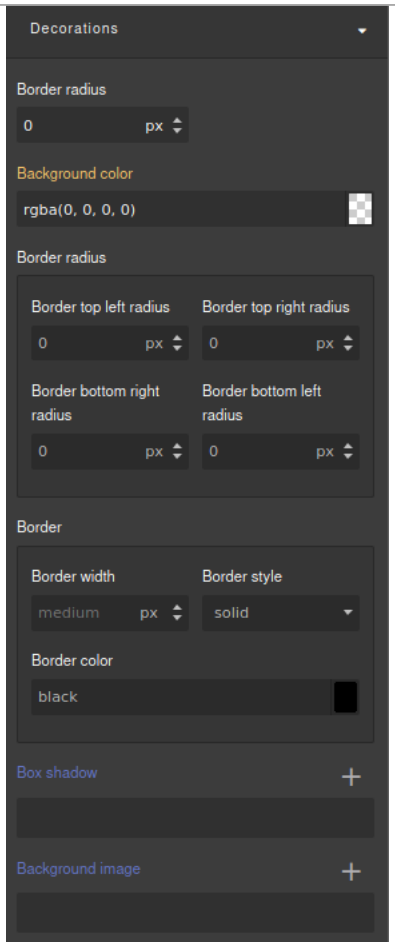
Decorations

You can set element border, background and shadows in this section:

- **Border radius**
- **Background color**
- **Border radius**
- **Border**

Box shadow — adds shadow for the element. You can configure shadow type, position, blur, spread, colo

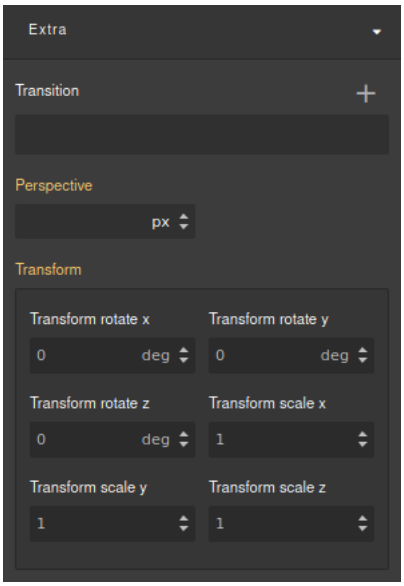
Background image — adds image as element background. You can upload your image or select it from th and repeat:



Extra

In this section, you can configure element animation during state changes. The following parameters are

- **Transition**
- **Perspective**
- **Transform rotate, Transform scale**



Layer manager

The last section in the right pane of the editor is Element Structure.

Here you will find a tree of elements (HTML DOM). Use it to get to any element in the template in a coupl useful, especially if you have to work with a template that you did not create. One look at the structure c understand the general structure of the template and adjust it without going into details.

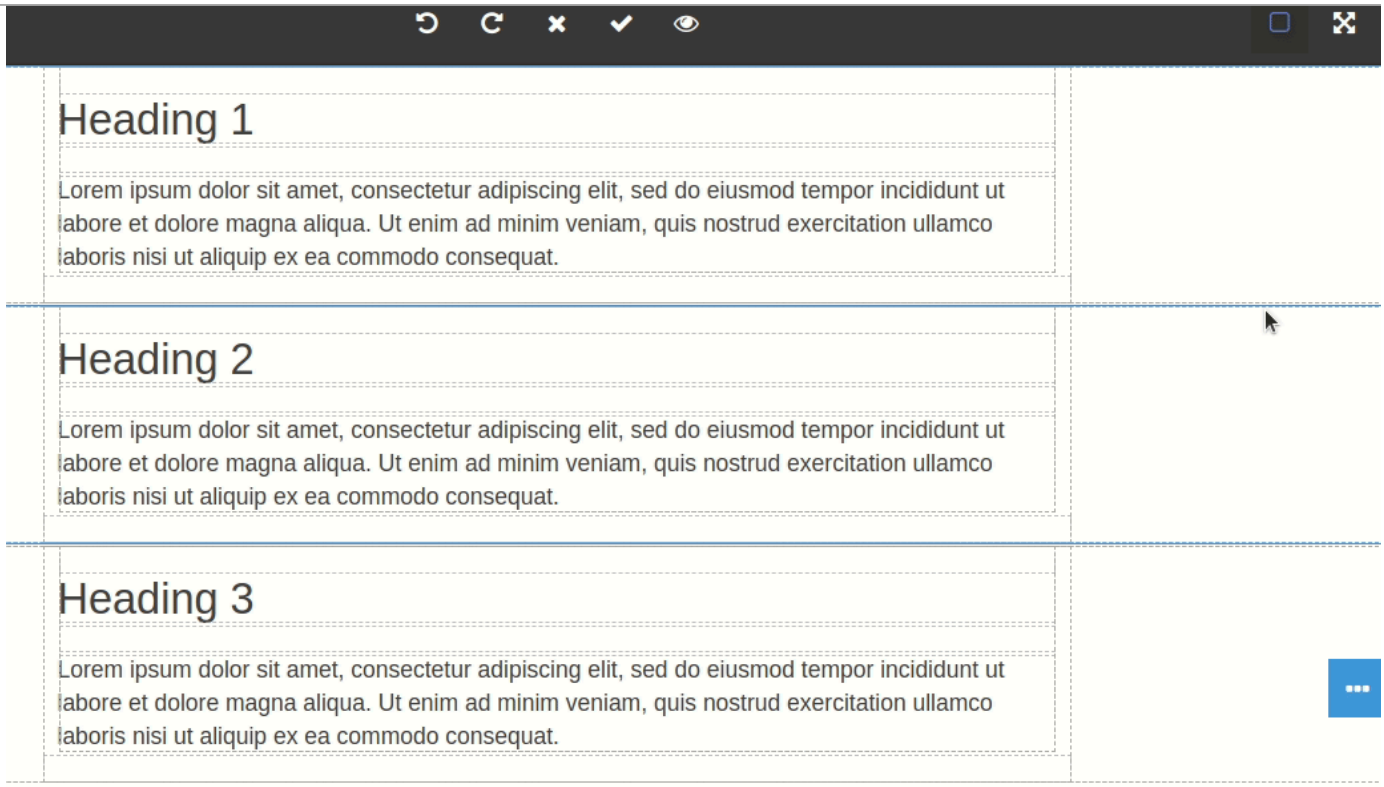
When you open the element structure, a list of all the template elements appears on the right side. Here elements are nested within each other.

The number opposite each item shows how many children the parent has.

If you hover over or click on an element, the area occupied by it in the template will be highlighted in blu

So how can we use the element structure?

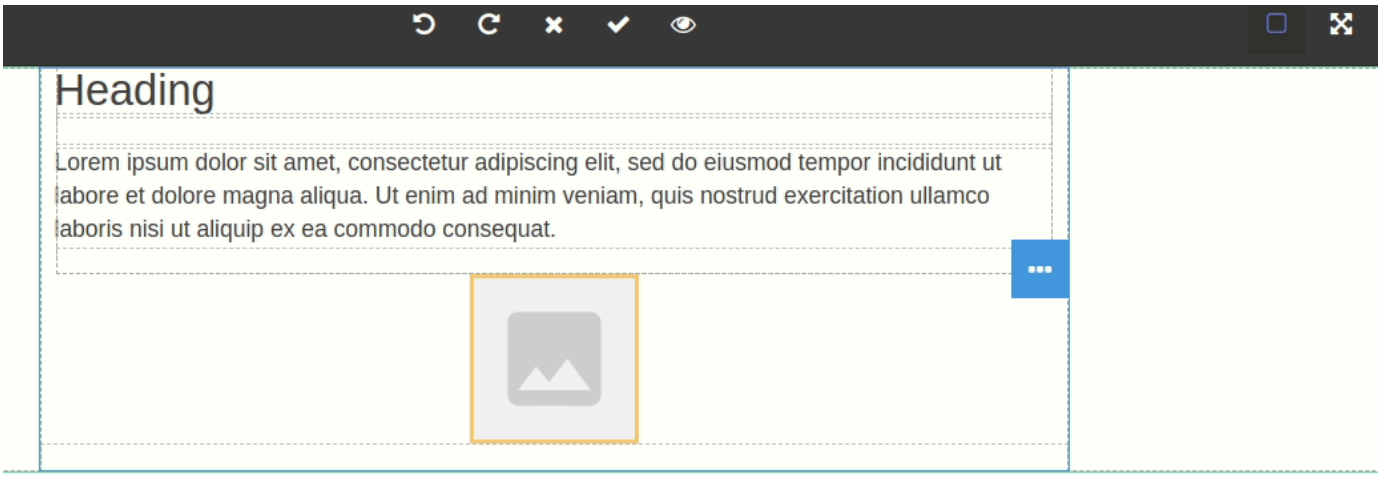
(1) Change the position of an element or the entire container. Drag and drop the element to its new posit



(2) Hide elements. Perhaps now you don't need a certain element of the message or a whole block, or yo look without it. Click

to hide the element. If you change your mind and want to return the element, click

.



Altcraft Variables and Functions

Profile Data

Description	Va
Email address	`{lead.email}`
Phone numbers	`{lead.phones}`
First name	`{lead._fname}`
Last name	`{lead._lname}`
Date of birth	`{lead._bdate}`
Gender	`{lead._sex}`
Registration date	`{lead._regdate}`
Registration IP	`{lead._regip}`
Registration city	`{lead._regcity}`
Registration country	`{lead._regcountry}`
Registration URL	`{lead._regurl}`
IP address	`{lead._ip}`
City	`{lead._city}`
Country	`{lead._country}`
Region	`{lead._region}`
Postal code	`{lead._postal_code}`
Time zone	`{lead._tz}`
Provider	`{lead._vendor}`
Profile ID	`{lead._id}`
XXM hash	`{lead._xxm}`
XX hash	`{lead._xxh}`
MD5 hash	`{lead._md5}`
*** Custom field**	`{lead.custom_field_name}`

Subscriptions

Subscription variables contain data related to a specific subscription and the delivery channel being use pass a phone number when sending via email using these variables.

Description	Variable
Subscription email	`{subscription.email}`
Email domain	`{subscription.email_domain}`
List of subscription emails	`{subscription.emails}`
Subscription phone	`{subscription.phone}`
List of subscription phones	`{subscription.phones}`
List of push subscriptions	`{subscription.pushes}`
MD5 hash of email	`{subscription.email_md5}`
MD5 hash of phone	`{subscription.phone_md5}`
Subscription hash	`{subscription.hash_id}`
Subscription field (by channel)	`{subscription_field("channel" "field_name" resource_id)}` **channel — string indicating the channel (“email”)** **field_name — name of the field to insert (“domain”)** **resource — resource ID (e.g., 10)**

The `subscription_field` function lets you insert fields from subscriptions regardless of the channel used by if a profile has an SMS subscription, you can insert their phone number into the email version of the mes

`{subscription_field("sms" "phone" 1)}`

Available fields:

Channel	Field	
email	**email**	**Subscription email address**
	domain	**Email domain of the subscription**
	domain_group	**Domain group**
	md5	**MD5 hash of the subscription email**
sms	**phone**	**Subscription phone number**
	md5	**MD5 hash of the subscription phone**
push	**subscription_id**	**Push subscription ID**
	bundle_id	**iOS APNs app bundle ID**
	provider	**Push provider**
telegram_bot/telegram_group	**id**	**Chat ID**

TIP

Similarly, you can insert data from custom channel subscriptions into templates. Fields of the main cha you set when creating the channel.

Additionally, the `subscription_field` function can access additional subscription fields created in the reso additional field name as `field_name`:

`{subscription_field("email" "custom_field_name")}`

UTM

If a profile has UTMs in their card, you can insert them into the message template.

Description	
Mailing UTM	`{lead._utm_campaign}`
Traffic source UTM	`{lead._utm_source}`
Medium UTM	`{lead._utm_medium}`
Content UTM	`{lead._utm_content}`
Keyword UTM	`{lead._utm_term}`

External JSON

External JSON content is a way to create dynamic messages and automate work. It lets you add informa the moment each message is sent.

Read more in "Using JSON content in messages".

In message text, JSON content appears as variables like `{json.variable}`. Data types `list` and `object` sup access elements by key: `{json.object.key_name}`.

Market

Using market variables

To use order and product information in mailings, add market variables to your message template. See "I for details.

Date and Time

Date and time variables

Description	Variable	Example
Formatted date	`{format}`	`{format datenow "%Y-%m-%D %H:%M:%S timezone, %I am"}` **2018-12-25 18:56:10 +03:00, 6 pm**
Raw date	`{datenow}`	**2018-12-25T18:56:10+0300**
Date offset	`{adddate()}`	`{adddate(datenow 0 0 3)}` **2018-12-27T18:56:10+0300**
Full year	`{yearlong}`	**2018**
Year	`{year}`	**18**
Month	`{month}`	**12**
Day	`{day}`	**10**
24-hour format hours	`{hours}`	**18**
12-hour format hours	`{hours12}`	**06**
AM/PM	`{ampm}`	**PM**
Minutes	`{minutes}`	**56**
Seconds	`{seconds}`	**10**

Output Functions

Output functions

Description	Variable	Example
Number formatting	`format`	`{format 1000000000 ","}` **1,000,000,000**
Variable length	`length`	`{length("abc")}` **3**
Array of values	`array`	`{array[1 "2" 3.3]}` **[1 2 3.3]**
Random array element	`randomize`	`{randomize(array[1 2 3])}` **1** **Example: a profile custom field `Favourite_genres` stores `["Detective", "Humor", "Sci-Fi"]`. `{randomize(lead.Favourite_genres)}` will output one rando of Objects, as well as arrays in API content and external JSON content.**

Description	Variable	Example
Limit array	`limit`	`{limit(array[1 2 3] 2)}` **[1 2]**
Number is even	`iseven`	`{iseven(2)}` **=> true** `{iseven(3)}` **=> false**
URL encode	`urlencode`	`{urlencode(lead._regurl)}` **http:%2F%2Ftest.testdomain.com%2Fsomething** **Use to encode invalid characters in URL parameters.**
URL decode	`urldecode`	`{urldecode(lead._regurl)}` **http://test.testdomain.com/something**
Uppercase	`uppercase`	`{uppercase("text")}` **TEXT**
Uppercase first letter	`uppercasefirst`	`{uppercasefirst("john")}` **John**

Encryption

Encryption and hashing

Template variable encryption is used when you need to pass data along with a click to a target site but d server logs or be visible to the user. For example, form prefill data can be encoded in base58 and decode

To pass template content and variables in encrypted or hashed form, use the `crypt` function. Click the `</` choose Encryption.

```
{crypt(algorithm field key)}
```

You can change encoding or encrypt data with these algorithms:

- **base58**
- **base64**
- **sha1**
- **sha256**
- **aes**
- **blowfish**

For sha1, sha256, aes and Blowfish you must specify a key that will be used on the other side to decrypt

Details for AES and Blowfish:

	AES	
Initialization vector	**First 16 bytes of the string**	**First 8 bytes of the string**
Cipher mode	**CFB (Cipher Feedback)**	**CBC (Cipher Block Chainir**
Key length	**128-bit — 16 chars**	**64-bit — 8 chars**
Padding	**zero**	**zero**

> **TIP**
>
> **Encryption/decryption uses Base64URL encoding.**

Examples of syntax:

```
{crypt("base58" lead._fname)}
{crypt("base64" lead._lname)}
{crypt("sha256" lead.custom_field "encrypt_key")}
{crypt("sha1" lead._city "encrypt_key")}
{crypt("aes" lead._fname "key1234567891234")}
{crypt("blowfish" lead._fname "key12345")}
```

If you need to encrypt a string, wrap it in quotes. For a number, quotes are not needed:

```
{crypt("base58" "value")}
{crypt("base64" 42)}
```

Hashing algorithms (each supports an optional salt):

- **md5 + salt**
- **xxh32 + salt**
- **xxh64 + salt**

Syntax matches the encryption examples:

```
{crypt("md5" subscription.email "salt")}
{crypt("xxh32" loyalty.new2.promocode "salt")}
{crypt("xxh64" json.text "salt")}
```

Note that encrypted data is already Base64-encoded.

If you need to encrypt variables inside a form link, use the `formcrypt` function:

```
{formcrypt(lead.email)}
```

The encryption algorithm is not specified for `formcrypt`.

Logical Expressions

Logical expressions

Usage is described in a separate article. For a simple condition:

```
{if lead.age gte 18}
    If 18 or older.
{else}
    If younger than 18.
{end}
```

Fragments

Message fragments

Fragments are message blocks reused across messages (e.g., header, footer, social links).

Call a fragment with `{fragment.<short_fragment_name>}`**.**

Call a random fragment from a list: `{randomfragment[fragment.name1 fragment.name2 fragment.name3]}`**.**

See more in this article.

> TIP: Fragments and Parts
>
> **The platform also includes similar functionality called parts. Parts differ from fragments in that they are single email. To start working with parts, add the variable** `{part.example_part}` **to the template code. After the bottom of the page, where you can configure the template parts.**
>
> **Read more about parts here.**

Parameters

Description	Variable
Sent message ID (SMID)	`{send_message_id}`
Resource token	`{resource_token}`
Template ID	`{msgid}`
Mailing ID	`{campid}`
Resource ID	`{resid}`
Database (profiles) ID	`{listid}`
Mailing SUBID	`{subid}`
Resource name	`{resname}`
Database name	`{listname}`
Message type	`{msg_type}`
Resource URL	`{resurl}`
Resource domain	`{resdomain}`
Message name	`{msgname}`
Mailing name	`{campname}`
SMS sender name	`{from_name_sms}`
Segment ID	`{segid}`
Segment name	`{segname}`
Attribute value	`{attribute_value("attribute")}` **where** `"attribute"` **is the API name of the attribute**

Service

Description	
API content	`{apicontent.field_name}`
Cancel message sending	`{cancel}`

Tracking

Description	
Tracking domain	`{trkdomain}`
Pixel recording an open	`{pixel}`
Pixel recording a read	`{read}`
Subscription manager link	`{preferences}`
Unsubscribe link	`{unsubscribe}`
Global unsubscribe link	`{globalunsubscribe}`
Add to suppression list	`{suppress}`
Web version link	`{webversion}`

Workflow

Description	
Workflow name	`{workflow_name}`
Workflow ID	`{workflow_id}`
Node (element) ID	`{node_id}`
Node (element) type	`{node_type}`

Events

Event variables let you work with user actions (page views, add-to-cart, purchases, etc.) collected via pi

How it works

- **Real data is populated only when a trigger actually fires or within a running workflow.**
- **Events are context-bound — in triggers they are events of that trigger; in workflows they are events**

> INFO
>
> **These variables are only available for trigger-based mailings and scenarios that use the “Pixel activatio**

Variables

Description	Variable	Notes
First event	`{event.first}`	**The oldest event in the current context**
Last event	`{event.last}`	**The most recent event in the current context**
All events	`{event.all}`	**Full list of events in reverse arrival order; index `[0]` is the last**
Event count	`{event.count}`	**Number of events in the queue**

Using in templates

To get a specific event field, append the parameter with a dot:

```
{event.first.goal} // goal of the first event
{event.last.value} // value of the last event
{event.first.geo_city} // city from the first event
{event.last.utm_source} // source from the last event
{event.first.browser} // browser from the first event
{event.first.categories} // categories from the first event
```

To iterate over all events, use a loop:

```
{for $index $event = event.all}
            {if $event.goal equal "cart"}
                  {formatdate $event.date "%Y-%m-%D to %H:%M"} added to the cart {$event.data.count}x {$event.data.product}.
              {end}
{else}
    <p>No events...</p>
{end}
```

Examples

Trigger message after add-to-cart:

```
You left an item: {event.first.data.product}
Price: {event.first.data.price}
Buy: {event.first.data.link}
```

Segmentation by geolocation:

```
A sale is happening in your city {event.first.geo_city}!
Source: {event.first.utm_source}
```

Mailing performance analysis:

```
Source: {event.last.utm_source}
Mailing: {event.last.utm_campaign}
Browser: {event.last.browser}
```

For use in scenarios with the Event Waiting node:

```
Hello, {user.name}!

You recently added a new item to your cart but didn’t complete the purchase.

Your item is waiting for you:
{event.last.product_name}
```

Event fields

Basic fields

Field	Type	Description	
`id`	**string**	**Unique event ID**	`{event.first.`
`type`	**string**	**Platform event type**	`{event.last.t`
`date`	**DateTime**	**Event date and time**	`{event.first.`

Field	Type	Description	
`test`	**bool**	**Test event flag**	`{event.last.t`

Geo data

Field	Type	Description	
`geo_country`	**string**	**Country (full name)**	`{event.first.geo_countr`
`geo_tld`	**string**	**Two-letter country code**	`{event.first.geo_tld}`
`geo_region`	**string**	**Region**	`{event.first.geo_region`
`geo_city`	**string**	**City**	`{event.first.geo_city}`
`geo_address`	**string**	**Address**	`{event.first.geo_addres`
`geo_zip`	**string**	**Postal code**	`{event.first.geo_zip}`
`geo_time_zone`	**string**	**Time zone**	`{event.first.geo_time_z`
`geo_lat`	**float**	**Latitude**	`{event.first.geo_lat}`
`geo_lon`	**float**	**Longitude**	`{event.first.geo_lon}`

UTM tags

Field	Type	Description	
`utm_source`	**string**	**Traffic source**	`{event.first.utm_source}`
`utm_medium`	**string**	**Traffic medium**	`{event.first.utm_medium}`
`utm_campaign`	**string**	**Mailing name**	`{event.first.utm_campaign}`
`utm_content`	**string**	**Content identifier**	`{event.first.utm_content}`
`utm_term`	**string**	**Keyword**	`{event.first.utm_term}`
`utm_keyword`	**string**	**Alternative keyword**	`{event.first.utm_keyword}`

Pixel event (Pixel: `goal`)

Field	Type	Description	
`pixel_id`	**int**	**Pixel ID**	`{event.first.pixel_id}`
`goal`	**string**	**Goal name**	`{event.first.goal}`
`value`	**float**	**Goal value**	`{event.first.value}`
`data`	**object**	**Custom goal data**	`{event.first.data.order_id}`
`browser`	**string**	**User's browser**	`{event.first.browser}`
`user_agent`	**string**	**Browser User-Agent**	`{event.first.user_agent}`
`accept_language`	**string**	**Browser language**	`{event.first.accept_language`
`count_items`	**int**	**Number of items**	`{event.first.count_items}`
`categories`	**string[]**	**Array of categories**	`{event.first.categories}`
`send_message_id`	**string**	**Message send ID**	`{event.first.send_message_id`

API call event (API: `api_call`)

Field	Type	Description	
`content`	**object**	**Data from API call**	`{event.first.content.order_id}`
`custom_data`	**object**	**Custom API data**	`{event.first.custom_data.field_name}`

Event types

Group	Type	Description
Pixel	`goal`	**Pixel/goal activation (including via API)**
API	`api_call`	**API call through a workflow or trigger start**

Note: All fields of different event types are available at the same level. For example, `{event.last.content}` r call event if that is the current event.

See the developer documentation for full technical details and limits. This helps you set up API data coll in templates.

Loyalty programs

When inserting loyalty program variables via the insert dialog in the editor, a context menu opens where a points type (as well as the transaction count limit and transaction activation period for the last two va

Description	
Member points balance	`{loyalty_program.<LOYALTY_PROG`
Member active points balance	`{loyalty_program.<LOYALTY_PROG`
Member inactive points balance	`{loyalty_program.<LOYALTY_PROG`
Balance with currency name	`{loyalty_program.<LOYALTY_PROG`

Description	
Active balance with currency name	`{loyalty_program.<LOYALTY_PROG`
Inactive balance with currency name	`{loyalty_program.<LOYALTY_PROG`
List of ближайших expirations (where x is the transaction count limit and y is the transaction activation period)	`{loyalty_expiration_points(<LO`
List of nearest points activations (where x is the transaction count limit and y is the transaction activation period)	`{loyalty_activation_points(<LO`

Escaping

The `escape` function is used to include strings that contain characters similar to the platform's variable s

```
{escape "..."}
```

Everything inside the quotes is passed without changes — the platform does not attempt to interpret it a

Examples

```
https://example.example/abcd?some={escape "{lead.email}"}
```

In this example, `{lead.email}` remains plain text and is not replaced with the profile's actual email:

```
https://example.example/abcd?some={lead.email}
```

If you use `escape` inside an HTML link, add the `no-track` attribute to prevent the link from breaking during

```
<a href="https://example.com/link?query={escape "{lead.email}"}" title="Go to link" no-track>Link with escaping</a>
```

> TIP: Escaping quotes and backslashes
>
> **If the string contains quotes, they must be escaped with a backslash `\"`. A backslash itself is written a**
>
> **Escaping quotes:**
>
> ```
> {escape "String with \"quotes\" inside"} = String with "quotes" inside
> ```
>
> **Escaping backslashes:**
>
> ```
> {escape "Backslash: \\\\"} = Backslash: \\
> ```

Promocode

Promocodes allow you to insert special codes into a message, generated according to the template setti

```
Your promocode: {loyalty.sale15.promocode}
```

Emoji

You can also insert emoji into your message templates. Choose the emoji picker in the variables menu ar

Profile Relations

General variable formula:

```
{relation.[relation_short_name].[direct for direct or reverse for reverse].[property_name].[count — count, total — sum,
```

Count of direct relations with `profit` property:

```
{relation.managers_loyal.direct.profit.count}
```

Top reverse relations by `profit`. A loop is used to output multiple values:

```
{for $index $item = relation.managers_loyal.reverse.profit.top}
{$item.lead._fname} {$item.lead._lname} ({$item.value} profit)
{else} There is no 'profit' for you :(
{end}
```

Sum of `profit` across direct relations:

```
{relation.managers_loyal.direct.profit.total}
```

Additional Editor Functions

Price formatting

A number formatting function splits numbers into groups of three with a given delimiter.

```
Price 8.000 RUR
```

```
<p>Price {format json.price "."} RUR</p>
```

To insert, click `</>` in the editor and choose Output functions —> Formatting.

Date offset

The `adddate(date, year, month, day)` function returns a date after adding the specified interval.

```
<p>Event starts on {adddate(lead.notify_date 0 0 10)}</p>
```

`adddate()` can be used inside `format()`:

```
<p>Event starts on {format adddate(lead.notify_date 0 0 10) "%Y-%m-%D"}</p>
```

To insert, click `</>` —> Date & time —> Date offset.

You can format and shift the date at the same time using the following function:

```
{format adddate(datenow 0 0 3) "%Y-%m-%D"}
```

Date formatting

Date values differ by locale. Use date formatting for subscriber-friendly dates.

For example, to show the subscriber's date of birth as:

```
Date of birth 02.11.1998
```

Use:

```
<p>Date of birth {format lead._bdate "%D.%m.%Y"}</p>
```

You can format and shift the date at the same time using the following function:

```
{format adddate(datenow 0 0 3) "%Y-%m-%D"}
```

All date formatting parameters:

```
# years
"%y": "06"
"%Y": "2006"
"yy": "06"
"YYYY": "2006"

# months
"%m": "01"
"MM": "01"
"%B": "January"
"month": "January"
"%ru_month": "январь"

# days
"%d": "2"
"%D": "02"
"DD": "_2"
"%A": "Monday"
"weekday": "Monday"

# hours
"%H": "15"
"%I": "3"
"hh": "15"
"hours": "15"
"twelve_h": "3"

# mins
"%M": "04"
"mm": "04"

# seconds
"%S": "05"
"ss": "05"

# am/pm indicator
"am": "pm"
"AM": "PM"
"%p": "pm"

# timezone
"tzone": "-07"
"timezone": "-07:00"
"%Z": "Z0700"
```

To insert, click `</>` —> Date & time —> Formatted date.

INFO

If the template uses content from external JSON (`apicontent` variable or dynamic JSON content), the pla into date/time.

Formatting example:

Mode	Format
JSON	`"date_time": "2023-02-21T11:10:35.141Z"`
Template	`{format apicontent.date_time "%Y-%m-%d %H:%M:%S"}`
Preview	`2023-02-21 11:10:35`

Logical expressions in messages

Description

TIP

If you're using multiple logical operators in a single `if` expression, make sure to wrap each pair of con required for correct parsing.

Correct syntax example:

```
<html>
{if param1 and (param2 and (param3 and param4))} <!--true-->
if true
{else}
if false
{end}
</html>
```

Example that will cause a parsing error:

```
<html>
{if param1 and param2 and param3 and param4} <!--false-->
if true
{else}
if false
{end}
</html>
```

The core of highly personalized targeted messages are conditional expressions.

They are based on if → then → else logic and display different content depending on conditions.

```
{if lead.field in json.promo_actions1}
```

You can use data variables both in conditions and in output blocks. This significantly reduces message d

Example: setting special discounts for VIP customers of a specific shop, while all other recipients receiv

- `Congratulations! Your premium discount is 42%!` **— VIP customers**
- `Special offer! Get a 20% discount for your second purchase!` **— all others**

Two conditions inside a single message are enough.

1. **Check if the customer has the `"best_buy_type"` field and belongs to the required shop.**
2. **Check whether premium products were purchased.**

```
<!-- Condition 1: profile has "shop_name" and "best_buy_type" -->
{if lead.shop_name and lead.best_buy_type}

    <!-- Condition 2: shop_name equals "my_shop" -->
    {if lead.shop_name equal "my_shop"}

        <!-- Condition 3: best_buy_type contains "premium" -->
        {if lead.best_buy_type contains "premium"}
            <p>Congratulations! Your premium discount is 42%!</p>
        {else}
            <p>Special offer! Get a 20% discount for your second purchase!</p>
        {end}

    {else}
        <p>Special offer! Get a 20% discount for your second purchase!</p>
    {end}

{else}
    <p>Special offer! Get a 20% discount for your second purchase!</p>
{end}
```

Examples

Conditions

Simple condition

```
{if param1}
         if true
{else}
         if false
{end}
```

Complex condition

Example 1:

```
{if param1 in array["par1" "par2"] and ( 5 equal length("par 3") or false )}
        if true
{else}
        if false
{end}
```

Example 2 using `apicontent`:

```
{if apicontent.field_1 not_equal apicontent.field_2 and ( apicontent.field_1 or apicontent.field_2 )}
        if true
{else}
        if false
{end}
```

Example 3:

```
{if crypt("md5" lead._fname "salt") like "123b003aa39d110d1333b37af9889e10" and crypt("base58" lead.lottery_id) like json.crypted_winner}
        if true
{else}
        if false
{end}
```

Example 4:

```
{if iseven(lead.count) and iseven(lead.count2) equal false}
        if true
{else}
        if false
{end}
```

Check field and cancel sending if missing

```
{if lead.some_field}
some_field is not empty
{else}
{cancel}
{end}
```

Loops

Simple loop

```
{for array[1 "2" 3.3]}
        {.}
{else}
        if empty array
{end}
```

Loop with variables

```
{for $index, $el = array[1 "2" 3.3]}
        {$index}: {$el}
{else}
        if empty array
{end}
```

Value matching conditions

Equal

```
{if param1 equal param2}
        if true
{else}
        if false
{end}
```

Not equal

```
{if param1 not_equal param2}
        if true
{else}
        if false
{end}
```

String is equal (case insensitive)

```
{if param1 like param2}
        if true
{else}
        if false
{end}
```

String is not equal (case insensitive)

```
{if param1 not_like param2}
        if true
{else}
        if false
{end}
```

String contains (case insensitive)

```
{if param1 contains param2}
         if true
{else}
         if false
{end}
```

String doesn’t contain (case insensitive)

```
{if param1 not_contains param2}
         if true
{else}
         if false
{end}
```

Numbers and dates comparison

More than number/date

```
{if 0 gt 1}
         if true
{else}
         if false
{end}
```

More than or equal to number/date

```
{if 0 gte 1}
         if true
{else}
         if false
{end}
```

Less than number/date

```
{if 0 lt 1}
         if true
{else}
         if false
{end}
```

Less than or equal to number/date

```
{if 0 lte 1}
         if true
{else}
         if false
{end}
```

Arrays and tags fields

Element is in array

```
{if param1 in array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

Element is in tags field

```
{if param1 in lead.TagsField}
         if true
{else}
         if false
{end}
```

Element is not in array

```
{if param1 not_in array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

Element is not in tags field

```
{if param1 not_in lead.TagsField}
         if true
{else}
         if false
{end}
```

All elements are in first array

```
{if array["par1" "par2"] all_in array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

Not all elements are in first array

```
{if array["par1" "par2"] not_all_in array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

At least one match in the arrays

```
{if array["par1" "par2"] any_of array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

No matches in the arrays

```
{if array["par1" "par2"] not_any_of array["par1" "par2"]}
         if true
{else}
         if false
{end}
```

Loops in messages

The message supports loops for JSON variables of type `"list" []` and `"object" {}`, as well as access to a {json.object.name}.
If additional profile fields are inserted as variables into the template, fields of the “Array of Objects” and be used in loops.
Nested loops are also possible.

Variables example

```
{
  "Addresses": ["Barclay street, 244 building", "Green street, 555 building"],
  "Information": {
    "LJ": [
      { "name": "Optimyss X1", "price": "$10" },
      { "name": "Optimyss X2", "price": "$20" }
    ],
    "Samsung": [{ "name": "Galaktiko 17", "price": "$50" }]
  }
}
```

Loops example

```
<!-- Loop by object elements -->
{for $vendor $models = json.Information}
    <p>{$vendor}</p>

    <ol>
        <!-- Nested loop -->
        {for $index $model = $models}

            <!-- Call elements by key -->
            <li>{$model.name} {$model.price}</li>

        {else}
            <p>Empty array!</p>
        {end}
    </ol>

{else}
    <p>Empty array!</p>
{end}

<h5>Addresses:</h5>
<ol>
<!-- Simple loop by strings list -->
{for $index $address = json.Addresses}
    <li>{$address}</li>
{else}
    <p>Empty array!</p>
{end}
</ol>
```

Result

```
<h5>New:</h5>

    <!-- Loop by object elements -->
    <p>LJ<p>
    <ol>
        <!-- Nested loop. Call elements by key -->
        <li>Optimyss X1 $10</li>
        <li>Optimyss X2 $20</li>
    <ol>

    <!-- Loop by object elements -->
    <p>Samsyng<p>
    <ol>
        <!-- Nested loop. Call elements by key -->
        <li>Galaktiko 17 $50</li>
    <ol>

<h5>Addresses:</h5>
    <ol>
        <!-- Simple loop by strings list -->
        <li>Barclay street, 244 building</li>
        <li>Green street, 555 building</li>
    </ol>
```

Market variables in templates

Variables

There are four variables for Market:

- `products` — function selects products

```
{products("<filter>" 10 "manufacturer_name" "desc")}
```

- `sku` — function selects SKUs

```
{sku("<filter>" 10 "manufacturer_name" "desc")}
```

- `orders` — the function selects orders from the list of profile orders

```
{orders("<filter>" 10 "status" "asc")}
```

- `order` — the function selects one order from the list of profile orders

```
{order("<filter>")}
```

Function arguments

`("<filter>" 10 "manufacturer_name" "desc")`

Argument	Required	Description
`"<filter>"`	Yes	The filter is a string of conditions. Allows you to select products, SKUs, orders, or order lines that meet specified conditions. For example, you can select products from the same manufacturer or orders that exceed a specific cost. See below for details.
`10`	Yes	Limit. How many array elements to iterate over. The maximum value is 30. For example, if you specify `10`, then the filter will select the first ten elements from the array.
`"manufacturer_name"`	No	Sort objects by the specified field. By default — no sort.
`"desc"`	No	Sort direction. `asc` — ascending (by default) `desc` — descending

Filter

The filter consists of fields and operators. Each object in the market has fields. For example, products ar name). Orders have a `total_price` field (total cost of the order). In a filter, you specify a field and a value t object matches the value, this object will be selected and its data can be substituted into the template.

Filter example

Let's add the "Orders" variable to the template:

```
{orders ("order.total_price == '200 USD' && (order.custom_fields.field == 'lead._region')" 10 "id" "desc")}
```

The bold part of the variable is the filter. Let's consider it in parts:

- `order.total_price == '200 USD'`: the condition will select all orders with a cost equal to $200.
- `order.custom_fields.field == 'lead._region'` - the condition will select orders with the additional 'field'. Yo field instead of `"field"`. The specified equality means that the value of the `"field"` must match the v profile (`"lead._region"`).
- `&&` is an operator. It indicates that both conditions must be met.
- The number 10 indicates the number of output objects, `"id"` indicates the filter field, and `"desc"` indi

Below are the fields and operators available in the filter.

Operators and functions in a filter

Operator	Description	Example
>	**greater**	`order.delivery_cost > '10 USD'`
>=	**greater or equal**	`order.delivery_cost >= '10 USD'`
<	**less**	`sku.count < '10'`
<=	**less or equal**	`sku.count <= '10'`

`==`	**equals**	`order.status == '1'`
		
		`order.custom_fields('date') == '2024-12-25'`

!=	**not equal**	`product.manufacturer_name != 'Apple'`
&&	**AND (both conditions match)**	`product.manufacturer_name == 'Apple' && product.price < '200 USD'`
||	**OR (one of the conditions matches)**	`sku.manufacturer_name == 'MSI'` || `sku.manufacturer_name == 'Acer'`
`in('<field_name>', 'value1', 'value2'...'valueN')`	**The field value (`<field_name>`) matches one of the specified values ('`value1`', '`value2`')**	`in(product.external_id,'1','2','3')`

Operator	Description	Example	
`contains('<field_name>', 'value1', 'value2'...'valueN')`	**One of the array elements matches one of the specified values.** `<field_name>` — **array** `'value1', 'value2'` — **values to match.** **Used if the field is an array, i.e. contains several elements (for example: categories, images, additional fields).**	`contains(product.categories,'smartphone')`	
`len('<field_name>', <operator>, <number> )`	**Checks the length of an array. Compares the number of elements in the array with the specified number.** `<field_name>` — **array field (e.g.,** `product.pictures` **)** `<operator>` — **comparison operator (** `>` , `>=` , `<` , `<=` , `==` , `!=` **)** `<number>` — **integer for comparison.**	`len(product.pictures, !=, 0)`	

Working with custom fields

Two syntaxes are available for filtering by custom fields:

Universal syntax (recommended)

```
(product.custom_fields('external_id'))
```

> **INFO**
>
> **We recommend using the universal syntax** `product.custom_fields('external_id')` **because it supports all type string, including special characters. This option guarantees correct filtering regardless of your custom**

Examples:

```
{for $product in products("product.custom_fields('123') == 'red'")}
{for $product in products("in(product.custom_fields('color'), 'red', 'blue')")}
{for $product in products("sku.custom_fields('45') > '100'")}
```

String syntax (limited)

```
(product.custom_fields.field_name)
```

> **INFO**
>
> **The old syntax** `product.custom_fields.field_name` **can only be used if the field's** `external_id` **consists exclusiv does not contain numbers, hyphens, or other special characters.**

Example:

```
{for $product in products("product.custom_fields.brand == 'Apple'")}
```

List of available fields in filters

Products

Prefix — `product.`

Field	Description
`product.external_id`	**Product ID**
`product.name`	**Product name**
`product.manufacturer_name`	**Manufacturer's name**
`product.url`	**URL of the product page on the store website**
`product.pictures`	**Product Image URL**
`product.count`	**Number of products available for sale**
`product.is_available`	**Is the product in stock at the store?**
`product.delivery`	**Possibility of delivery (for all regions to which the store delivers)**
`product.barcode`	**List of product barcodes from the manufacturer**
`product.categories`	**Product category ID**
`product.custom_fields.<field_external_id>`	**Custom product field ID. For numeric IDs use** `custom_fields('123')`

SKU

Prefix — `sku.`

Field	Description
`sku.external_id`	**SKU ID**
`sku.name`	**SKU name**
`sku.manufacturer_name`	**Manufacturer's name**
`sku.url`	**URL of the SKU page on the store website**

Field	Description
`sku.pictures`	**SKU image URL**
`sku.count`	**Number of SKUs available for sale**
`sku.is_available`	**Is the SKU in stock at the store?**
`sku.delivery`	**Possibility of delivery (for all regions to which the store delivers)**
`sku.barcode`	**List of SKU barcodes from the manufacturer**
`sku.categories`	**SKU category ID**
`sku.custom_fields.<field_external_id>`	**Custom sku field ID. For numeric IDs use** `custom_fields('123')`

Orders

Prefix — `order.`

Field	Description
`order.external_id`	**Order ID**
`order.status`	**Order status ID** **1 - new** **2 - delivered** **3 - paid** **4 - cancelled** **5 - partial refund**
`order.delivery_cost`	**Shipping cost**
`order.total_price`	**Total cost of the order.** **It consists of the number of each item in the order and the cost of delivery.**
`order.custom_fields.<external_id поля>`	**Custom order field ID**
`order.real_create_time`	**Order creation time**
`order.real_update_time`	**Order update time**
`order.region_external_id`	**Region ID**

Sorting by Custom Fields

Sometimes you need to sort objects by their custom field values (the `custom_fields` object).
Custom fields are stored in the database as a JSON string. Therefore, to access a field inside the JSON f
operator.

Example of accessing a key:

```
custom_fields -> 'priority'
```

Usage in a sort parameter function (using `products` as an example):

```
{for $index $el = products("contains(product.categories,'smartphone')" 3 "custom_fields -> 'priority'" "desc")}
{$el.name} — {$el.price}
{else}
if empty array
{end}
```

In this example, the product list is first filtered by the `smartphone` category, then sorted by the value of th
descending order, and the first 3 results are displayed.

> INFO
>
> **Important**
>
> - **The `->` operator is used only in the sort parameter to access a JSON key.**
> - **For filtering by custom fields, use the dot notation from the "Filter" section, for example: `order.custo`**
> - **The `->` operator is not a market operator — it is a database operator (PostgreSQL) used for the que**
>   **considered a workaround.**

Use cases

Any variable in the market is an array. For example, `products` contains an array of all products uploaded f
an array of orders for one profile. To iterate over an array, use loops.

Searching the array of products (products)

Outputs 5 Apple products in an email:

```
You might be interested in these products:
{for $index $el = products("product.manufacturer_name == 'Apple'" 5 "external_id" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

7 cheapest smartphones:

```
The best smartphone prices for you and your loved ones:
{for $index $el = products("product.categories == 'smartphone'" 7 "price" "asc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Goods with delivery:

```
Delivered to your door:
{for $index $el = products("product.delivery == true" 10 "name" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Search SKU (sku) array

Smartphones under 20000:

```
Smartphones you will like:
{for $index $el = sku("contains(sku.categories,'smartphone') && sku.price < '20000 RUB'" 3 "price" "asc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

IPhone with more than 128GB of storage:

```
Enough memory for everything:
{for $index $el = sku("contains(sku.name,'IPhone') && sku.custom_fields.memory >= '128'" 10 "name" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Searching the array of all orders (orders)

Orders at pickup points:

```
Orders are waiting for you at the pickup point:
{for $index $el = orders("order.status == '2'" 10 "external_id" "desc")}
Order # {$el.external_id}
Due: {$el.total_price}
{else}
if empty array
{end}
```

List of new orders:

```
New orders in the last week:
{for $index $el = orders("order.status == '1'" 20 "external_id" "desc")}
Order # {$el.external_id}
Due: {$el.total_price}
{else}
if empty array
{end}
```

Search for items in order with id = '0001' (order + lines)

```
{for $index $el = order("order.external_id == '0001'")}
 {if $index equal "lines"}
  {for $i $line = $el}
    {$line.name} position name: {$line.name}
    {$line.price} Item price: {$line.final_price_of_line}
  {end}
 {end}

 {if $index equal "delivery_cost"}
   {$el} {$delivery_cost} {$el}
 {end}
 {if $index equal "total_price"}
   To pay: {$el}
 {end}

{else}
if empty array
{end}
```

List of available fields for substitution in the template

products/sku

Substitution field	Description
external_id	**Product/SKU ID**
name	**Product/SKU name**
manufacturer_name	**Manufacturer's name**
url	**URL of the product page on the store website**
pictures	**List of URL links to product images (array)**
count	**Quantity of products/SKU available for sale**
available	**Is the product/sku in stock at the store?**
delivery	**Possibility of delivery (for all regions to which the store delivers)**
barcode	**List of product barcodes from the manufacturer (array)**
categories	**List of product/SKU category IDs (array)**
custom_fields.<external_id поля>	**Custom field ID. For numeric IDs and special characters use custom_fields('external_id') syntax.**
price	**Current product price**
old_price	**Previous product price**
regional_data.<region_external_id>.count	**Quantity of products/SKU available for sale in the specified region.** `regional_data` **is product/sku data that varies by region.** `<region_external_id>` **is region ID.**
regional_data.<region_external_id>.delivery	**Possibility of delivery to the specified region**
regional_data.<region_external_id>.available	**Is the product/SKU in stock in the specified region?**
regional_data.<region_external_id>.price	**Product/SKU price in the specified region**

orders / order

Substitution field	Description
external_id	**Order ID**
status	**Order status ID**
delivery_cost	**Shipping cost**
total_price	**Total cost of the order.** **It consists of the cost of each item in the order and the cost of delivery.**
custom_fields.<external_id поля>	**Custom field ID. For numeric IDs and special characters use custom_fields('external_id') syntax.**
create_time	**Order creation time**
update_time	**Order update time**
region.external_id	**Region ID**
lines	**Order line. Only for the** `order` **function.**

lines

Substitution field	Description
id	**Line ID**
external_id	**Order line additional ID**
custom_status_external_id	**Order line status ID**
base_price_per_item	**Base price**
final_price_of_line	**Final price**
count	**Quantity of products or skus in the line**
custom_fields.<field_external_id>	**ID for the custom field of the order line. For numeric IDs and special characters use custom_fields('external_id') syntax.**
product_external_id	**Line product ID**
name	**Line name**
manufacturer_name	**Manufacturer's name of the line product**
url	**URL of the page of the line product**
pictures	**List of URL links to images of the line product**
balance	**Stock quantity**
available	**Is the line product in stock at the store**
delivery	**Possibility of delivery of the line product (array)**
barcodes	**List of barcodes of the line product (array)**
categories	**List of category IDs of the line product**

INFO

Custom field override priority:
(defines which field values are applied first)

SKU > Product > Order-item

Using the JSONPath functionality

The Altcraft platform supports the functionality of the JSONPath query language when working with JSO you to quickly and easily select objects within code and use them message templates. It can be used wh with external JSON content.

API content allows you to change the content of the trigger message each time it is sent. The required v API request. You can learn more about how to run triggered mailings and how to work with API content i

To use external JSON content, go to the Options section when creating or editing a template and provide source. The platform will automatically determine the fields of the external JSON file. More information message templates can be found here.

The JSONPath variable on the Altcraft platform is specified as follows:

```
{jsonpath(json.<root_object> "<path_to_object>" "<output_format>")}
```

Function arguments

Argument	Description
Root object	**Accessing a JSON object that contains the desired parameter or object**
Path to object	**The path to the desired object can consist of segments separated by dots, logical operators, filters or functions**
Output format	**The final variable can be represented in two versions: item or array. `Item` will display the first result found for the request, and if there is no result, it will retur This mode always outputs the result as an array, even if there is only one or no result.**

Usage examples

Let's look at examples of using JSONPath. Suppose we have a JSON file with the following content:

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "images": ["AbeLincoln.png","Churchill.jpg","Thatcher.png"]
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "images": ["MenAtArms.png","Gentlemen.jpg","Officers.png"]
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "images": ["Ahab.png","Ishmael.jpg","whale.png"]

      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ]
  }
}
```

With JSONPath functionality, a small expression can return an array containing all the authors of the boo

```
{jsonpath(json.store "$.book[*].author" "array")}
```

Another expression will return one object, the author of the first book:

```
{jsonpath(json.store "$.book[*].author" "item")}
```

You can build a more complex function, for example, in order to find out the name of a book whose price the following code:

```
{jsonpath(json.store "$.book[?(@.price > 10)].title" "item")}
```

To find out the arithmetic average of the price of all books, we can use this function:

```
{jsonpath(json.store "$.book[*].price.avg()" "item")}
```

JSONPath variables can also be run in loops:

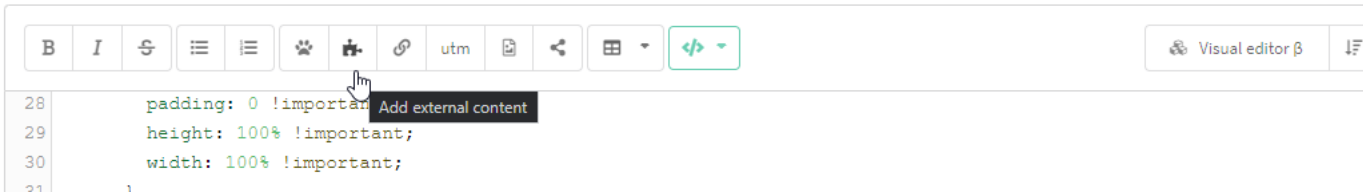
```
{for $index $imgs = jsonpath(json.store "$.book[*].images" "array")}
    <li>{$imgs}</li>
{else}
    <p>Empty array!</p>
{end}
```

The full functionality of JSONPath on the Altcraft Platform platform is listed in the table below:

Argument	Description	Use example	
`$`	Root element	`{jsonpath(json.store "$.book[0].title" "item")}`	Sayings of the Centu
`@`	Current element	`{jsonpath(json.store "$.book[?(@.price > 10)].title" "item")}`	Sword of Honour
`..`	Recursive descent	`{jsonpath(json.store "$..title" "array")}`	[Sayings of the Cent
`*`	Match all objects	`{jsonPath(json.store "$.book[*].price" "array")}`	[8.95 12.99 8.99 22.9
`?()`	Filter	`{jsonpath(json.store "$.book[?(@.category == 'reference')].title" "item")}`	Sayings of the Centu
Boolean operator for filtering	Description	Use example	
`==`	Equal to	`{jsonpath(json.store "$.book[?(@.category == 'reference')].title" "item")}`	Sayings of the Centu
`!=`	Not equal to	`{jsonpath(json.store "$.book[?(@.category != 'reference')].title" "array")}`	[Sword of Honour Mo
`<`	Less than	`{jsonpath(json.store "$.book[?(@.price < 10)].title" "array")}`	[Sayings of the Cent
`>`	Greater than	`{jsonpath(json.store "$.book[?(@.price > 10)].title" "array")}`	[Sword of Honour Th
`<=`	Less than or equal to	`{jsonpath(json.store "$.book[?(@.price >= 12.99)].title" "array")}`	[Sword of Honour Th
`>=`	Greater than or equal to	`{jsonpath(json.store "$.book[?(@.price <= 12.99)].title" "array")}`	[Sayings of the Cent
Function	Description	Use example	
`max`	Maximum value in array	`{jsonPath(json.store "$.book[*].price.max()" "item")}`	22.99
`min`	Minimum value in array	`{jsonPath(json.store "$.book[*].price.min()" "item")}`	8.95
`first`	First array element	`{jsonPath(json.store "$.book[*].title.first()" "item")}`	Sayings of the Centu
`last`	Last array element	`{jsonPath(json.store "$.book[*].title.last()" "item")}`	The Lord of the Ring
`sum`	Sum of numbers in the array	`{jsonPath(json.store "$.book[*].price.sum()" "item")}`	53.92
`avg`	Average of the numbers in the array	`{jsonPath(json.store "$.book[*].price.avg()" "item")}`	13.48
`length`	Number of elements in the array	`{jsonPath(json.store "$.book.length()" "item")}`	4

Dynamic HTML content

External HTML content is one of the ways to create dynamic messages and automate your job. Using it, generated on a remote server exactly in the mailing moment.
To add the link to dynamic HTML content, press “Add external content” button in the message editor too



For example, if the server generates latest news, the message will be automatically maintained up to da
You can use several links to HTML content in the message.

```
{[http://www.example.news?index=1&news_group=movies]}

{[http://www.example.news?index=2&news_group=movies]}
```

You can include different variables into dynamic HTML content requests. It can be for example profile da `{lead._fname}` **or a containing database name** `{listname}`**. Thus instead of a static** `&news_group=movies` **you ca field variable:** `&news_group={lead.NewsGroup}`**.**
Each time such message is sent it will request HTML content from the server and generate it for differen personal preferences.

Dynamic JSON content

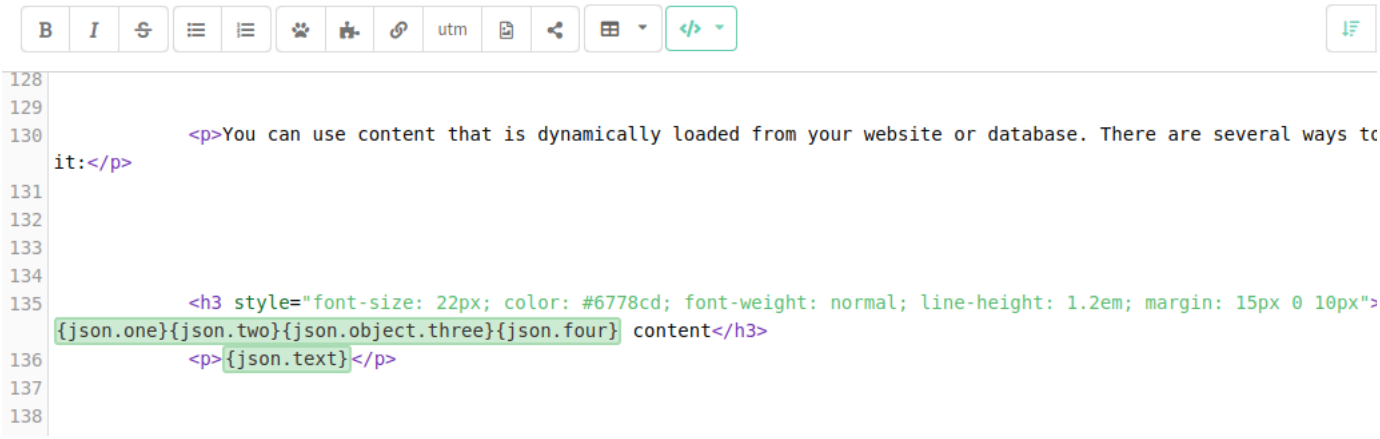
Dynamic JSON content is good way to create dynamic messages and automate your work. It allows you at the moment messages are sent.
You can include variables into JSON content queries, for example profile data fields like `{lead._fname}`.
Set the source of your external JSON content in message options tab.

> CAUTION
>
> **By default, the maximum size of dynamic JSON is 1 MB. In the On-premises version, this limit is set in t `MAX_REMOTE_CONTENT_BYTES` parameter.**

JSON data looks like this:

```
{
    "number": 10,
    "text": "Hello!",
    "list": ["One", "Two", "Three"],
    "object": {"model": "x17", "price": "$99"},
    "mixed": [
        {"city": "Los Gamburger", "street": "Florence Avenue"},
        {"city": "St. Potato", "street": "Glendale Boulevard"}
    ]
}
```

To add dynamic JSON content to your message template click the green `</>` variables button and select
Thus JSON content is inserted into a message as a special variable type* `{json.variable}:`



Two data types: “list” and “object” are available.**

- **Objects are inserted as a one-value variable, like this: `{json.object.key_name}`**
- **Lists are JSON arrays — they can be inserted with the help of loops, like this one:**

```
{for $index $products= json.products}
    <li>{$products}</li>
{else}
    <p>Check for new products next week!</p>
{end}
```

Content from SQL database in templates

Description

You can use dynamic content from external data tables, which can be stored on MySQL, PostgreSQL, MS database servers.

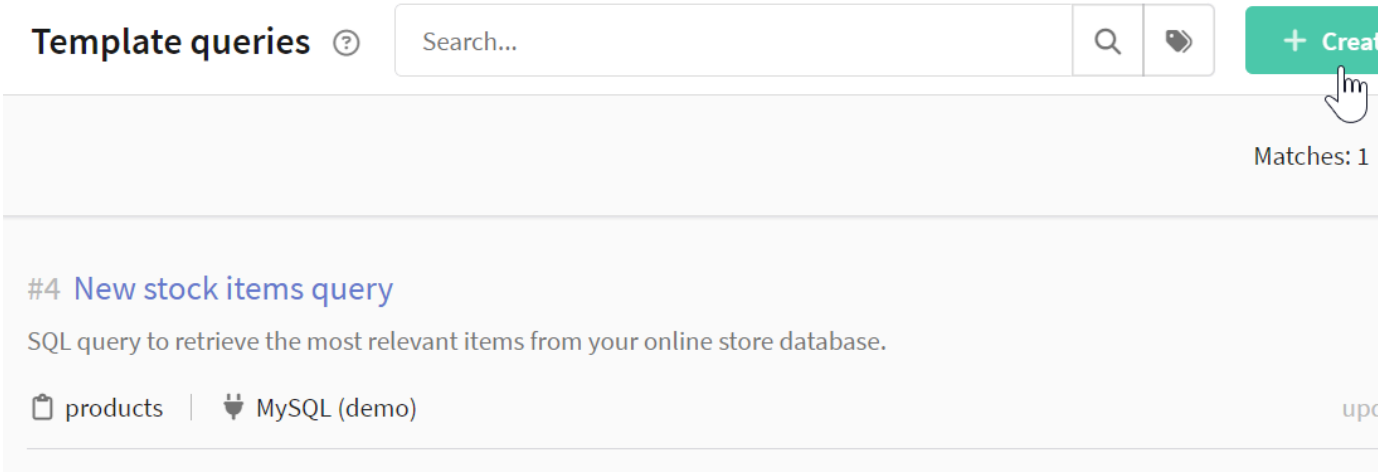
Connecting to SQL server

To connect to your database you would need an account with proper access privileges. The process dep Platform:

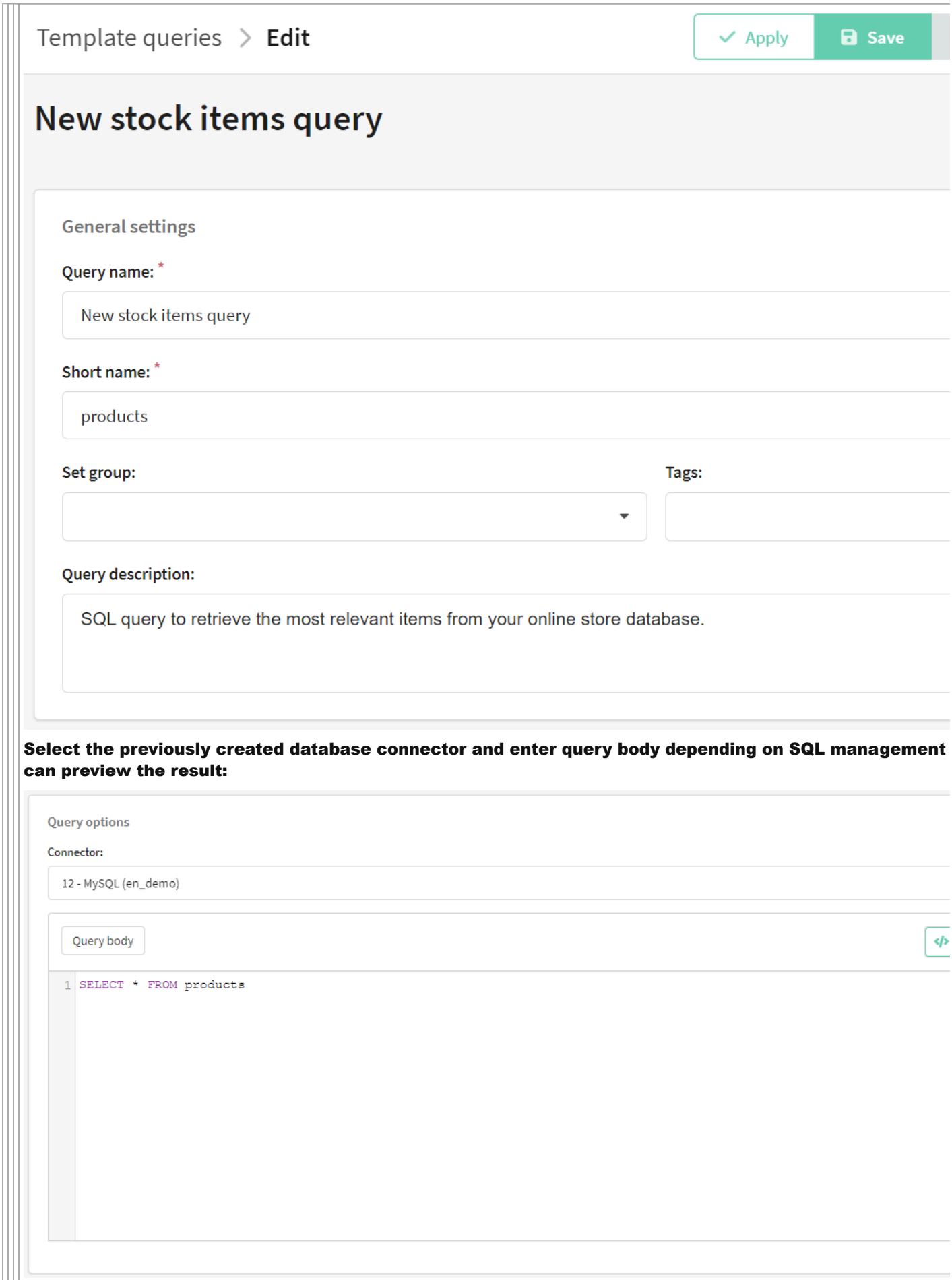
- **If you are a cloud account user — send access credentials to our support — support@altcraft.com. Y any questions related to connecting and configuring external services and databases.**
- **If you have an On-premises installation, platform administrator should create a database connector i it to your platform account.**

Creating query to external data table

To create a query, go to Data —> Template queries and create a new query:



In general settings, you need to set Query name and Short name for using in the templates editor. If nece groups to restrict other users from accessing the request, and attach tags for quick search.



Select the previously created database connector and enter query body depending on SQL management can preview the result:

TIP

Also, you can create the query by selecting Data tables in the Data menu. Select Save as template quer

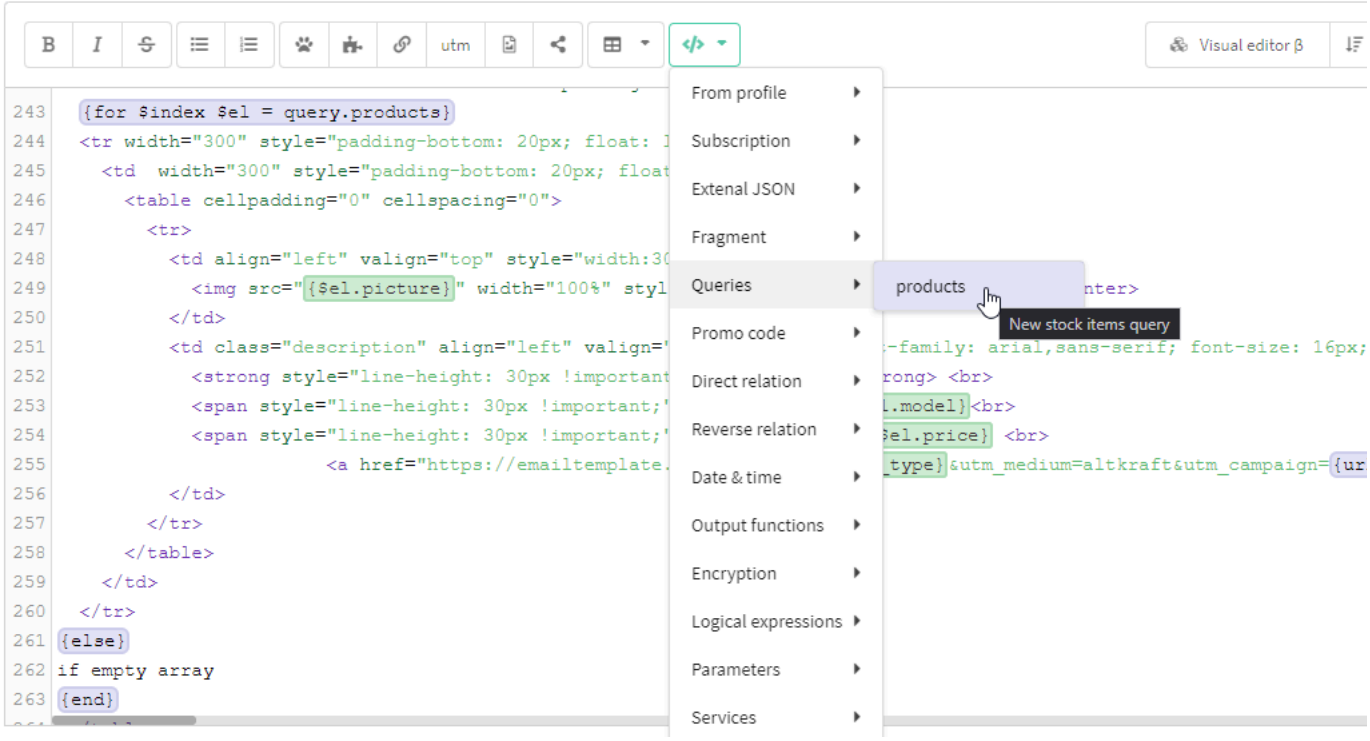
Using SQL query in message templates

In the message template queries are inserted like this: `query.your_query_name`. To insert query contents into (`</>`).

TIP

You can also include queries into template loops and logical expressions.

HTML VERSION



Then you can style your query to be displayed properly:

```
{for $index $el = query.products}
  <tr width="300" style="padding-bottom: 20px; float: left">
    <td  width="300" style="padding-bottom: 20px; float: left">
      <table cellpadding="0" cellspacing="0" class="doc-table">
        <tr>
          <td align="left" valign="top" style="width:30%;"><center>
            <img class="img" src="{$el.picture}" width="100%" style="display: block; border: 0;" /></center>
          </td>
          <td class="description" align="left" valign="middle" style="font-family: arial,sans-serif; font-size: 16px; color: #333; padding: 0
            <strong style="line-height: 30px !important;">{$el.type} </strong> <br>
            <span style="line-height: 30px !important;">Model: </span>{$el.model}<br>
            <span style="line-height: 30px !important;">Price: </span>$ {$el.price} <br>
                    <a href="https://example.com style="display: block; text-decoration: none; font-weight: bold; margin-top: 15px; text-
name="link1" >Buy now!</a>
          </td>
        </tr>
      </table>
    </td>
  </tr>
{else}
if empty array
{end}
```

Subject: F - for fresh awesome offers

Only the hottest AWESOME items

Dress

Model: Orient-rosso

Price: $ 650

Buy now!

T-shirt

Model: Retoro-SE

Price: $ 220

Buy now!

T-shirt

Model: Skel-hand

Price: $ 260

Buy now!

Sweatshirt

Model: Tomato soup

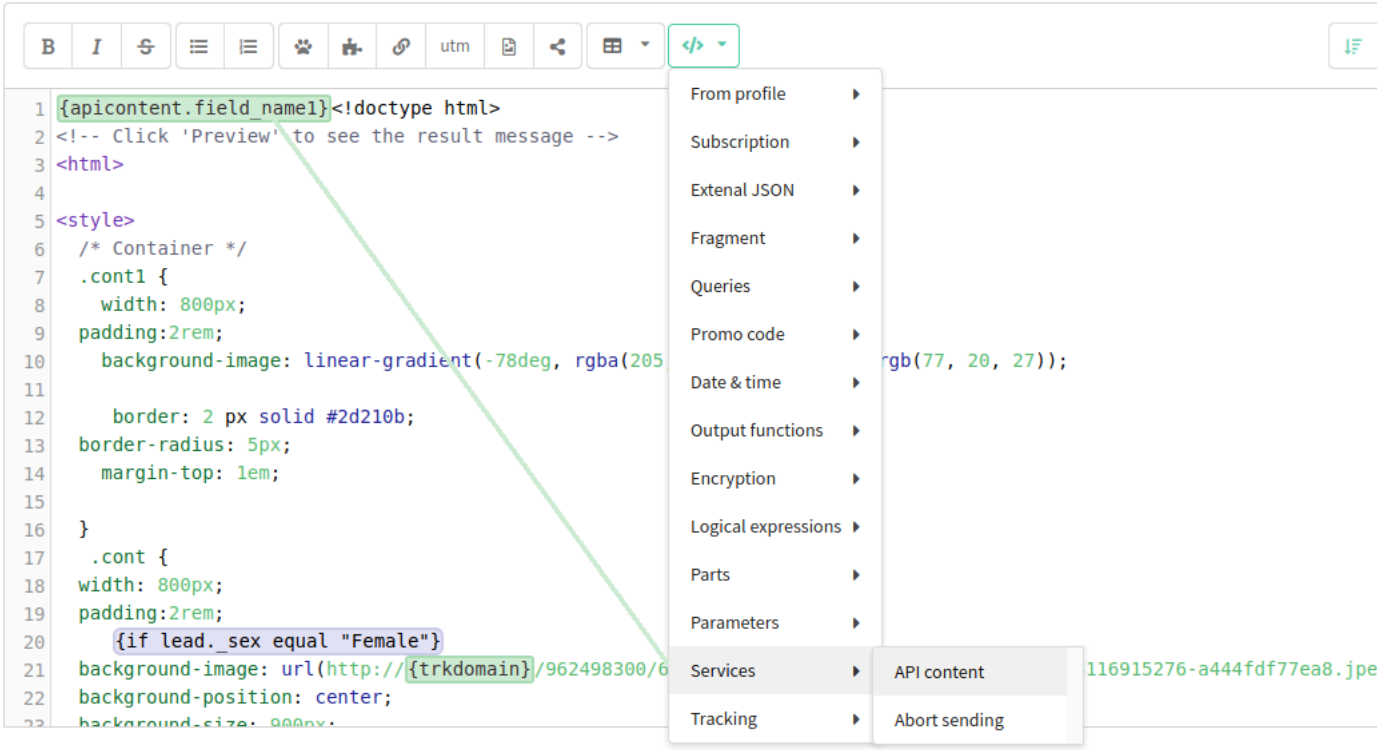
Price: $ 800

Buy now!

PayPal VISA pay Pay

Dynamic API content

When sending messages using the [API Call Trigger](), starting from version 1.1, it is possible to transmit da API request.



Using `apicontent`, one message can be automatically personalized for each client. Using `apicontent`, you c completed transactions, purchases, customer history, etc. This information is transmitted as a json-obje An example of what an object for apicontent might look like:

```
{
  "product_id": 11,
  "product_name": "Graphic design: a beginner course",
  "product_price": 1000
 }
```

How to transfer API content to the platform

API request

Data can be transferred if you run a trigger mailing or scenario using an API request. In the request, the Examples of requests in which the content field is available:

- **Trigger launch**
- **Profile Import + trigger mailing launch**
- **Bulk trigger launch**
- **Bulk profiles import + trigger mailing launch**
- **Engage profile in scenario**
- **Import and engage profile in scenario**
- **Batch import and engage profiles in a scenario**

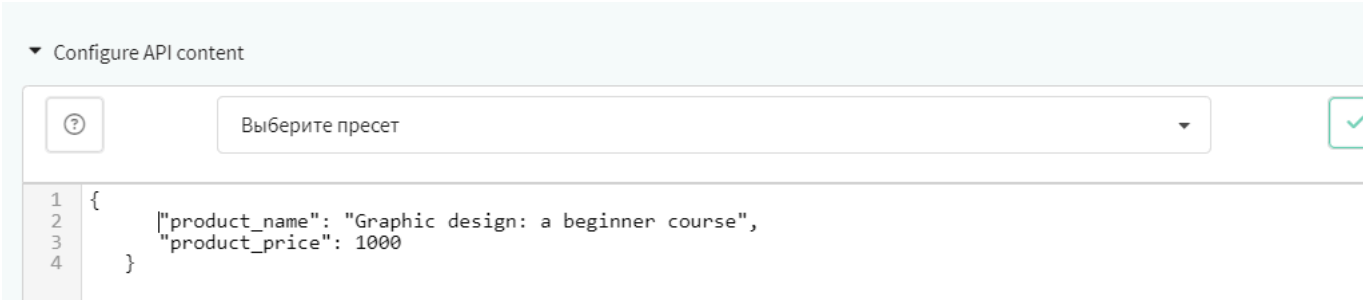
Next, this data can be inserted into the template using the `apicontent` variable:

HTML TEXT

```
<table align="center" role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%"
<tr>
<td style="background-color: #ffffff;">
<table role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%">
<tr>
<td style="padding: 20px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20
<h2 style="margin: 0 0 10px 0; font-family: sans-serif; font-size: 28px; color: #6778cd; font-wei
<h2 style="font-size: 28px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0
<p>Thank you for choosing our course:</p>
<ul>
    <li>{apicontent.product_name}</li>
    <li>{apicontent.product_price}</li>
</ul>
</td>
<tr>
<td style="padding: 0px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20p
<h2 style="font-size: 15px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0
   <p>Thank you and good luck!</p>
</td>
</tr>
</table>
```

TIP

The preview of the message with `apicontent` is available only after configuring the section Configure AP preview section and paste the object containing the `apicontent` into the editor. Click on the check mark



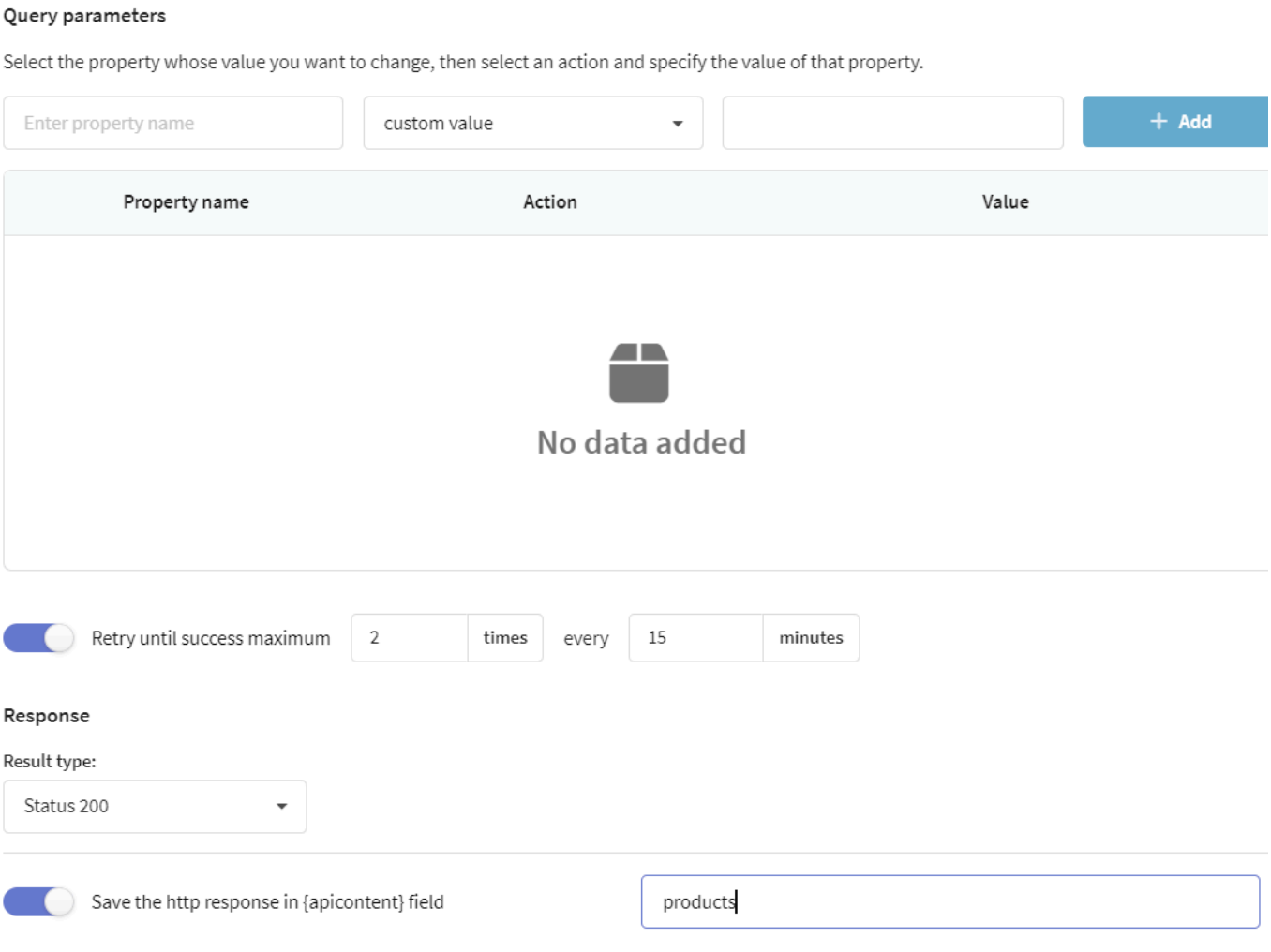
“API Request” element in the scenario

In the API Request scenario element, you can get data from an external source using a request and save variable. The information from the request can later be used in the scenario.

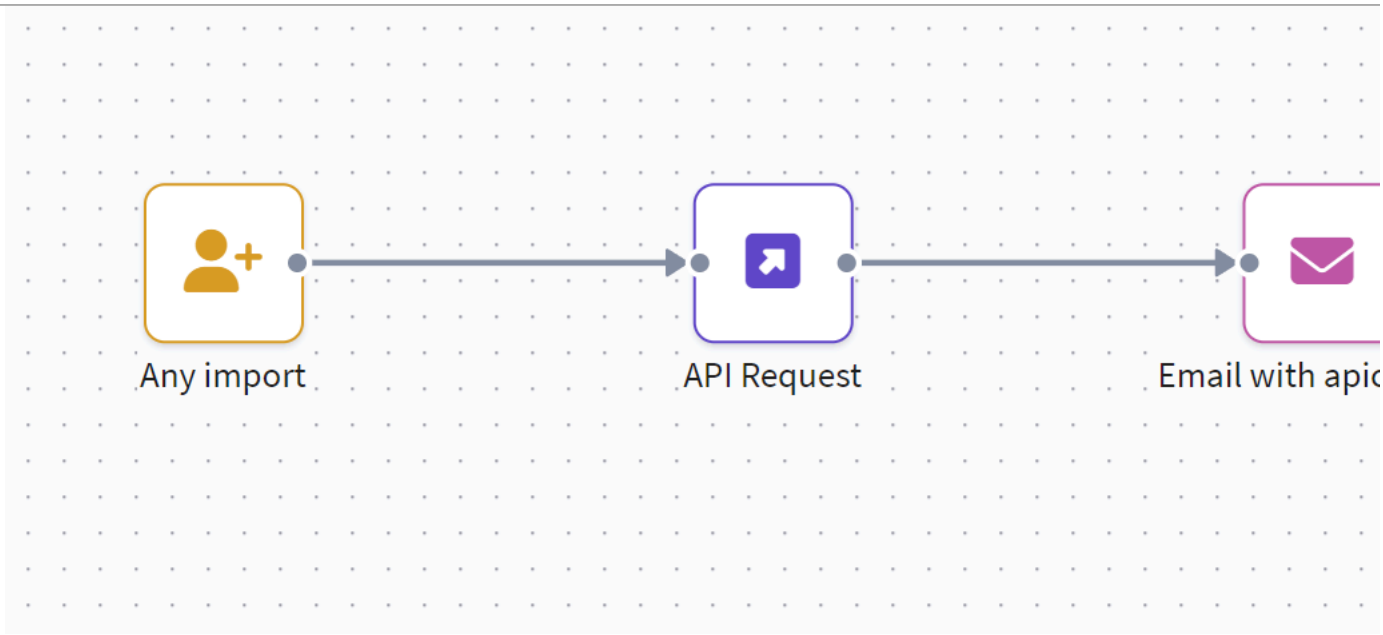
You can read about how to configure the “API Request” element [in this article](#).

Select the type of request and paste the link to the service that will return the json-object:

Click on the switch Save the http response in apicontent field and enter the name of the field under whic will be recorded and which will be used in the message template in the future:



In order to understand how information will be transmitted to `apicontent` in the future, let's look at the fol element in the required place in order to send the data received from the external service to the client v



The path to the data in the template will change slightly:

HTML TEXT

```
105 v style="border: 1px solid #6778cd; -webkit-border-radius: 5px; -moz-border-radius: 5px;border-radius: 5px; pa
106  <table align="center" role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%" style="marg
107  <tr>
108  <td style="background-color: #ffffff;">
109  <table role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%">
110  <tr>
111  <td style="padding: 20px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #
112  <h2 style="margin: 0 0 10px 0; font-family: sans-serif; font-size: 28px; color: #6778cd; font-weight: 800; li
113  <h2 style="font-size: 28px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">Thank
114  <p>Thank you for choosing our course:</p>
115  <ul>
116      <li>{apicontent.products.product_name}</li>
117      <li>{apicontent.products.product_price}</li>
118  </ul>
119  </td>
120  <tr>
121  <td style="padding: 0px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #
122  <h2 style="font-size: 15px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">We app
123     <p>Thank you and good luck!</p>
124  </td>
125  </tr>
```

TIP

The preview of the message with `apicontent` is available only after configuring the section Configure AP preview section and paste the object containing the `apicontent` into the editor. Click on the check mark

Using a loop in a template with apicontent

You can also transfer `apicontent` variable into json-object with an array.

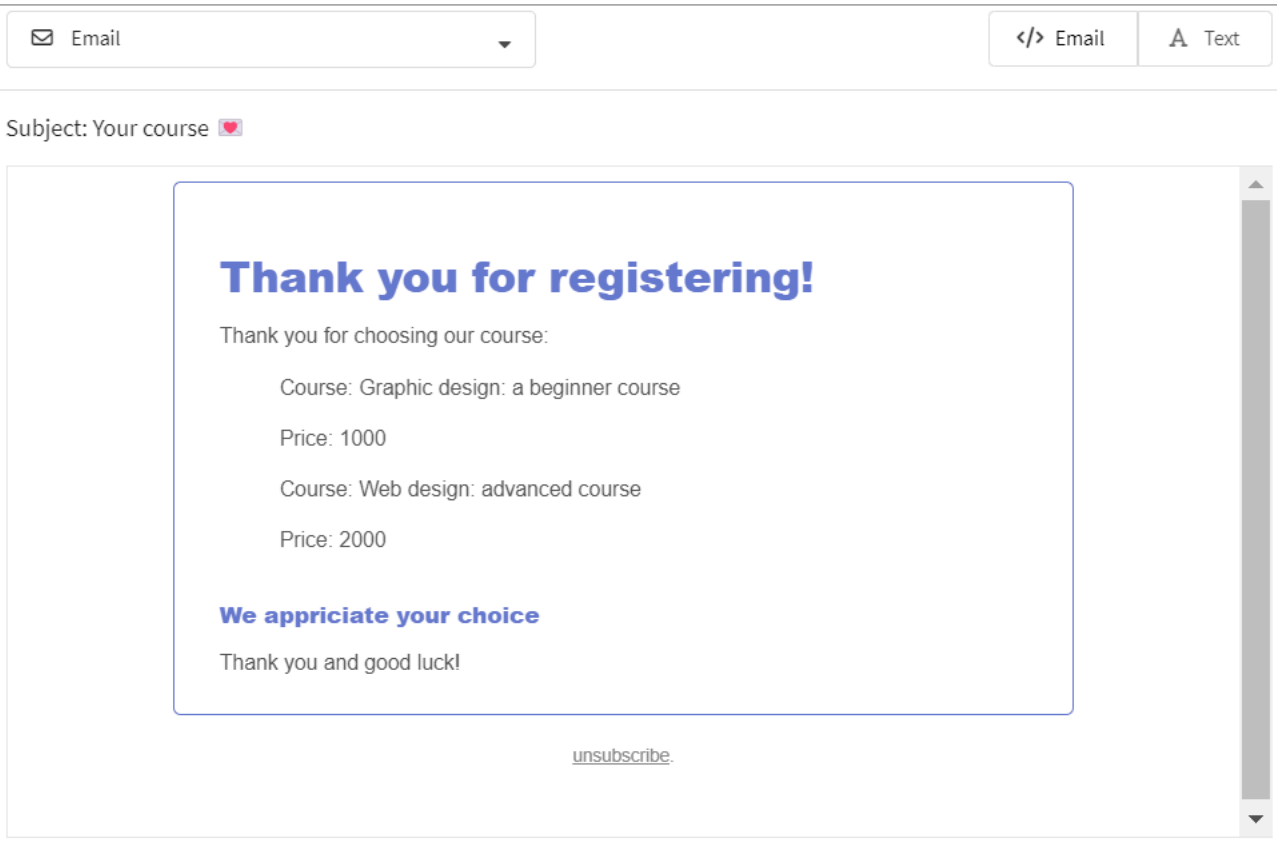
An example:

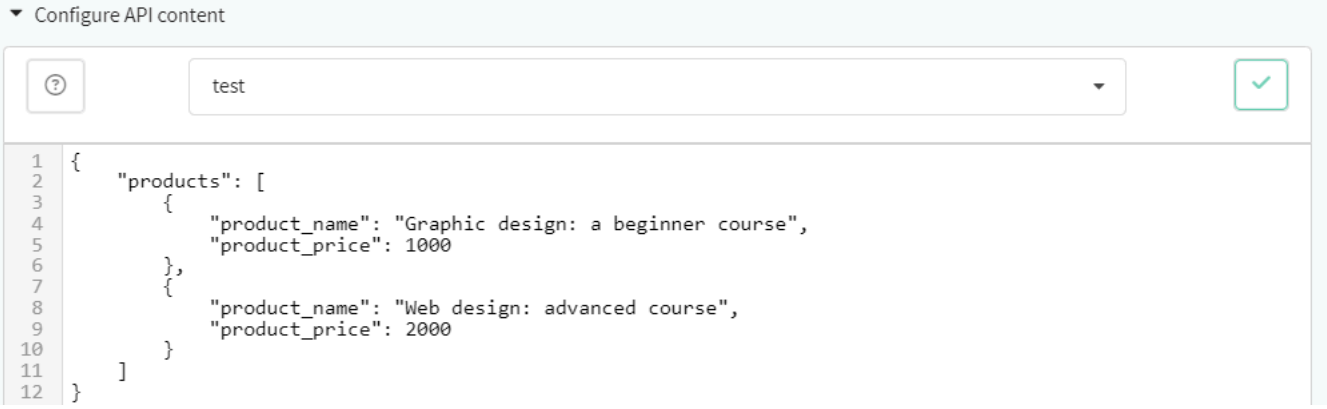
```
{
    "products": [
      {
        "product_id": 11,
        "product_name": "Graphic design: a beginner course",
        "product_price": 1000
      },
      {
        "product_id": 12,
        "product_name": "Web design: advanced course",
        "product_price": 2000
      }
  ]
}
```

To insert data from an array into a template, you need to iterate through your array. To do this, use the lo

```
{for $index $el = apicontent.products}
<div>
  <p>Course: {$el.product_name}</p>
  <p>Price: {$el.product_price}</p>
</div>
{else}
  if empty array
{end}
```

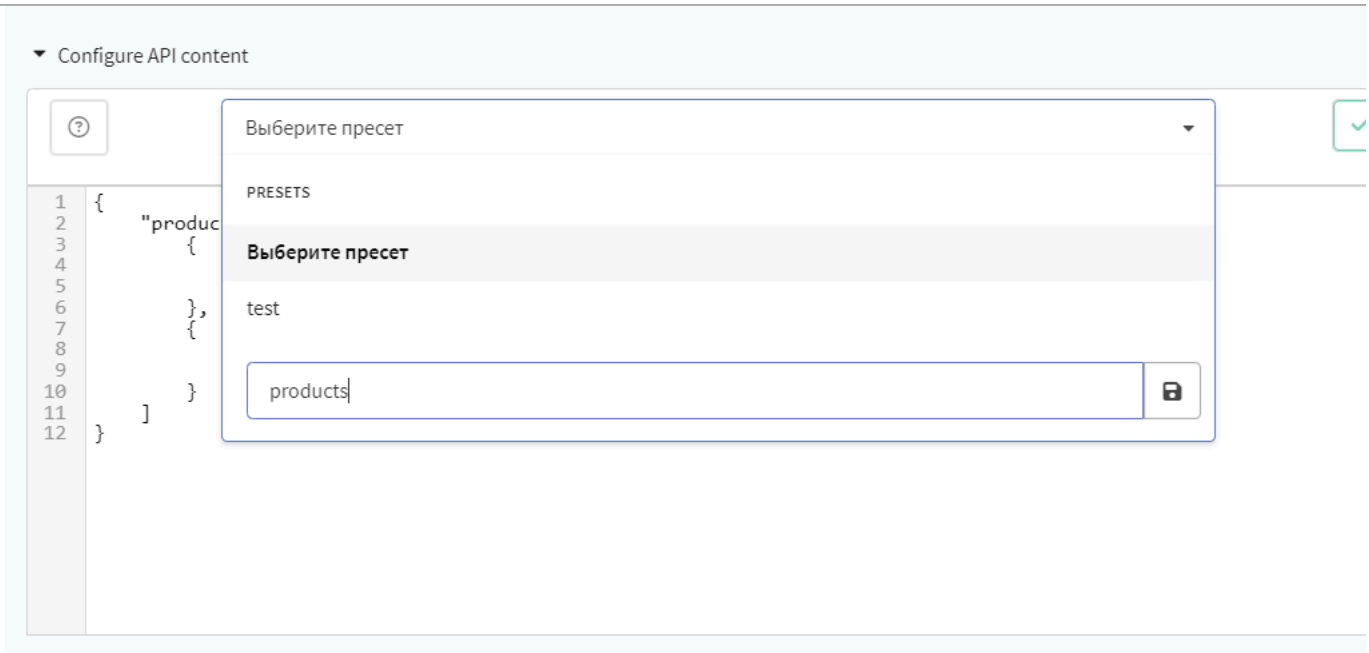
This code will insert the following data into the email:





TIP

If you regularly use the same json-bject in a variable, then you can save it as a preset and then select it



Cancel sending message if field is missing

You can configure the cancellation of the message sending if a specific field in the `apicontent` object is m do this, add a check for the field's existence and use the `{cancel}` variable.

Example:

```
{if apicontent.field_2}
field_2 не пустое
{else}
{cancel}
{end}
```

In this case, the mailing will not send the email if the `apicontent` object contains one of the following:

```
{
  "field_1": "example"
}
{
  "field_1": "example",
  "field_2": "" / or 0, null, false
}
```

This approach helps to avoid sending irrelevant emails and blocks sending if incomplete data was receiv

Use case example

Suppose you have a marketing mailing intended only for users who have started or completed a certain c `apicontent` object with information about the course progress status:

```
{
  "user_id": 123,
  "course_id": "abcadefg",
  "course_name": "Graphic Design: Beginner Course",
  "course_status": "completed"
}
```

However, if the `course_status` field is missing or empty in the request, it may indicate that the user has no this case, the email should not be sent. The email template:

```
{if apicontent.course_status}
<div>
  <span>You showed interest in the course {apicontent.course_name}.</span>
  <span>You might also like these courses: <course list></span>
</div>
{else}
{cancel}
{end}
```

Mailings

Description

In Altcraft Platform, Mailings are used for sending messages to the customers.

Mailing creates message for certain resource subscribers, based on selected template, and adds custom dynamic content, parts and fragments.

If dynamic segment is used as mailing audience, it will be counted just before every launch. Also, before profiles in suppression lists and blocked by sending policy.

Mailing types

![img] (/img/attachments/3114285/9408316.png)	Broadcasts This mailing performs one mailing to certain subscribers audience. If the audience is too big, mailing can be divided into several p as audience.
![img] (/img/attachments/3114285/9408317.png)	Regular mailings This mailing sends certain content to certain audience regularly. This mailing will never stop by itself. Subscribers base or segme
![img] (/img/attachments/3114285/9408319.png)	Multivariate testing This mailing allows you to test marketing efficiency of sent messages. You can add up to 16 messages in the mailing.
![img] (/img/attachments/3114285/9408318.png)	Trigger mailings This mailing allows you to send messages to subscribers after certain action. Mailing launch conditions can be changing subscrib

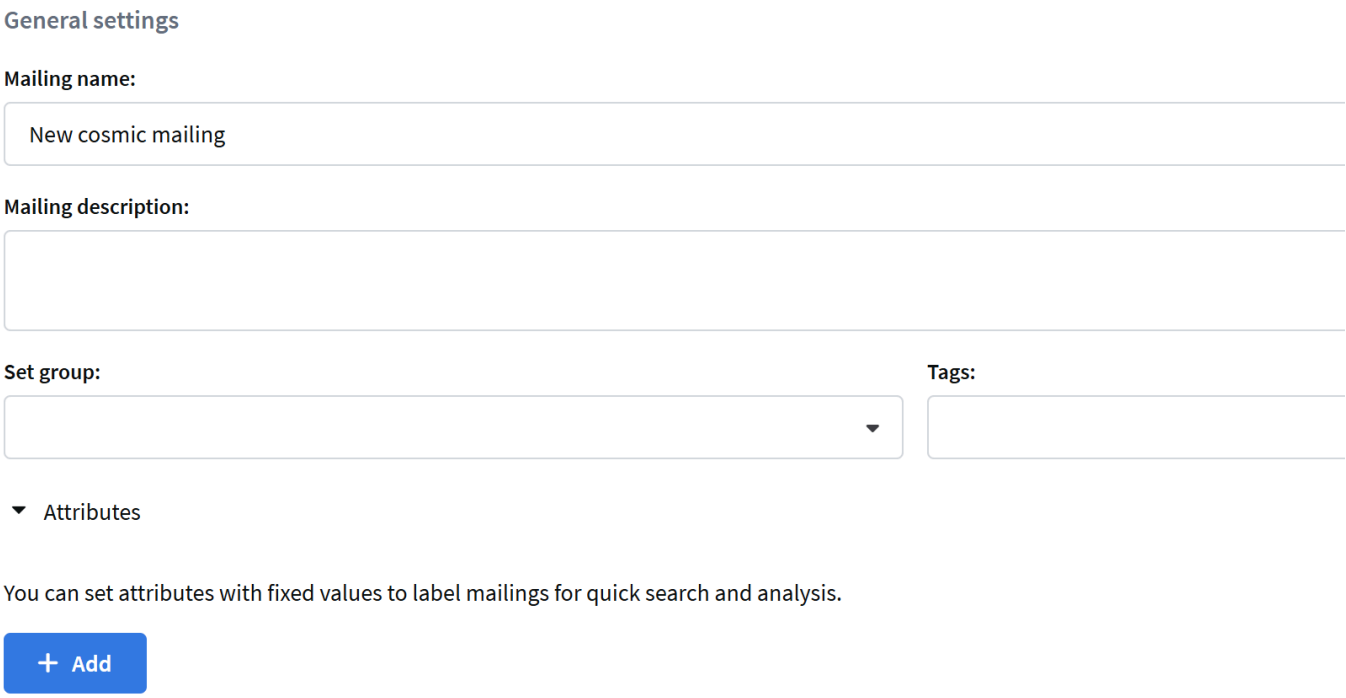
Broadcasts

Description

![img] (/img/attachments/3114538/56250462.png)	**Broadcast mailings are used when you need to communicate to a specific audience one single time. It can be a seasonal offer en sent to an audience rather than to separate customers on a non-regular basis. At the end of a broadcast mailing, every recipient** Thus broadcast messages mostly use static content, as it is sent once. Personalization options can still be a great asset to make your mass communications more human-like. To create a new broadcast, go to Mailings → Broadcasts and click + Create button. Also, you can use the calendar to create a new broadcast. </td> </tr>

General settings

In the main settings, specify the name and description of the new mailing, and if needed, set the access additional grouping. You can also specify a statistics SUBID for the mailing here. In the upper right corne mailing.



Mailing settings

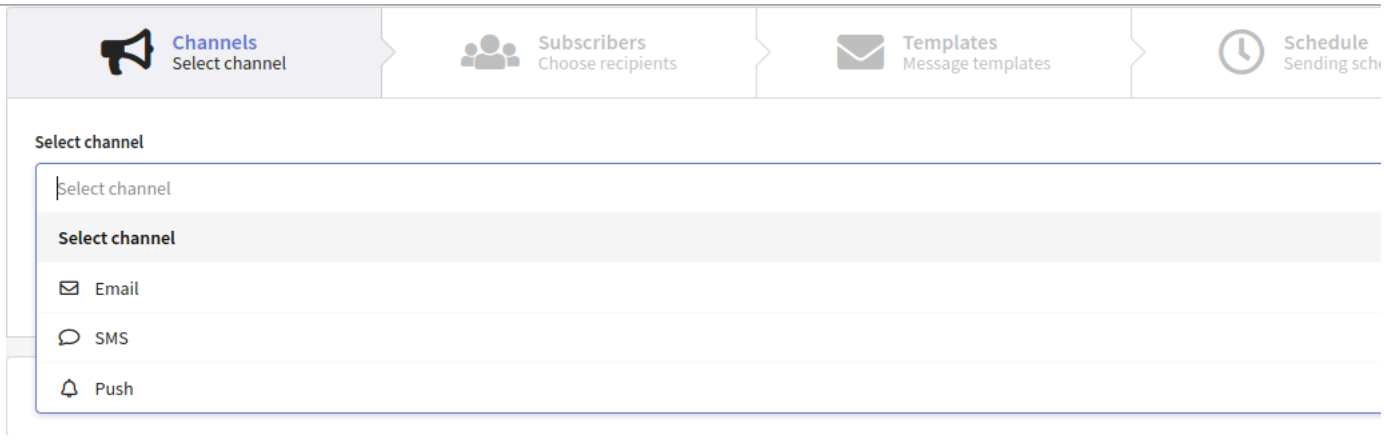
The following information is defined in mailing settings: communication channels, subscribers, template mailing.

Channels

On the first tab, you can select a communication channel you want to use. Channel you want to use mus subscriptions resource, which you can select on the next tab.

INFO

For Email sending, you can use the field with the main subscriber contact information – the Email field.



TIP

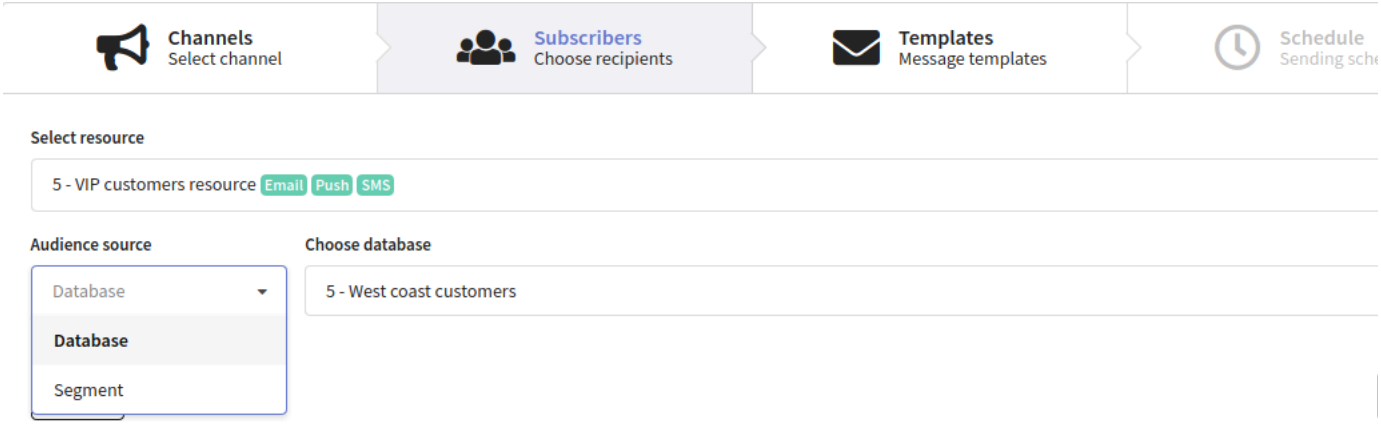
When using a push channel, you can select specific push providers for sending, as well as the sending r subscriptions taking into account filters, or last, sending to the subscription with the most recent creat these (created at the same time), then one will be selected randomly. If the mailing has filters by provid the filters are applied first, and then the most recent subscription is searched.

Subscribers

On this tab, select the resource form drop-down menu. You can use a database or segment as an audien

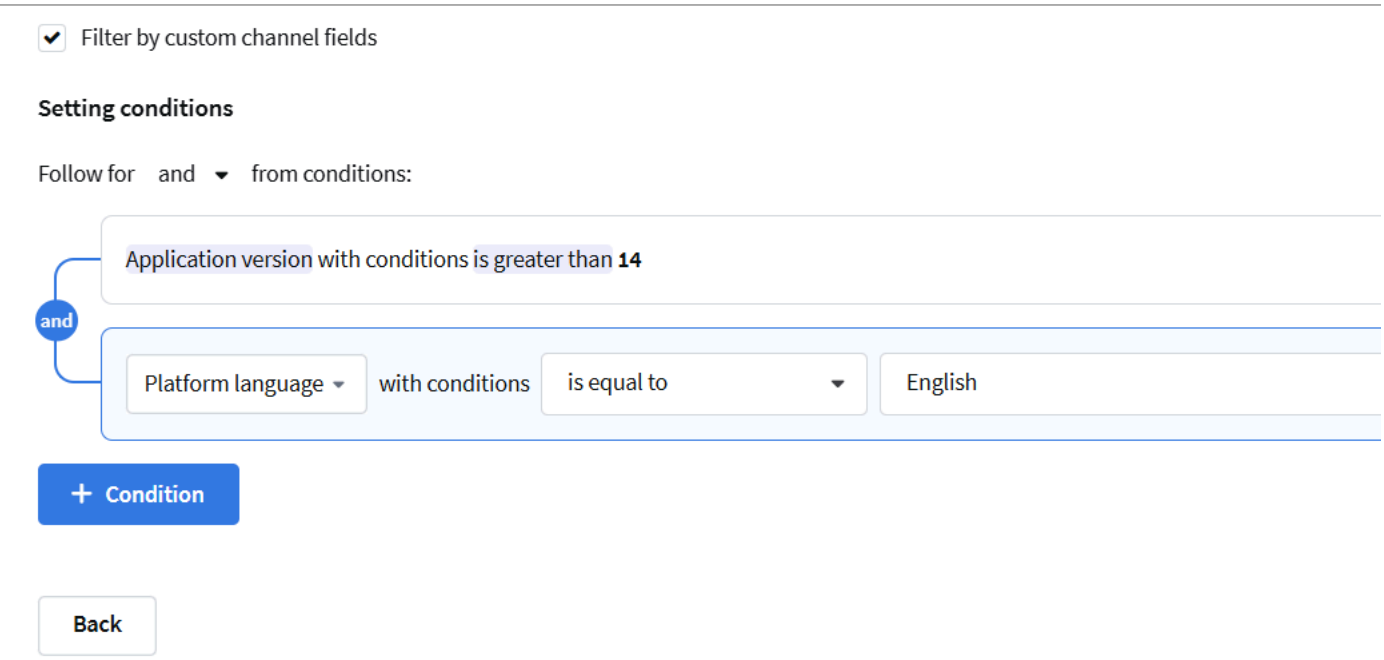
TIP

Mailings are sent to customers who subscribe to a certain resource. Resources can be a separate webs direction of your communications with customers. Select a resource. Make sure the recipients in your c



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to ha resource. In mailing settings, select the categories to which the profile should be subscribed to receive

You can also filter recipients by custom channel fields. To do this, select either a standard or custom sul condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using be met) and OR (at least one condition must be met).



Templates

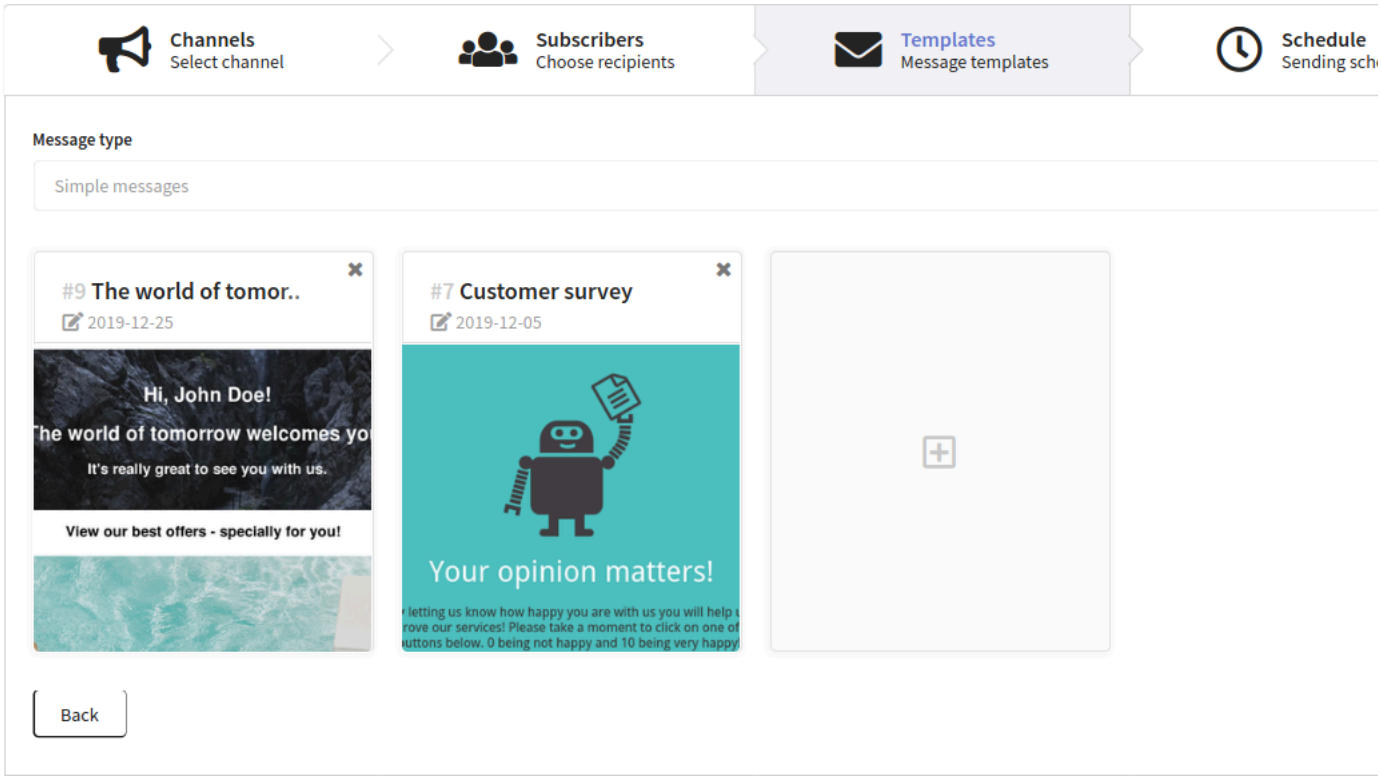
On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templa available channels.

TIP

Please, make sure, that the template has content for the selected channel.

Clcik

and select a prepared template. Any mailing type can use up to 8 templates. The broadcast mailing will using selected templates in random order in equal parts. Click on the thumbnail to preview, or click on th editor:



Schedule

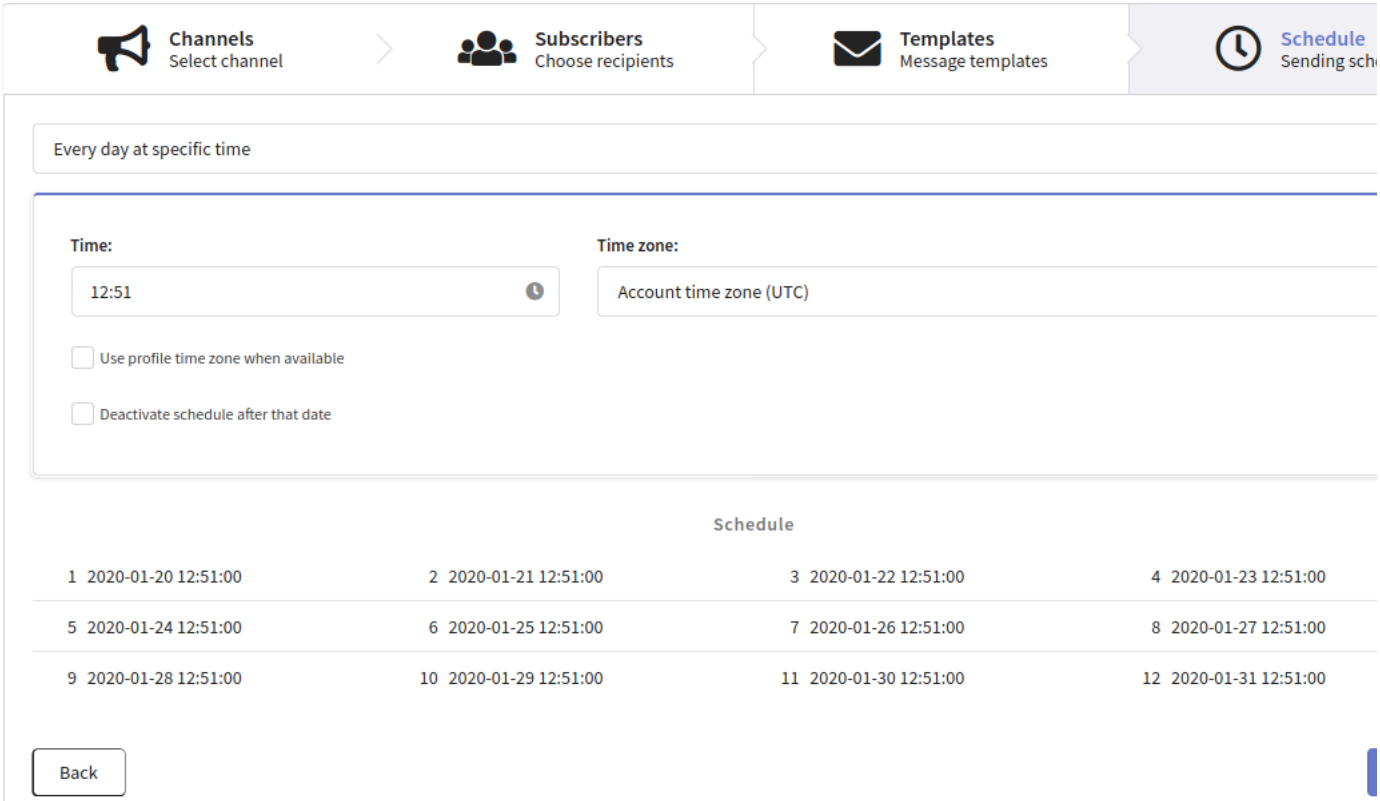
TIP

The mailing schedule is described in detail on this page.

On this tab, you can configure sending policy. During massive mailings, you need to remember about sen limits for massive mailings that depend on sender reputation and the number of sent messages. Usually, received in 24 hours.

Sending all messages at once

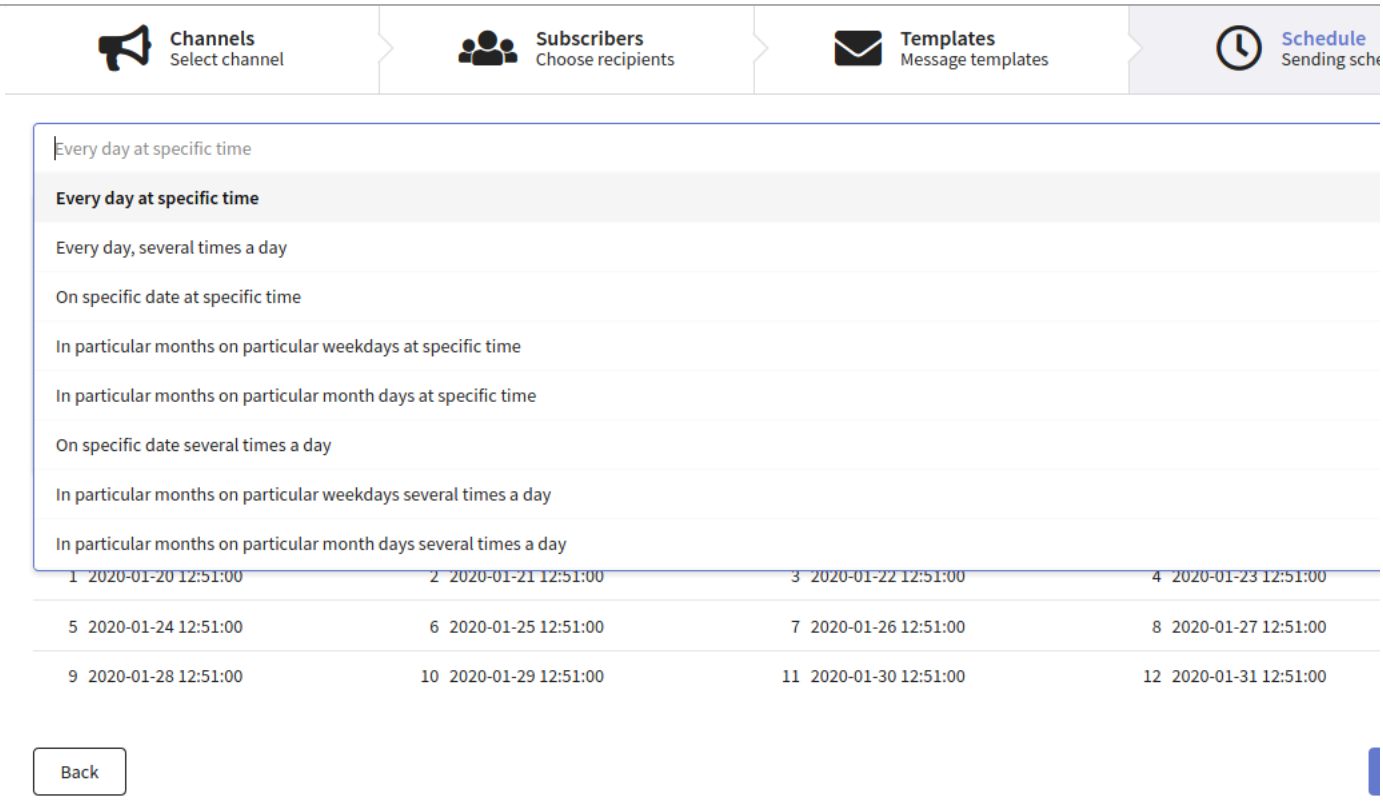
If your sender reputation is high enough, you can send all messages in 24 hours. To do this, select On sp



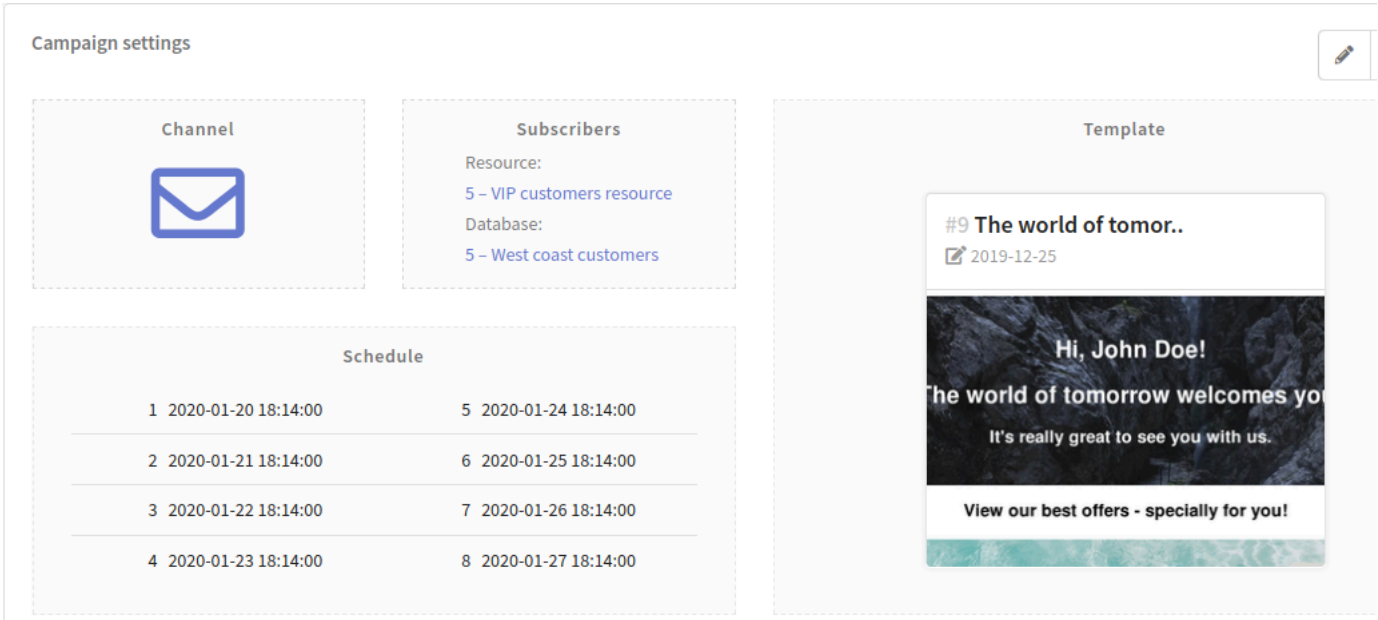
Also, you can select a time zone for sending, or Use profile time zone when available.

Partial sending

Most email providers limits are valid for 24 hours. If your mailing is over limits, we recommend you to us You can configure the number of sent messages in the Limitations block.



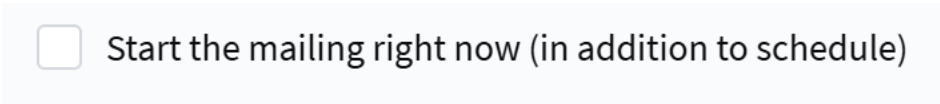
After you made your configurations, click the Finish button. Selected parameters will be displayed in the appropriate buttons to edit settings or go to the [mailing report](#).



Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part o

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



Limitation

If the number of messages in the mailing is more than the number of messages received by email provid broadcast mailing into several parts.

INFO

You can also use mailing limitations to control the loading of your sender or your company network arcl

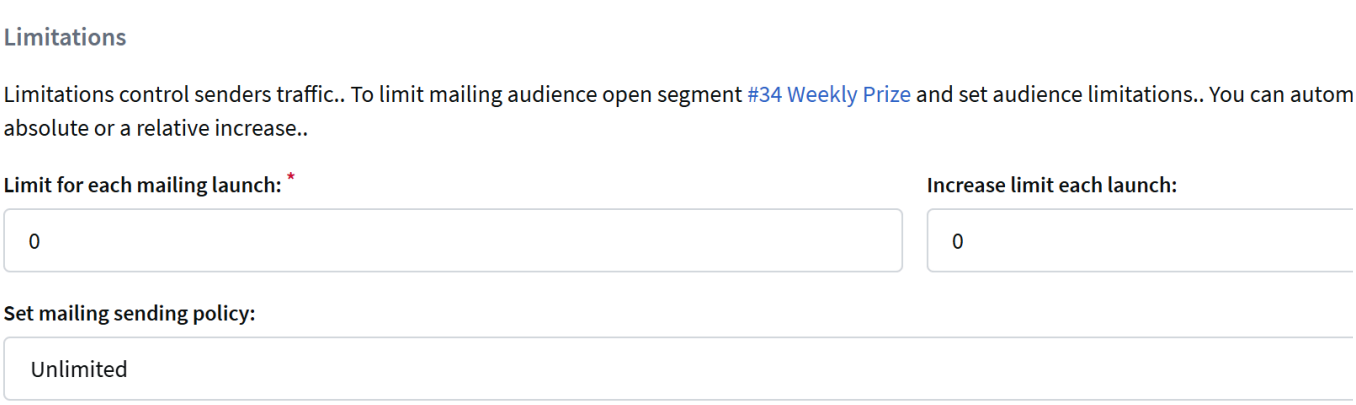
You can configure limits using these options:

- **Limit for each mailing run — the number of messages, that will be sent during the next launch by sch**
- **Increase limit each run — the number of messages or percent value that will be added during next la**

CAUTION

When the “Use profile time zone when available” option is enabled, the limit settings and instant mailin

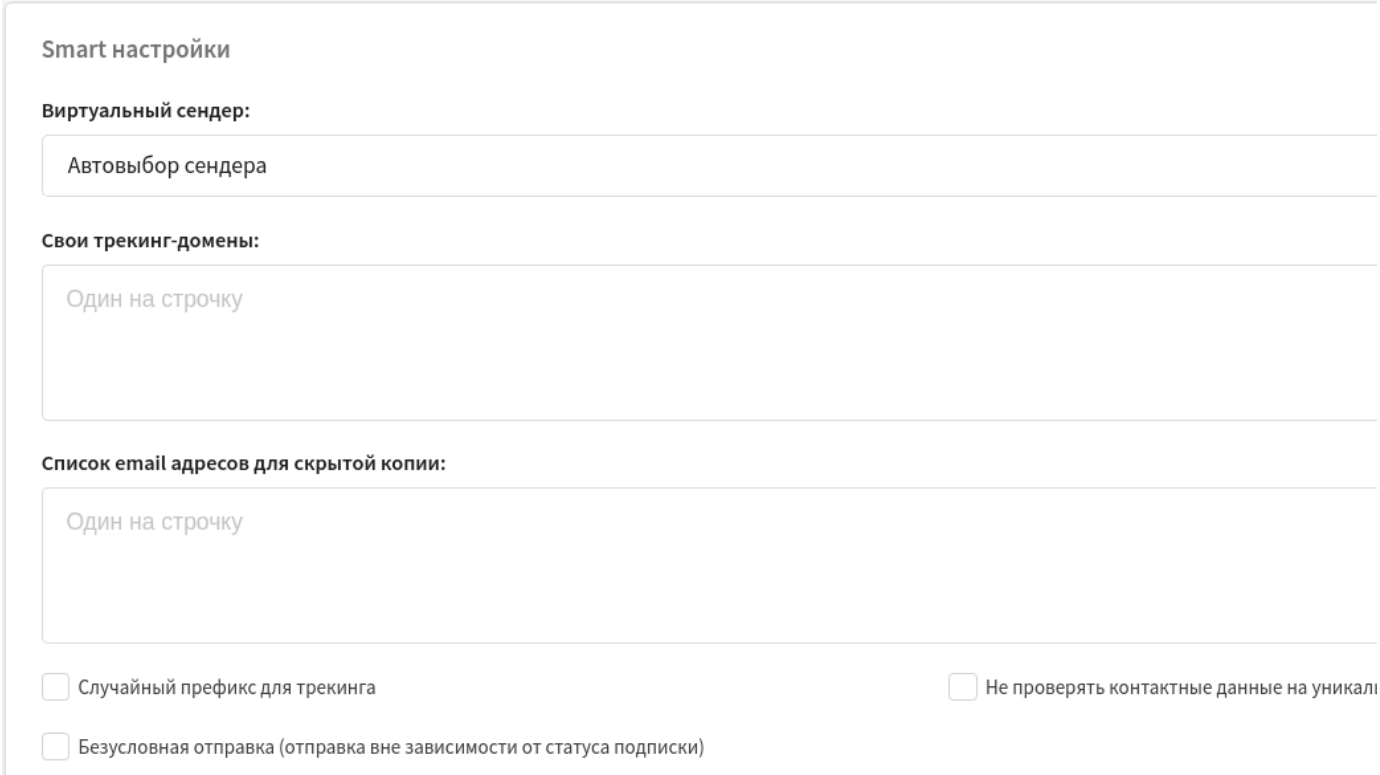
Communication policy sets limits for communication with one subscriber. By default, the mailing will use



Smart settings

The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of campaign recipients b**



Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mai watchers database. It is created the same way as a regular database, but you must enable the correspo

In Smart Settings, select the watchers database and set the message sending limit for each mailing laur

Наблюдатели за рассылкой

Наблюдателям будут отправлены письма каждый запуск рассылки если лимит не установл
порция наблюдателей в пределах лимита.

Письма будут отправлены на основной email профиля наблюдателя, подписки не учитыва

БД наблюдателей:

2 - Mailing watchers database 1000

Broadcast mailing sequence

**The Activate this mailing automatically after another one is finished option is available only for broadcas
launch a mailing after the previous mailing is finished. For this, select the mailing, after:**

Broadcast mailing sequence

Activate this mailing automatically after another one is finished:

Select a mailing

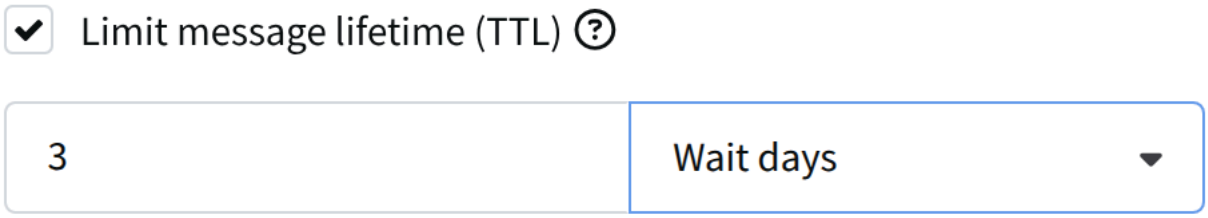
CAUTION

If you have massive mailings, please, make sure that your mailing fits the limits of email service provid

Message Time-To-Live (TTL) Restriction

**Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the
delivered within this time, sending stops and the message is removed from the queue.**

**TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is
reminder, a one-time code, or an offer that quickly loses relevance.**



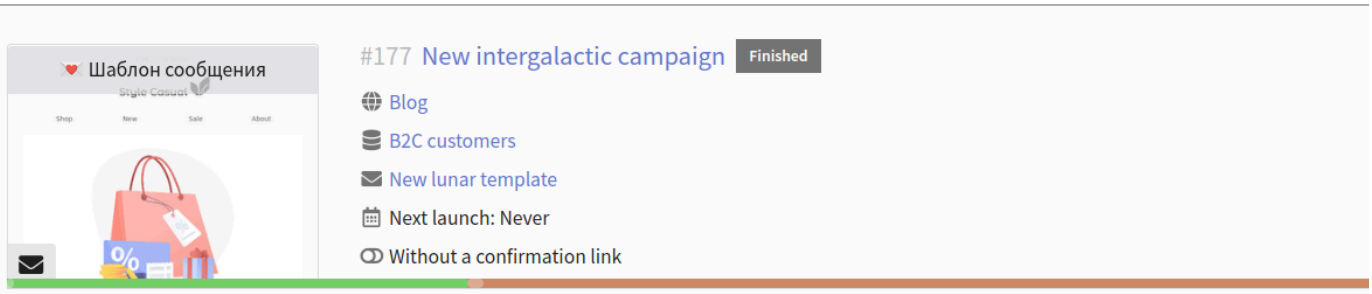
**To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, h
After the specified time expires, the system stops delivery attempts for the undelivered message and re**

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration par**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API para
Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefor
different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and**

Mailing progress

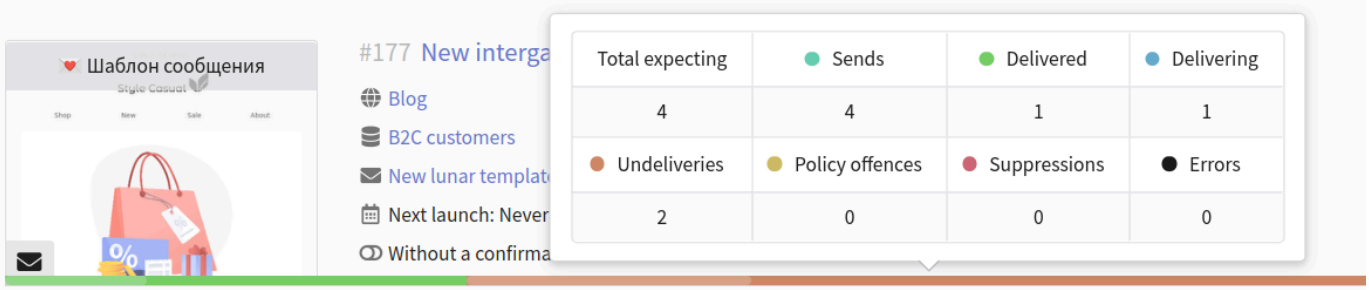
In the list of broadcast mailings, you can see the progress bar of each mailing.



Each message status has its own colour:

- **sends – the number of profiles for which messages were generated**
- **delivered – the number of profiles to which the message was delivered**
- **delivering – the number of messages in the sender queue**
- **undeliveries – the number of profiles that failed to deliver the message**
- **policy offences – the number ofprofiles affected by sending policy (see more)**
- **supressions – the number of profilesin suppression lists (see more)**

Click on the progress bar to open a table with mailing statistics:



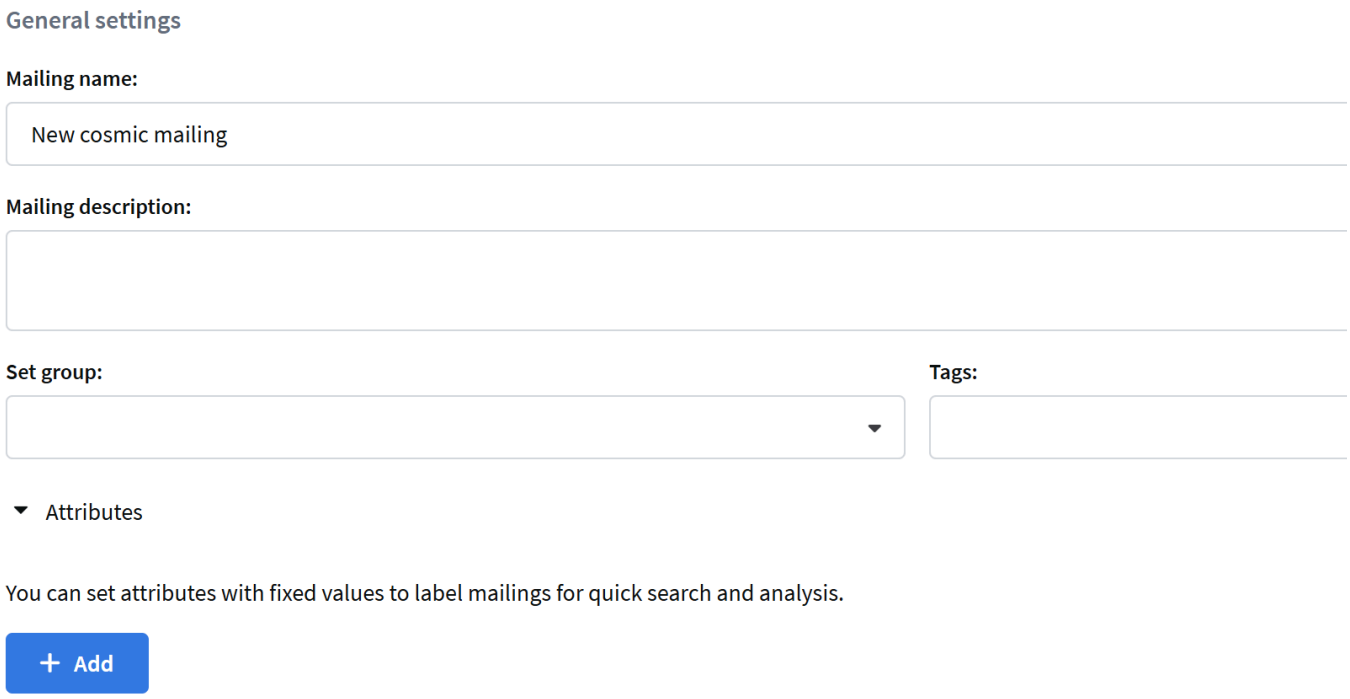
Trigger mailings

Description

![img] (/img/attachments/3114522/56250464.png)	**Trigger mailing is our ultimate tool for composing transactional messaging. In other words, if you select this mailing, a marketing has occurred. Trigger mailings send one message to each subscriber who fits trigger launch conditions.** It can be used as a standalone feature or as a part of a Scenario tool – a state of the art solution for composing marketing automation and creating customer journey paths. To create a new mailing, select Mailings → Triggers and click on the "Create" button, or you can use the mailings calendar. </td> </tr>

General settings

In the main settings, specify the name and description of the new mailing, and if needed, set the access additional grouping. You can also specify a statistics SUBID for the mailing here. In the upper right corne trigger, putting it into standby mode:



Mailing settings

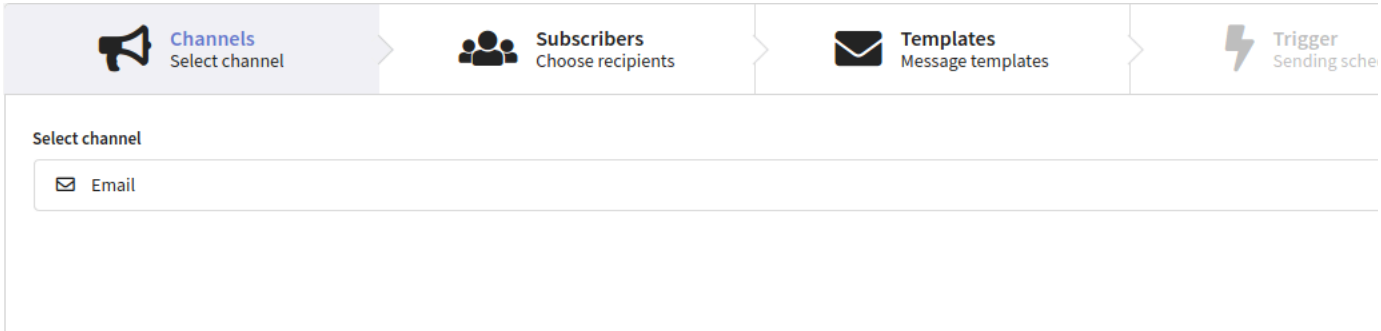
The following information is defined in mailing settings: communication channels, subscribers, template mailing.

Channels

On the first tab, you can select a communication channel you want to use. Channel you want to use mus subscriptions resource, which you can select on the next tab.

INFO

For Email sending, you can use the field with the main subscriber contact information – Email field.



Subscribers

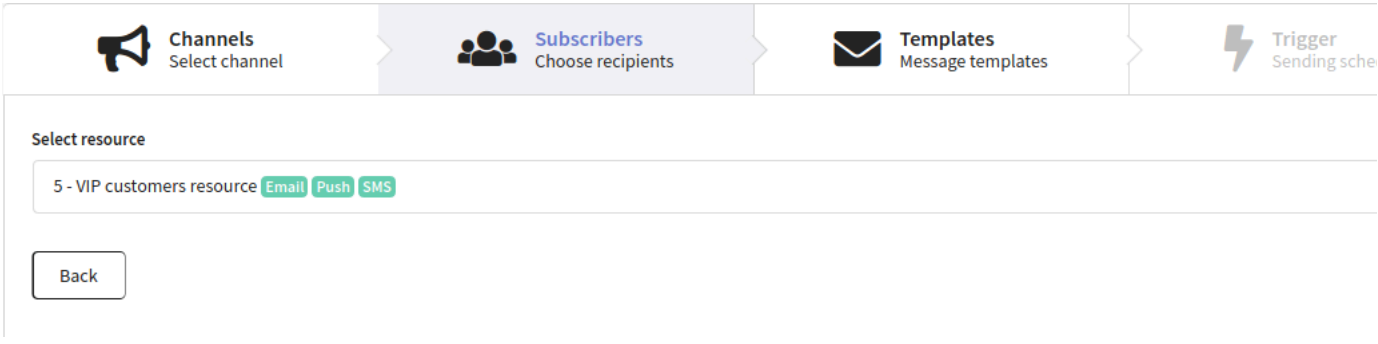
On this tab, select the resource form drop-down menu. There is no audience source in trigger mailings. T event trigger.

INFO

Mailings are sent to customers who subscribe to a certain resource. Make sure that recipients are subs

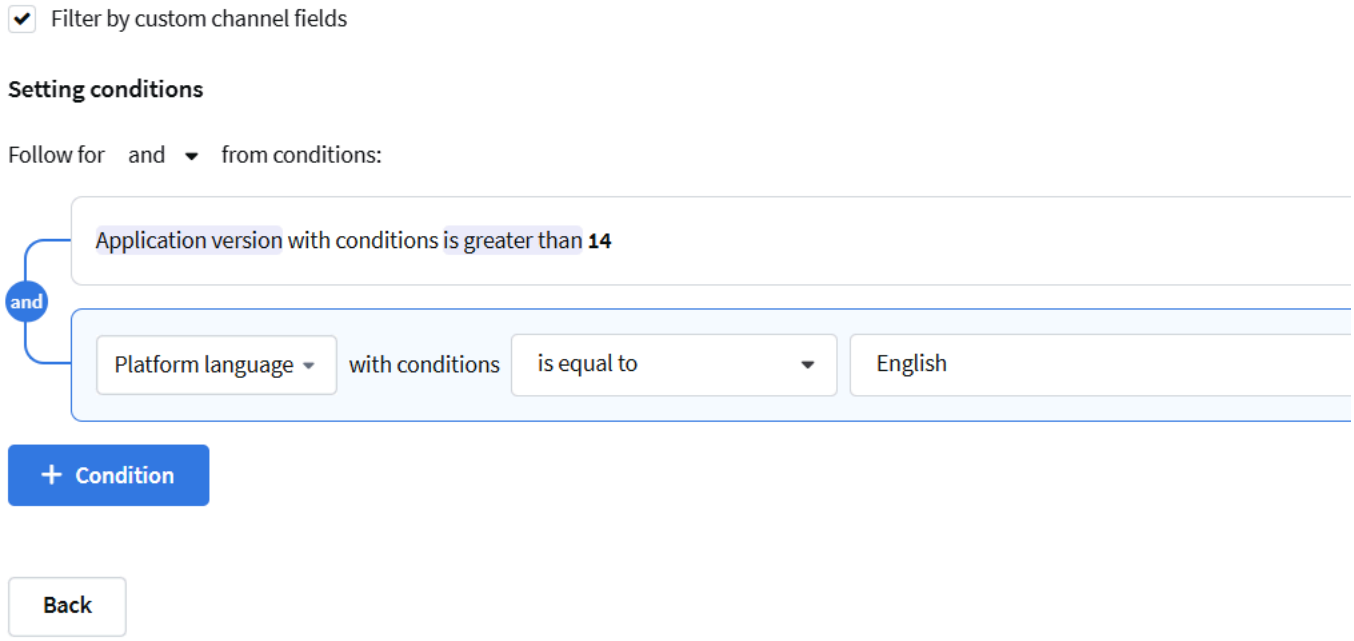
CAUTION

If database import is used as a trigger event, make sure that the database is available in the resource s messages will not be sent.



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to ha resource. In mailing settings, select the categories to which the profile should be subscribed to receive

You can also filter recipients by custom channel fields. To do this, select either a standard or custom sul condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using be met) and OR (at least one condition must be met).



Templates

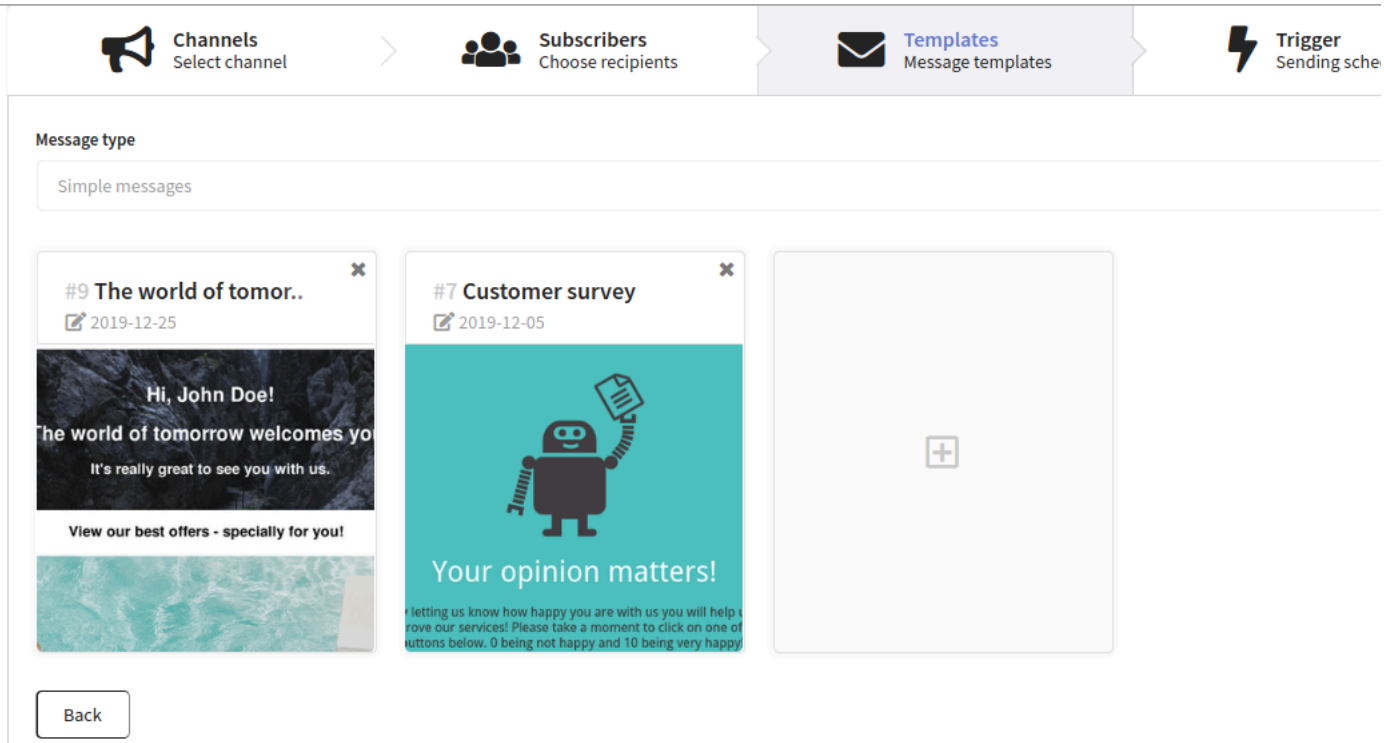
On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templa available channels.

TIP

Please, make sure, that the template has content for the selected channel.

Click the "Add" button and select a prepared template. Any mailing type can use up to 8 templates. The each customer using selected templates in random order in equal parts.

Click on the template miniature to see a preview, or click on the template name to open the template edi



Trigger

Trigger mailings use the sender's priority channel to deliver messages instantly after a trigger event occ

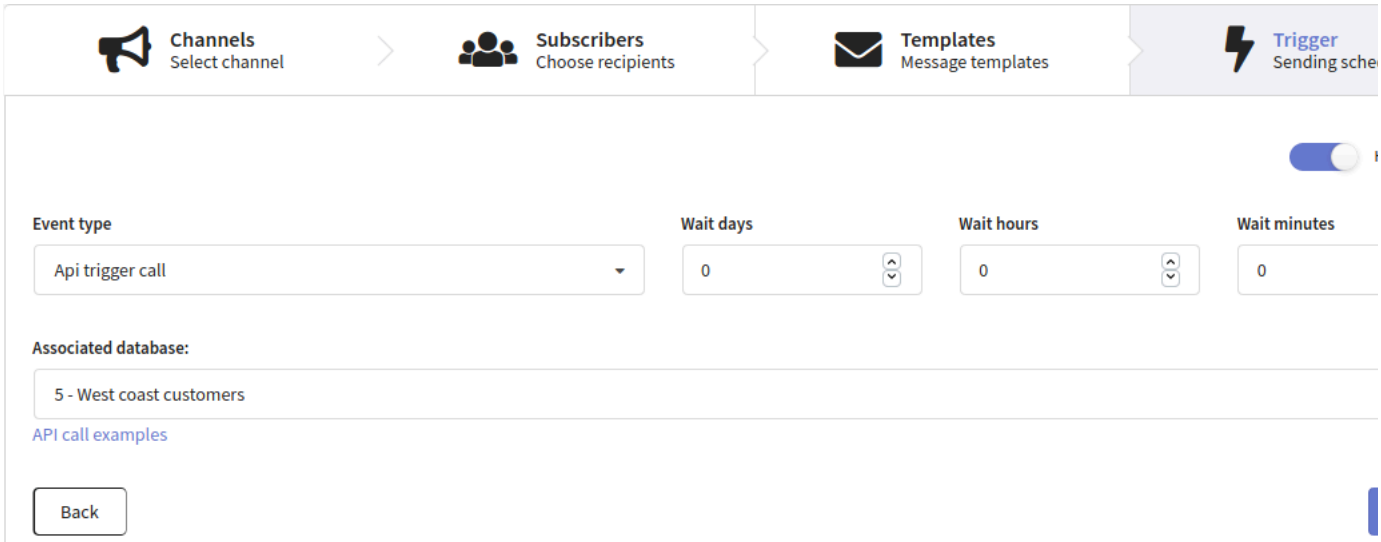
You can also enable the Highest priority — ideal for urgent transactional messages. In this case, content priority queue in the message broker (`oxy_triggers_prior`), so the message is prepared faster.

After content is generated, the message is moved to the general sending queue, where priority is no long in the order they arrive. The only exception is the email channel with AKMTA. In this case, a dedicated v side to deliver high-priority messages faster. For all other channels, priority doesn't affect the sending or

INFO

If the mailing is created in a high-priority scenario, the Highest priority option is automatically enabled, cannot turn it off manually. The toggle value is not overwritten: as soon as the scenario is no longer hig restored.

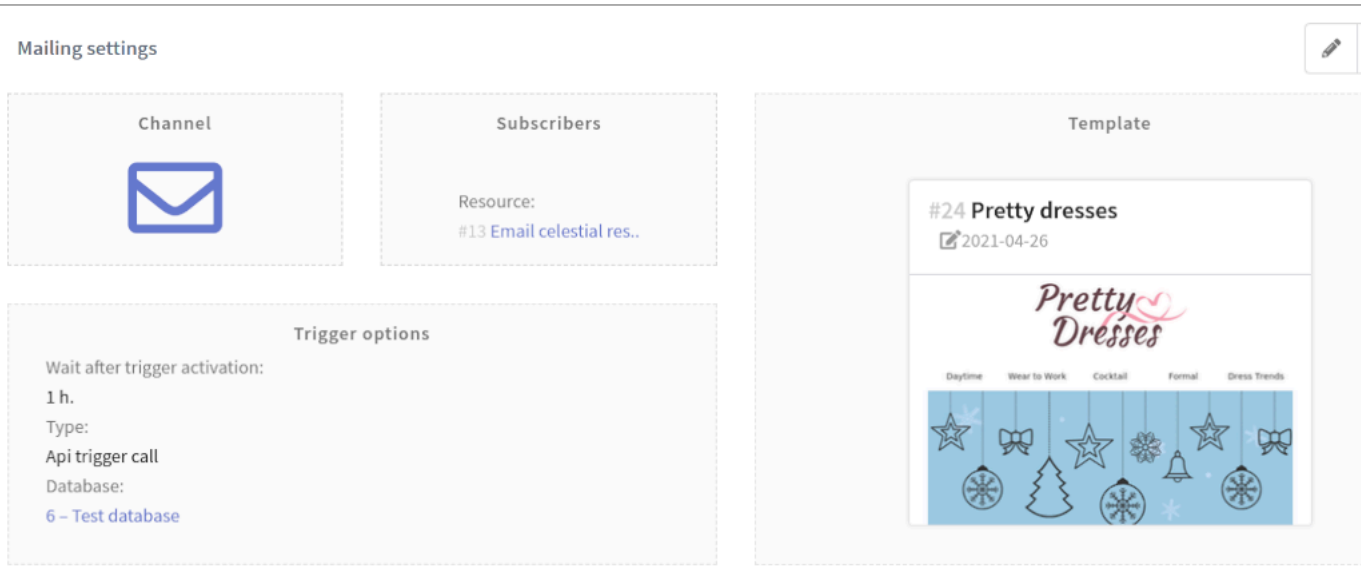
Additionally, you can set a delay for each trigger in days, hours, and minutes.



TIP

The types of trigger events are described in the next paragraph of this article.

When you've done with settings, click "Finish". The parameters you've chosen will be displayed in the "T buttons in the upper right corner to edit settings, check the log of the mailing or see the channel report.



NOTE

In the mailing log for trigger mailings, the Total time spent field may display a value of 30 seconds, ever immediately after the triggering event occurred.

Time	Type	Value
2026-01-19 11:29:53	Trigger activated	
2026-01-19 11:29:53	Database ID	6
2026-01-19 11:30:23	Number of subscribers received from the source	1
2026-01-19 11:30:23	Number of subscribers found	1
2026-01-19 11:30:23	Number of email recipients	1
2026-01-19 11:30:23	Trigger finished	
2026-01-19 11:30:23	Mailing finished	
2026-01-19 11:30:23	Total time spent	30.066781361s

For trigger mailings, the Total time spent value is recorded when the mailing is completed, not at the m actually sent. During this interval, the trigger continues to accept new events, after which the campaig statistics for the period are recorded.

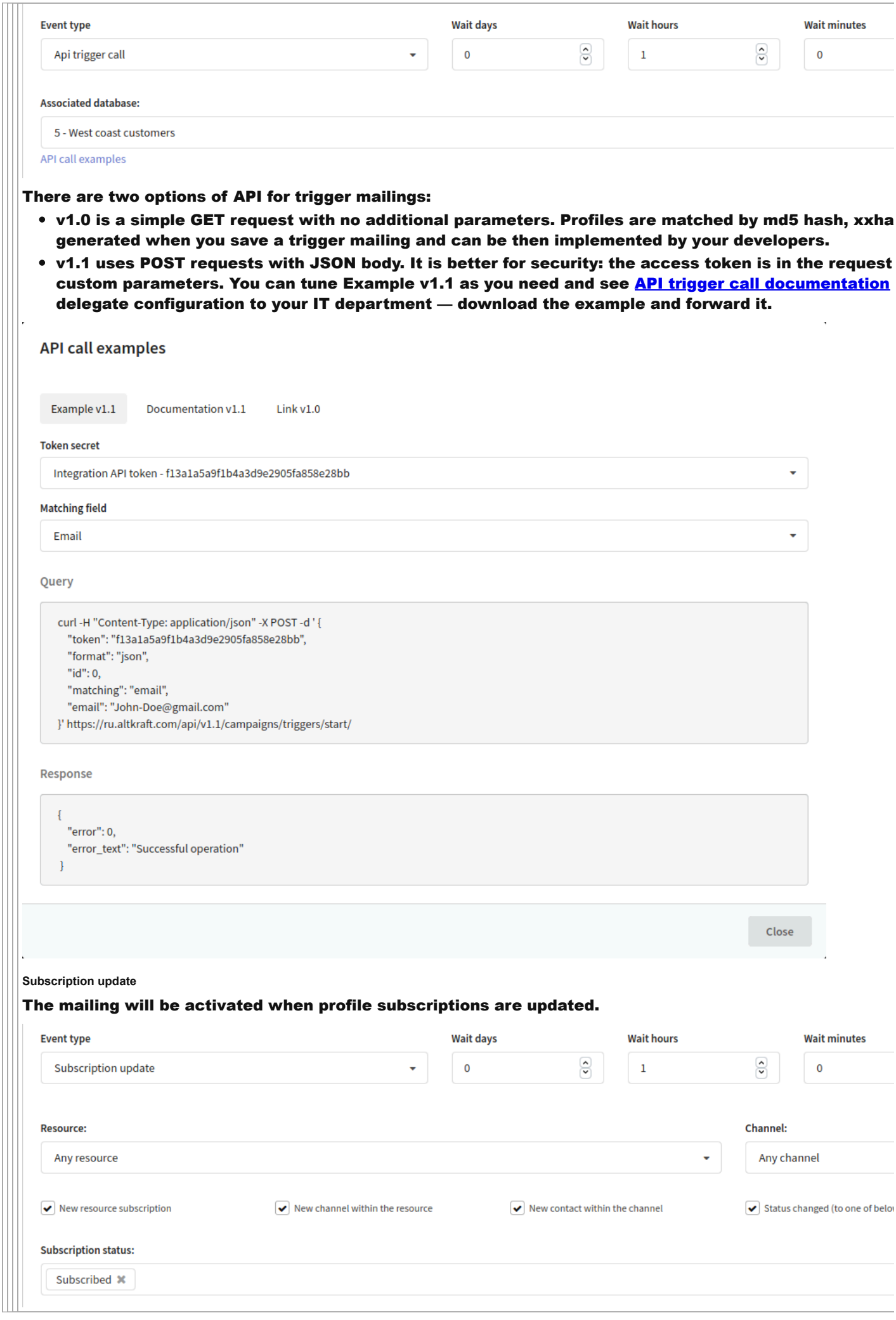
As a result, the same time value is shown in the log regardless of when a particular message was actu

To find the actual time of message generation and sending, use the profile history, where the correspor

Trigger types

API trigger call

API-triggered mailings are effective means of integration with your websites, apps, CRMs or other netwo specific steps of your digital workflow. Such triggers can work with one specific profiles database.



There are two options of API for trigger mailings:

- **v1.0 is a simple GET request with no additional parameters. Profiles are matched by md5 hash, xxha generated when you save a trigger mailing and can be then implemented by your developers.**
- **v1.1 uses POST requests with JSON body. It is better for security: the access token is in the request custom parameters. You can tune Example v1.1 as you need and see [API trigger call documentation] delegate configuration to your IT department — download the example and forward it.**

Subscription update

The mailing will be activated when profile subscriptions are updated.

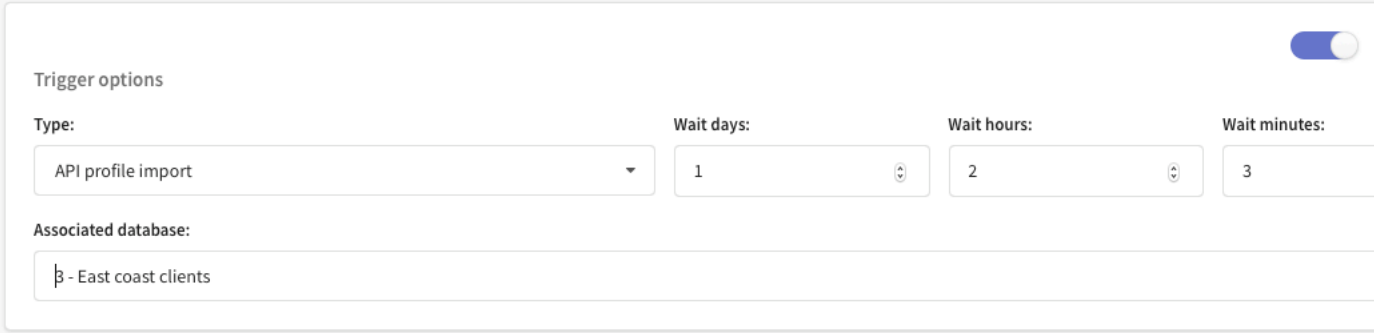
- **New resource subscription** — when a profile gets subscribed to a resource for the first time.
- **New channel within the resource** — when a profile gets subscribed to another channel of commu for the first time.
- **New contact within the channel** — works for each new contact entry within a channel of a resou this channel.
- **Status changed** (to one of the below) — when subscription status is changed for a profile.
- **The first 3 options are selected** — when any subscription entry is added to a profile.
- **No option selected** — nothing would happen.

CAUTION

Please, take note that this trigger works only for subscriptions with a “subscribed” status.

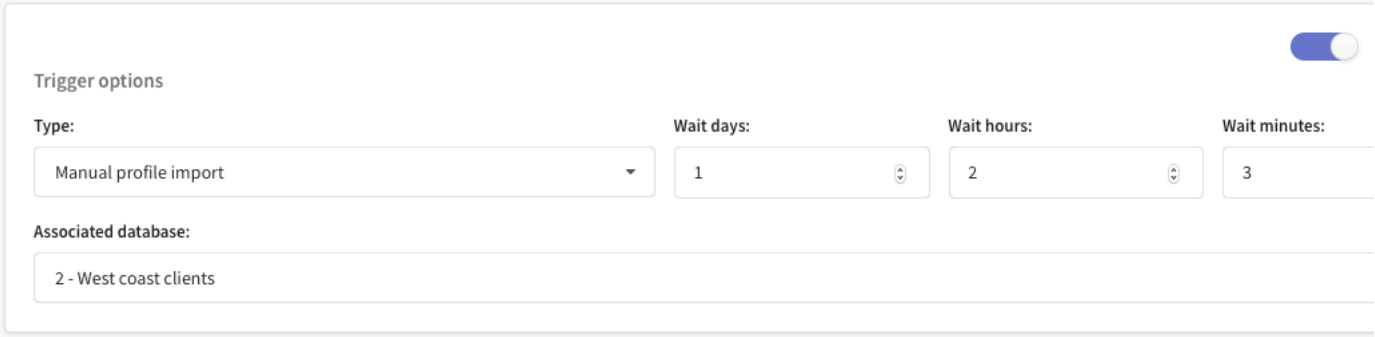
API profile import

The mailing will be activated when a new profile is created by your site, app or CRM API request.



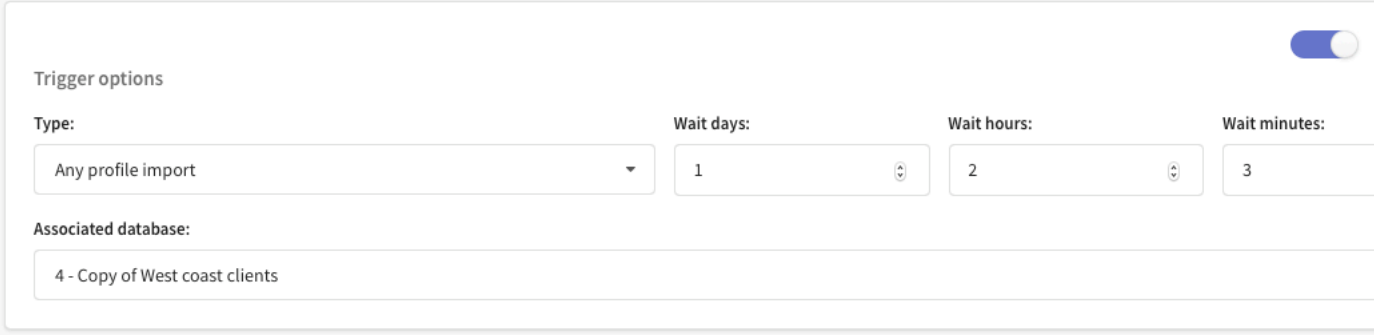
Manual profile import

This trigger will activate a mailing when you add customer profiles manually.



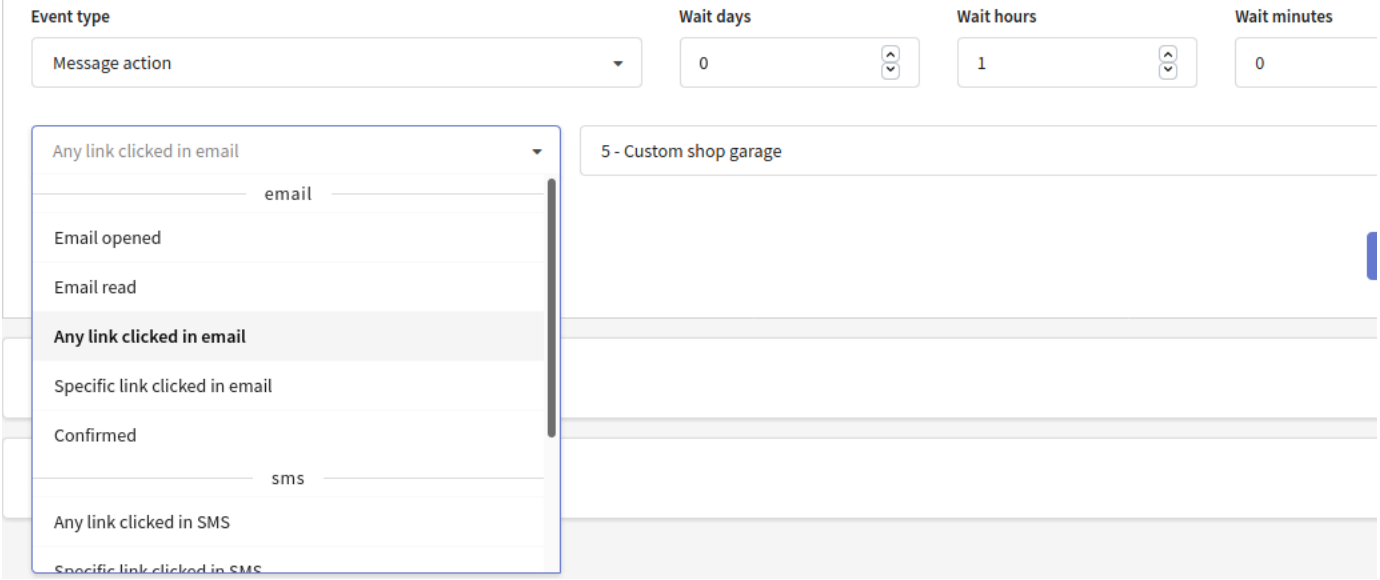
Any profile import

This trigger will activate a mailing when you import a profile to a certain database by any means.



Message action

This trigger is based on default Altcraft message tracking. Mailings can be activated by various systems link clicks, you can specify a link or select Any link clicked.



CAUTION

The Any Click trigger is activated for the first click on any link in the message within a 24-hour period. T first click on the specified link within the last 24 hours.

INFO: Popup form submission

If you want to launch a mailing using the "Popup Submission" trigger, you need to provide conditions for configured in the `matching_mode` and `matching` parameters, for example:

```
acpopup('create', 'AP-6eVbo2dP2Nnh0wtBp', {
    matching_mode: 'email',
    matching: {
        email: 'john@example.com',
        db_ids: [1]
    },
    smid: 'send_message_id',
})
```

If matching is not configured and you still need to set up a trigger campaign for popup submissions, use action is performed — Popup custom event.

Add the following code to the JS section of your popup settings:

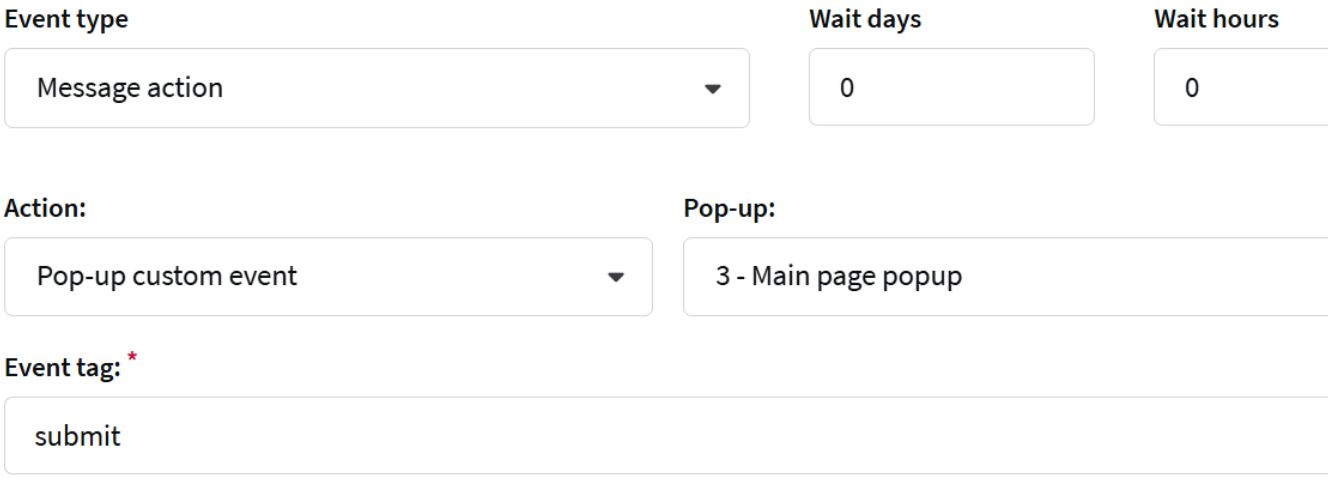
```
document.querySelector('[data-ac-popup-container]')?.shadowRoot?.getElementById('ac-form')?.addEventListener('submit', (event) => {
    const formData = new FormData(event.currentTarget);
    const email = formData.get("{{popup('form' 'email' 'fieldData')}}");

    if (email) {
        acpopup('sendEvent', 'AP-484uAZEXfQkR7d9ix', {
            web_event_name: 'submit', // custom event identifier
            matching_mode: 'email',
            matching: {
                email,
                db_ids: [1], // profile database ID
            },
        });
    }
});
```

What you need to change in the code:

- `AP-484uAZEXfQkR7d9ix` **— replace with your popup ID**
- `db_ids: [1]` **— specify your profile database ID**
- **If the email input field has a different name, replace `'email'` in `formData.get()`**

Trigger setup: in the "Event name" field, specify the same value as in `web_event_name` (in this example —

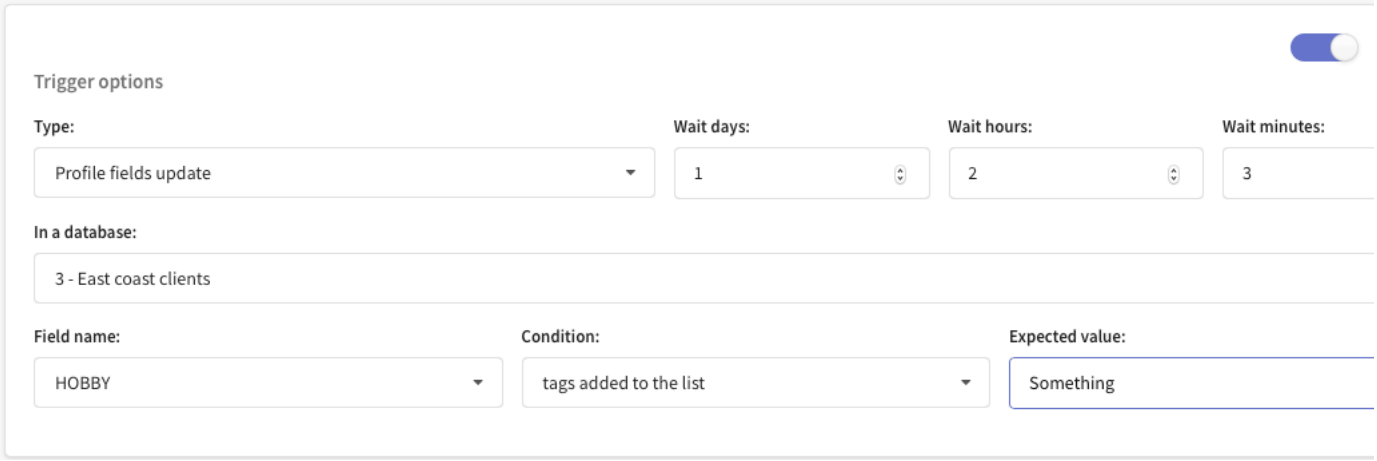


Please note that the code works when manually installing the popup on your website. Additional configu installing via a tag manager.

For more information about the `acpopup` **function and how to manually place a popup on your website, se [management via script]**

Profile fields update

Data update trigger detects changes in a customer profile. When a field value is changed to meet specifi launched.



Conditions vary depending on the type of field:

Condition	Field type	Description
changed	**all types**	**Any change in the profile field**
equal	**all types except *tags, object, array of objects***	**The new value is equal to the expected one**
not equal	**all types except *tags, object, array of objects***	**The new value is not equal to the expected one**
not empty	**all types except *tags***	**The field takes any non-empty value**
empty	**all types except *tags***	**All values have been deleted for the profile field**
tags added to the list	***tags***	**The list of tags is formed in the trigger**
tags removed from the list	***tags***	**The list of tags is formed in the trigger**
property changed	***object***	**The property of the field whose changes should be observed is specified.**
property equal	***object***	**The field property and the expected value are specified.**
property not equal	***object***	**The field property and the expected value are specified.**

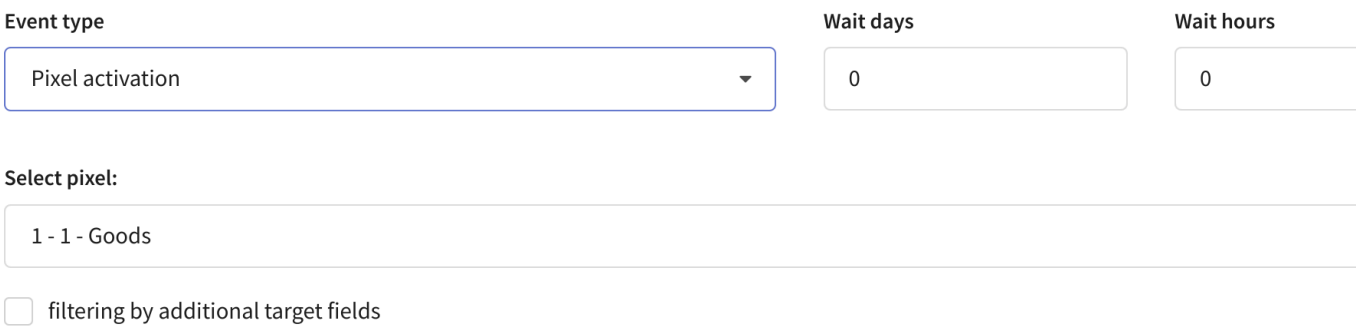
Condition	Field type	Description
element's property changed	***array of objects***	**The path to an array element is specified. If the property of this element changes, the trigger will be activated.** **When choosing the path to the element property, you also select the data type to which the compared value should be converted. Compa bool. It is not possible to compare complex data types (object, array). In the example below, you won't be able to track changes in the p** `[ { "key_1": [1,2,3,4,5], "key_2": 2 }, { "key_3":"value", "key_4": [1,2,3,4,5] } ]` **Please note that the dropdown list contains available paths to the field property, but the index of any element is set to 0 by default — `$[` manually enter the index in the JSONPath format — `$[1].key_3`.**
element's property equal	***array of objects***	**The field property and the expected value are specified.**
element's property not equal	***array of objects***	**The field property and the expected value are specified.**
property added	***array of objects***	**An element is added to the array.**
property deleted	***array of objects***	**An element is deleted from the array.**

CAUTION

This type of trigger can be used with links in templates that are configured to update profile fields. Plea are attached to the link clicks (see above), they will also work.

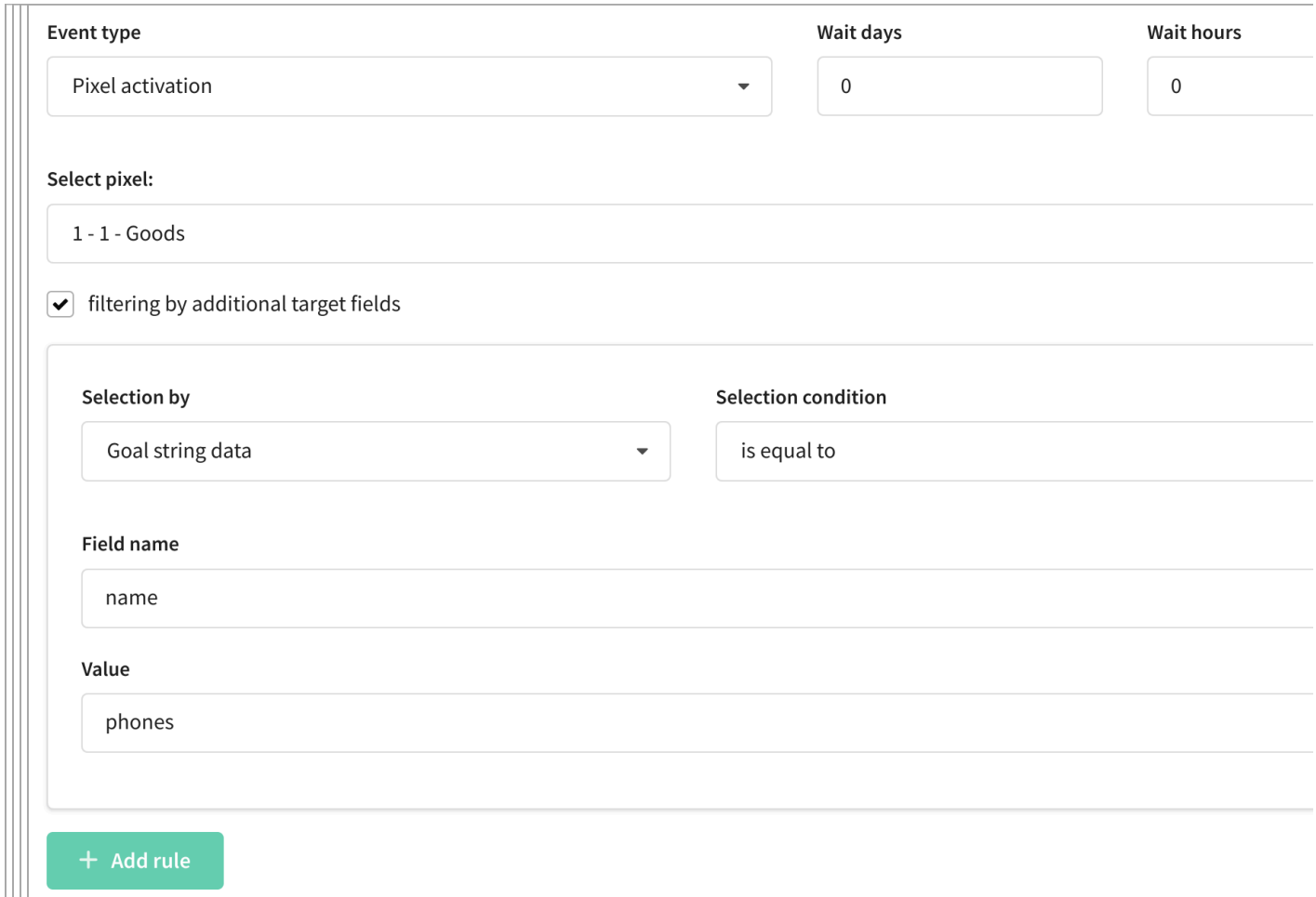
Pixel activation

Altcraft pixel is a highly customizable tracking technology stack, that can be easily integrated with your can as well trigger a mailing: when your customers browse your website or use your applications pixels mailings and workflows when it is needed:



It is also possible to track not only the general event of pixel activation. When activated, the pixel can re registration event, you can pass the goal value and other additional information to the platform.

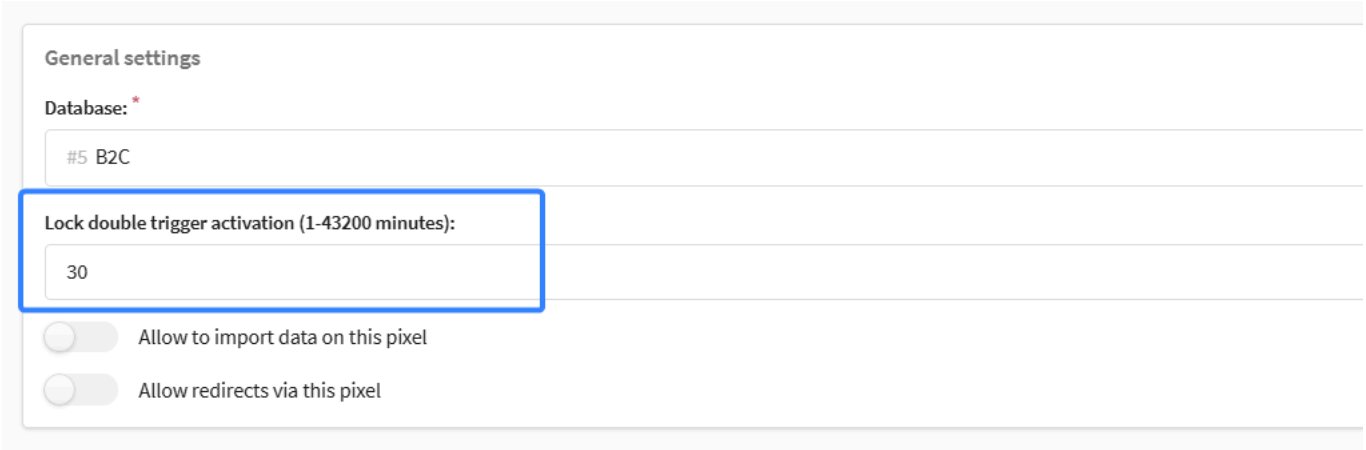
You can specify selection rules for a goal and its additional fields, for example, the values of some fields allows you to more accurately customize a mailing for a specific group of customers or specific products



When sending messages in templates, you can add variables with data about the pixel trigger event that more about these variables here.

TIP

If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is re interval.

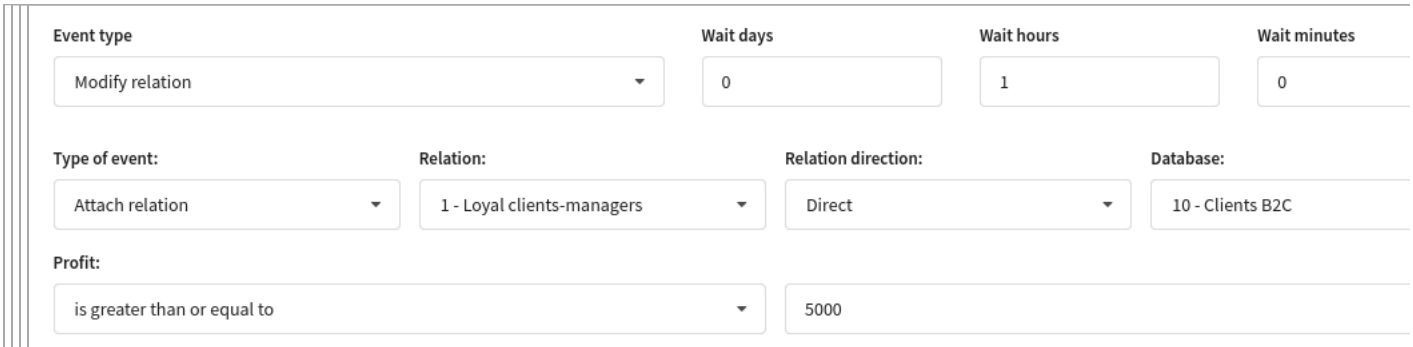


Modify relation

The event can be triggered when the relations between the profiles are changed: both direct and reverse

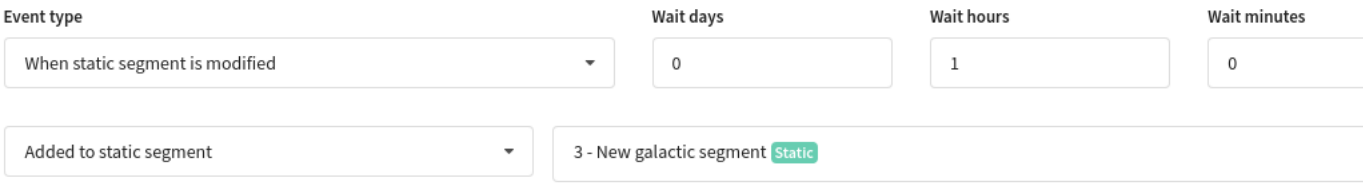
The event types:

- **attach relation,**
- **change relation,**
- **reaching relation "x" values,**
- **delete relation.**



When static segment is modified

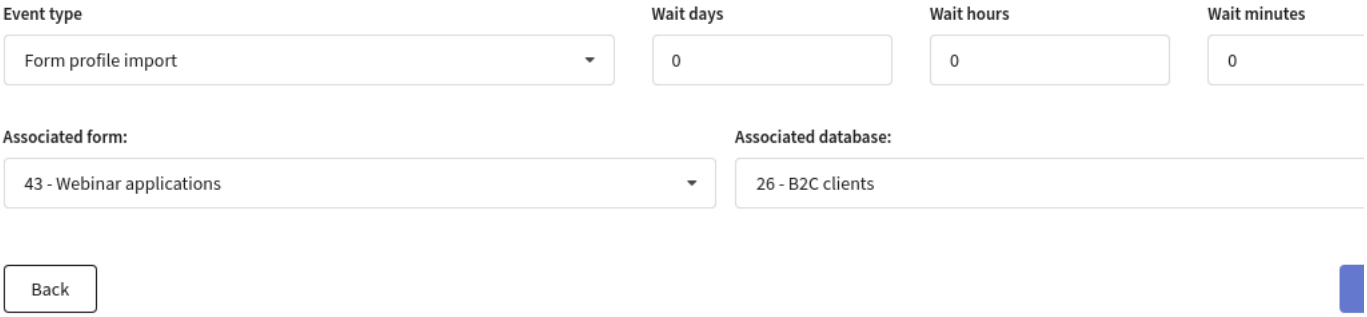
The event can be triggered when a change occurs in a static segment: a profile can be added to the sele it.



Form profile import

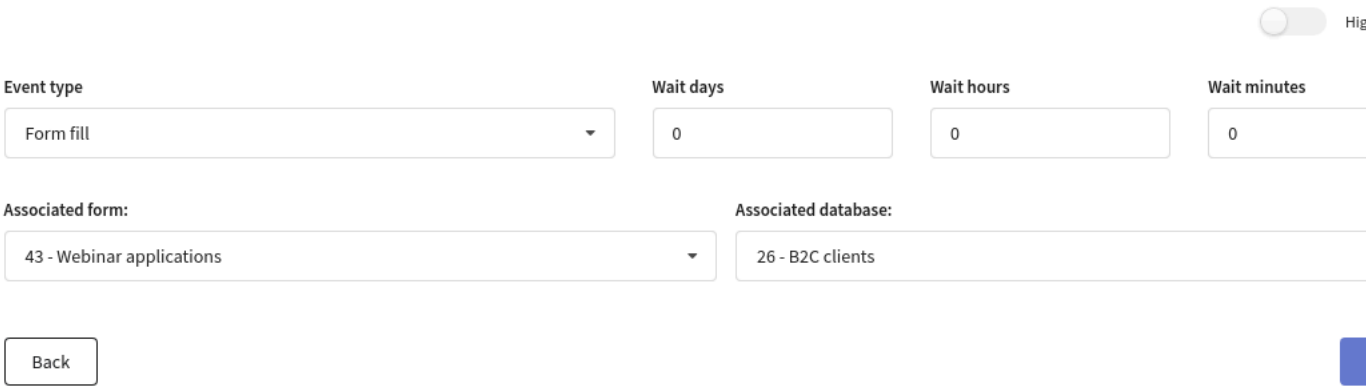
The trigger is activated if the profile has filled out the form and for this reason, has been added to the da the form, activate the Import Profiles option on the Actions tab.

Select the form and a profile database:



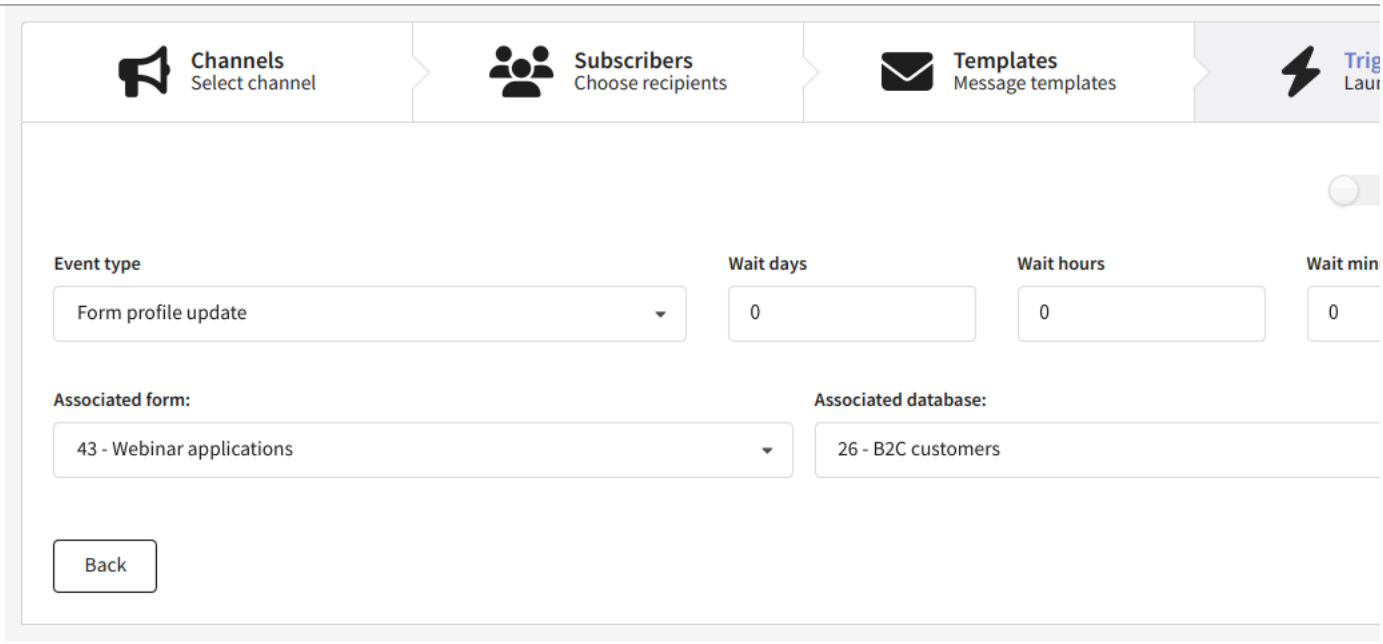
Form fill

The trigger is activated only if the profile fills out the form after clicking a link in an email. Select the for profiles that will fill out the form:



Form profile update

The trigger is activated when the data of a profile, which is already in the database, is changed as a resu If you want to track changes to profile data only in a specific database, also specify the database.



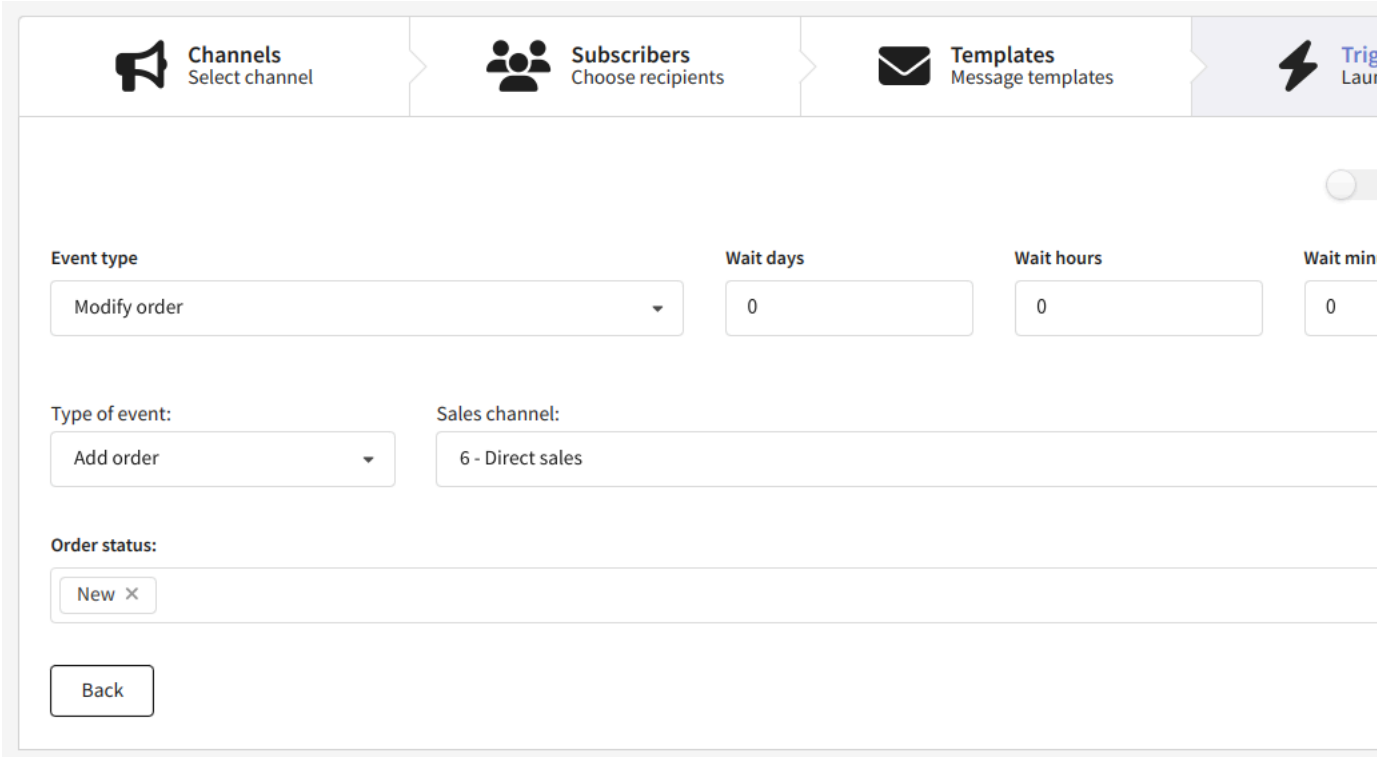
Modify order

The trigger be activated if order in the Market has been changed. Select a sales channel and an event ty

- **add order (a new order has been added to the Market)**
- **change order (the status of an existing order has changed)**

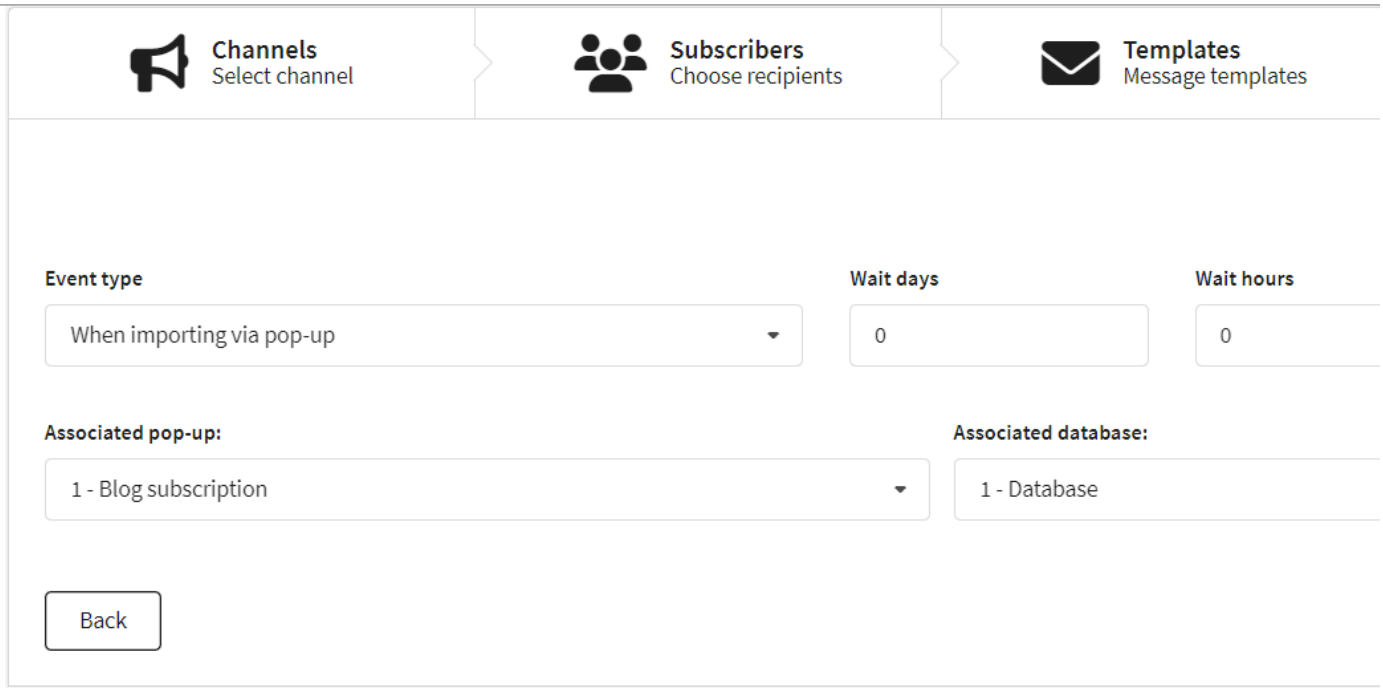
Available order statuses:

- **new**
- **delivered**
- **paid**
- **cancelled**
- **partial refund**
- **in delivery**
- **in process**



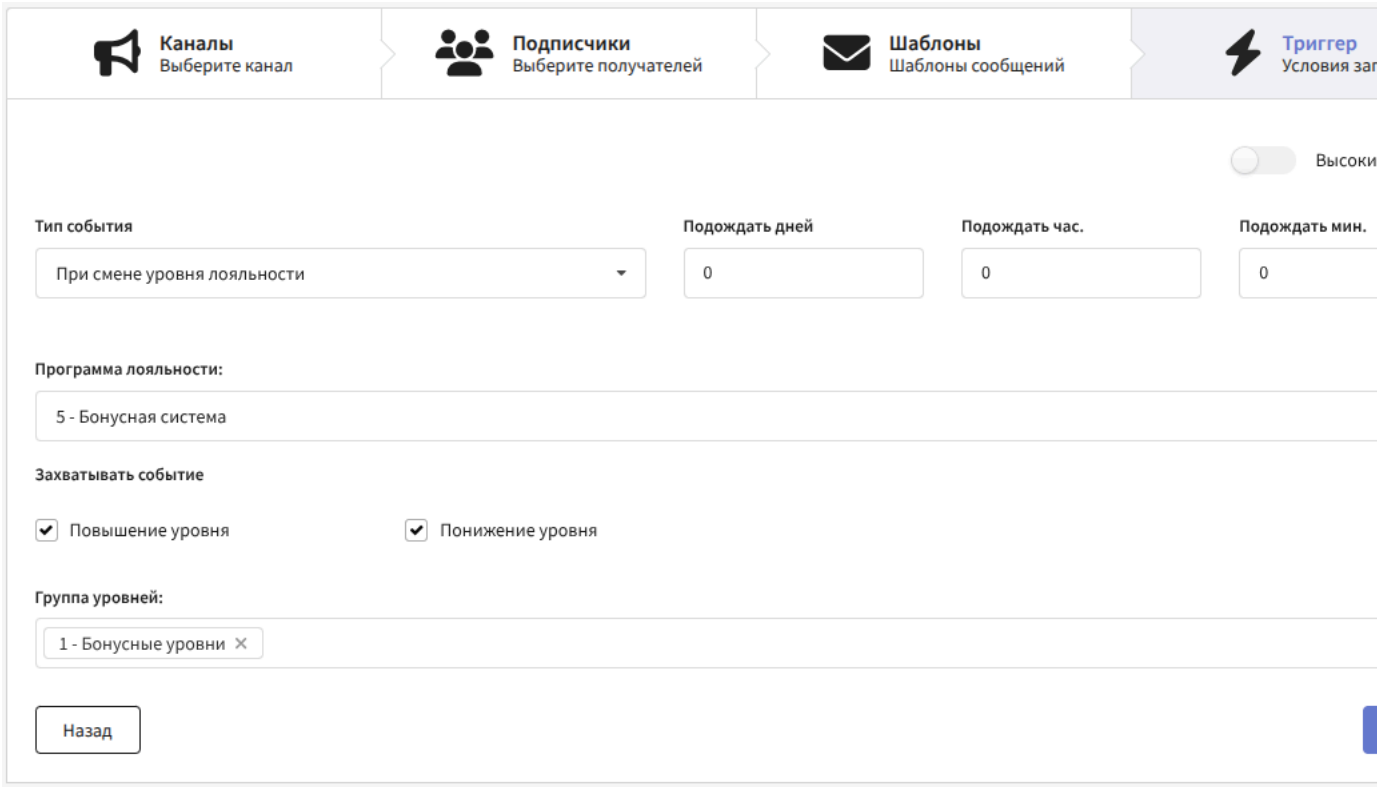
Import via pop-up

The trigger will be activated if the profile data gets into the database as a result of filling out and submit popup you want to track:



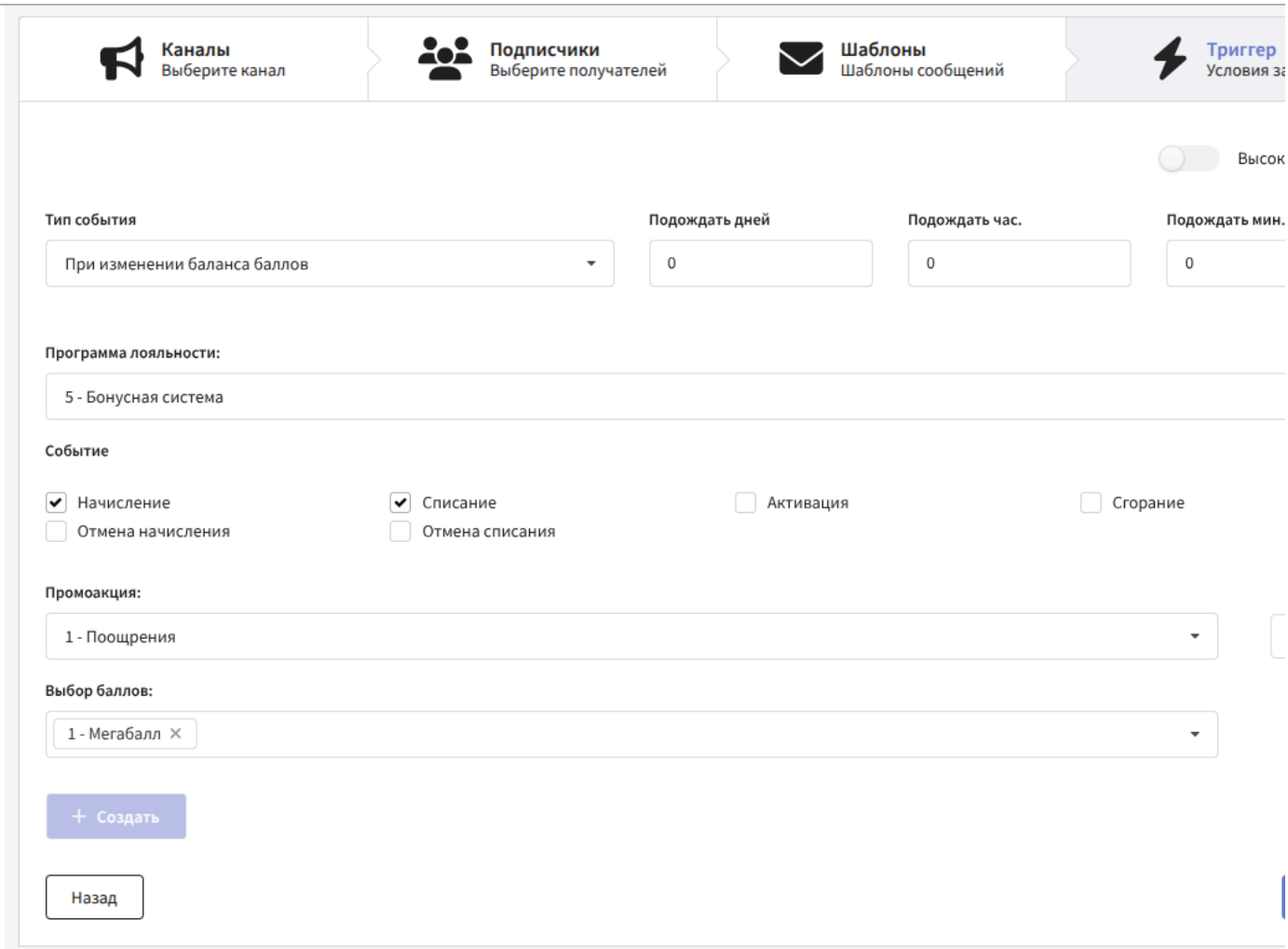
On loyalty level change

The trigger fires when a profile moves to a different level within a loyalty program.



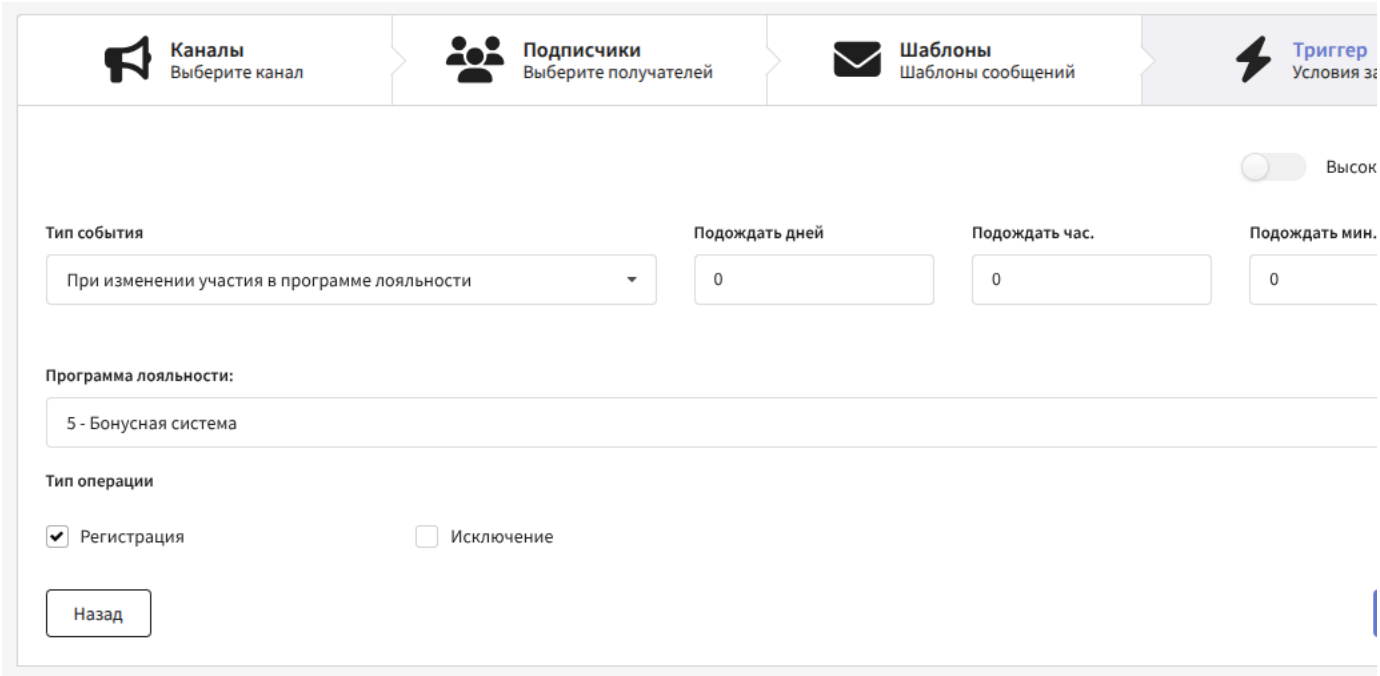
On points balance change

The trigger fires when the number of bonus points on a profile changes in a loyalty program.



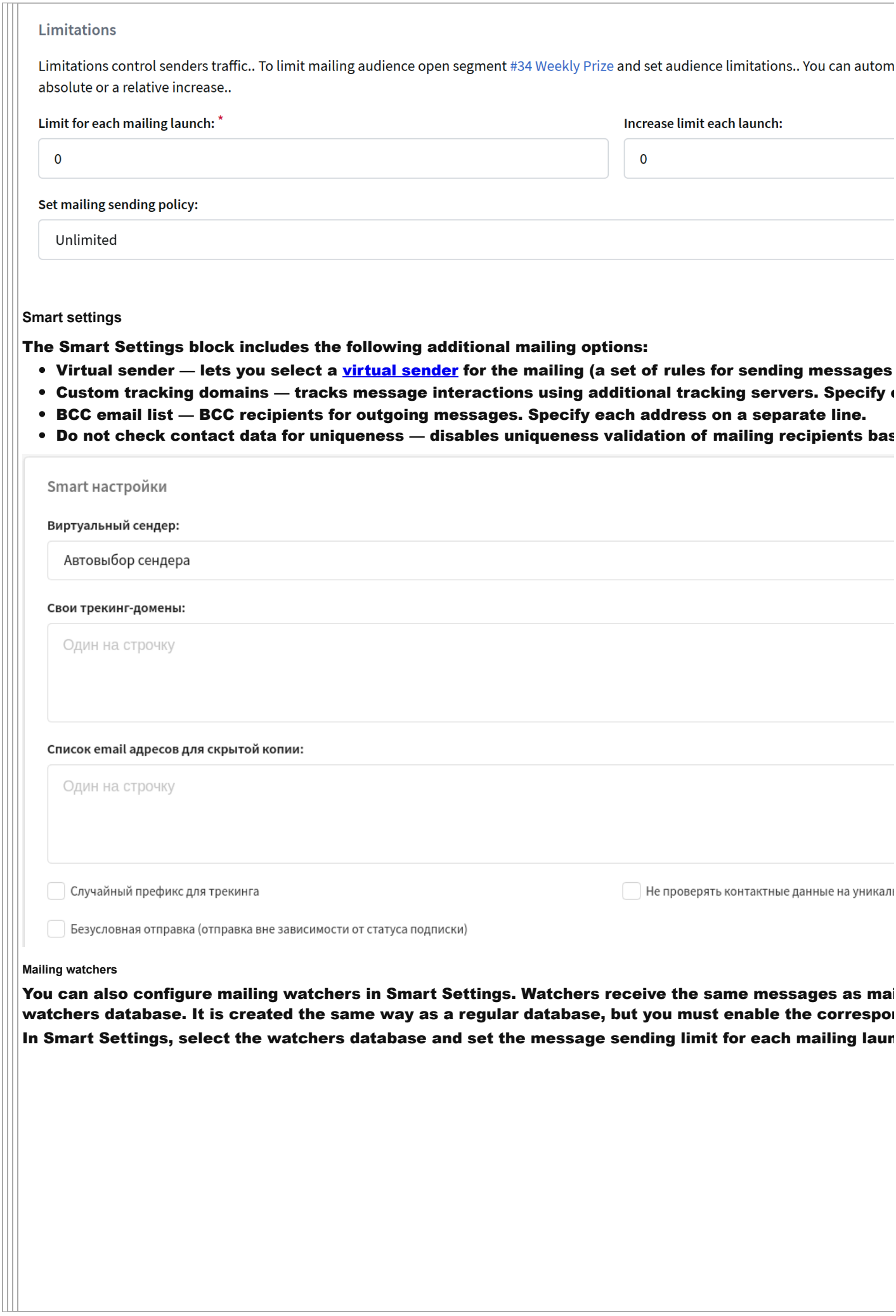
On loyalty program participation change

The trigger fires when a profile is registered in or removed from a loyalty program.



Limitations

As no mass audience is processed by trigger mailings — no limitations are even necessary except for co individual messaging per time period.



Smart settings

The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients bas**

Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mai watchers database. It is created the same way as a regular database, but you must enable the correspo

In Smart Settings, select the watchers database and set the message sending limit for each mailing laun

Наблюдатели за рассылкой

Наблюдателям будут отправлены письма каждый запуск рассылки если лимит не установл
порция наблюдателей в пределах лимита.

Письма будут отправлены на основной email профиля наблюдателя, подписки не учитыва

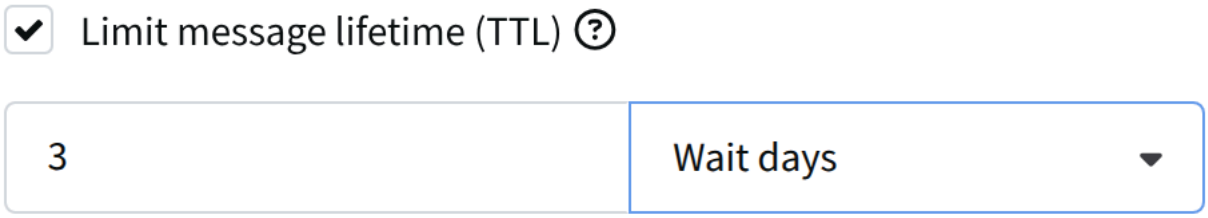
БД наблюдателей:

2 - Mailing watchers database 1000

Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is reminder, a one-time code, or an offer that quickly loses relevance.



To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, ho
After the specified time expires, the system stops delivery attempts for the undelivered message and reg

TIP: Message TTL specifics

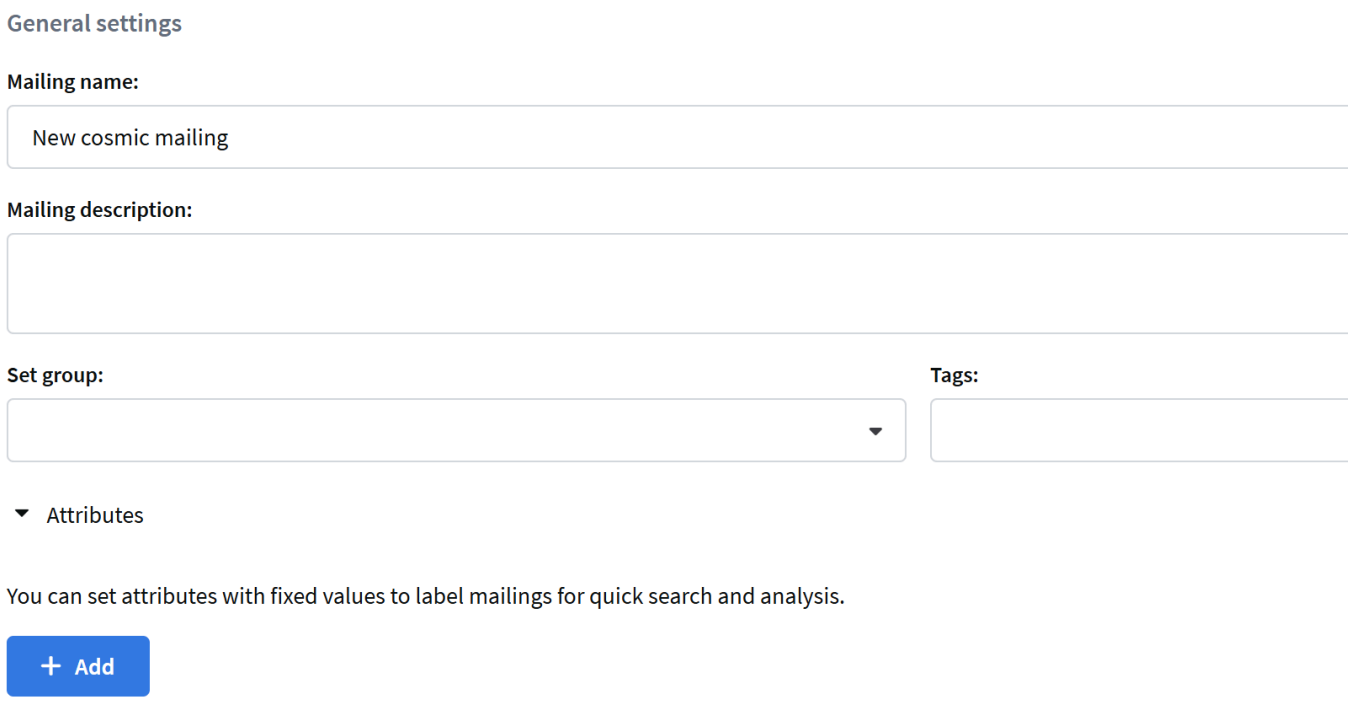
- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration par**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API para Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefor different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and**

Regular mailings

![img] (/img/attachments/9408348/56250456.png)	**Regular mailing** is a tool for organizing** massive multiple** mailings to your subscribers. **At every launch moment,** this mailing sends one message to each subscriber from the selected audience. Unlike a broadcast mailing, which processes the database **only once**, a regular mailing never finishes by itself. To create a new mailing, select **Mailings → Regulars** and press **+ Create** button, or you can use the mailings calendar. </td> </tr>

General settings

In the main settings, specify the name and description of the new segment, and if needed, set the acces additional grouping:



Mailing settings

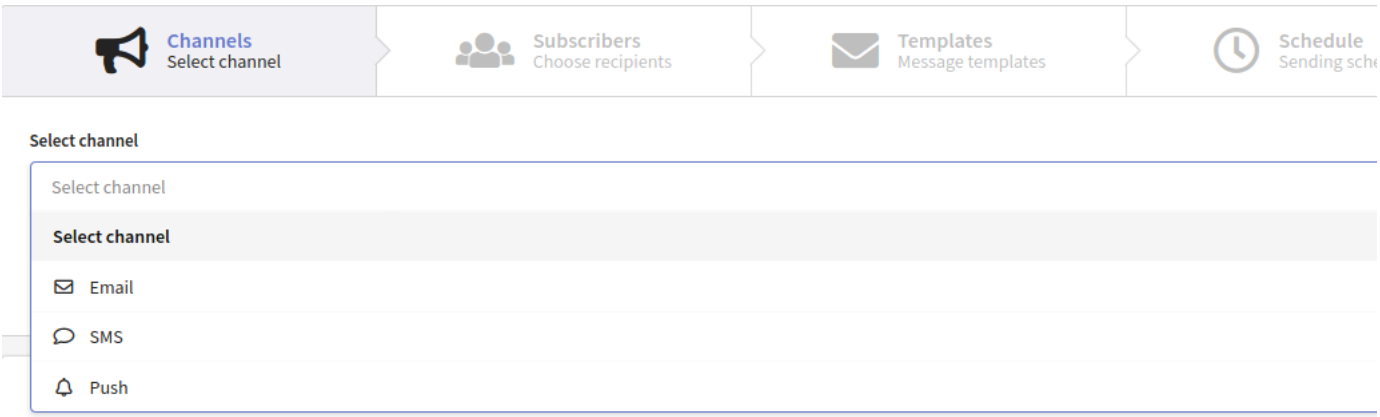
These settings are communication channel, messages recipients, templates with content and mailing sc

Channels

On this tab, you can select a communication channel for the mailing. This channel must be available in a select on the next tab.

NOTE

For Email sending, you can use the main subscriber contact data, which is the Email field in the databas



TIP

When using a push channel, you can select specific push providers for sending, as well as the sending r subscriptions taking into account filters, or last, sending to the subscription with the most recent creat these (created at the same time), then one will be selected randomly. If the mailing has filters by provid the filters are applied first, and then the most recent subscription is searched.

Subscribers

On this tab, you can select Resource for mailing from the drop-down menu. The resource must contain a previous step.

Also, you can select an Audience source, which can be a database or segment with profiles from one or

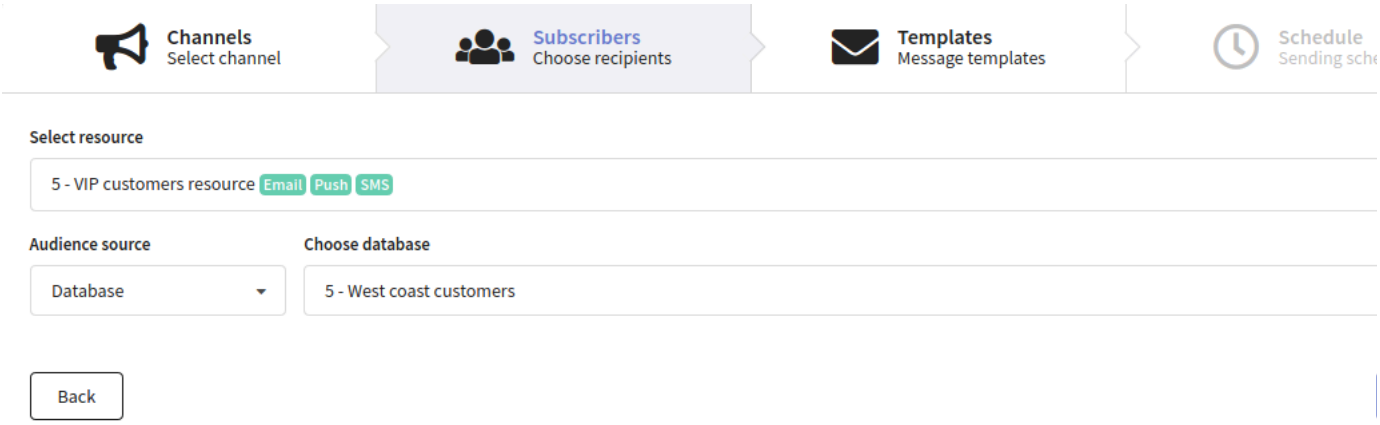
TIP

The audience source for a regular mailing depends on its purpose:

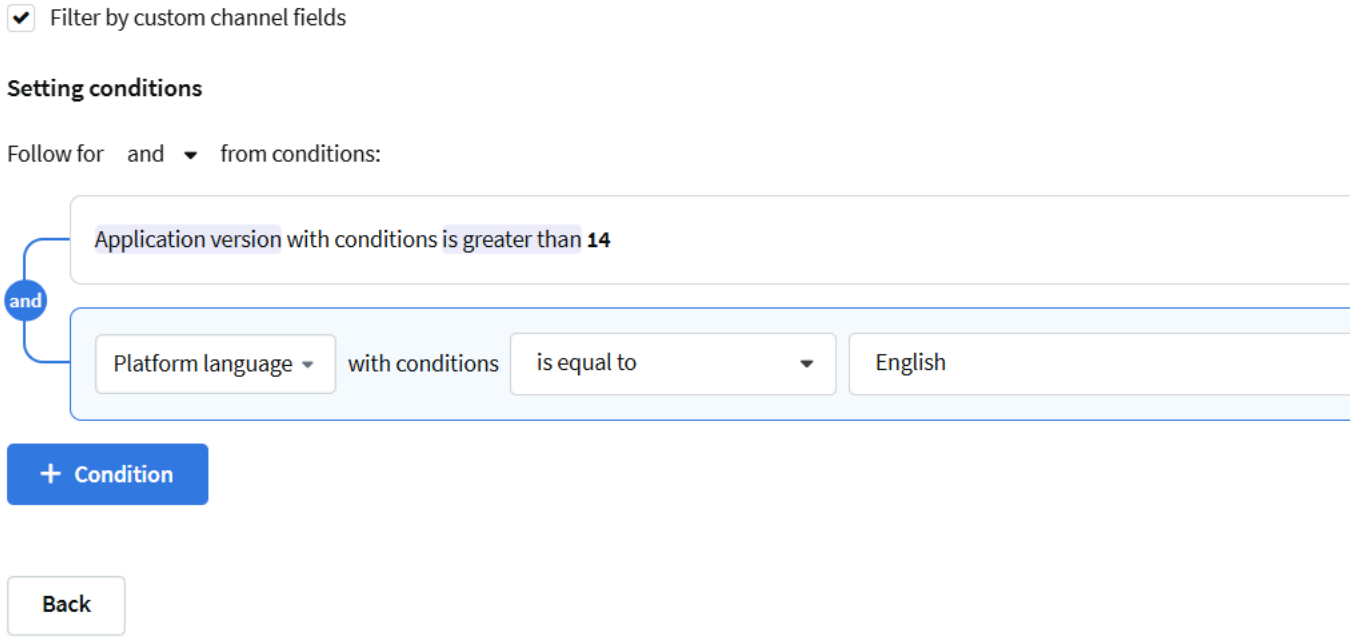
- **For periodic mailings, you can use a database, a static or dynamic segment. The mailing will send r configured audience.**
- **For activity-based mailing, it is necessary to use a dynamic segment with selection based on some**

TIP

Mailings are sent to customers who subscribe to a certain resource.Please, make sure your customers before launching a mailing.



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to ha resource. In mailing settings, select the categories to which the profile should be subscribed to receive You can also filter recipients by custom channel fields. To do this, select either a standard or custom sul condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using be met) and OR (at least one condition must be met).



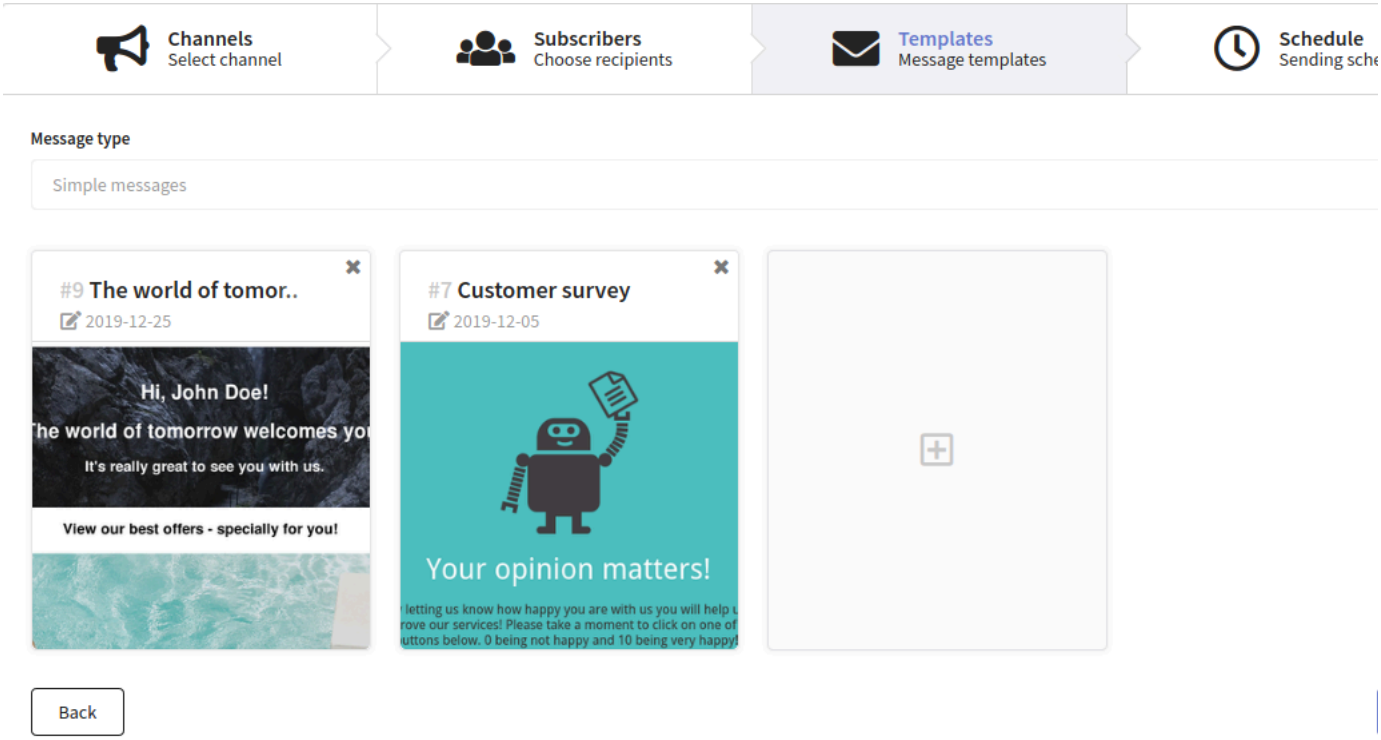
Templates

On this tab, you can select the content you want to use in your mailing. In Altcraft, templates combine c channels.

> **TIP**
>
> **Before mailing, please, make sure that the content you want to use is contained in the selected templat**

Press Add button and select the template. You can use up to 8 templates in any type of mailing. A regula each subscriber in random order in equal parts.

Click on the template miniature to open the preview.** Click on the template name to open it for editin**



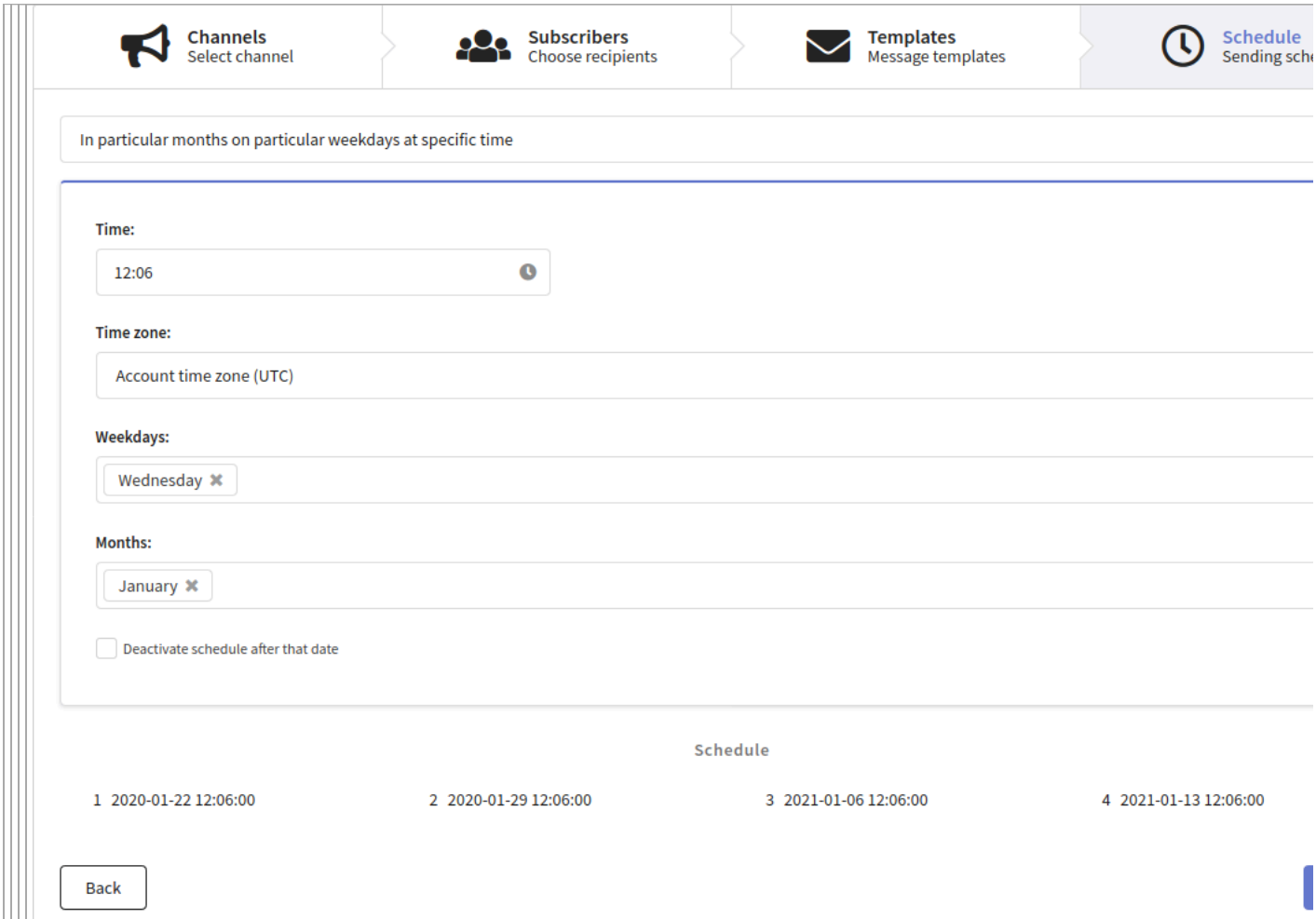
Schedule

> **TIP**
>
> **The mailing schedule is described in detail on this page.**

On this tab, you can configure sending policy. During massive mailings, you need to remember about sen limits for massive mailings that depend on sender reputation and the number of sent messages. Usually, received in 24 hours.

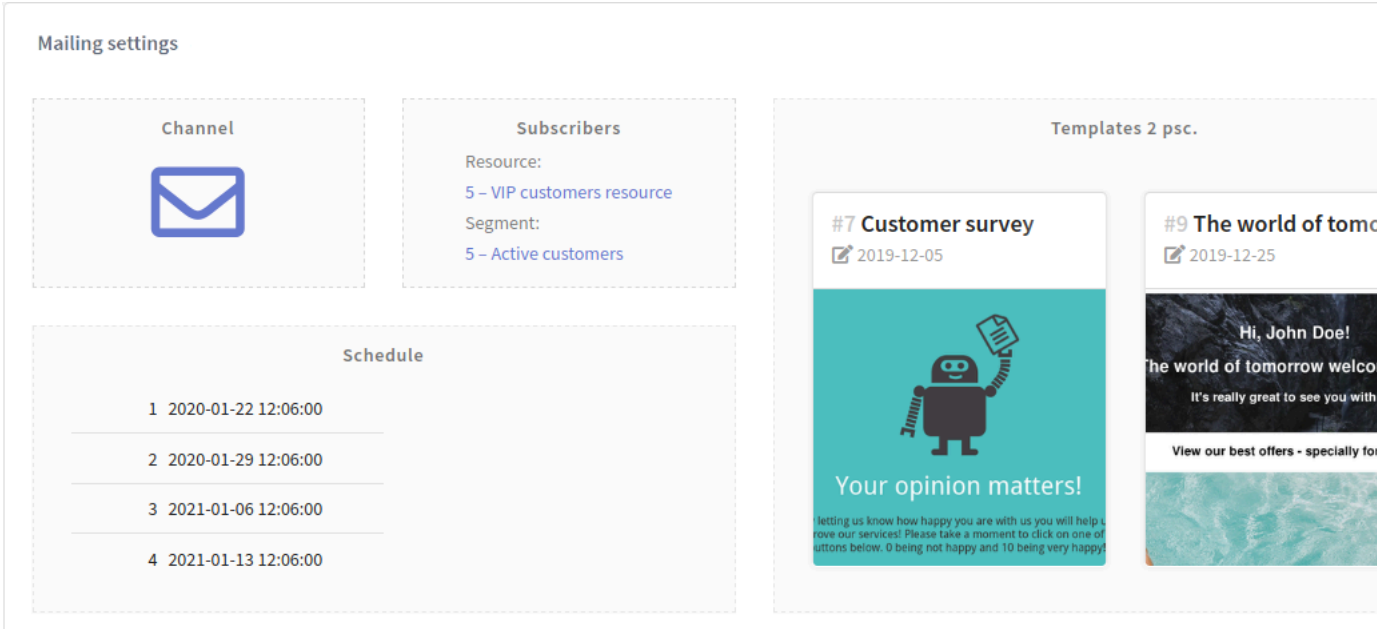
Regular mailing schedule

A regular mailing schedule is based on what kind of mailing you want to organize. You can set minutes, h even certain days of the week relative to the beginning of the month:



Also, for some schedule types, you can set a time zone, or use profile time zone, if this information is ava mailings after a certain time, you can use Deactivate schedule after that date option.

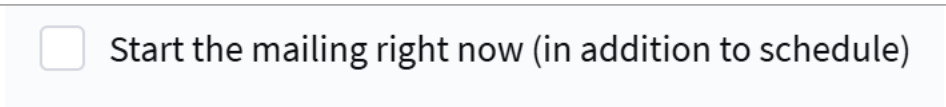
After you made your configurations, click the Finish button. Selected parameters will be displayed in the appropriate buttons to edit settings or go to the [mailing report](#).



Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part o

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



Limitations

In this block, you can configure limits and sending policy.

Using limits in the regular mailing

> **TIP**
>
> **The regular mailing will not be stopped automatically after sending one message to each subscriber. If mailing that must be sent in parts, we recommend that you use a broadcast mailing.**

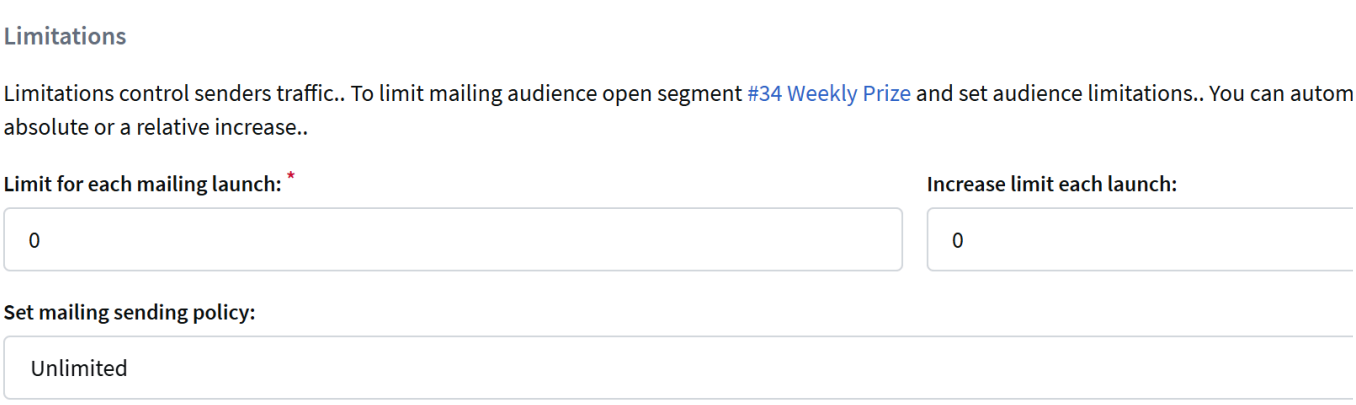
You can use limits in regular mailings when the number of mailings does not fit email providers limits.

> **NOTE**
>
> **You can also use limits to control the loading of your sender or your company network.**

Set limitations for the mailing:

- ****Limit for each mailing run** — the number of messages that will be sent during the next mailing ru**

> **DANGER**
>
> **Increase limit each run option is not recommended to use in regular mailings.**



In this case, during launch, the mailing will send as many messages, as you selected in the limit. When t than one set in the limit, mailing will start from the beginning of the list. To exclude re-sending one mess

> **CAUTION**
>
> **Please, remember that limitation can act for no more than 16 days, and the mailing period must not be**

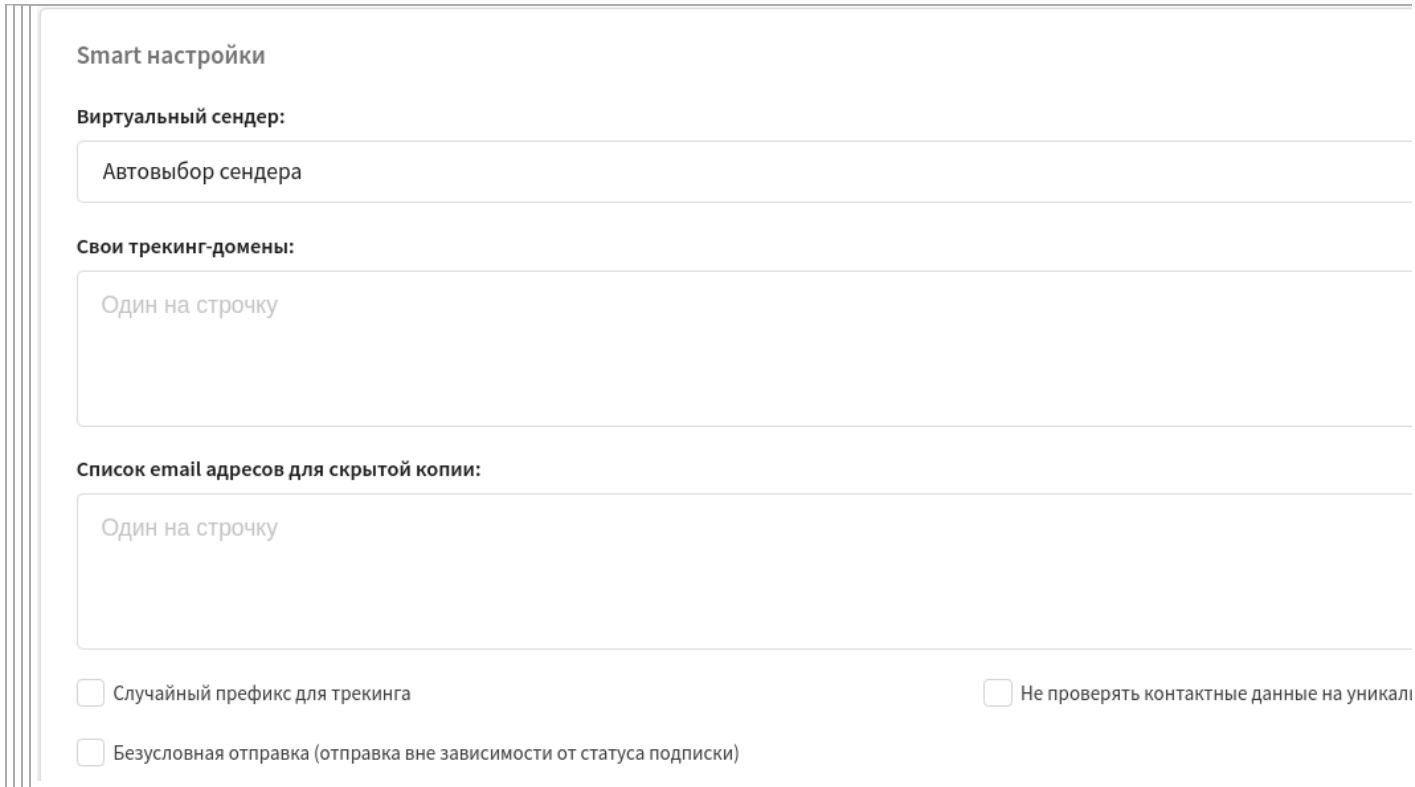
> **CAUTION**
>
> **When the “Use profile time zone when available” option is enabled, the limit settings and instant mailin**

Communication policy sets limits for communication with one subscriber. By default, the mailing will use

Smart settings

The Smart Settings block includes the following additional mailing options:

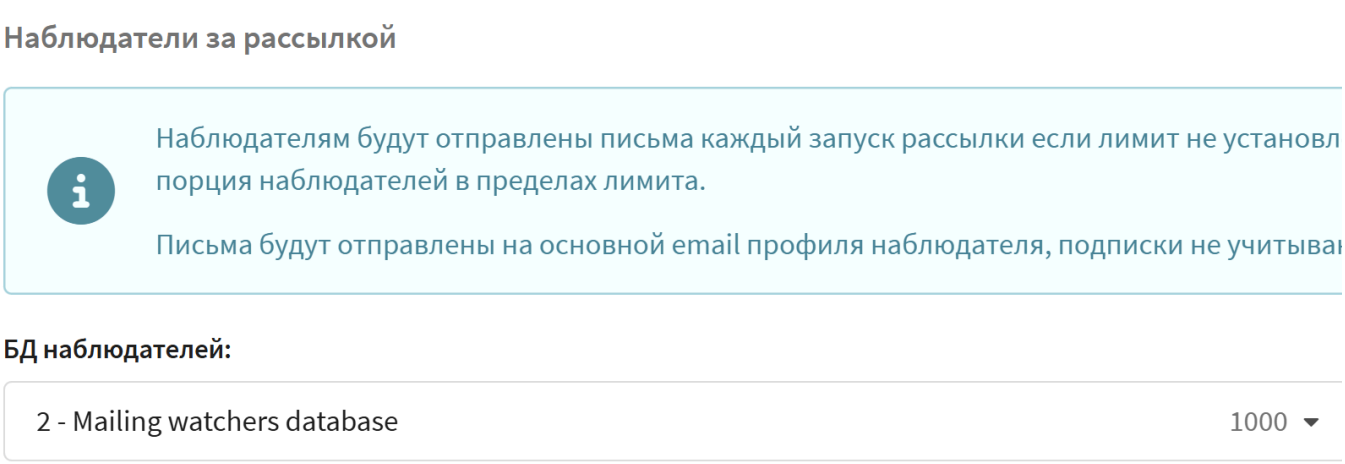
- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients bas**



Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mai watchers database. It is created the same way as a regular database, but you must enable the correspo

In Smart Settings, select the watchers database and set the message sending limit for each mailing laun



Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is reminder, a one-time code, or an offer that quickly loses relevance.

To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, h

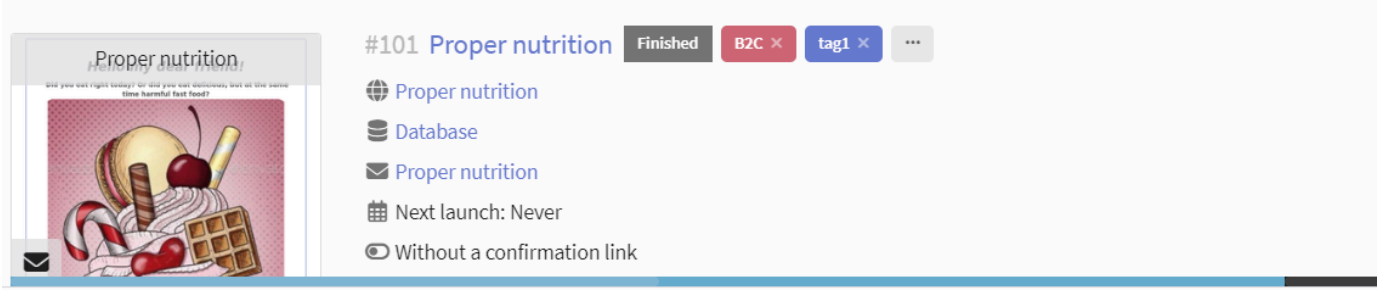
After the specified time expires, the system stops delivery attempts for the undelivered message and re

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration par**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API para Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefor different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and**

Mailing progress

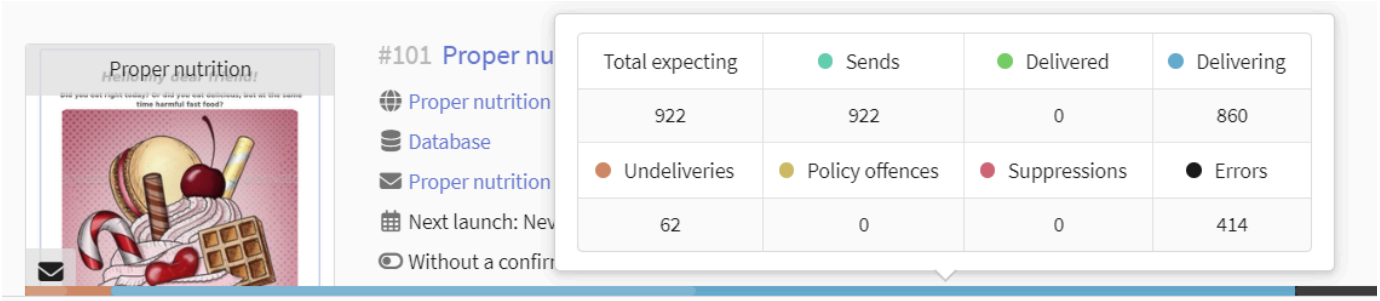
In the list of regular mailings, you can see the progress bar of each mailing.



Each message status has its own colour:

- **sends – the number of profiles for which messages were generated**
- **delivered – the number of profiles to which the message was delivered**
- **delivering – the number of messages in the sender queue**
- **undeliveries – the number of profiles that failed to deliver the message**
- **policy offences – the number of profiles affected by sending policy (see more)**
- **supressions – the number of profiles in suppression lists (see more)**

Click on the progress bar to open a table with mailing statistics:



Deactivating a mailing

You can deactivate a regular mailing in two ways: the Deactivate button in the mailing list and the Active

The “Deactivate” button in the mailing list

If the mailing is not currently sending, it is deactivated immediately.

If the mailing is currently sending, a confirmation window “Messages are currently being sent.” opens w default:

- **Interrupt the current sending — if the mailing is currently sending messages, abort sending.**
- **Clear the sending queue — clear the list of messages waiting to be sent.**

Click Deactivate to confirm the action or Cancel to dismiss it.

The “Active” toggle in the mailing settings

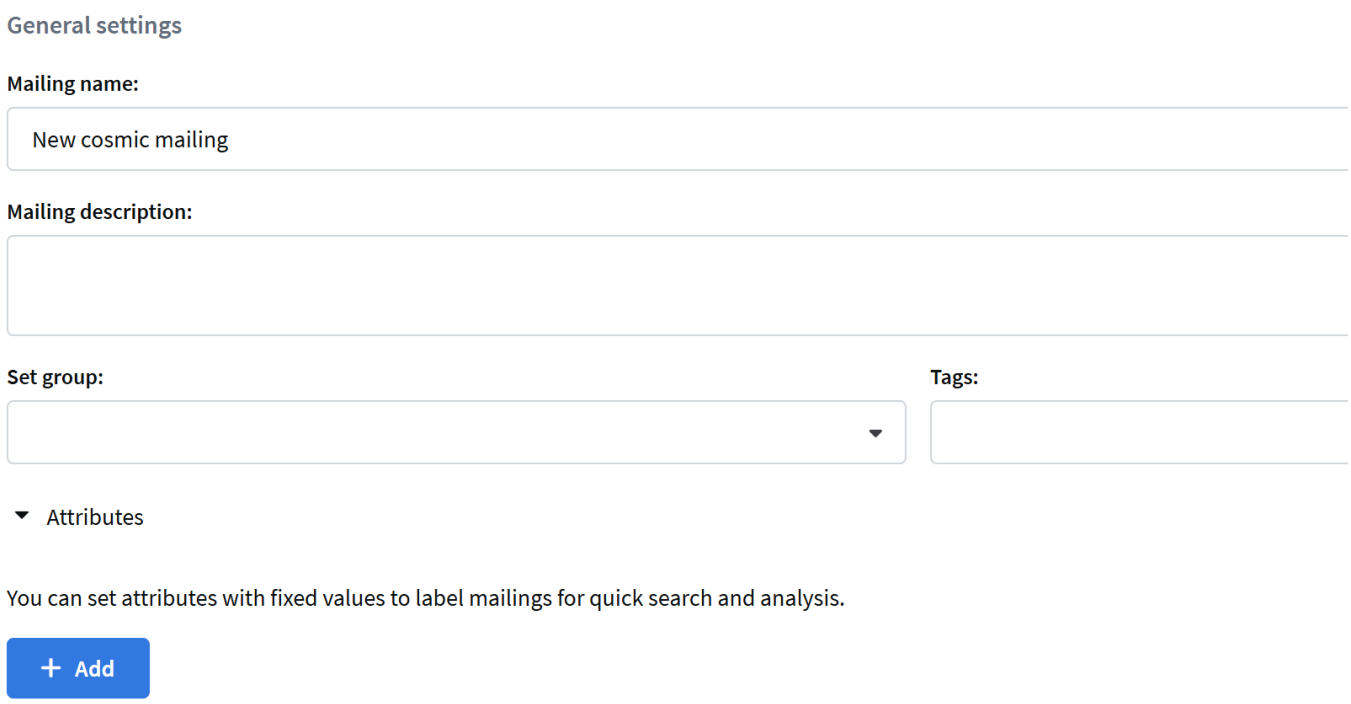
The Active toggle in the General settings block stops only the mailing schedule. Sendings that have alre scheduled launches do not occur.

Multivariate testing

![img] (/img/attachments/3114557/56250466.png)	**Multivariate testing, also known as MVT, or A/B/n testing, is a mailing that allows you to send multiple message templates (up to After that, the winning template is selected and sent to the rest of the audience. When the message is sent to each of the recipie** To create a new multivariate test mailing, select Mailings → Multivariate testing and press the + Create button. Also, you can use the mailings calendar to create a new multivariate testing mailing. </td> </tr>

General settings

In the main settings, specify the name and description of the new segment, and if needed, [set the acces](#) additional grouping:



Mailing settings

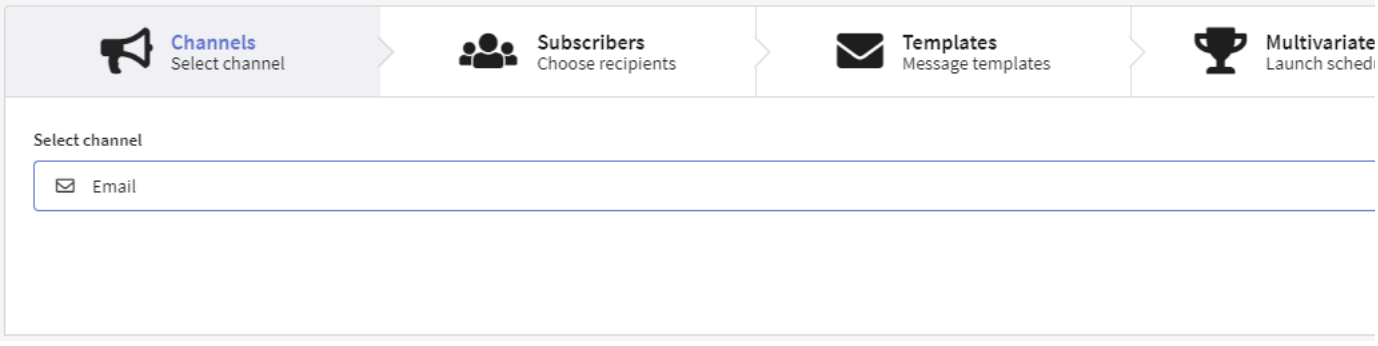
The following information is defined in mailing settings:

- **communication channels (Email, SMS or Push),**
- **subscribers,**
- **several templates with content,**
- **audience size for testing,**
- **selection of winner template.**

Channels

On the first tab, you can select the communication channel you want to use. This channel must be availa [resource](#). The resource is selected on the next tab.

> **NOTE**
>
> **For Email sending, you can use the field with the main subscriber contact information — Email field.**



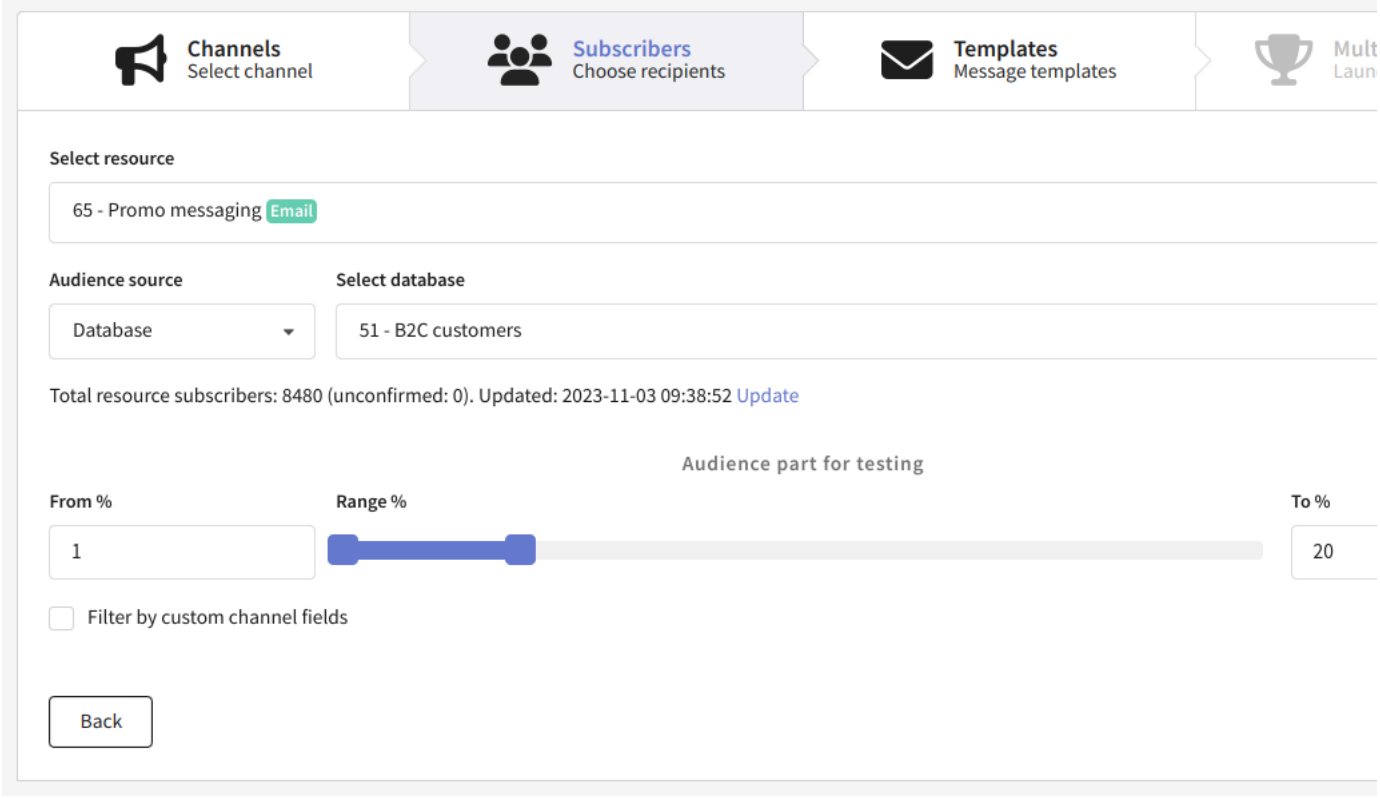
Subscribers

On the Subscribers tab you can choose a resource to send messages from and an audience you want to
or a segment.

TIP

Mailings are sent to customers who subscribe to a certain resource. Please, make sure that message r
resource before mailing.

In the Audience part for testing block, you set an audience part, that will receive several content kinds f



Templates

On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templa
available channels.

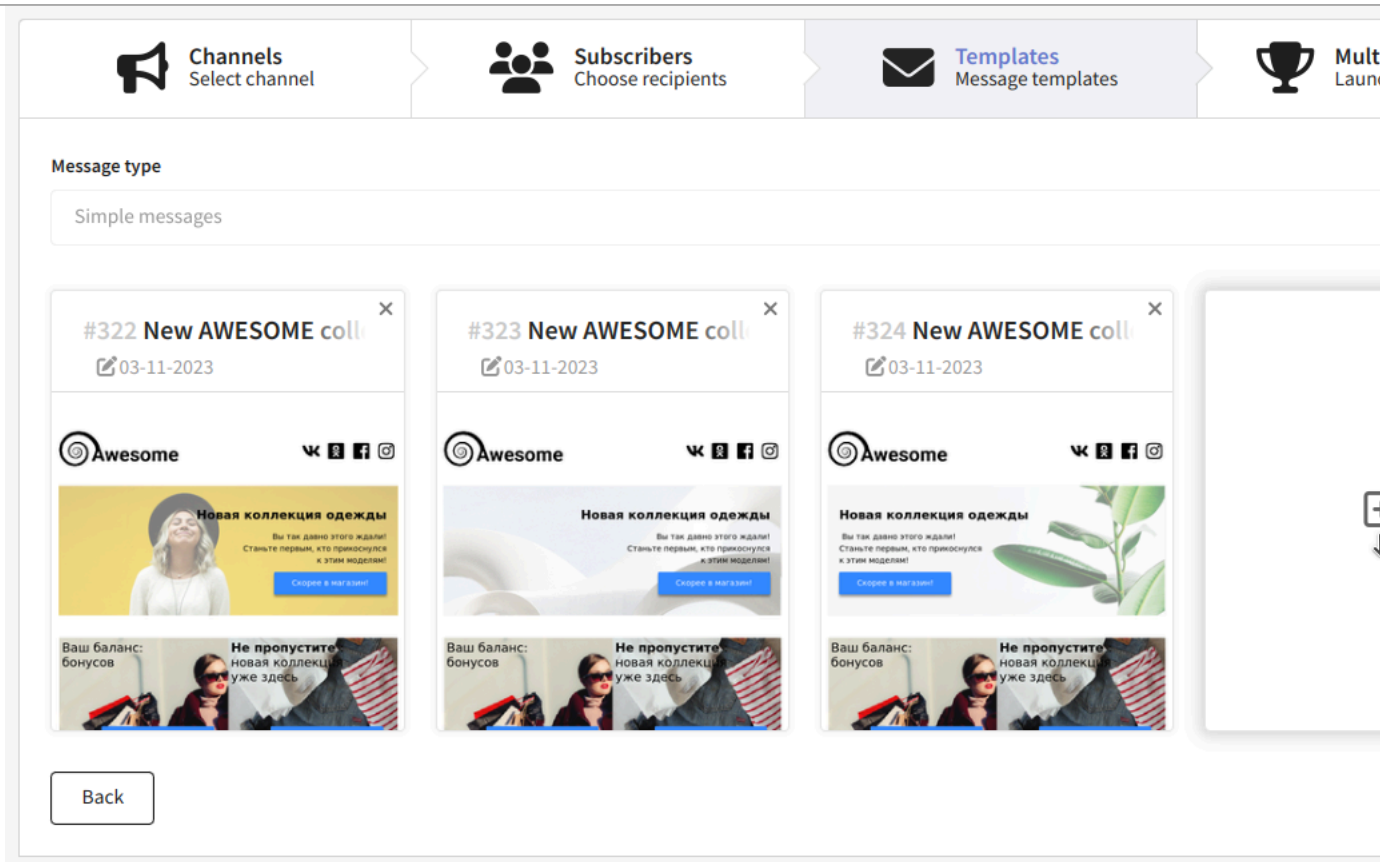
TIP

Please, make sure, that the template has content for the selected channel.

Press Add button

and select the template. Any mailing type can use up to 8 templates. A/B testing mailing will send one m
selected templates in random order in equal parts.

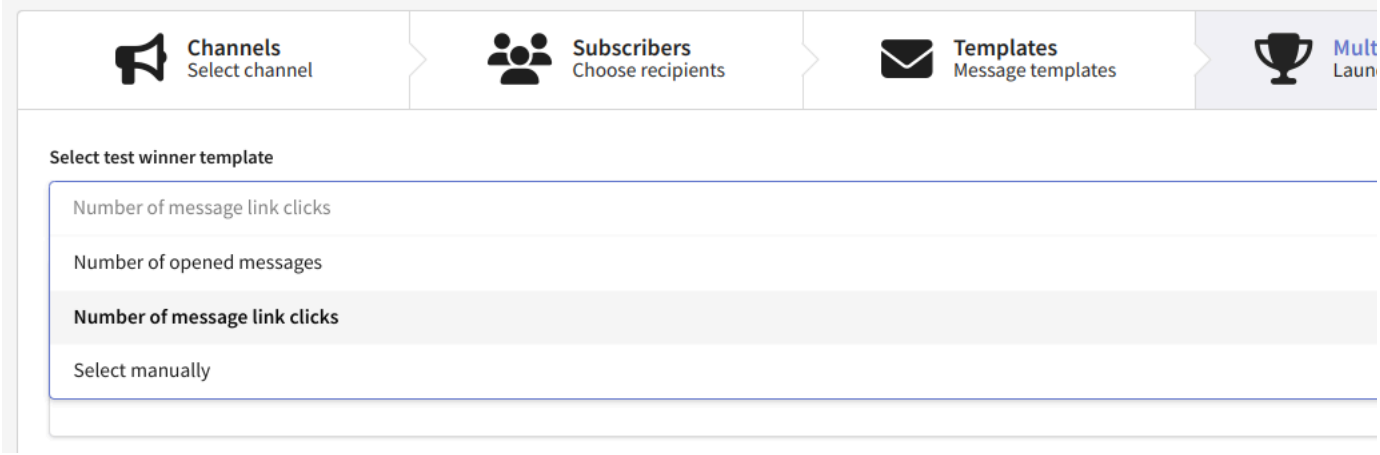
Click on the template miniature to see a preview, or click on the template name to open the template edi



Test settings

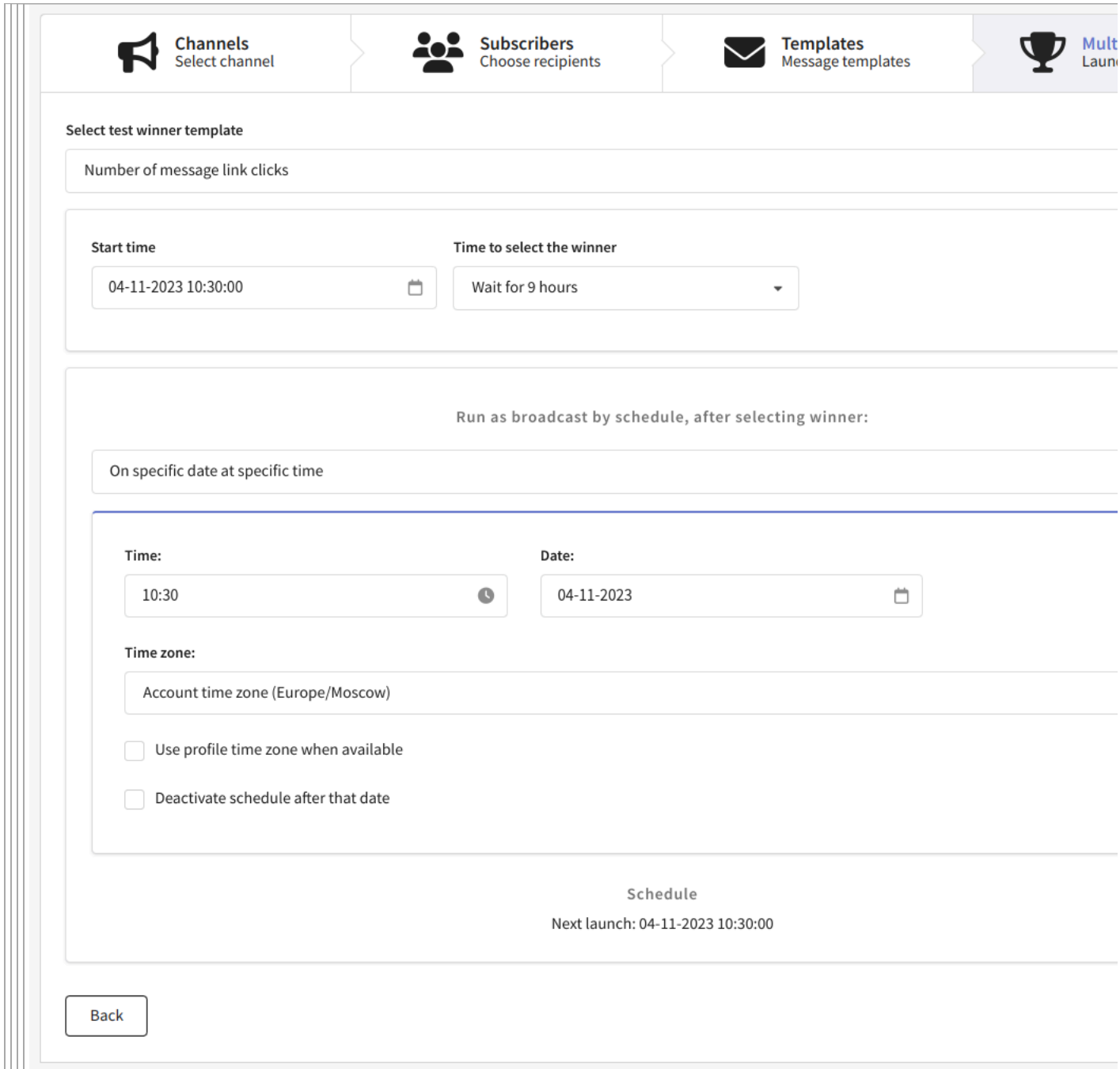
On this tab, you can configure how the winning template will be selected. The following options are avail

- **Number of opened messages**
- **Number of message link clicks**
- **Select manually**



Selection winner by openings/clicks

If you want to select the winning template automatically, set the mailing Start time. This is the time and
After launch, the mailing will automatically collect statistics by the total number of messages openings (collecting statistics using Time to select the winner parameter.
After the mailing collects statistics on opens or clicks, the message template that is most effective base sent to the main part of the audience according to the specified schedule:



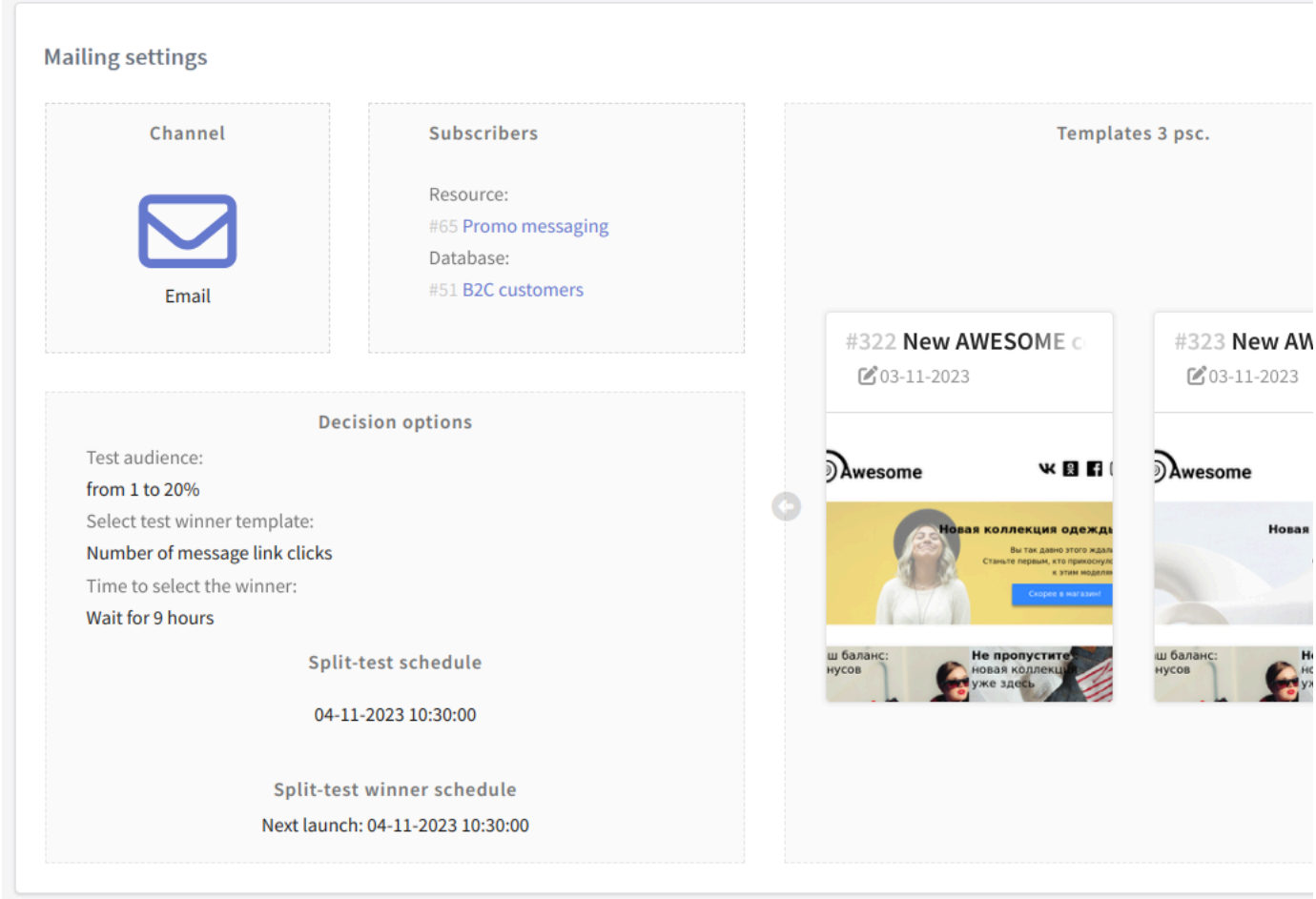
TIP

The mailing schedule is described in detail on this page.

CAUTION

If you select a time zone other than your account zone in the schedule settings, only the winner templa time zone. Test messages will be sent according to the account's time zone.

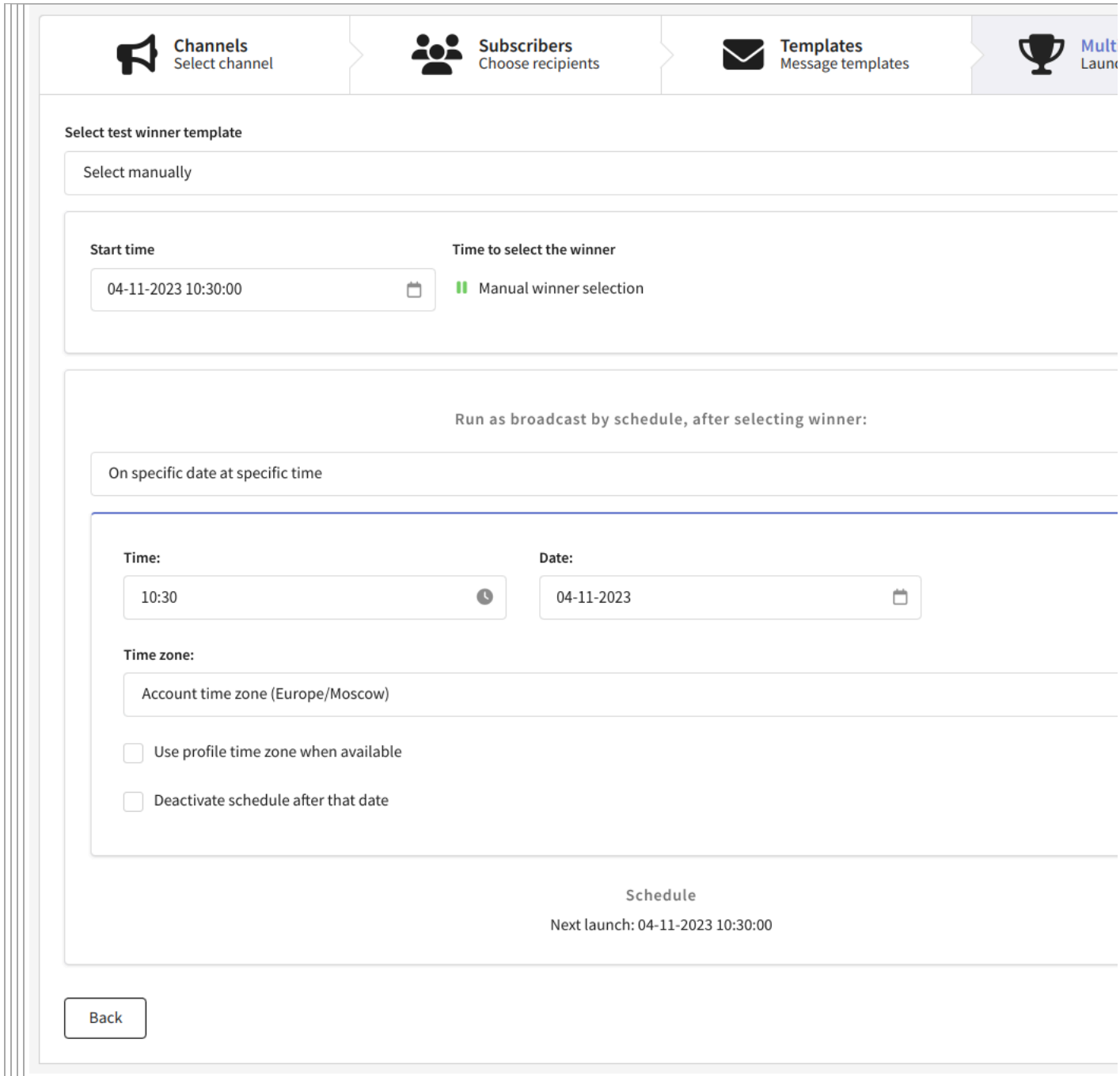
To complete the setup, click Finish. The options you select will be displayed in the Mailing settings block corner of the block you can edit the settings:



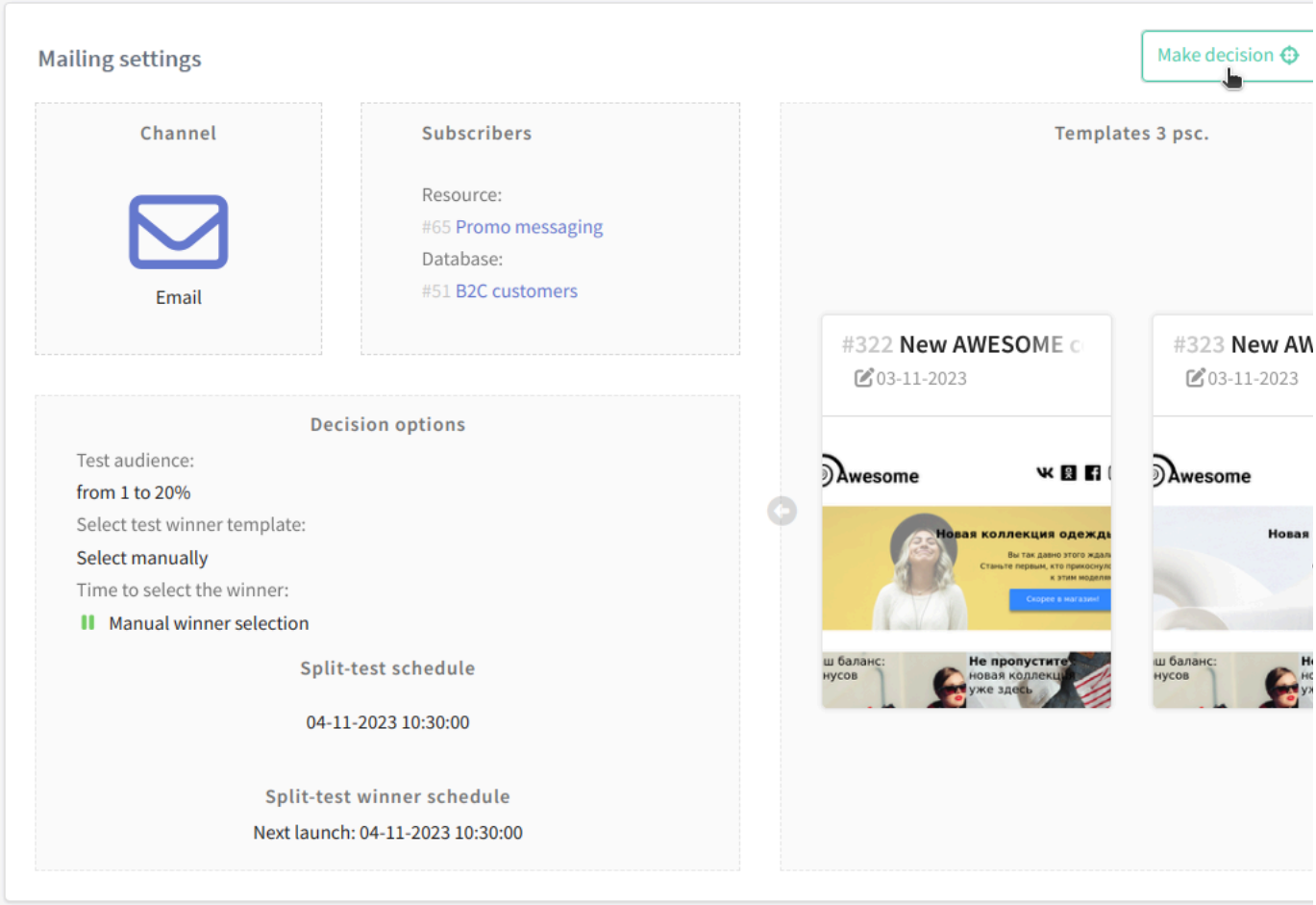
Save the mailing to activate it.

Manual winner selection

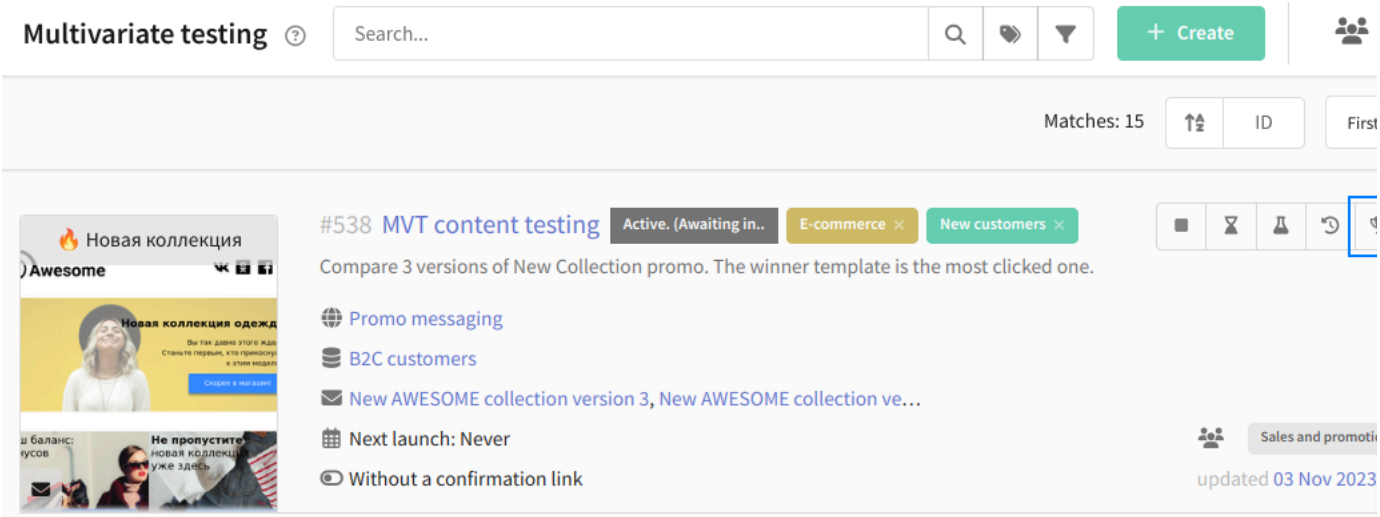
If you want to use the Select manually option, set the Start time. Mailing start is the time and date of lau
Next, set the mailing schedule. According to this schedule, the winner template will be sent to the main
Save the mailing to activate it. After the mailing sends test messages, it will switch to Active (Waiting fo



At this step, you need to analyze the performance of each template and choose the winner template. To button in the mailing editing window:

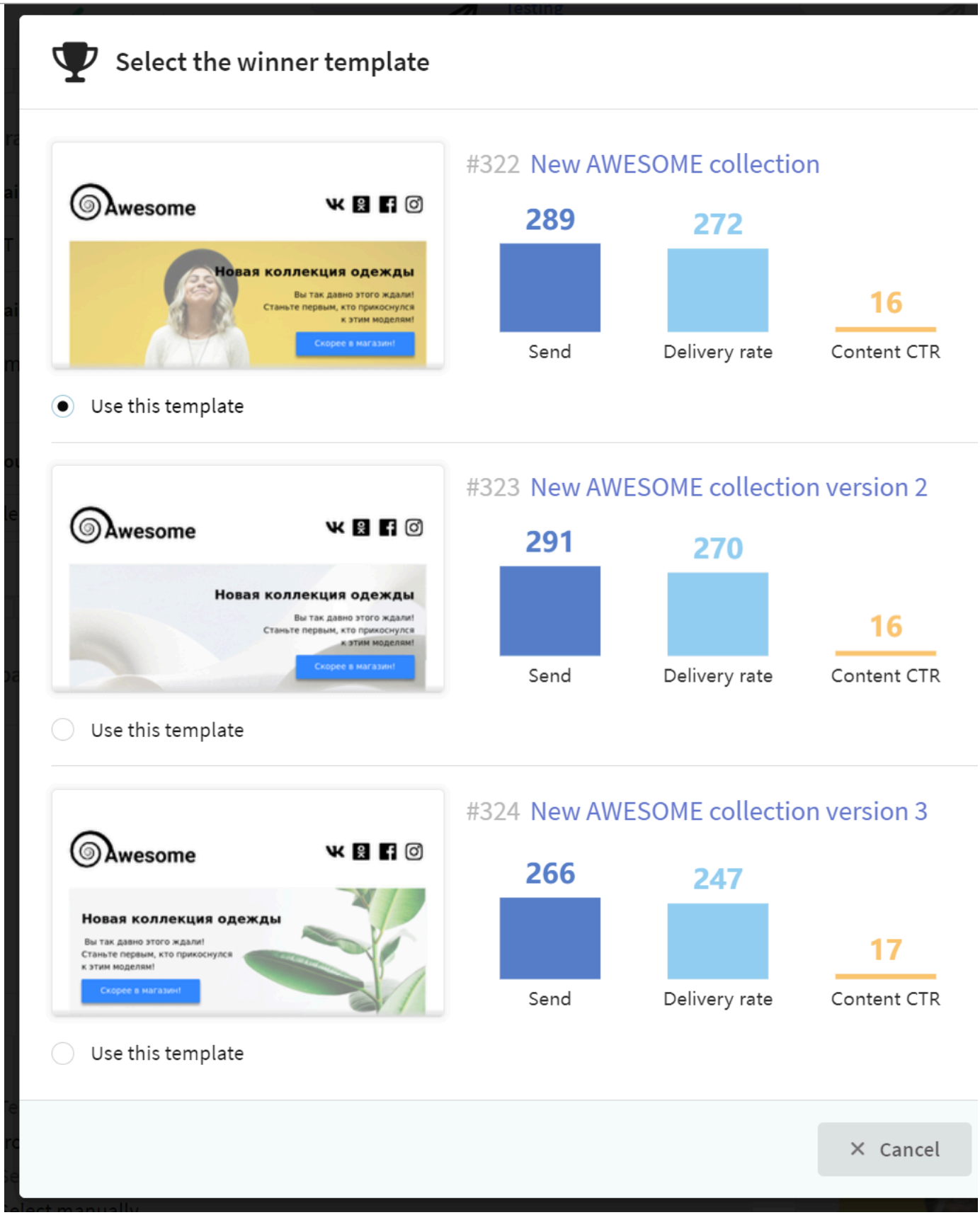


You can also select the winning template in the window with the list of mailings by clicking on the corre

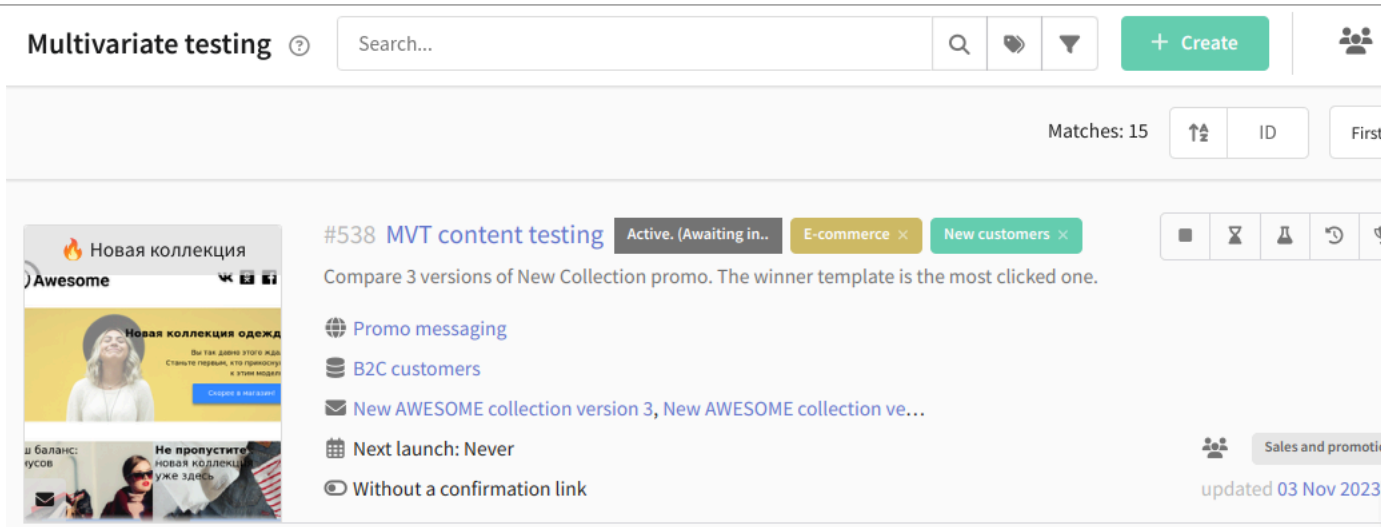


You will see a window with statistics for each template:

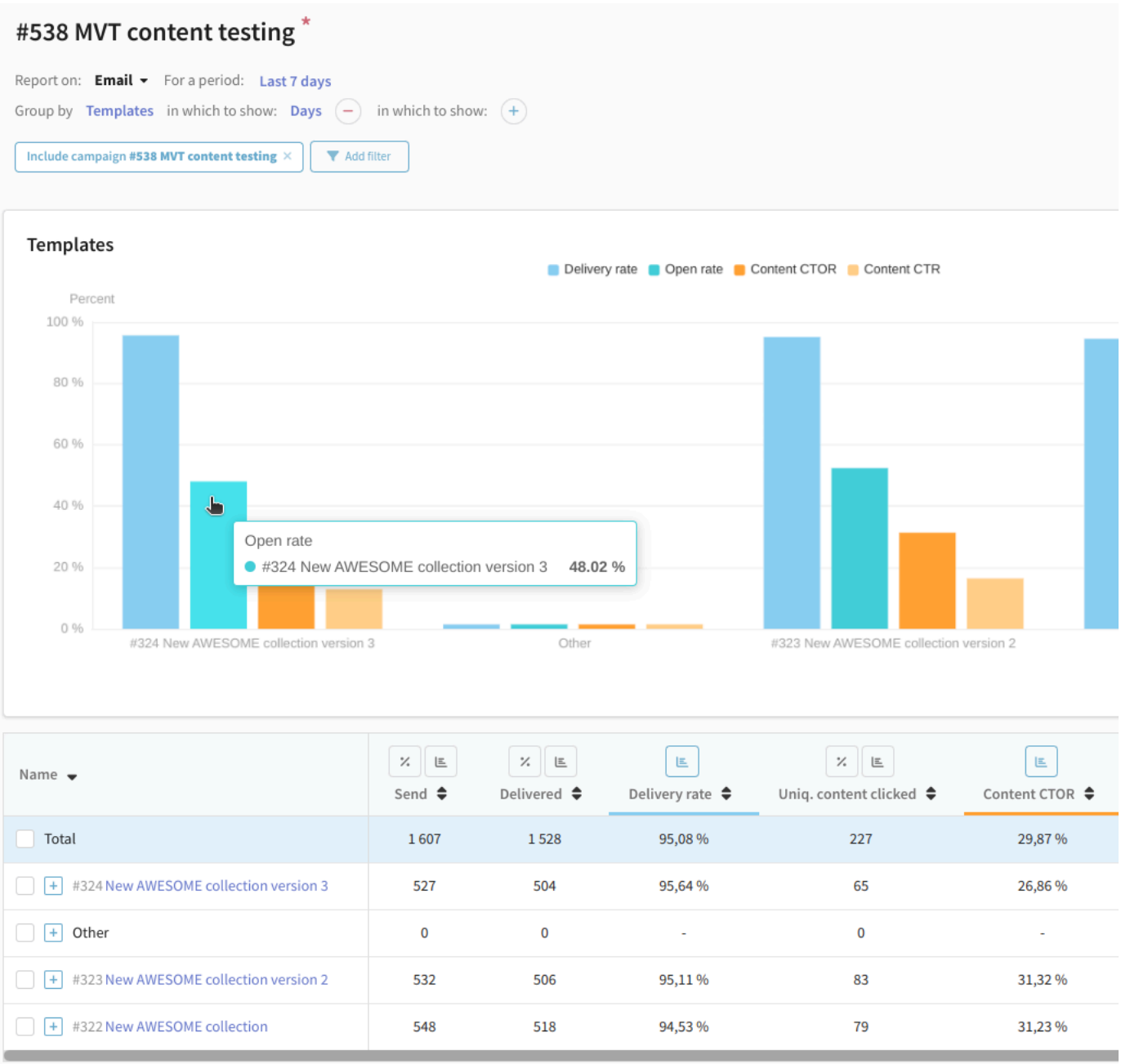
- **Send is the number of messages sent.**
- **Delivery rate is the number of messages delivered.**
- **Content CTR is the percentage of clicks in a message. Calculated using the formula: (Number of clic 100%.**
- **Content OR is the percentage of openings in a a message. Calculated using the formula: (Number of messages delivered) * 100%.**



If you don't have enough data to choose the winning template, you can use the Channel report. Click on t



In the report, select the parameters necessary to evaluate performance. These can be absolute or relati CTR/CTOR). You can also check negative reactions by the number of complaints and unsubscriptions.

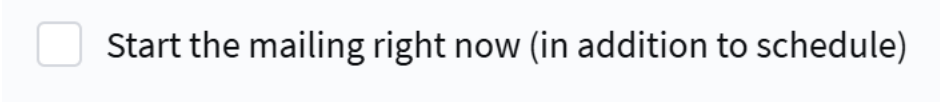


Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTOR
Total	1 607	1 528	95,08 %	227	29,87 %
#324 New AWESOME collection version 3	527	504	95,64 %	65	26,86 %
Other	0	0	-	0	-
#323 New AWESOME collection version 2	532	506	95,11 %	83	31,32 %
#322 New AWESOME collection	548	518	94,53 %	79	31,23 %

Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part o

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



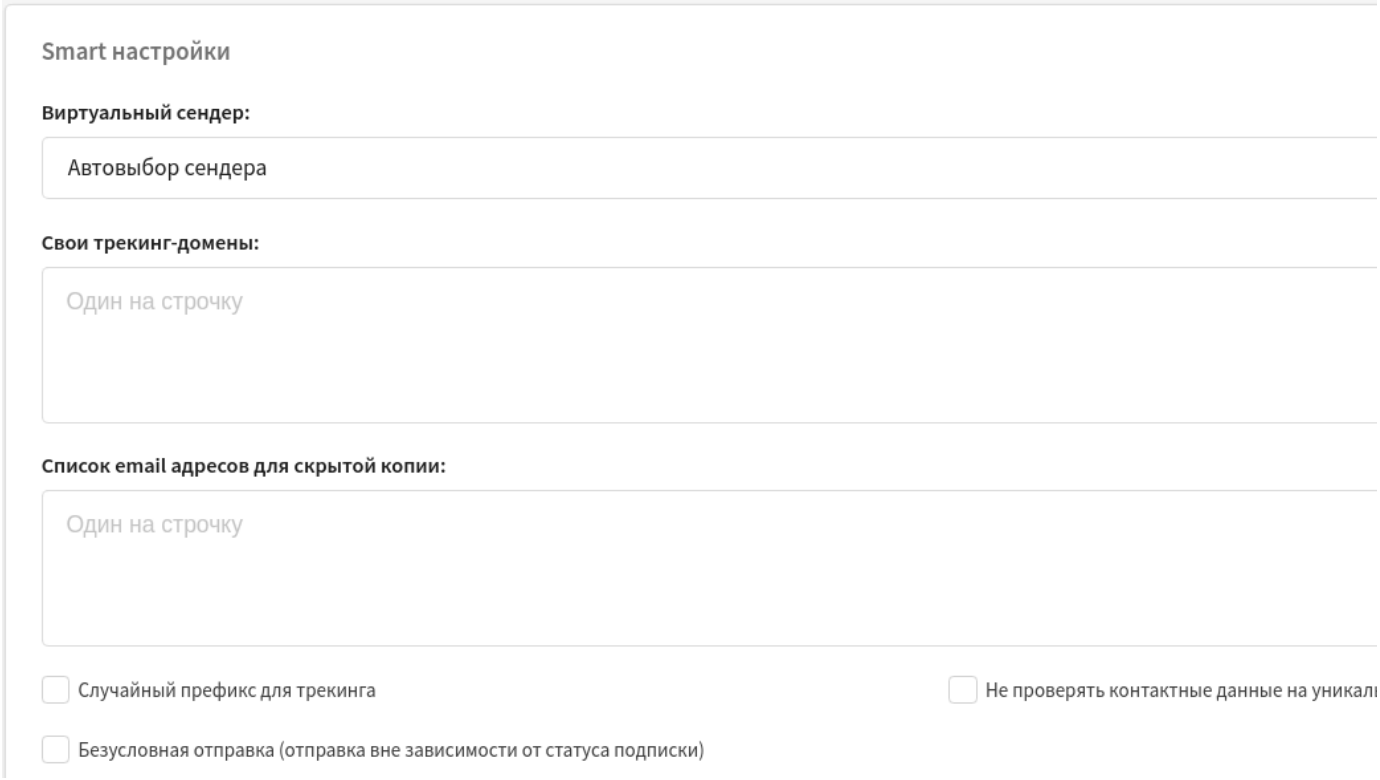
Communication policy

Communication policy sets limits for communication with one subscriber. It is configured in the Limitatio use account policy or resource policy.

Smart settings

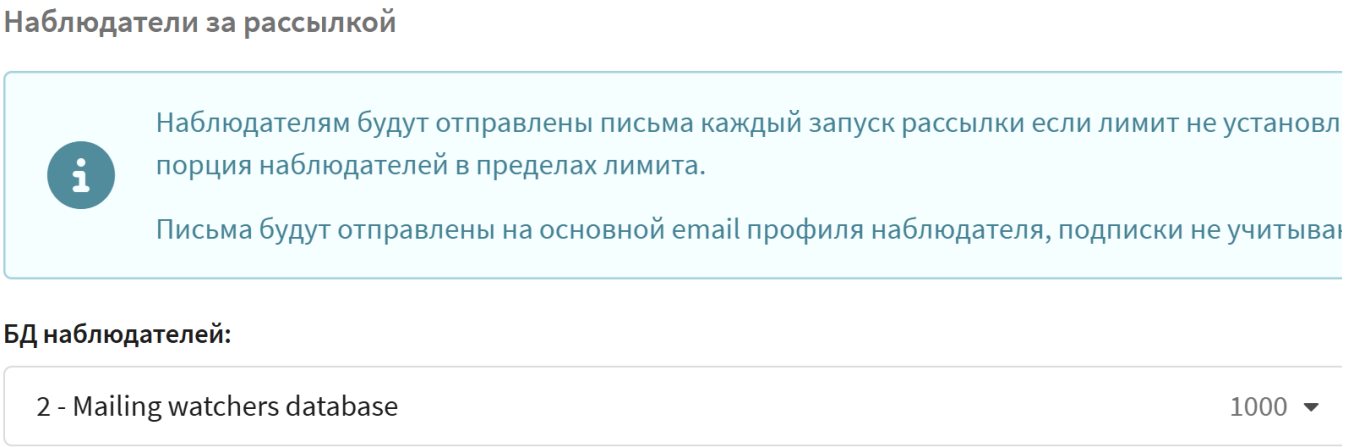
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients bas**



Mailing Watchers

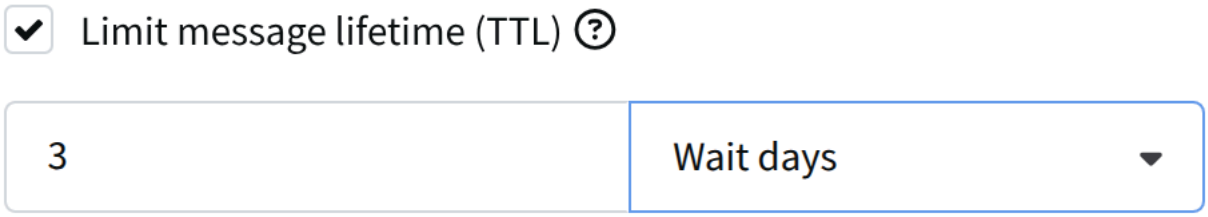
You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mai watchers database. It is created the same way as a regular database, but you must enable the correspo In Smart Settings, select the watchers database and set the message sending limit for each mailing laun



Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is reminder, a one-time code, or an offer that quickly loses relevance.



To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, h

After the specified time expires, the system stops delivery attempts for the undelivered message and re

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration par**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API para Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefor different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and**

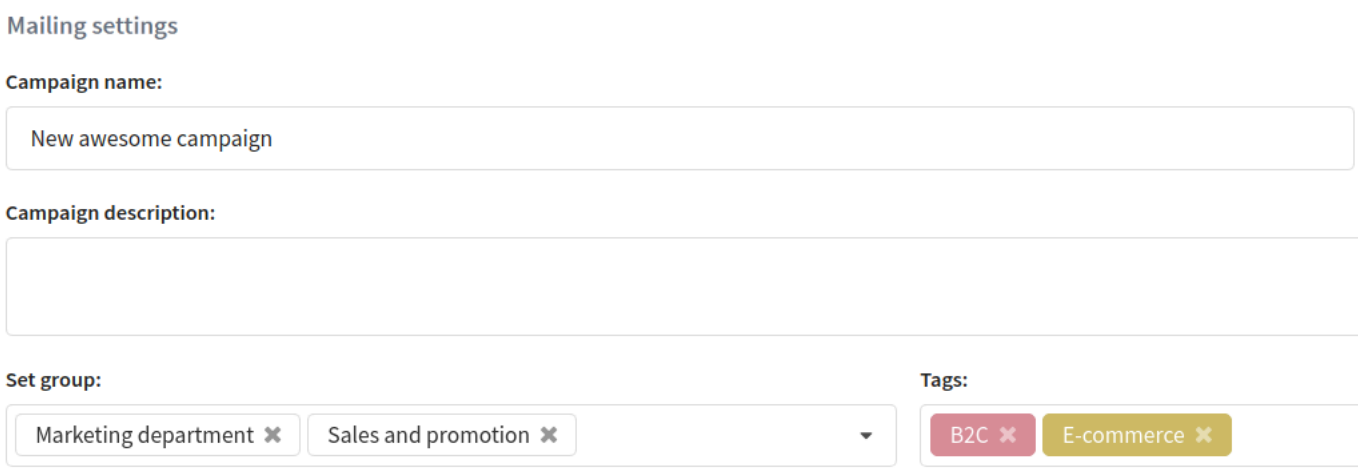
Placement

Description

![img] (/img/attachments/56250257/56250359.png)	**Placement mailing is a tool for placing content for subscribers on a specific resource with the possibility of cancellation at any** After launch, the mailing sends content to each subscriber in the data source (in the case of a scheduled start) or to a specific lead that initiated a certain event (in the case of a triggered start condition). If the channel is configured to unsubscribe, the placed content can be removed from the resource manually or upon reaching a certain date and time. To create a new mailing, go to "Mailings" → "Placements" in the main menu. Click + Create in the top panel. </td> </tr>

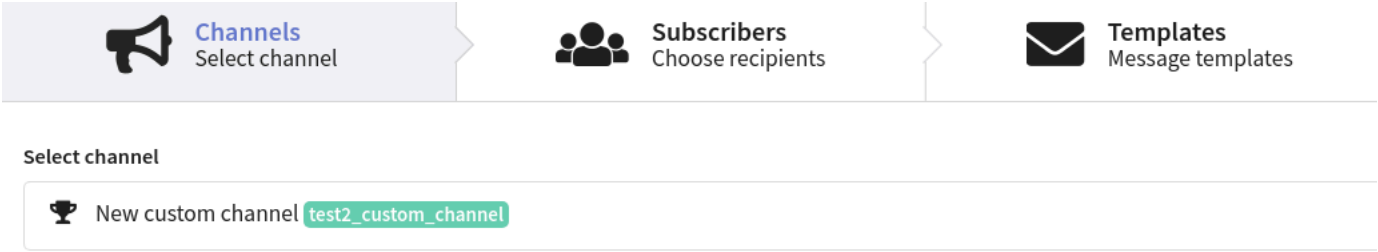
General settings

In the main settings, specify the name and description of the new segment, and if needed, set the acces additional grouping:



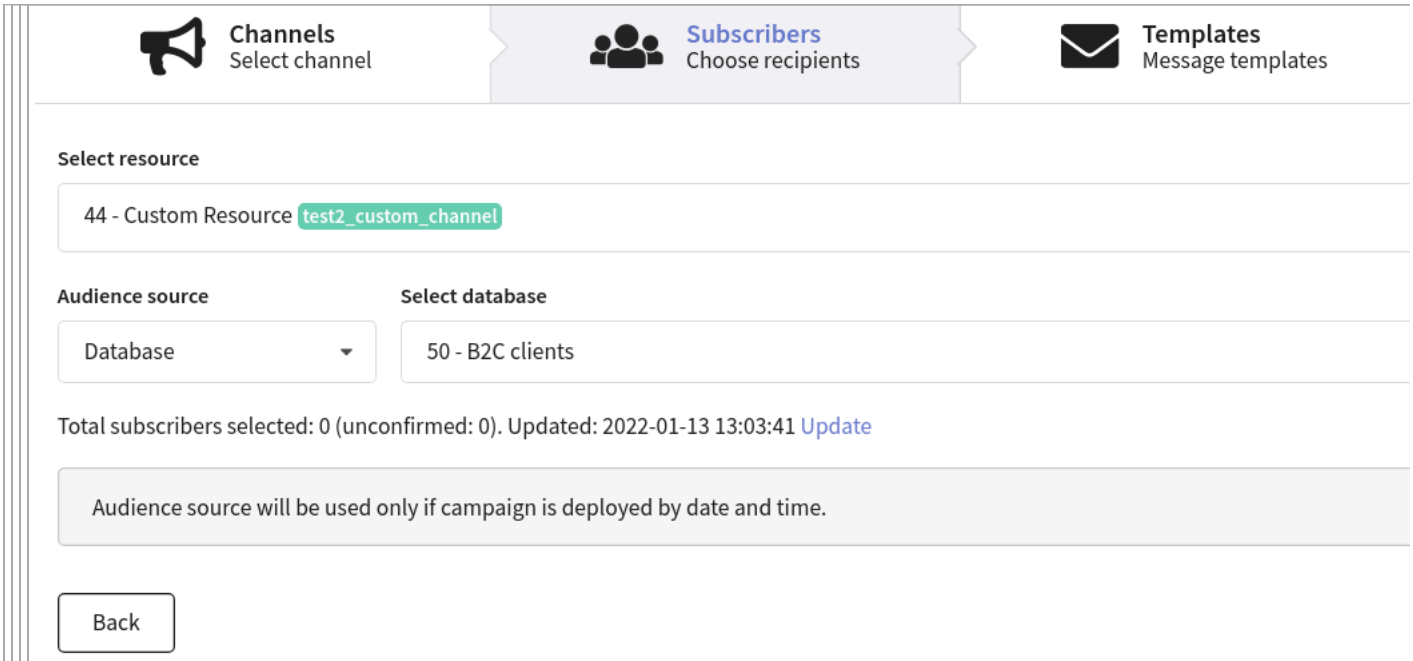
Channel

In the drop-down list, select a channel that can be used to send messages for this mailing type.

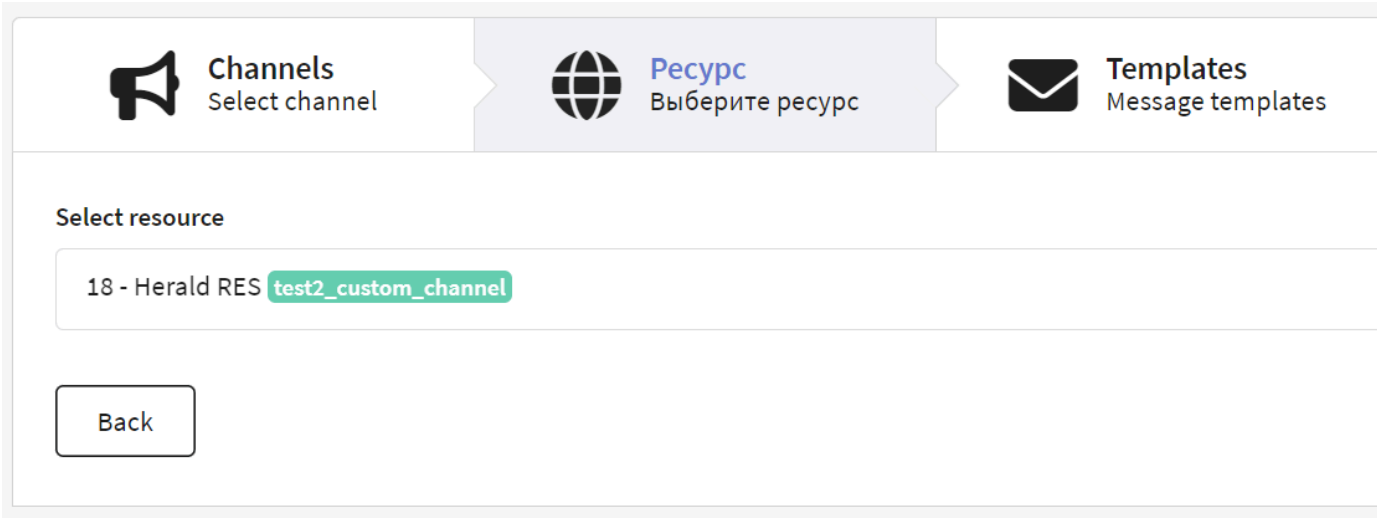


Subscribers

Mailings are sent to customers who subscribe to a certain resource. Resources can be a separate websi direction of your communications with customers. Select a resource. Make sure the recipients in your da Select an audience source. Placement mailing recipients can be a profile database or a segment consist databases.



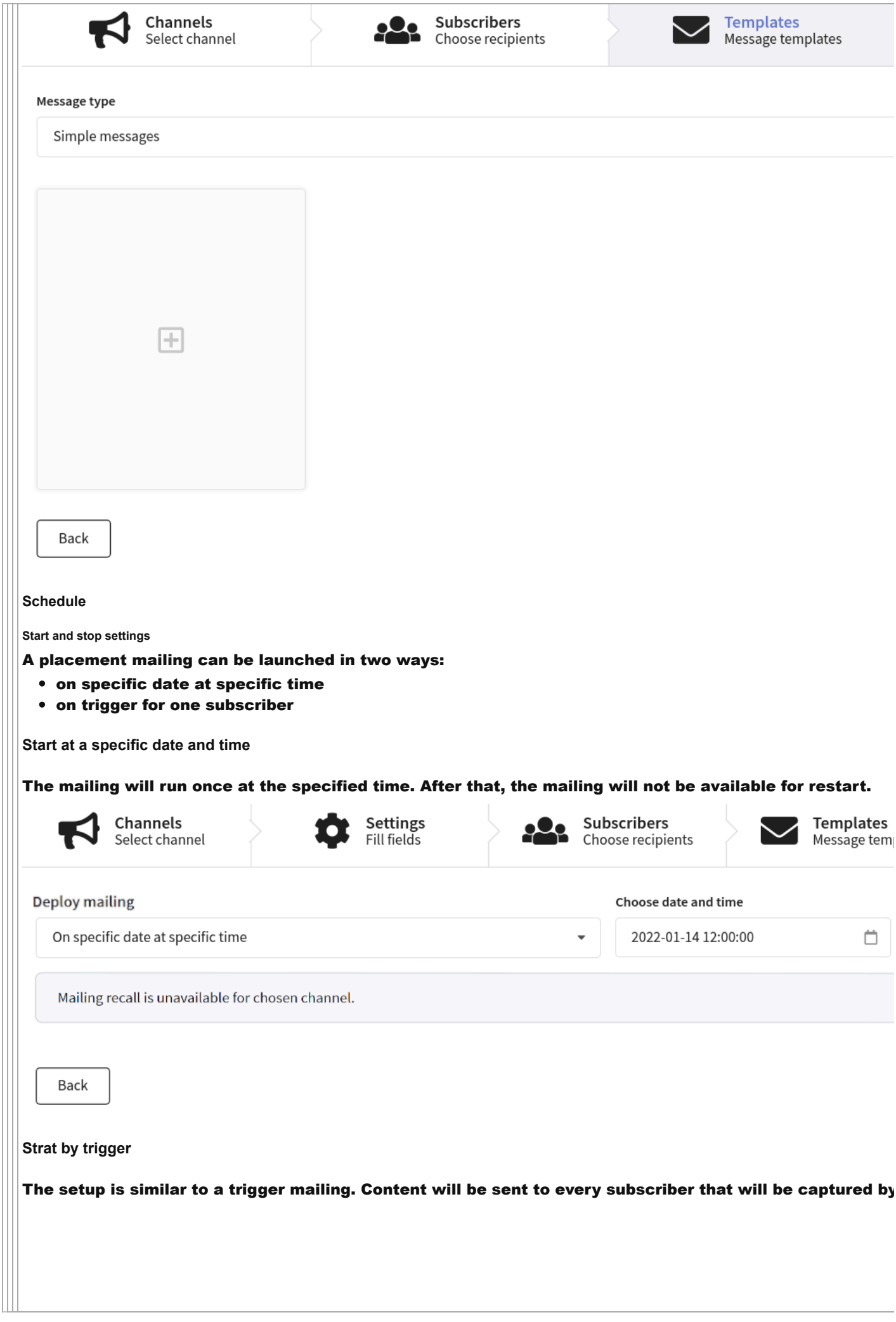
If you are using a placement mailing to send messages through a custom channel, the mailing settings n configuration. For example, some channels may require selecting only the resource without specifying th



Templates

Select the content you want to use in your placement mailing. Click

to add a new template. Any mailing type supports up to 8 message templates. The mailing will send eac message in equal proportions. If you want to preview the message, click on the template thumbnail.



Schedule

Start and stop settings

A placement mailing can be launched in two ways:

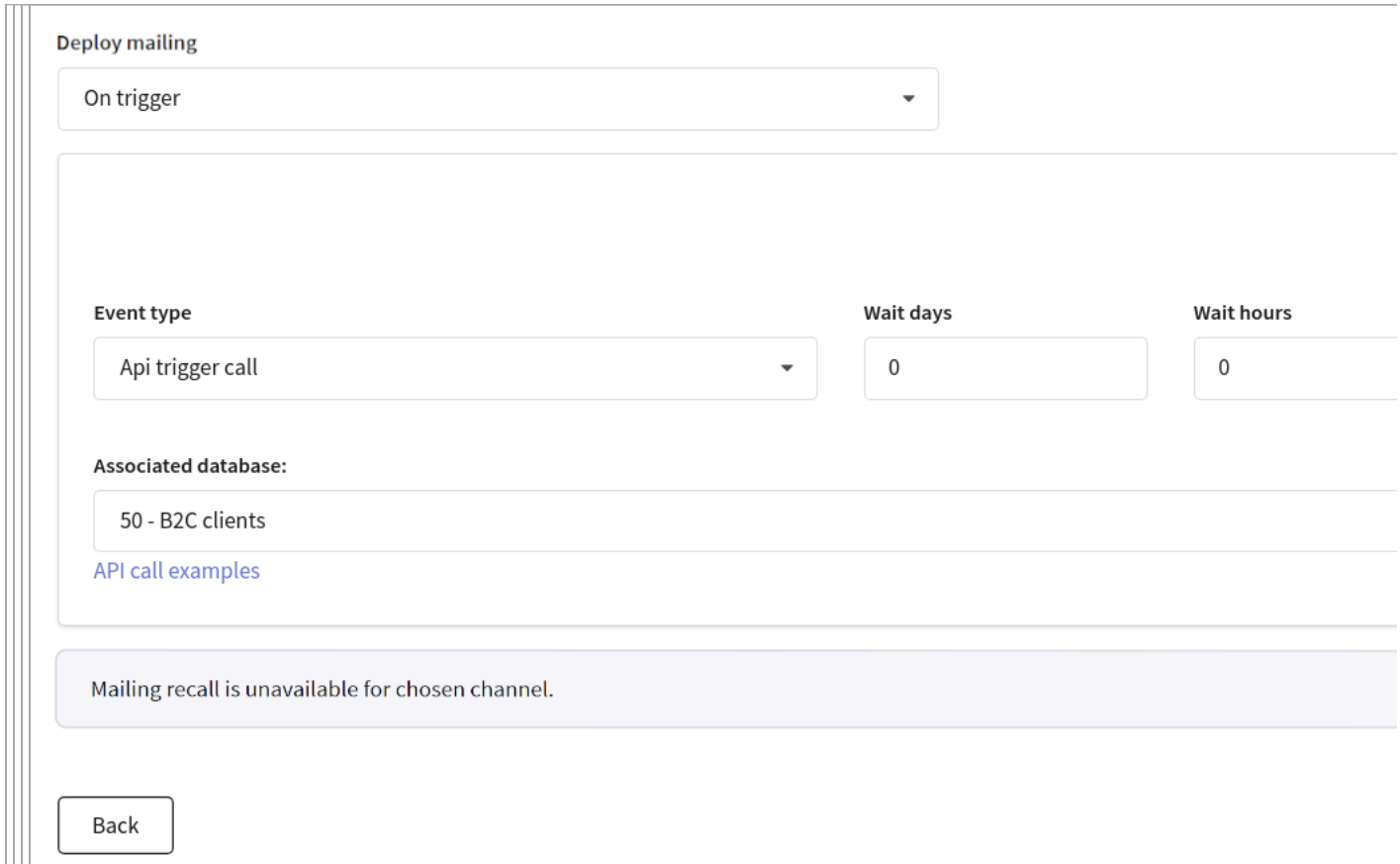
- **on specific date at specific time**
- **on trigger for one subscriber**

Start at a specific date and time

The mailing will run once at the specified time. After that, the mailing will not be available for restart.

Strat by trigger

The setup is similar to a trigger mailing. Content will be sent to every subscriber that will be captured by



INFO: Direct communications and trigger launch for placement

Event-based launch is available only for placements in direct channels. Telegram Group and MAX Group based launch is not available for them.

You can read more about the differences between direct and indirect channels here.

Cancellation of placement

Cancellation of placement is only available for channels where it's configured. If the cancellation was nc configuring the channel, this option will not be available for editing in the mailing.

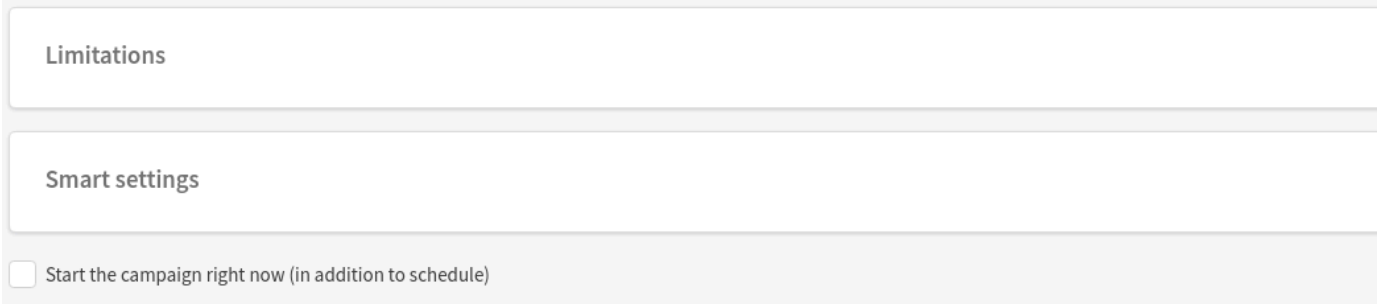
There are two types of cancellation:

- **on specific date at specific time**
- **manually**

Both options will cancel placements for all subscribers to whom the content was sent. It does not depen set to a trigger, the trigger will be deactivated before cancelling.

Placement mailing can be canceled manually if its status is "Deployed".

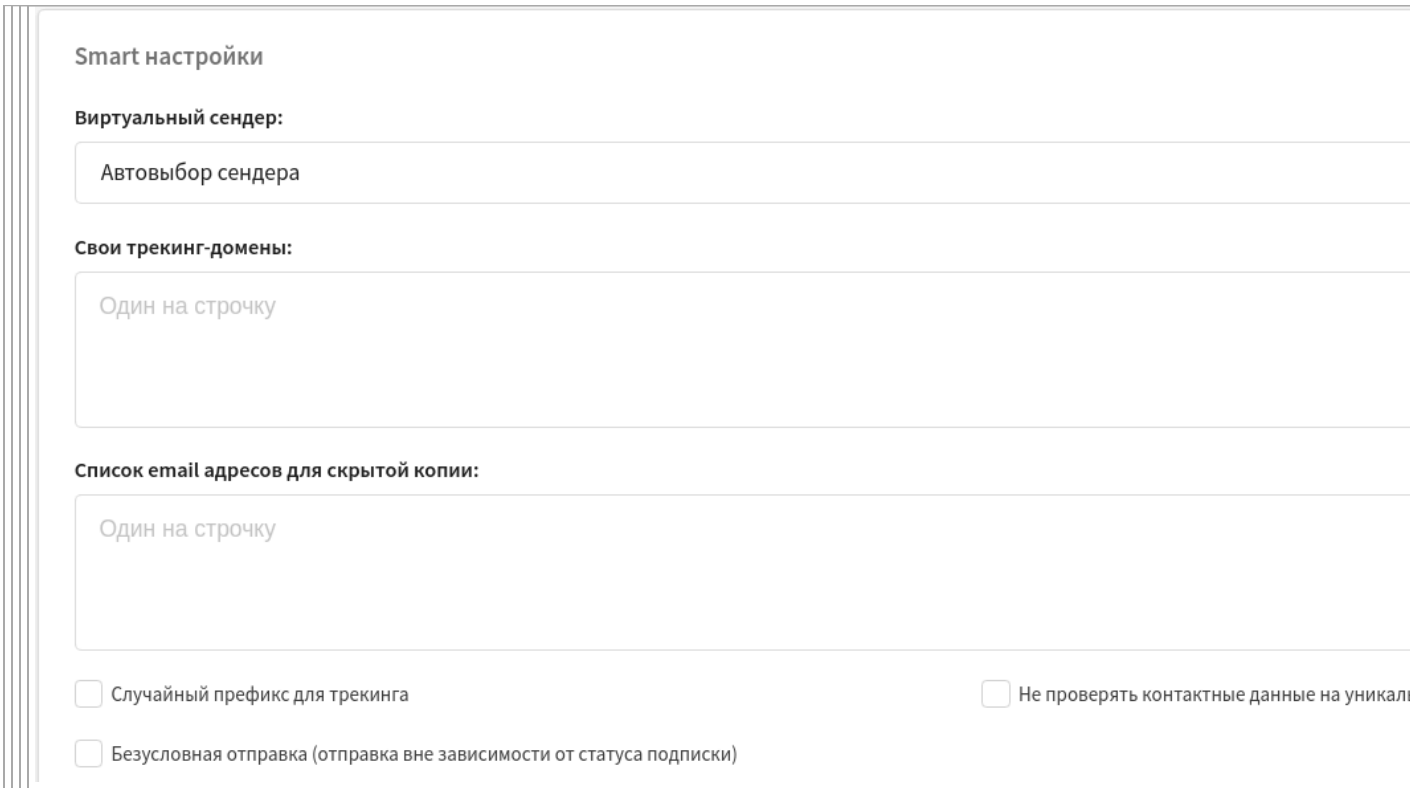
Additional settings



In the Limitations block, you can configure limits and sending policy.

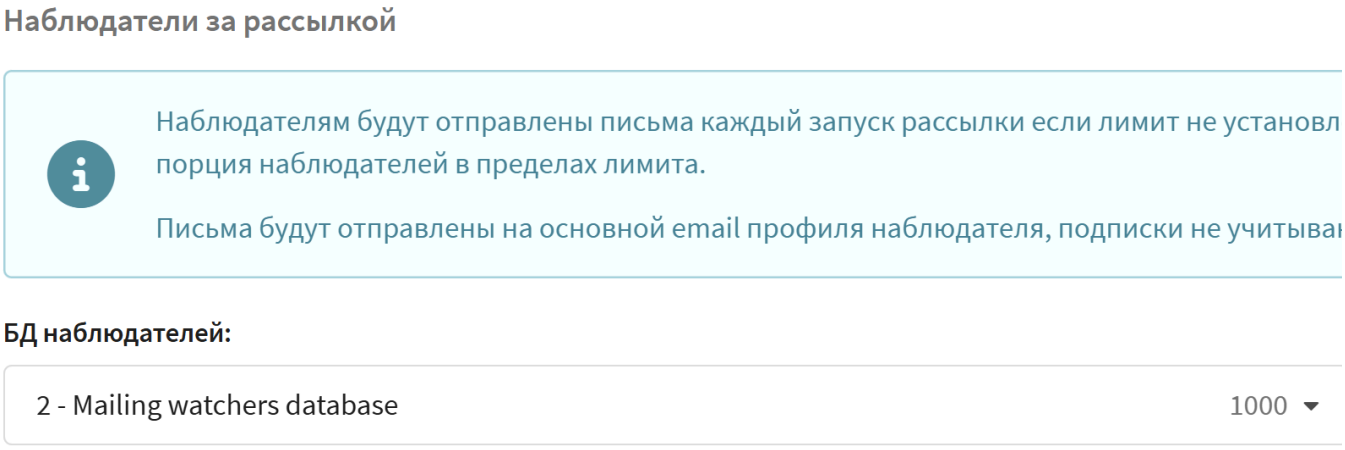
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients bas**



You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mai watchers database. It is created the same way as a regular database, but you must enable the correspo

In Smart Settings, select the watchers database and set the message sending limit for each mailing laun



If you want to launch a mailing right after its creation, toggle the option Start the mailing right now at th

Mailing testing

Description

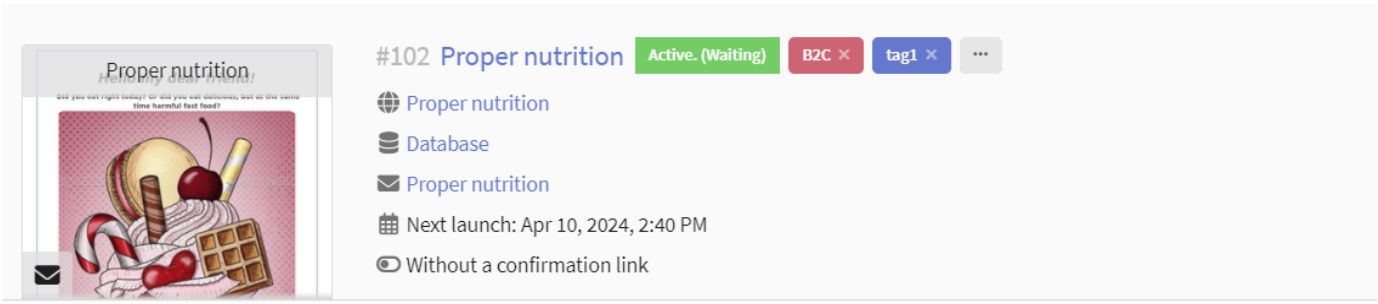
Sometimes you need to test your mailings before activation due to several reasons. This way, you can be

- **content will be properly displayed in different mail clients;**
- **the technical part of sending and delivering the message is all right;**
- **all substitutions and complex logic inside the template work correctly;**
- **email and SMS providers do not block messages because of the content.**

Using mailing testing, you can send letters for verification to the person responsible for your mailings or of messages, delivery infrastructure, etc.

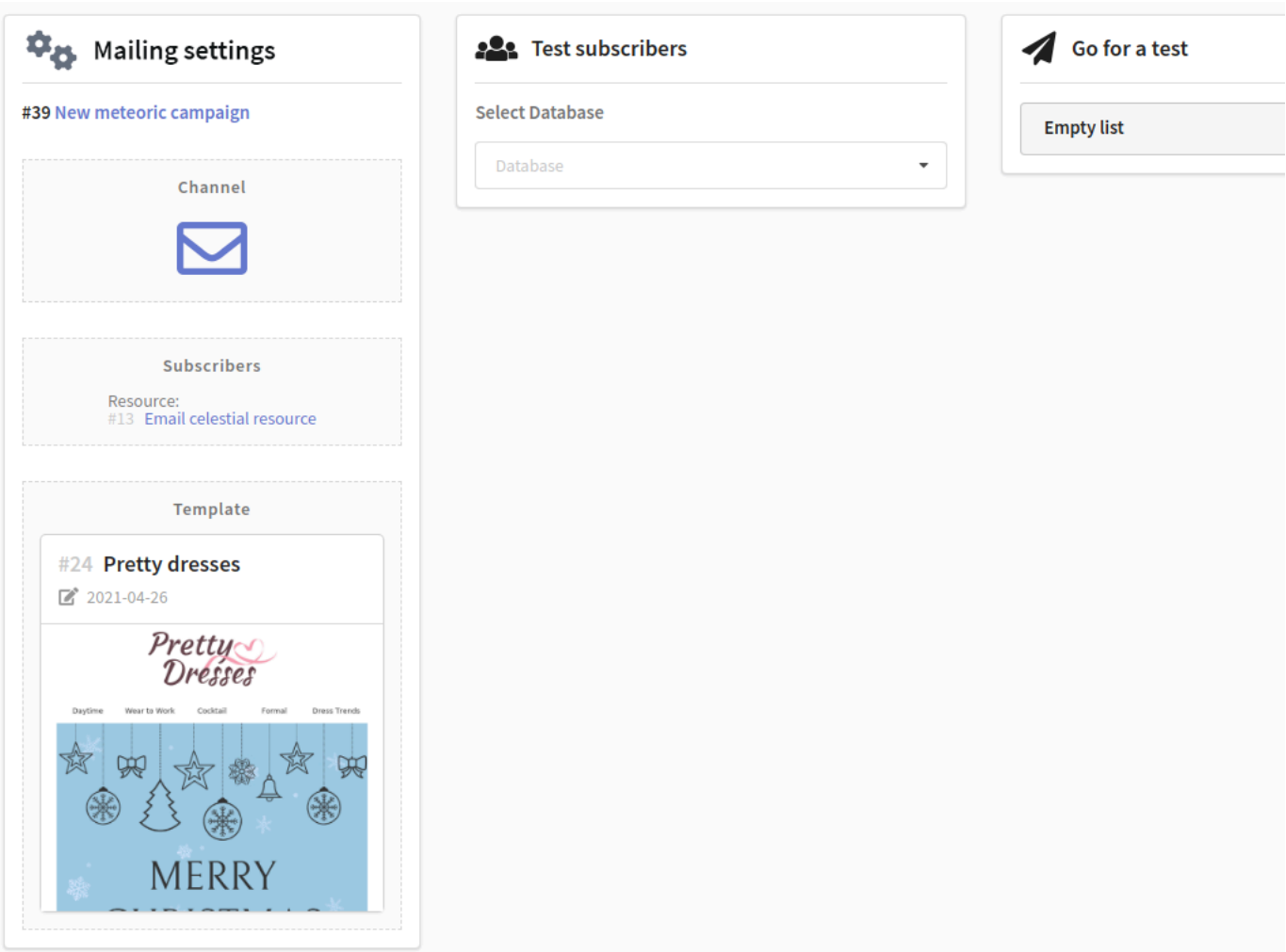
Choose a mailing of a certain type and click the test icon

:

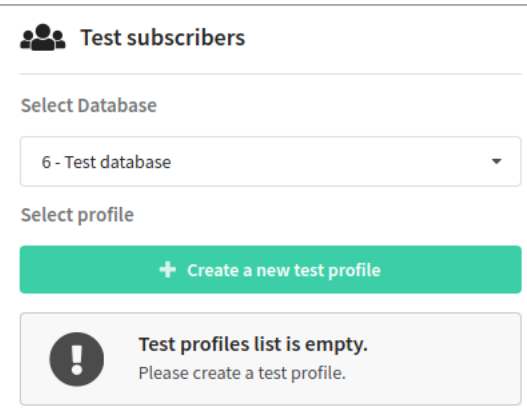


Contacts of test mailing recipients

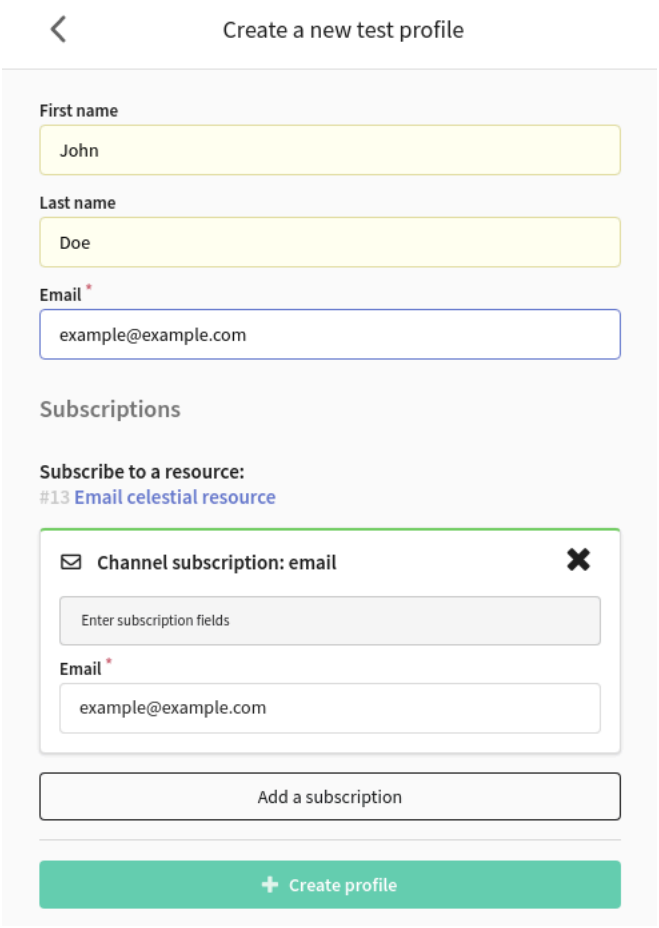
In the new window, you'll see the settings for the mailing you are testing and test subscribers.



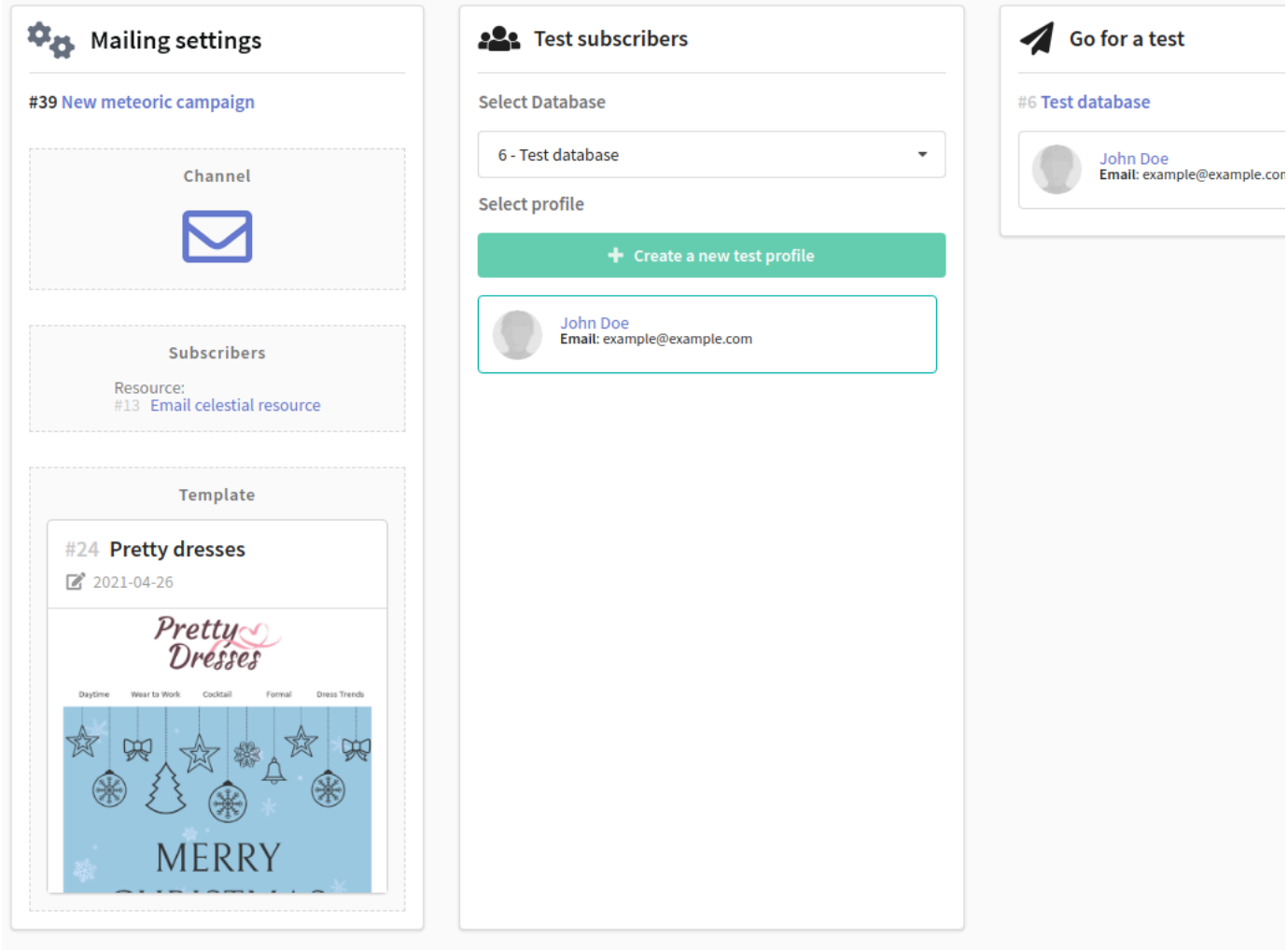
If you don't have test subscribers yet, you can add them. To do this, select the profile base in the drop-do create a test subscriber profile in this database if it has not yet been created. To do this, click on the Cre



Then you'll see the interface for creating a test profile:



After all the data has been entered in the appropriate fields, click the Create profile button. The test prof and will be available for selection:



Click the Test button and a test mailing will be sent. You'll get the test result in the new window:

Test results: #39 New meteoric campaign

Time	Type	Value
2021-04-26 18:22:25	Trigger activated	
2021-04-26 18:22:25	Database ID	0
2021-04-26 18:22:25	Number of subscribers processed	1
2021-04-26 18:22:25	Number of email recipients	1
2021-04-26 18:22:25	Trigger finished	
2021-04-26 18:22:25	Campaign finished	
2021-04-26 18:22:25	Total time spent	351.126175ms

Close

TIP

It is definitely a good practice to have test contacts for all most used email and SMS providers.

Mailing schedule

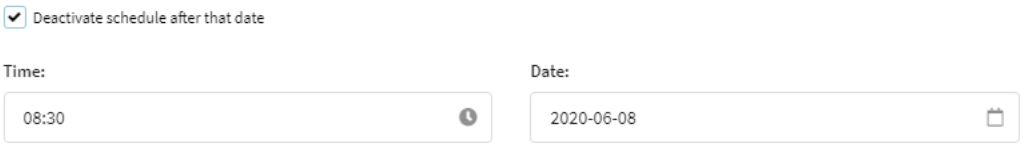
The schedule is applied to broadcast and regular mailings, MVT mailings, scenarios, scheduled imports a
Several plans are available — for different cases. They are described in this article.

INFO

For every schedule plan, you can override the account time zone.

INFO

Deactivate schedule option will help you stop mailing on specific time:

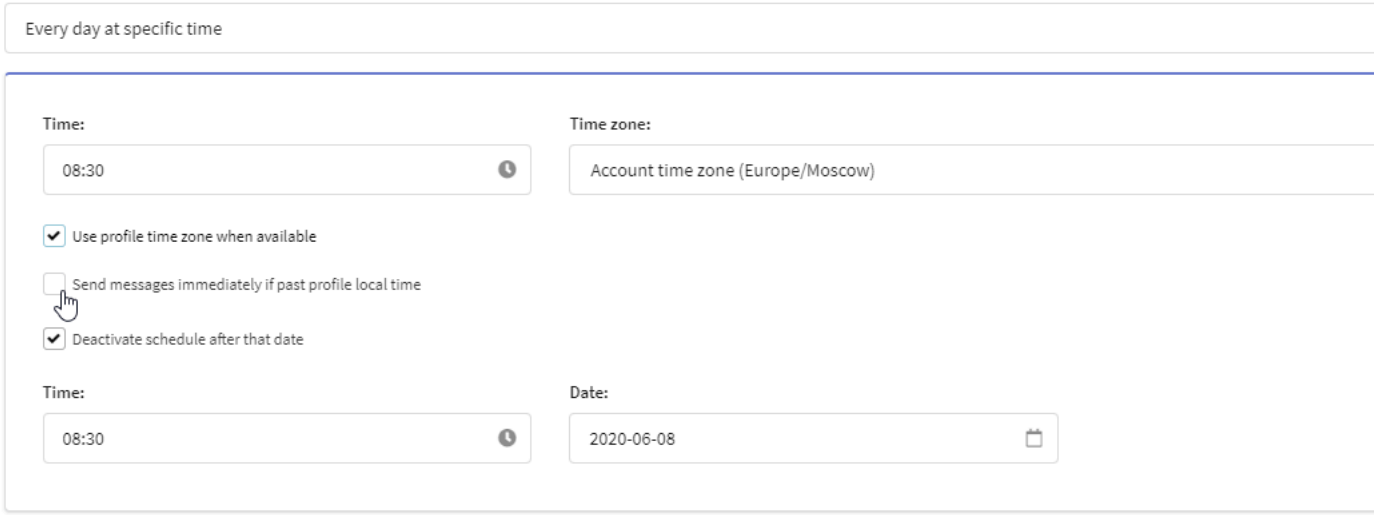


Every day at specific time

This plan will help organize dynamic regular mailings and scenarios: the mailing will start every day for s
birthday or pay day.

TIP

This schedule plan can use customer profile time zone - for delicate mailings. If you are past local time
to send immediately.

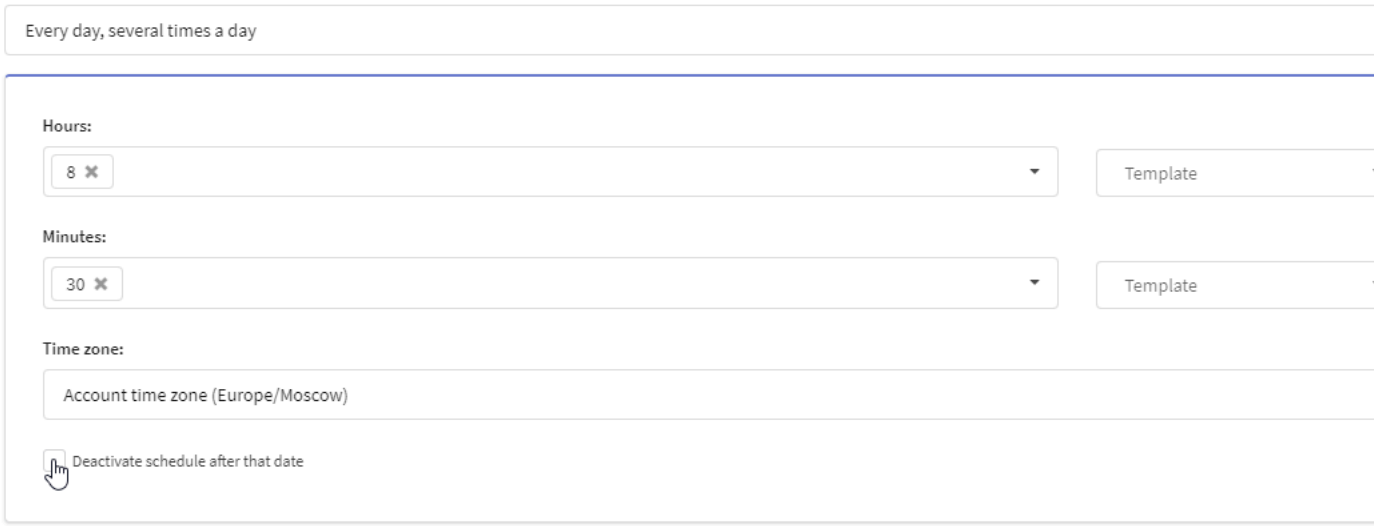


TIP

This plan is also good for large broadcasts, divided by audience limitations.

Every day, several times a day

Can be launched several times every day: specify hours and minutes. You can use templates at the right

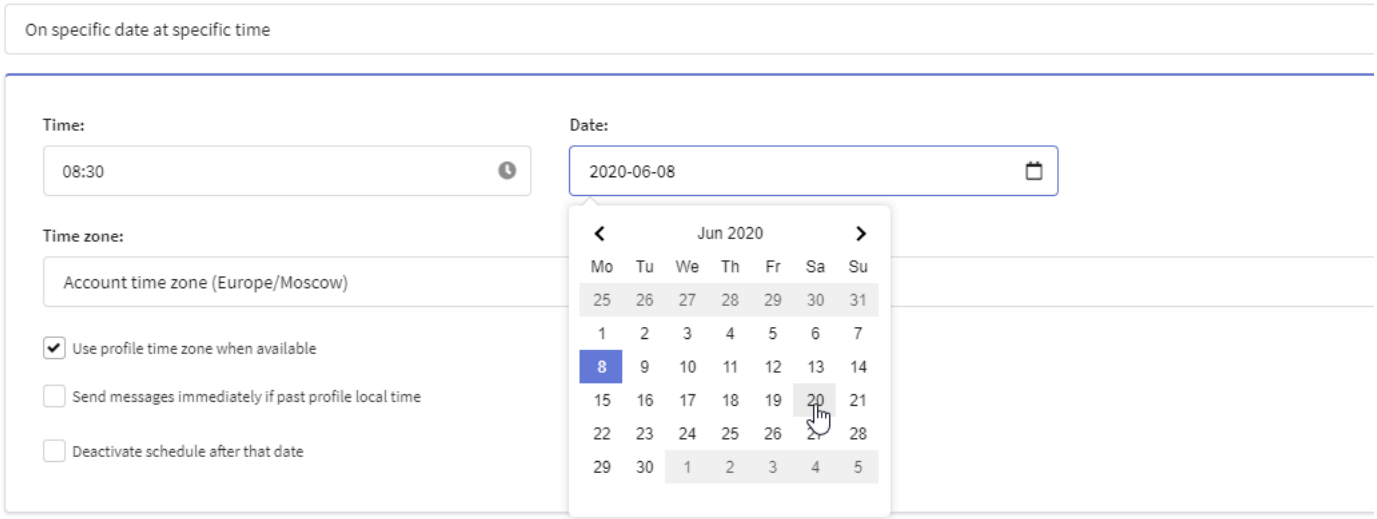


On specific date at specific time

This plan is great for one-time mailings.

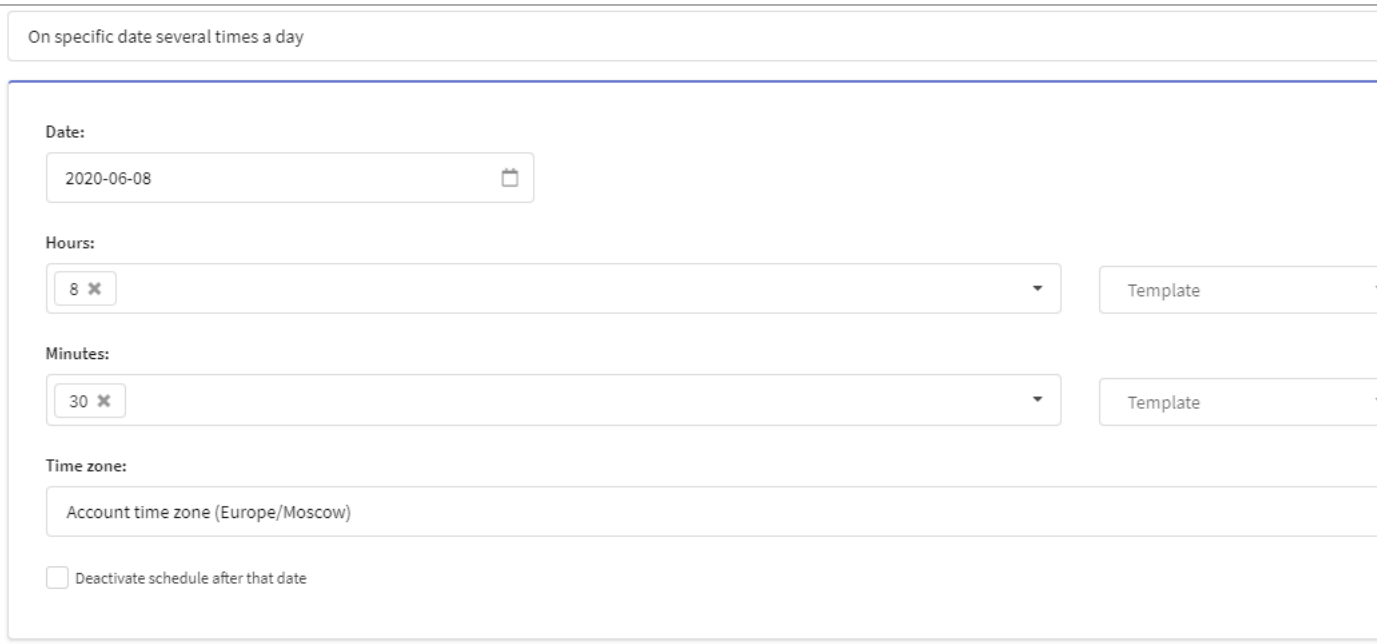
TIP

This schedule plan can use customer profile time zone - for delicate mailings. If you are past local time to send immediately.



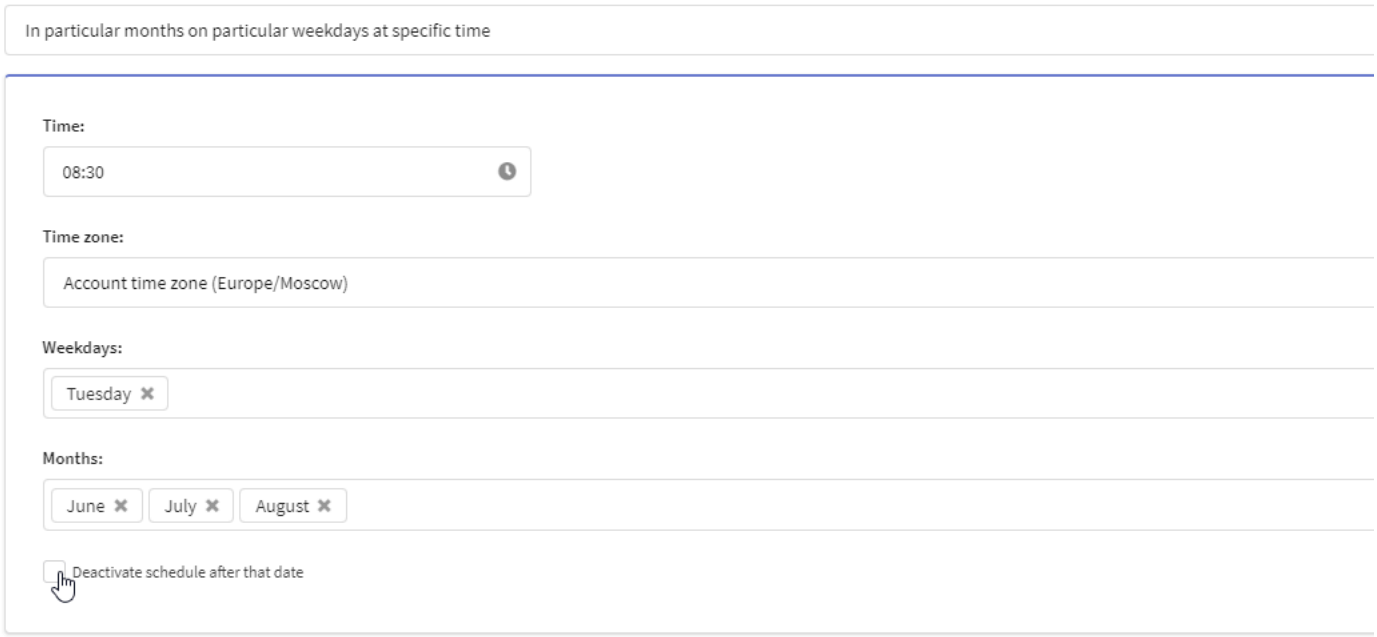
On specific date several times a day

With this plan you can launch mailings several times on one day.



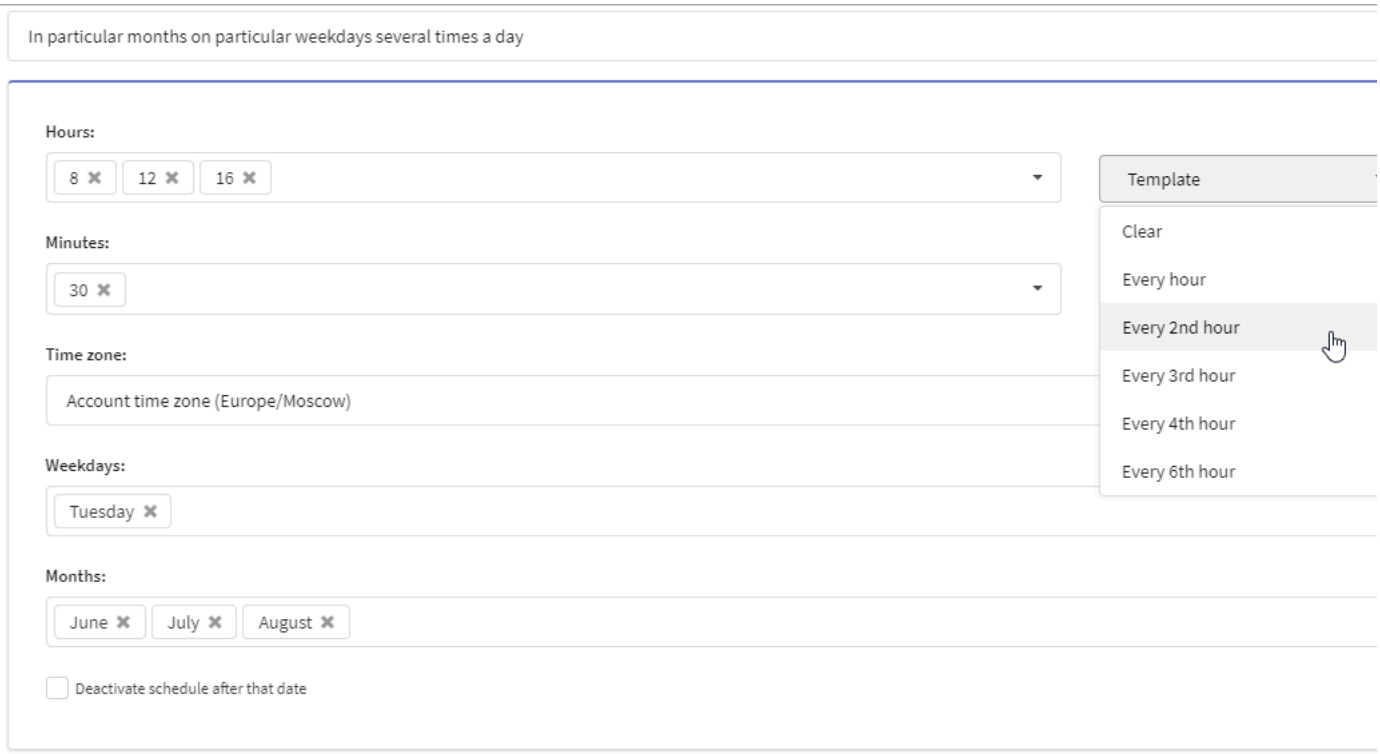
In particular months on particular weekdays at specific time

Use this plan for your weekly mailings: the mailing will start on every weekday specified - in specific mo



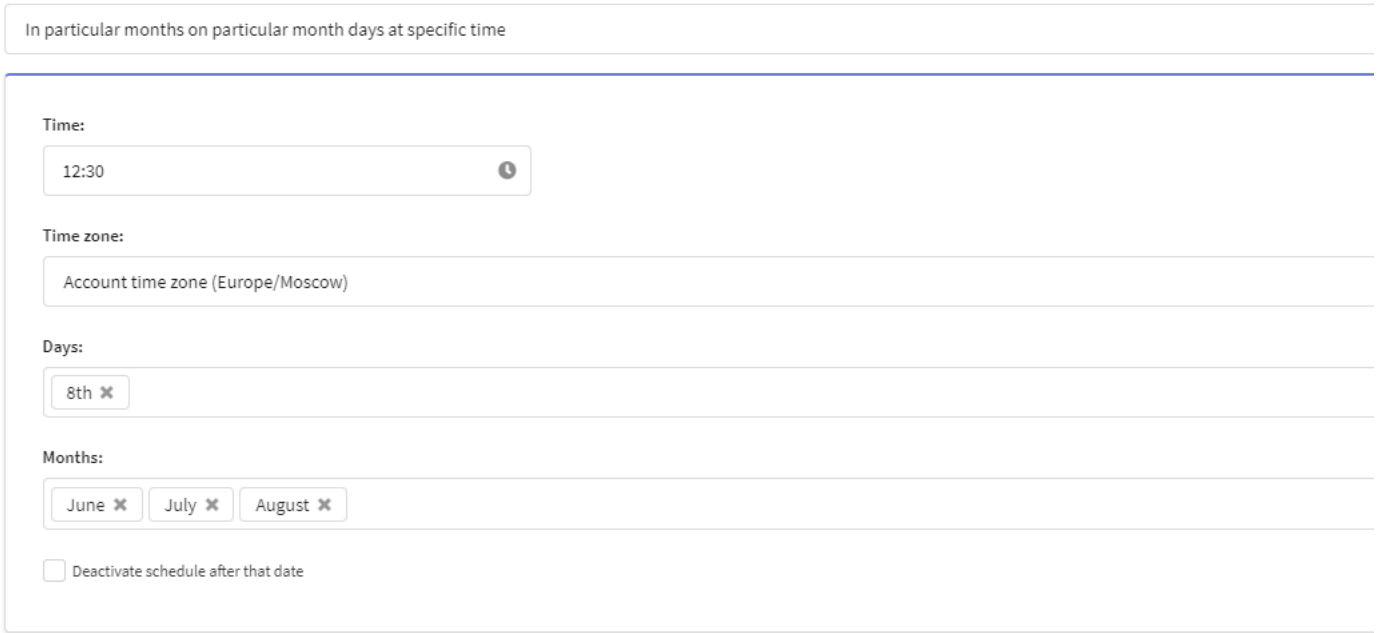
In particular months on particular weekdays several times a day

Here you can also specify multiple hours and minutes.



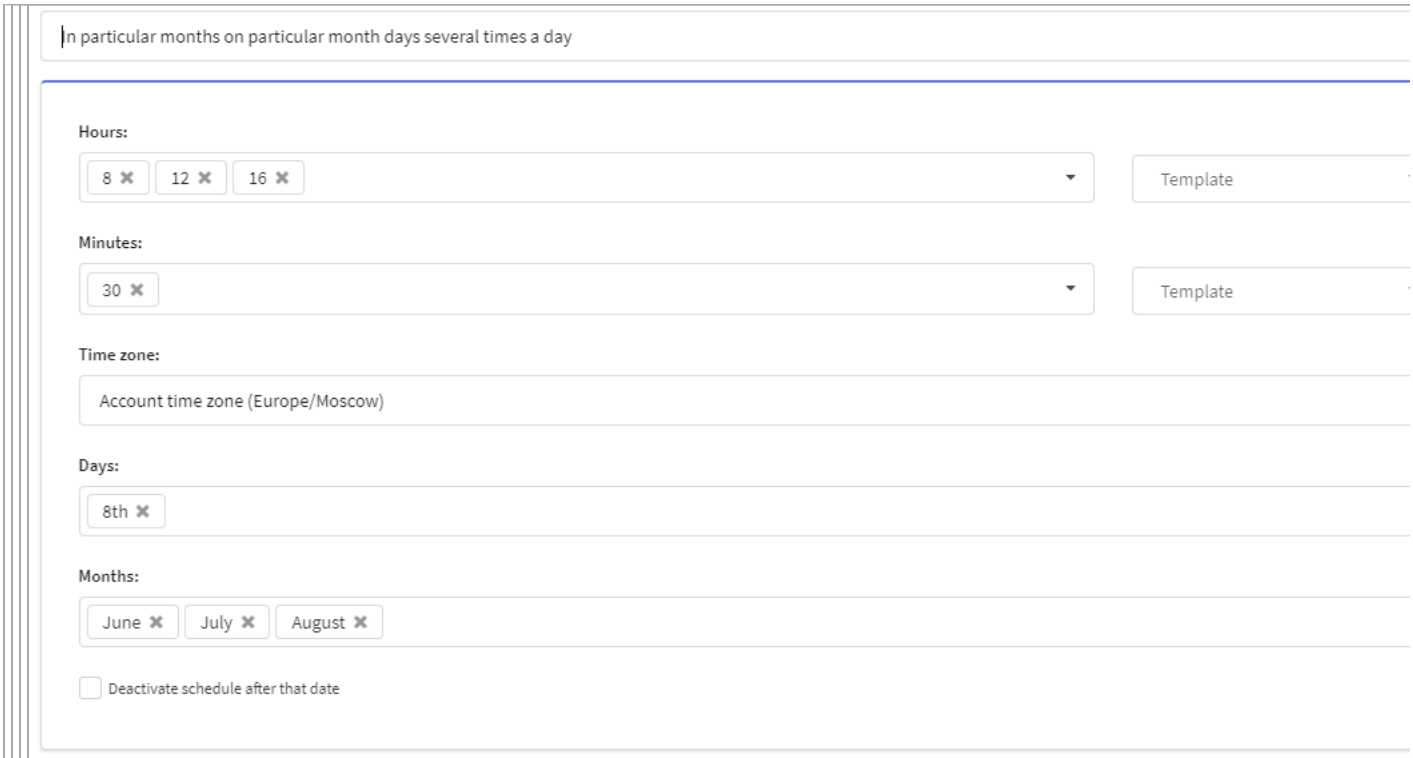
In particular months on particular month days at specific time

This plan uses not weekdays but month days. Thus it is quite good for monthly or quarterly mailings.



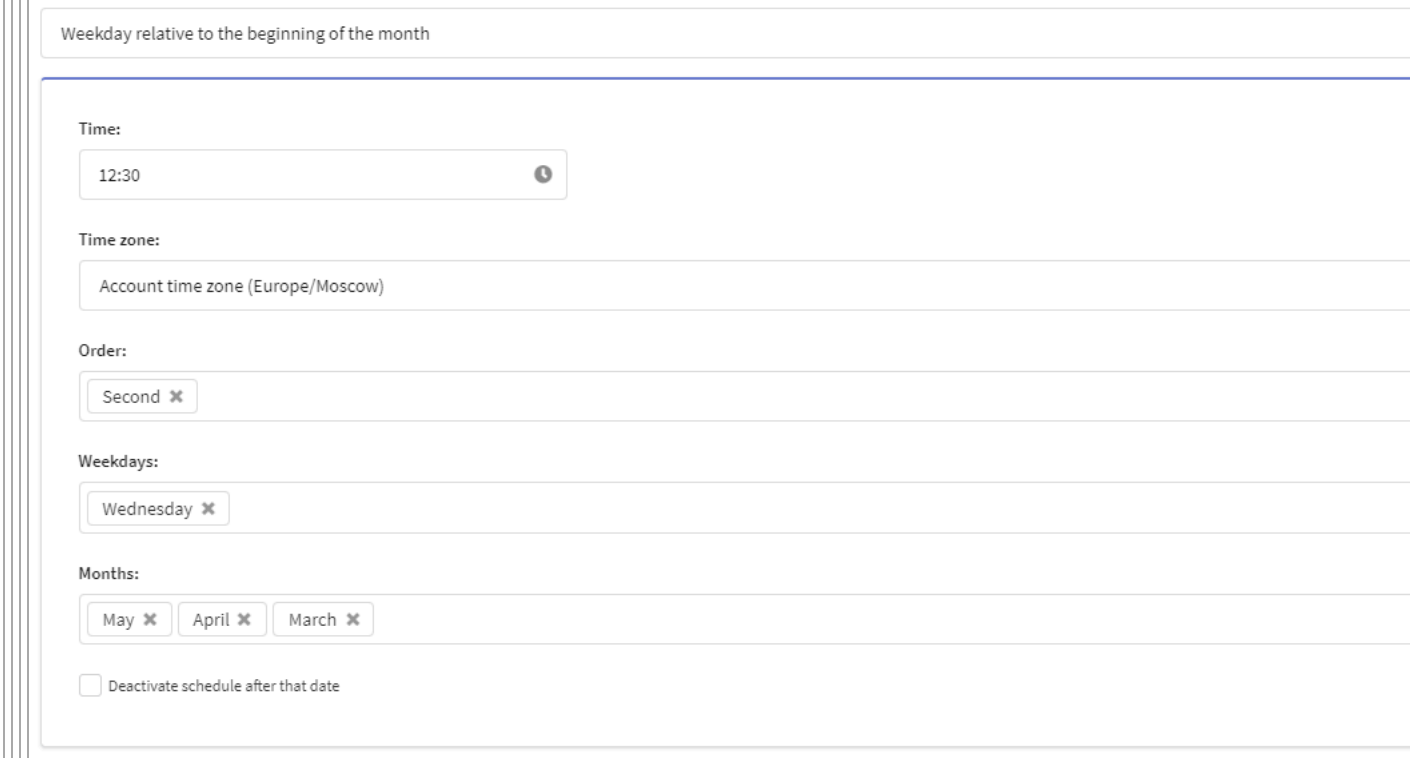
In particular months on particular month days several times a day

Here you can organize several launches a day.



Weekday relative to the beginning of the month

If you need a mailing for e.g. every second Wednesday during springtime - this plan can fit you best:



Mailings calendar

Altcraft mailings calendar is our ultimate solution for organizing your marketing mailings.
It empowers your business with a strategical overview of the situation and an efficient marketing comm

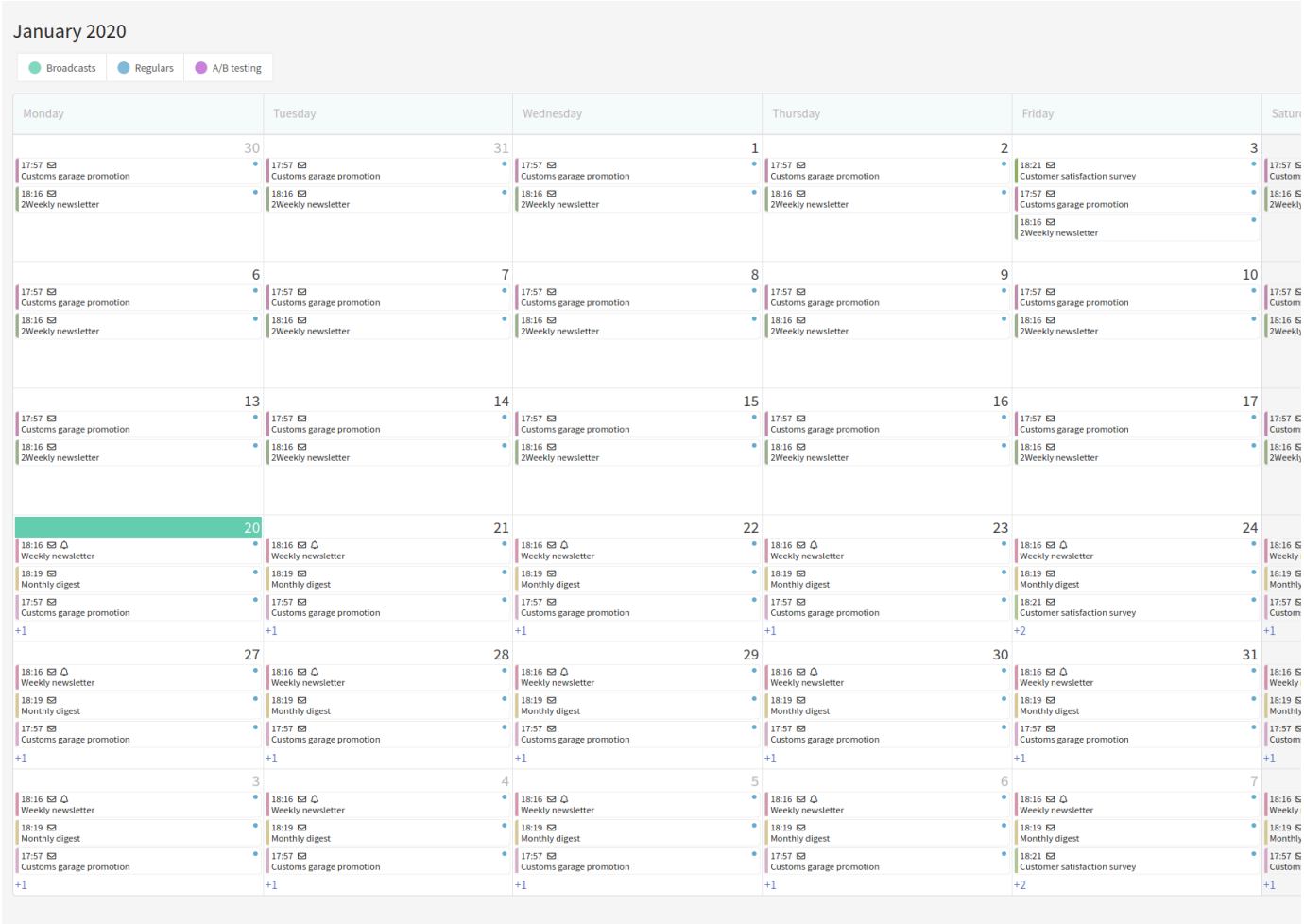
![img] (/img/attachments/3114286/56250420.png)	**Mailings calendar в displays launch time for all your marketing mailings during the custom period of time:** • **Broadcasts — one-time massive mailings** • **Regulars — periodic mailings** • **Multivariate testing (A/B/n) —massive mailings that allow you to select content mailing by best efficiency**

Also, from the calendar, you can create a new mailing of any type, including a trigger mailing, which doe
To access the calendar, in the main menu select Mailings → Calendar.

Viewing mailings information

The calendar displays information about future and past mailings. If you perform massive mailing divided displayed in the calendar. It can help you to control the loading of your sender and network, and email pr
For every mailing, launch time, mailing type and name, and used channels are displayed. Three display m

- **Month — this option displays information about all launches in a month. For every day, three first mai than three launches in a day, a link to detailed information will be shown in the bottom part of the til**
- **Week — this option displays information about all launches in a week. Mailings are grouped by launc**
- **Day — this option displays information about all launches on a separate day.**

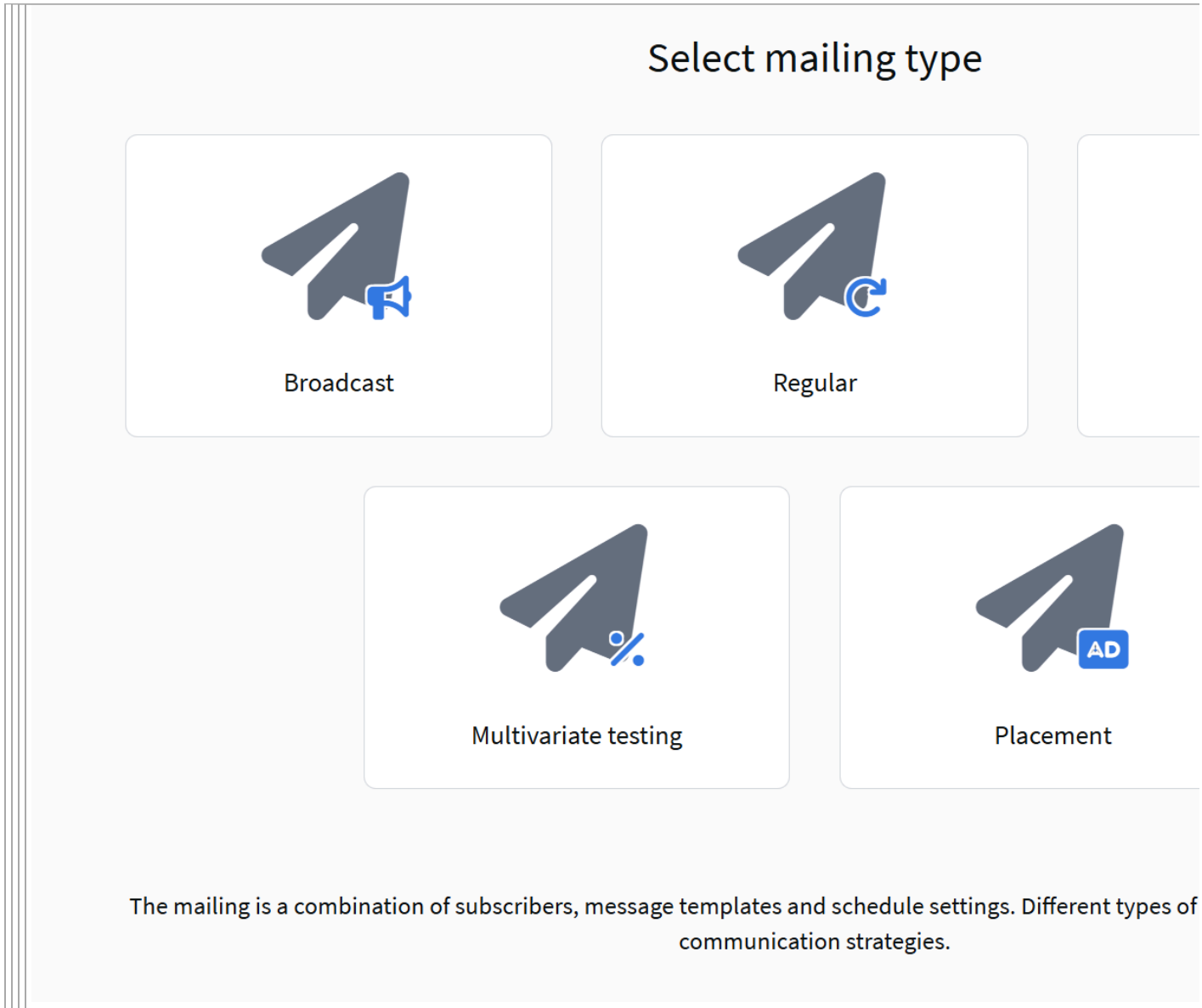


![img] (/img/attachments/3114286/56250424.png)	**Click a mailing label to view details of a mailing. You can clone or edit the mailing and see the mailing summary:** • **Mailing name** • **Message template** • **Database or segment** • **Activation date and time** • **Communication channel** • **Mailing creator** • **Creation date** • **Resource** **In month view you can see all active mailings for a specific month. Use < and > buttons to switch months.**

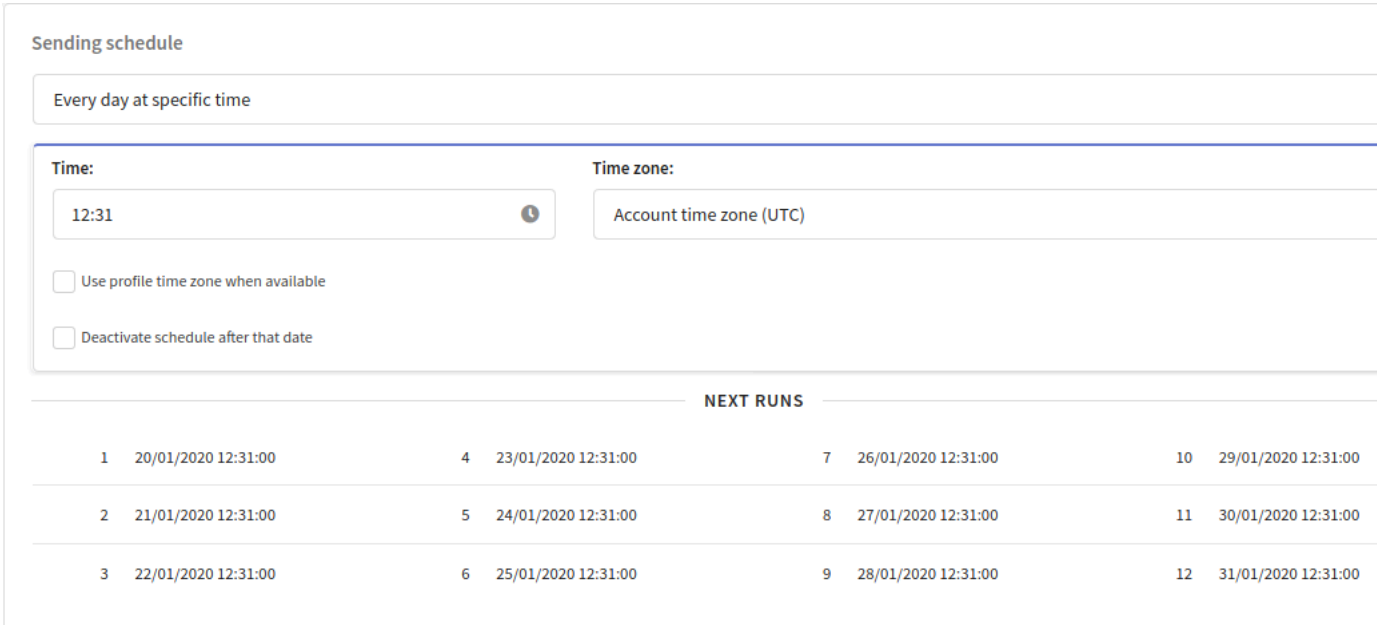
Creating a new mailing

![img] (/img/attachments/3114286/56250415.png)	**You can create a mailing at any time from the calendar interface. Select first launch day, click an empty space in the calendar fo**

The selection panel with all 4 mailing types will open. When you press on the type you want, the editing



Sending date will be preset in the mailing editor, though you can change it as you wish.



Mailing log — errors and troubleshooting

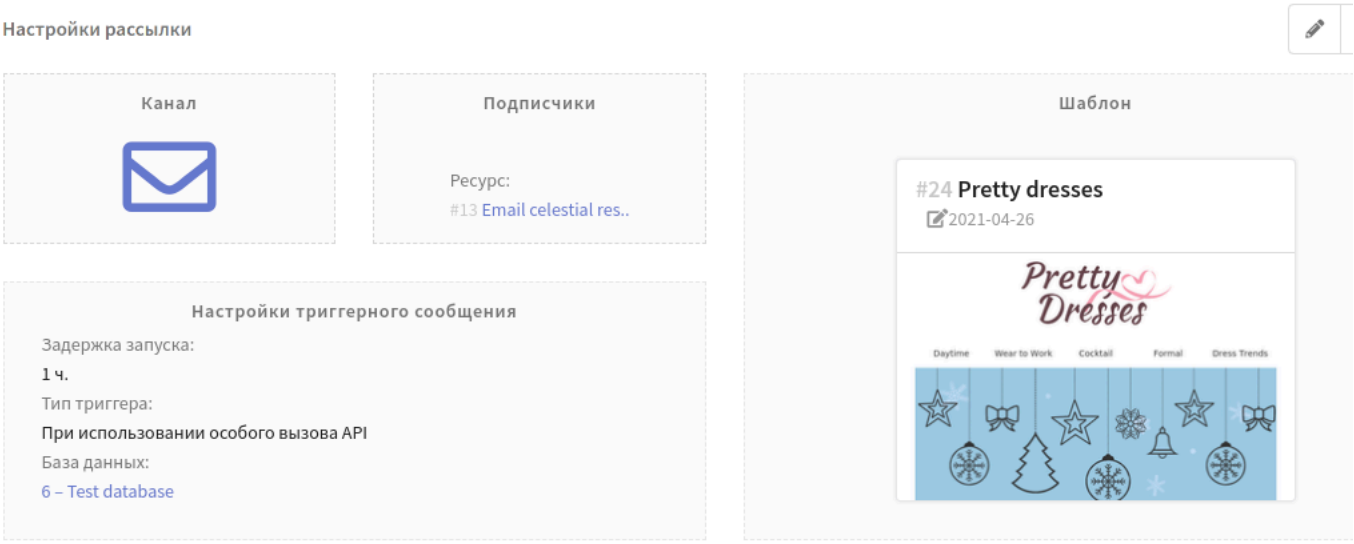
Description

When a mailing is launched, the platform runs a series of checks: validates settings, loads entities (data processes the content templates. If a problem occurs at any stage, the mailing stops and an error appea

This article groups mailing log errors by their cause. Each group has a single troubleshooting approach - combined to avoid repeating the same steps for every key.

Where to view the mailing log

The mailing log is available on the mailing page in the Mailing settings block — the log button is in the up fills in when the mailing starts (including test sends): it shows the preparation stages, sending progress,



Mailing Log:

#61 Welcome email

Time	Type	Value
2026-08-20 09:39:25	Failed templating HTML content	
2026-08-20 09:39:25	Error	template: html:1 unexpected <.> i operand
2026-08-20 09:39:25	Number of subscribers received from the source	12345678
2026-08-20 09:39:25	Number of subscribers found	12344110
2026-08-20 09:39:25	Number of email recipients	0
2026-08-20 09:39:25	Mailing finished	
2026-08-20 09:39:25	Total time spent	194.108313ms

Errors by category

Content templating error

Cause: the message template has a syntax error — an invalid placeholder, an unclosed tag, or an incorre

How to fix:

1. Open the message template and fix the variable syntax — placeholders must be written correctly (fo
2. Make sure the loop operator `{.}` is used only inside loop blocks.
3. Check the HTML markup for unclosed tags.
4. If AMP content is used, make sure the AMP-HTML is valid.
5. Save the template and run the mailing again.

Errors in this group:

Error key	Message
`ERR_PARSE_HTML_CONTENT`	HTML content templating error
`ERR_PARSE_TEXT_CONTENT`	Text content templating error
`ERR_PARSE_SMS_CONTENT`	SMS content templating error
`ERR_PARSE_MSG_SUBJECT_CONTENT`	Message subject templating error
`ERR_PARSE_SUBJECT_CONTENT`	Mailing subject templating error
`ERR_PARSE_MSG_FROM_EMAIL_CONTENT`	Message from email templating error
`ERR_PARSE_MSG_FROM_NAME_CONTENT`	Message from name templating error
`ERR_PARSE_MSG_REPLY_TO_EMAIL_CONTENT`	Message reply-to email templating error
`ERR_PARSE_MSG_REPLY_TO_NAME_CONTENT`	Message reply-to name templating error
`ERR_PARSE_CAMP_FROM_EMAIL_CONTENT`	Mailing from email templating error
`ERR_PARSE_CAMP_FROM_NAME_CONTENT`	Mailing from name templating error
`ERR_PARSE_CAMP_REPLY_TO_EMAIL_CONTENT`	Mailing reply-to email templating error
`ERR_PARSE_CAMP_REPLY_TO_NAME_CONTENT`	Mailing reply-to name templating error
`ERR_PARSE_RESOURCE_FROM_EMAIL_CONTENT`	Resource from email templating error
`ERR_PARSE_RESOURCE_FROM_NAME_CONTENT`	Resource from name templating error
`ERR_PARSE_RESOURCE_REPLY_TO_EMAIL_CONTENT`	Resource reply-to email templating error
`ERR_PARSE_RESOURCE_REPLY_TO_NAME_CONTENT`	Resource reply-to name templating error
`ERR_PARSE_PUSH_BODY_CONTENT`	Push body templating error
`ERR_PARSE_PUSH_TITLE_CONTENT`	Push title templating error
`ERR_PARSE_PUSH_ICON_CONTENT`	Push icon templating error
`ERR_PARSE_PUSH_CLICKURL_CONTENT`	Push link templating error
`ERR_PARSE_PUSH_CUSTOMJSON_CONTENT`	Custom JSON templating error in push message
`ERR_PARSE_AMP_CONTENT`	AMP content templating error
`ERR_PARSE_SITE_CONTENT`	Site content templating error
`ERR_PARSE_JSON_URL_CONTENT`	JSON request URL templating error
`ERR_PARSE_FORM_ACTIONS`	Form templating error
`ERR_PARSE_LOYALTY_ACTIONS`	Loyalty templating error
`ERR_PARSE_RELATION_ACTIONS`	Relation templating error
`ERR_PARSE_CAMP_ATTACH_CONTENT`	Failed to substitute variables in the attached file name/content
`ERR_UNSUBSCRIBE_NOT_EXIST`	Unsubscribe link not found in the message body
`ERR_PARSE_POINT_CONTENT`	Restricted `{.}` operator found outside of a loop
`ERR_CSS_INLINE`	Failed to inline CSS — styles will not be embedded into the email

TIP

The part of the key after `ERR_PARSE_` indicates the affected field: subject, body, from email, reply-to, etc.

Read more about templates and variables: [Altcraft Variables and Functions](#).

Custom channel error (fields, settings, bindings)

Cause: the custom channel settings are invalid — required fields are missing, or the bound entities are u

How to fix:

1. Check the custom channel settings in the mailing and fill in all required fields.
2. Check that the bound entities (resources, forms) exist and have not been deleted.
3. If a resource is used, make sure it has options configured for the required sending type (email, SMS,

Errors in this group:

Error key	Message
`ERR_CANT_PREPARE_CHANNEL_FIELDS`	Failed to prepare the custom channel fields
`ERR_CHECK_CUSTOM_CHANNELS_CAMP_FIELDS`	Invalid custom channel fields
`ERR_CHECK_CUSTOM_CHANNELS_RESOURCE_FIELDS`	Required fields for the custom channel are missing in the resource
`ERR_CAMPAIGN_GET_EXTERNAL_ENTITIES`	Error processing custom channel fields with bound entities in mailings
`ERR_MESSAGE_GET_EXTERNAL_ENTITIES`	Error processing custom channel fields with bound entities in messages
`ERR_UPDATE_TEMPLATE_FIELDS`	Failed to update the custom channel template fields
`ERR_VALIDATE_CHANNEL_ACCOUNT_FIELDS`	Failed to validate the custom channel settings in the account
`RESOURCE_WITHOUT_EMAIL_OPTIONS`	The specified resource has no email options
`RESOURCE_WITHOUT_PUSH_OPTIONS`	The specified resource has no push options
`RESOURCE_WITHOUT_SMS_OPTIONS`	The specified resource has no SMS options

Error key	Message
RESOURCE_WITHOUT_CUSTOM_CHANNEL_OPTIONS	The specified resource has no custom channel options

Read more about custom channel setup: Custom Channels.

File or attachment loading error

Cause: the custom channel template file is not loaded or cannot be read, or the attached files are invalid cannot be compressed.

How to fix:

1. **Make sure the attached files are not empty and do not exceed the total size limit.**
2. **Try re-uploading the files if they are corrupted.**
3. **(For custom channel errors) Check that the custom channel template file is loaded and available.**

Errors in this group:

Error key	Message
ERR_LOAD_CUSTOM_FILE	Custom channel file loading error
ERR_READ_CUSTOM_FILE	Error reading the custom channel template file
ERR_GET_CAMP_ATTACH_CONTENT	Error getting the attached files
ERR_CAMP_ATTACH_FILES_SUM_SIZE	Total attached files size exceeded
ERR_CAMP_INVALID_ATTACH	Invalid or empty attached files
ERR_CAMP_ATTACH_COMPRESS_FAILED	Failed to compress the mailing attached file

Channel, sender or SMS gateway error

Cause: the SMTP sender is not initialized, the SMS gateway does not support the template content type, outdated, or the Viber*™ connection is missing.

How to fix:

1. **Check that the correct virtual sender is selected in the mailing settings. Real senders are configured users do not have direct access to them.**
2. **Check the SMS gateway connection in Settings — Connections. If the gateway does not support the Unicode), select another gateway or change the content.**
3. **If WhatsApp* is used, check the template message (HSM) — its version in the platform may be outda**
4. **Check that the Viber*™ connection exists in the account integrations.**

Errors in this group:

Error key	Message
ERR_INIT_SENDER	Sender initialization error
ERR_NOT_SUPPORTED_IN_DEVINO	The Devino SMS gateway does not support the content type used in the template
ERR_NOT_SUPPORTED_IN_MTS	The MTS SMS gateway does not support the content type used in the template
ERR_NOT_SUPPORTED_IN_SMS_TRAFFIC	The SMS Traffic gateway does not support the content type used in the template
ERR_NOT_SUPPORTED_IN_RAPPORTO	The Rapporto SMS gateway does not support the content type used in the template
ERR_HSM_OBSOLETE	The template message (HSM) in the platform is outdated
ERR_HSM_CHECK	Unable to check the template message (HSM) relevance in the platform
ERROR_VIBER_CONNECTION_NOT_FOUND_IN_INTEGRATION	Viber*™ connection not found in the integration

Read more about channel setup: SMS channel, Viber*™, Connections.

Data source or audience error

Cause: the data source is not found, its type is not recognized, the database or segment is deleted, the S contains no profiles.

How to fix:

1. **Check that the database, segment or SQL query exists and has not been deleted.**
2. **Make sure read access to the segment data is granted.**
3. **Check that the segment returns profiles — if the audience is empty, the mailing will not start.**
4. **If timezone filtering is used, make sure at least one profile matches the selected timezones.**
5. **Check the profile field structure — it must match the database schema.**
6. **If an SQL query is used, make sure the syntax is correct and all tables exist.**

Errors in this group:

Error key	Message
ERR_BAD_DATA_ID	Data source unavailable
ERR_BAD_DATA_TYPE	Unknown data source type
ERR_BAD_LIST_ID	Invalid profile database ID
ERR_LOAD_NON_DELETED_LIST	Profile database missing
ERR_LOAD_ONE_NON_DELETED_LIST	Profile database unavailable
ERR_LOAD_NON_DELETED_SEGMENT	No access to the segment data, check access rights
ERR_VALIDATE_NON_DELETED_SEGMENT	Invalid segment structure
ERR_NO_PROFILES_IN_SOURCE	No profiles found in the source segment
ERR_NO_PROFILES_FOR_TIMEZONES	No profiles found for the selected timezones

Error key	Message
ERR_INVALID_LEAD_FIELDS	Invalid subscriber fields passed
ERR_GET_LL_COUNTER	Error getting the precomputed profile count for the mailing
ERR_GET_CURSOR	Error getting the mailing cursor
ERR_LOAD_QUERIES	Error processing SQL query results
ERR_LOAD_NON_DELETED_QUERIES	Data table query unavailable
ERR_QUERY_PREPARE	Failed to prepare the SQL query for execution
ERR_PARSE_QUERY_RAW	SQL query parsing error

Read more: Segmentation, Managing databases, Managing Data Tables.

Empty content or missing required fields

Cause: the content of the selected type (email/SMS/push) is not filled in, or the required sender fields are

How to fix:

1. **Open the template and fill in the content for the selected channel: HTML body, text version and subje title for push.**
2. **Check that the sender address (From email) is specified — it is set at the mailing, message or resour**
3. **Make sure the sender name (From name) for email or Caller ID for SMS is specified.**

Errors in this group:

Error key	Message
ERR_EMPTY_CONTENT	No content of the required type
ERR_LOAD_EMPTY_CONTENT	The message content is empty
ERR_EMPTY_SOME_EMAIL_CONTENT	Part of the email content is missing
ERR_EMPTY_SOME_PUSH_CONTENT	Part of the push content is missing
ERR_EMPTY_SOME_SMS_CONTENT	Part of the SMS content is missing
ERR_ALL_EMPTY_FROM_EMAIL	Sender address not specified
ERR_ALL_EMPTY_FROM_NAME	Sender name not specified
ERR_ALL_EMPTY_CALLER_ID	Caller ID not specified

Read more about filling in templates: Email template, SMS template, Push template.

Message template or fragment unavailable

Cause: the message template or template fragment has been deleted or is unavailable.

How to fix:

1. **Check that the message template exists and has not been deleted.**
2. **If fragments are used, make sure all of them exist and have not been deleted.**

Errors in this group:

Error key	Message
ERR_LOAD_NON_DELETED_MESSAGE	Failed to load the template: it may have been deleted
ERR_LOAD_NON_DELETED_FRAGMENT	Fragment unavailable

Read more about message templates: Message templates.

Resource, channel, form or connector unavailable

Cause: an entity referenced by the mailing (resource, channel, form, connector, placement) has been del link between two profiles in a database (for example, “employee — company”) that can be used in a mes trigger placement that a trigger mailing is bound to.

How to fix:

1. **Check that the resource specified in the mailing exists and has not been deleted.**
2. **Make sure the channel exists and is active.**
3. **If the mailing uses a form, check that the form has not been deleted.**
4. **For trigger mailings, check that the placements exist and are available.**
5. **Check that the connectors (links between entities) are active.**
6. **If the message template uses a relation between profiles, check that the relation exists.**

Errors in this group:

Error key	Message
ERR_LOAD_RESOURCE	Resource loading error
ERR_RESOURCE_IS_MISSING	Sending error without a resource
ERR_GET_CHANNEL	Error getting channel data
ERR_READ_CHANNEL	Failed to read the channel data when processing a placement
ERR_LOAD_CONNECTORS	Connector loading error
ERR_LOAD_FORM	Form unavailable
ERR_LOAD_RELATION	Relation unavailable
ERR_READ_PLACEMENTS	Placements reading error
ERR_READ_PLACEMENT	Failed to read the placement

Suppression list error

Cause: the suppression list has been deleted or is unavailable, or the platform cannot count the address

How to fix:

1. **Check that the suppression lists (global, database or message) exist and have not been deleted.**
2. **Make sure the global suppression list is available.**
3. **If the error persists after checking the lists, contact technical support.**

Errors in this group:

Error key	Message
`ERR_LOAD_GLOBAL_SUPPRESS`	**Unable to load the global suppression list**
`ERR_LOAD_ONE_NON_DELETED_MESSAGE_SUPPRESS`	**Suppression list loading error**
`ERR_LOAD_ONE_NOT_DELETED_LIST_SUPPRESS`	**Suppression list unavailable**
`ERR_COUNT_GLOBAL_SUPPRESS_EMAILS`	**Error counting emails in the global suppression list**
`ERR_COUNT_GLOBAL_SUPPRESS_PHONES`	**Error counting phone numbers in the global suppression list**
`ERR_COUNT_GLOBAL_SUPPRESS_DOMAINS`	**Error counting domains in the global suppression list**
`ERR_COUNT_LIST_SUPPRESS_EMAILS`	**Error counting emails in the database suppression list**
`ERR_COUNT_LIST_SUPPRESS_PHONES`	**Error counting phone numbers in the database suppression list**
`ERR_COUNT_LIST_SUPPRESS_DOMAINS`	**Error counting domains in the database suppression list**
`ERR_COUNT_MESSAGE_SUPPRESS_EMAILS`	**Error counting emails in the message suppression list**
`ERR_COUNT_MESSAGE_SUPPRESS_PHONES`	**Error counting phone numbers in the message suppression list**
`ERR_COUNT_MESSAGE_SUPPRESS_DOMAINS`	**Error counting domains in the message suppression list**
`ERR_PRECHECK_TARIF_LIMIT`	**Tariff calculation error**

Read more about suppression lists: Suppression lists.

VK Notify template error

Cause: the VK Notify connection is unavailable, or the template (regular or base) has not passed the cor

How to fix:

1. **If the error is about the connection, make sure the VK Notify integration is configured and the accou**
2. **If the error is about the template, contact technical support — users do not have access to the VK N**

Errors in this group:

Error key	Message
`ERR_NOTIFY_TEMPLATE_CHECK`	**Failed to check the VK Notify template for correctness**
`ERR_ NOTIFY_CHECK`	**Failed to check the connection to VK Notify**
`ERR_NOTIFY_TEMPLATE_OBSOLETE`	**The VK Notify template is outdated**
`ERR_BASE_TEMPLATE_CHECK`	**Failed to check whether the VK Notify base template has been updated**
`ERR_BASE_TEMPLATE_OBSOLETE`	**The VK Notify base template is outdated**

`ERR_ NOTIFY_CHECK` — checking the connection to VK Notify. `ERR_NOTIFY_TEMPLATE_CHECK` — checking the messag `ERR_BASE_TEMPLATE_OBSOLETE` — checking the VK Notify base template: the platform compares the saved copy VKontakte; if the base template has changed, the mailing must be updated.

Multivariate test error

Cause: the multivariate test cannot continue: the waiting period has not elapsed, the winner has not bee is specified.

How to fix:

1. **If the test is paused, wait for the waiting period between waves to end.**
2. **If a manual winner selection is required, open the multivariate test settings and select the winner.**
3. **Check that the multivariate test type is correct.**

Errors in this group:

Error key	Messag
`ERR_SPLIT_WAIT_TIME`	**Waiting time has not elapsed**
`ERR_SPLIT_NO_MANUAL_WINNER`	**Winner template not selected**
`ERR_CHOISE_WIN_MSG`	**Winner template not selected**
`ERR_BAD_SPLITTEST_TYPE`	**Invalid multivariate test type**

Read more about multivariate testing: Multivariate testing.

Timezone data loading error

Cause: the platform cannot load the timezone map or the account timezone.

How to fix:

1. **Check that the account timezone is set correctly.**
2. **If the error persists, contact technical support.**

Errors in this group:

Error key	Message
`ERR_GET_TZ_INFO_MAP`	**Error getting the timezone map**

Error key	Message
ERR_GET_TZ_OFFSET_MAP	Failed to load the timezone offset map
ERR_LOAD_ACCOUNT_TIMEZONE	Failed to load the account timezone

Cron scheduler error

Cause: failed to create or delete a cron job for a scheduled or recurring mailing.

How to fix:

1. **Check that the mailing start time is set correctly.**
2. **If the error persists, contact technical support.**

Errors in this group:

Error key	Message
ERR_CRONTAB_NEW	Cron job creation error
ERR_CRONTAB_REMOVE	Failed to delete the schedule

Read more about launch scheduling: Mailing schedule.

Sending policy unavailable or violated

Cause: the sending policy attached to the mailing is not found, or the mailing violates a policy rule (time

How to fix:

1. **Check that the sending policy exists and is attached to the mailing.**
2. **If the mailing violates the policy, change the settings (sending time, frequency) or ask the administra**

Errors in this group:

Error key	Message
ERR_LOAD_POLICY	Sending policy unavailable
ERR_POLICY_RULE_OFFENCE	Sending is prohibited by the policy

Account not found or unavailable

Cause: the account is not found by ID or is inactive.

How to fix: Contact the platform administrator and check the account status.

Errors in this group:

Error key	M
ERR_LOAD_ONE_ACTIVE_ACCOUNT	Account unavailable
ERR_CANNOT_FIND_ACCOUNT	Failed to find the account

Tariff limits or billing error

Cause: the tariff plan limit is exhausted, or the tariff data cannot be loaded.

How to fix:

1. **Check the current tariff plan and remaining limits.**
2. **If the limit is exhausted, wait for the reset or upgrade the tariff.**
3. **If the calculation error persists, contact technical support.**

Errors in this group:

Error key	Message
ERR_CANT_GET_TARIFF	Unable to get the account tariff information
ERR_TARIF_LIMIT	Tariff limit reached

Loyalty program unavailable

Cause: the loyalty program specified in the mailing is not found, is inactive, or is unavailable for promocc

How to fix: Check that the loyalty program exists, is active, and is attached to the mailing.

Errors in this group:

Error key	Mes
ERR_LOAD_ONE_ACTIVE_ACTUAL_LOYALTY	Unable to load the loyalty program
ERR_LOAD_ONE_PROMOCODE_LOYALTY	Loyalty program unavailable

Read more about loyalty programs and promo codes: Loyalty programs.

Account frozen

Cause: the account is frozen (debt, blocking). The mailing cannot start.

How to fix: Pay the debt or contact the administrator to unblock the account.

Errors in this group:

Error key	Message
ERR_ACC_FROZEN	Mailing start error: account is frozen

Template in draft status

Cause: the message template is in the “Draft” status. Sending is impossible.

How to fix: Publish the message template — change its status from “Draft” to “Published”.

Errors in this group:

Error key	Message
`ERR_MESSAGE_IS_DRAFT`	**The message template is marked as a draft**

Read more about template statuses: Working with message templates.

Unsubscribe link missing in the template

Cause: the HTML template does not contain the unsubscribe link placeholder.

How to fix: Add the unsubscribe link placeholder to the HTML template.

Errors in this group:

Error key	Message
`ERR_NO_UNSUB_LINK`	**No unsubscribe link in the template**

Read more about subscription management: Subscription Manager.

Push notifications disabled

Cause: push notifications are disabled at the platform or account level.

How to fix: Contact the administrator to enable push notifications.

Errors in this group:

Error key	Message
`ERR_PUSHES_IS_DISABLED`	**Push notifications disabled**

Read more about the push channel: Push Channel.

Invalid JSON in the push message template

Cause: after variable substitution, the JSON in the push message turned out to be invalid.

How to fix: Check the JSON in the push message settings — after variable substitution, the result must b

Errors in this group:

Error key	Message
`ERR_MARSHAL_PUSH_CUSTOMJSON_CONTENT`	**Error generating JSON in the push message**

Too many tracking pixels in the template

Cause: too many tracking pixels are specified in the template.

How to fix: Reduce the number of tracking pixels in the template.

Errors in this group:

Error key	Mess
`ERR_MANY_PIXELS`	**Too many pixels!**

Read more about pixels: Pixels.

Link processing error (tracking)

Cause: failed to process a link in the content (for example, for tracking).

How to fix: Check the links in the template — they must be correct and reachable.

Errors in this group:

Error key	Message
`ERR_LINK_PROCESSING`	**Failed to process the link**

Read more: How email tracking works.

Placement trigger deactivation error

Cause: failed to deactivate the placement trigger when the mailing finished.

How to fix: Contact technical support.

Errors in this group:

Error key	Messag
`ERR_DEACTIVATE_PLACEMENT_TRIGGER`	**Placement trigger deactivation error**

Read more: Trigger mailing, Placement mailing.

Global control group settings error

Cause: failed to get the ignore settings in the Global control group.

How to fix: Contact technical support.

Errors in this group:

Error key	Message
`ERR_GET_IGNORE_GCG_SETTINGS`	**Failed to get the ignore settings in the Global control group**

Read more: Global control groups.

Platform system errors

The errors below relate to internal platform processes and do not depend on the mailing settings. Conta

Errors in this group:

Error key	Message
`ERR_BAD_CAMPAIGN_TYPE`	**Unknown mailing type**
`ERR_SAVE_CAMPAIGN`	**Mailing save error**
`ERR_GET_CAMPAIGN`	**Error getting the mailing data**
`ERR_FAILED_START_PUBLISHER`	**Publisher initialization error**
`ERR_FAILED_GET_PUSH_TEMPLATES`	**Failed to get the push templates**
`ERR_FORCE_REGEN_OFF`	**Error disabling the forced preview generation**
`ERR_LOAD_ALL_NON_DELETED_VSENDERS`	**Virtual senders unavailable**
`ERROR`	**Error**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

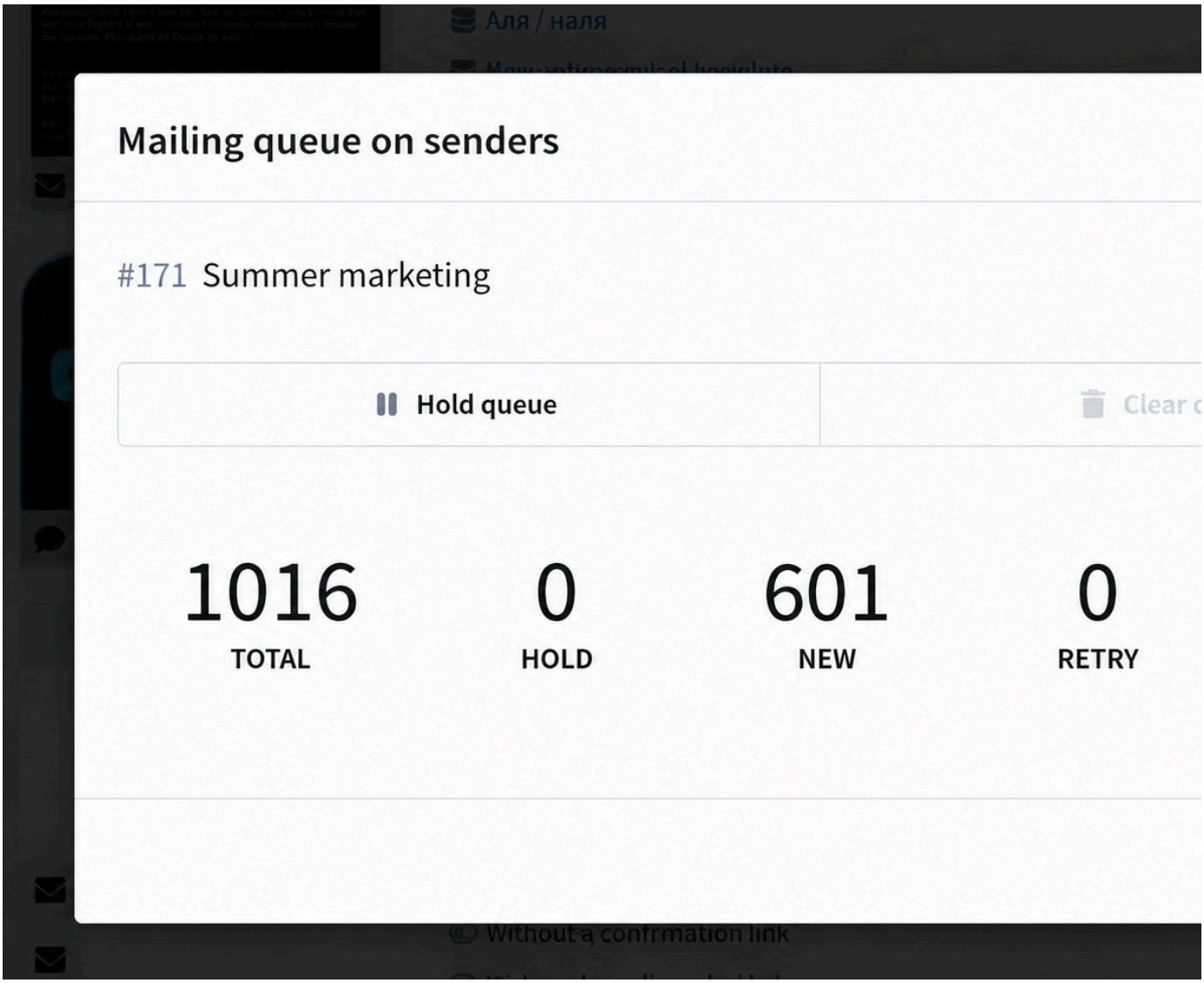
Managing the Sender Queue

After launching an email mailing, messages are passed to the sender queue for delivery. You can monitor pause sending if needed.

To open the queue management panel, select a mailing and click **Sender Queue in th**

![img](/img/attachments/managing-a-queue/en/quick-actions.png)

Queue Metrics



The management panel displays five key queue metrics:

Metric	Description
Total	**The total number of mailing messages across all sender queues**
Frozen	**Messages that have been paused — they are not sent until unfrozen**
New	**Messages that have not yet been attempted for sending**
Retries	**Messages that had sending attempts but did not complete successfully (received a Soft Bounce). They are waiting for a retry**
Sending	**Messages that are currently being sent to the recipient's SMTP server**

Total equals the sum of New, Retries, Sending, and Frozen messages.

Freezing the Queue

The Freeze Queue button allows you to temporarily pause message sending for the mailing on senders.

When frozen, messages in the New and Retries states transition to the Frozen state. Messages that are a (Sending) are not affected by the freeze — they will be delivered.

Frozen messages do not participate in sending until you unfreeze them. After unfreezing, they return to t sending.

Clearing the Queue

The Clear Queue button allows you to remove messages from the mailing queue. When clearing:

- **Messages in the New, Frozen and Retries states are marked as deleted and will no longer be sent.**
- **Messages in the Sending state are not affected — they will be delivered.**

The counters are updated to reflect the current queue status.

Automation scenarios

Description

Scenarios are sequences of actions used for marketing workflow automation. Scenarios can be convenie and their actions during the certain time periods. You can also use them to make your marketing proces Welcome sequences, abandoned carts, reactivation, cross-sales and many other mechanics can be easil

Scenarios can be launched:

- **Individually for each customer after a certain event, same as a trigger mailing.**
- **On schedule for a segment of customers, same as a regular mailing.**

In this section

Managing scenarios

INFO

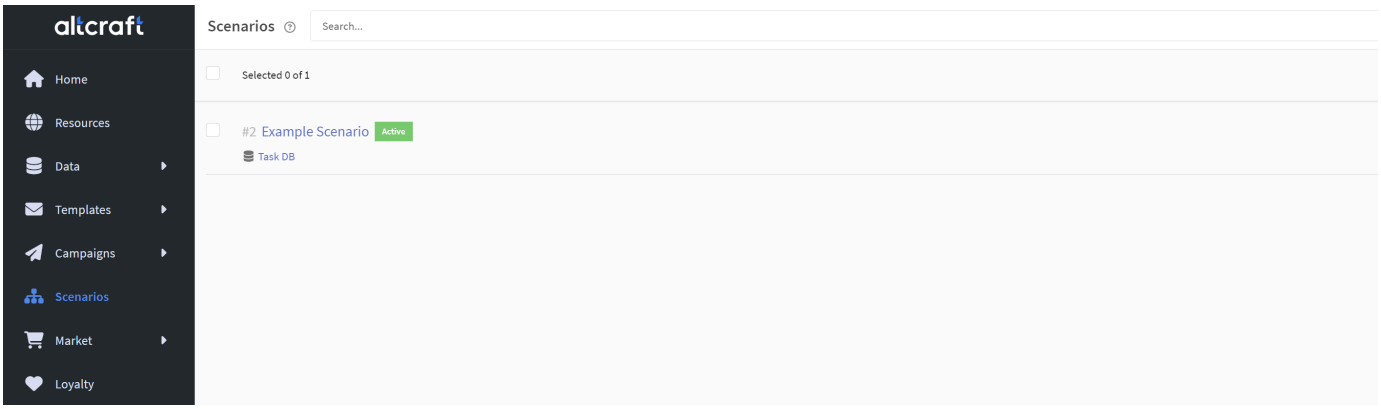
Elements (steps) of the scenario are described in a separate article.

INFO

Automation scenarios can be managed via API. Information is available at this link.

Scenario management

To start working with customer interaction scenarios, select Scenarios from the main menu. The list tha existing scenarios, their activity statuses, and the number of profiles that are currently in the scenarios.



You can use the context menu to:

— activate the scenario.

— stop the scenario. If there are still profiles in the scenario at the moment of stopping, they can be sav they will “wake up” and continue to go through the scenario from the element where they stopped.

— send profiles from the dynamic segment to the scenario (the segment must use the same profile datal

— clone the scenario.

— move the scenario to the trash (available only for inactive scenarios).

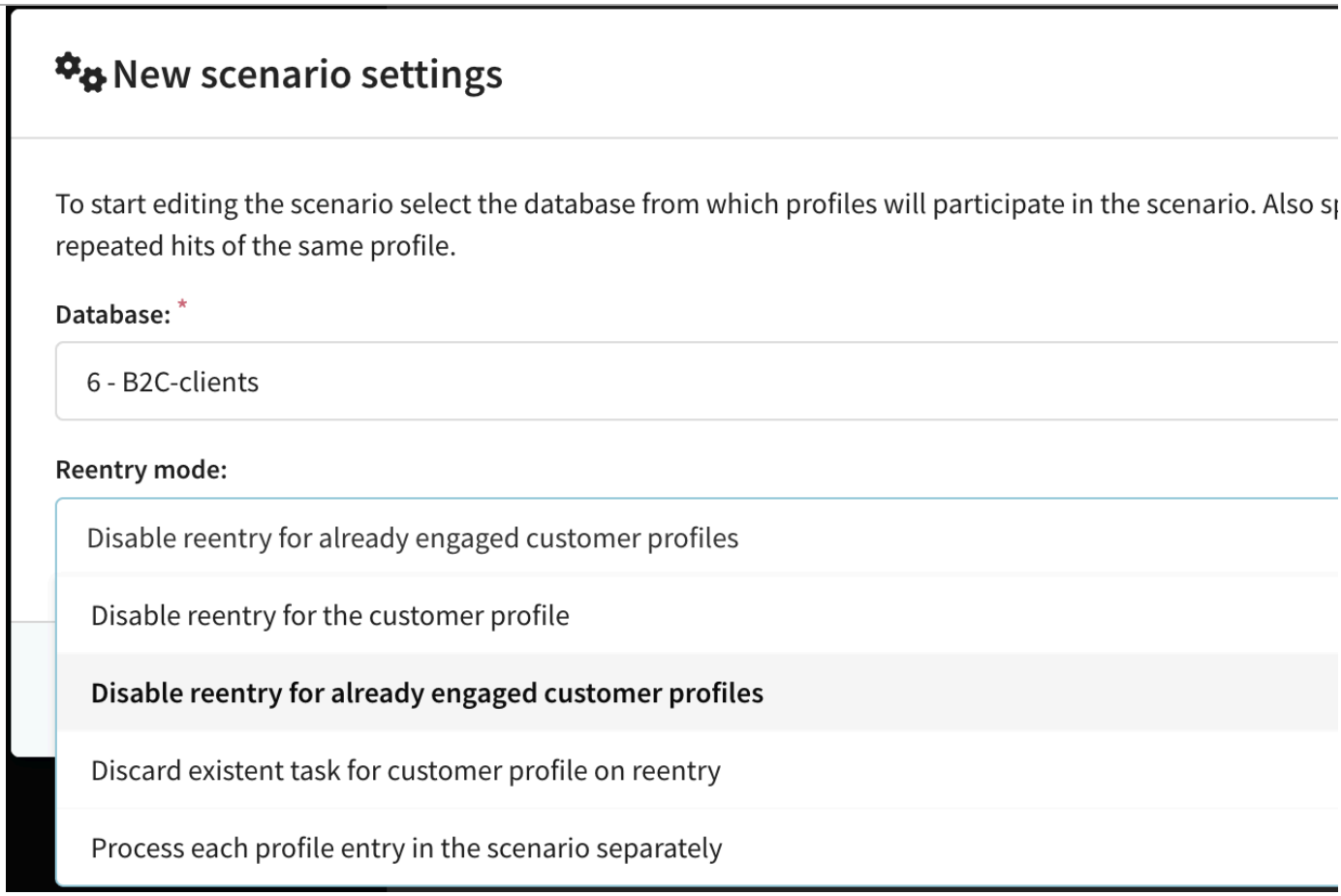
— go to the scenario report.

Getting started with the scenario

To add a new scenario, click + Create at the top of the page.

Pre-setting

In the scenario pre-configuration window that opens, specify the database that will be captured in the sc be changed). Then press the Continue button. You can change these settings later.



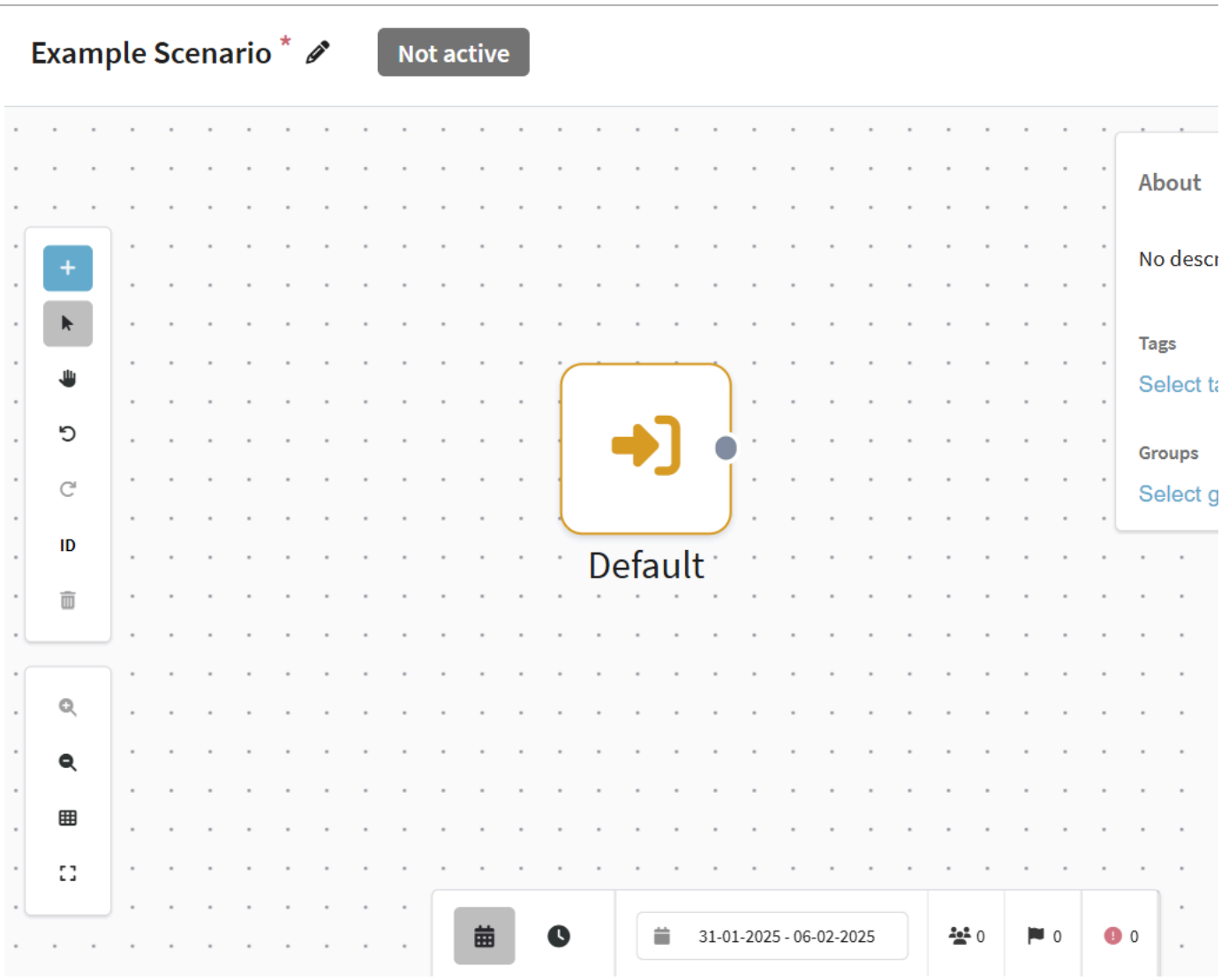
Reentry mode defines the behavior of the scenario when repeatedly capturing a client profile. Select one

- **Disable reentry for the customer profile — in this case, the profile can only enter the scenario once, enter it again. Never.**
- **Disable reentry for already engaged customer profiles — If the profile is currently already in the scen soon as it exits, it can.**
- **Discard existent task for customer profile on reentry — if a client profile is recaptured by the scenar will be immediately terminated and replaced by a new one. A task is an individual process of going t profile. All data associated with the current job (`apicontent`, etc.) will be deleted. The profile will star beginning, under the new task in the scenario.**
- **Process each profile entry in the scenario separately — no matter how many times the profile is pres captured again when the capture condition is met. Both captures will be processed completely.**

General settings

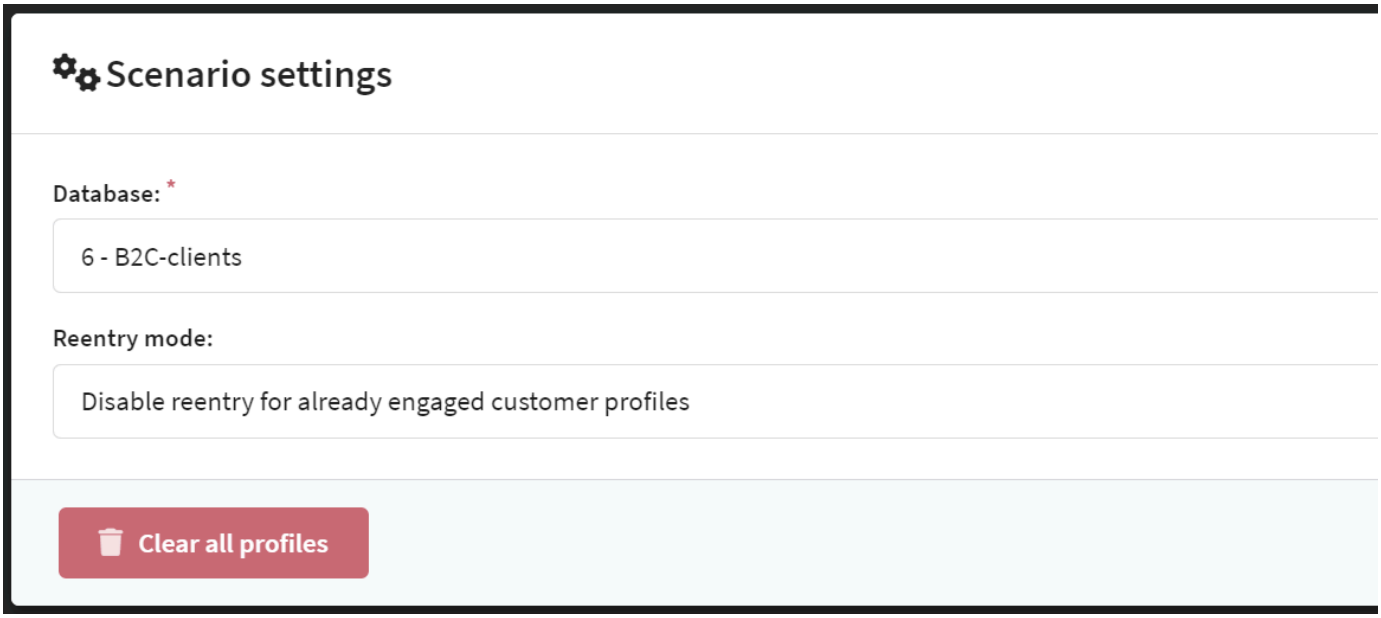
You will be taken to the editor. In the top panel, enter the name of the scenario. In the About section you attributes for quick search and assign groups for access.
To display or hide the About section, use the

button on the toolbar:



If you want to change the base of profiles in the scenario or choose a different re-entry mode, then on th

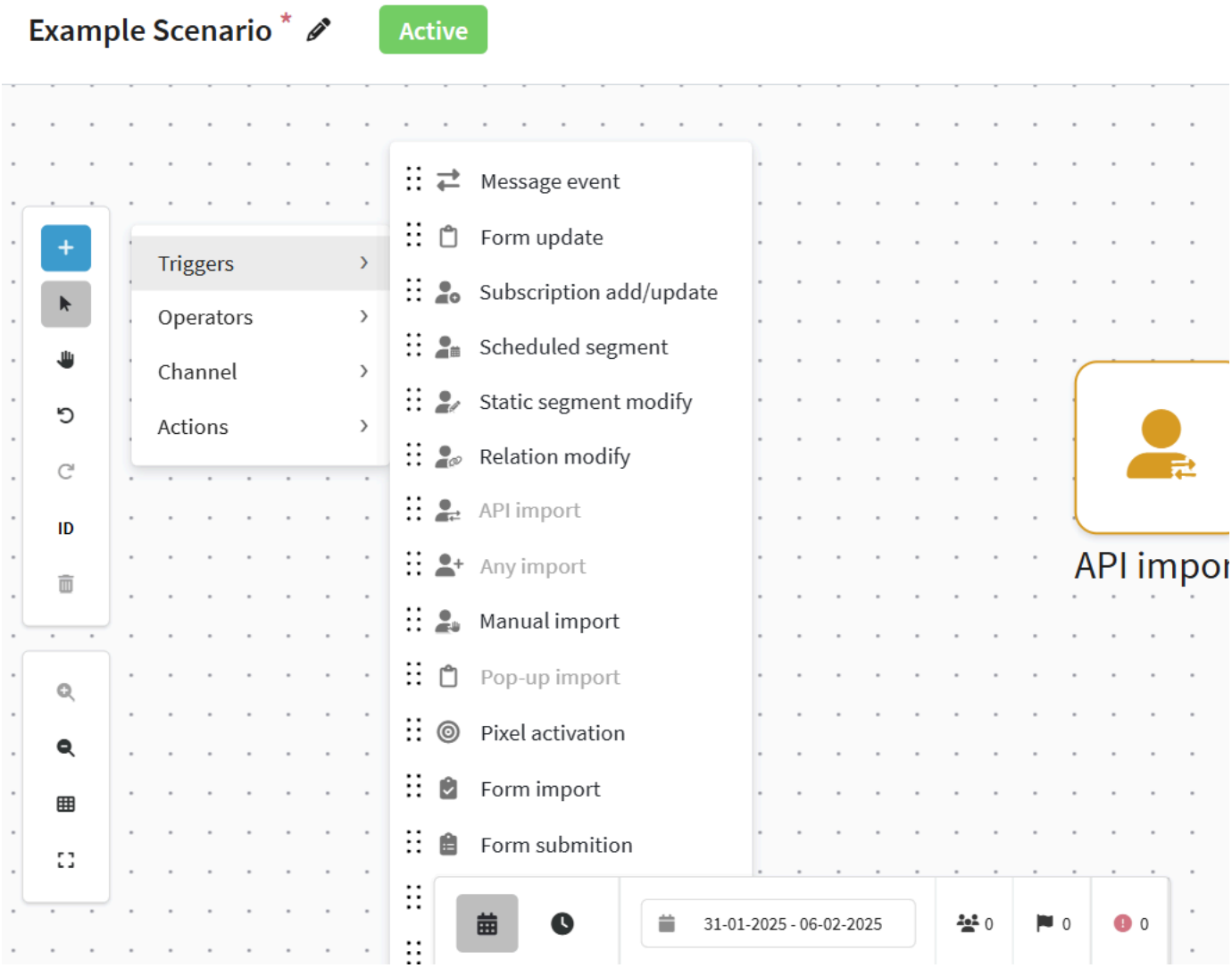
and edit the settings. The Clear all profiles option is also available in the same window - it allows you to there from the scenario.



If any required attribute is enabled for scenarios, you must select its value right away.

Configuring the capture of profiles to the scenario

The conditions for a profile to get into the scenario are determined by a trigger. A Trigger is a start elem
configure up to 10 triggers in one scenario.
To add a trigger, click + Element and select the desired item from the Triggers category:



TIP

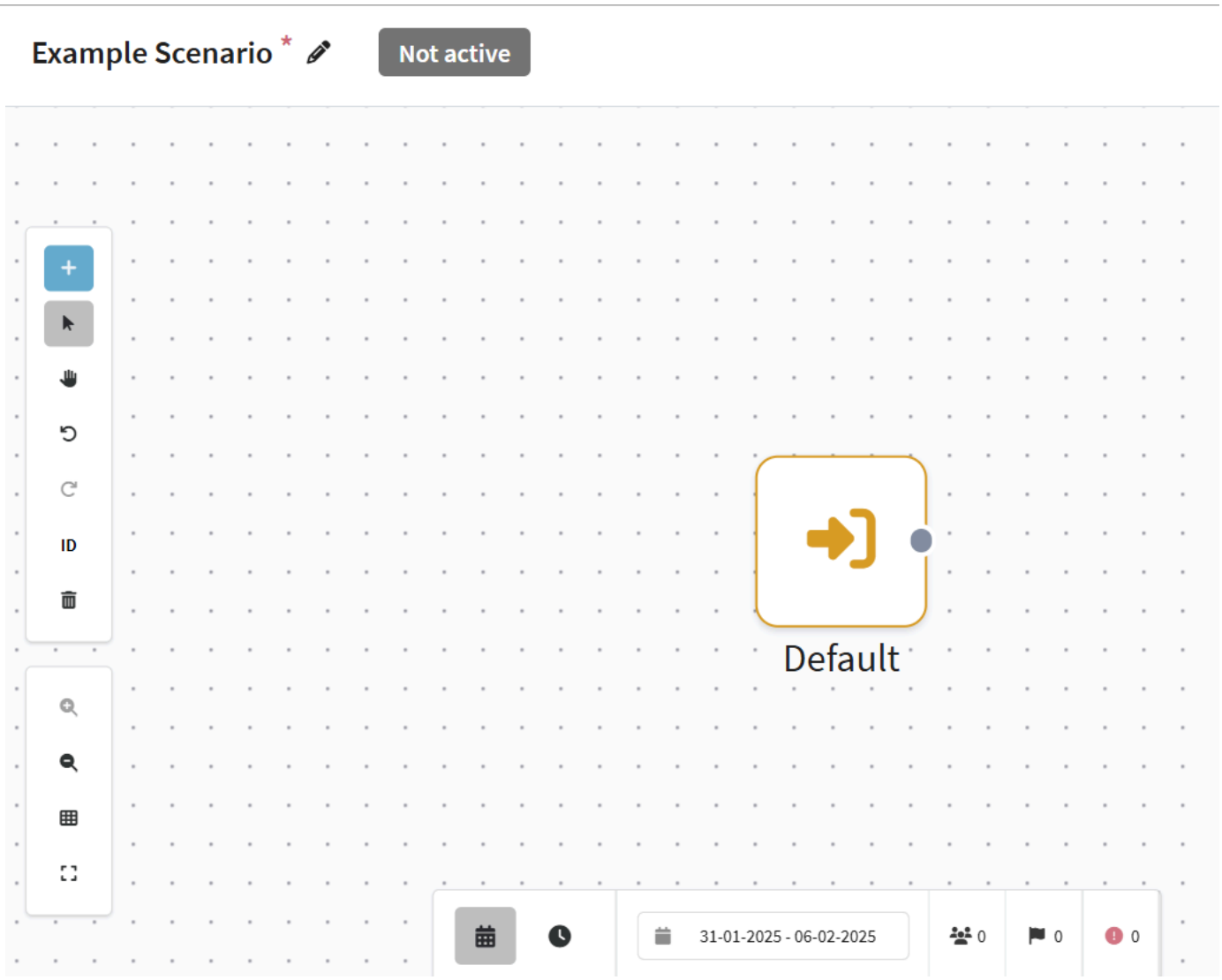
Detailed information on each trigger can be found in this article.

CAUTION

In order to save your scenario, you need to associate all start elements with any element.

Default profile capture

Any scenario has an element Default profile capture. This element is a built-in feature of the scenario an
to read the description.



With the "Default" element, profiles can enter the scenario from different points in the platform:

1. When sending a profile to the scenario via API. When the scenario is saved, the item window will displ
However, it is recommended to use a newer version of the API. See documentation for more information.

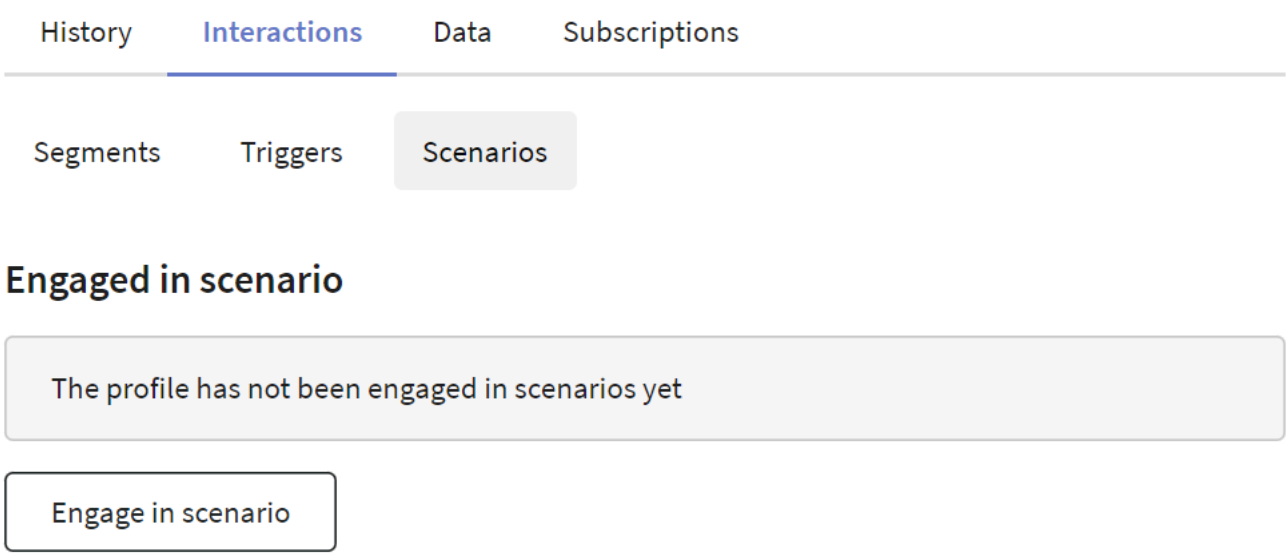
Default profile capture

This trigger enables automatic routing of profiles into a scenario from various entry points within the platform. This ir
the card, quickly sending a segment to the scenario, jumping from another scenario via a node, and re-entering the sc
start node is activated.

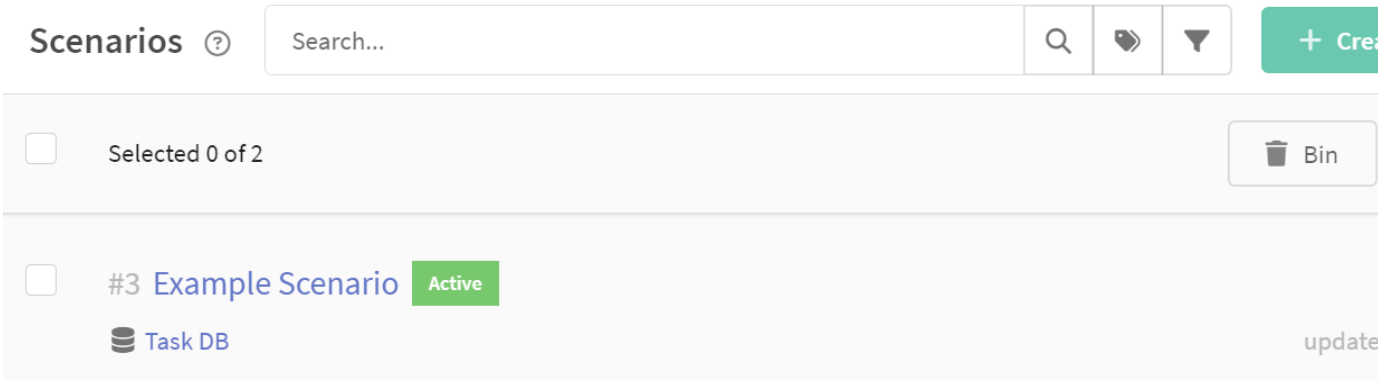
Profiles sent to the scenario through the API will also enter this node. For more details on triggering scenarios via the .

▸ Deprecated version API ⓘ

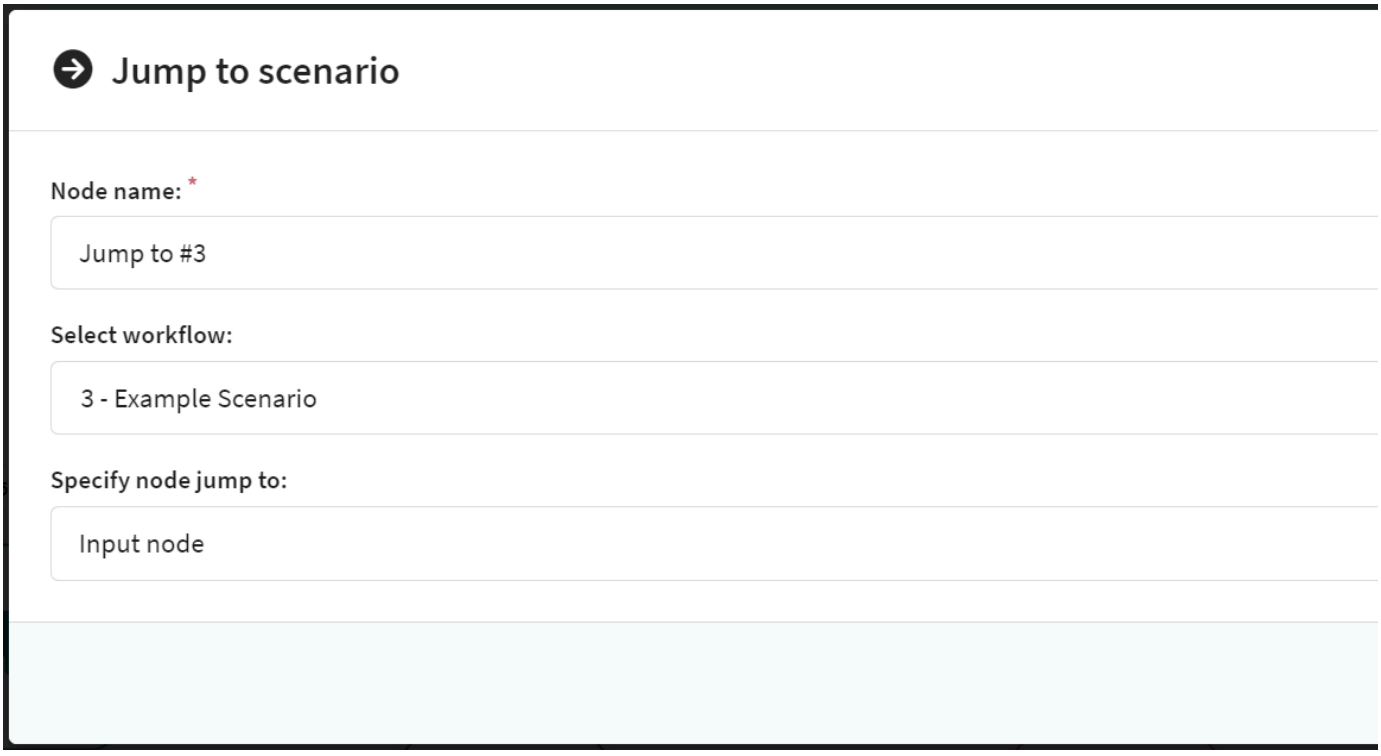
2. When adding a profile to a scenario manually:



3. When quickly sending a segment to a scenario:



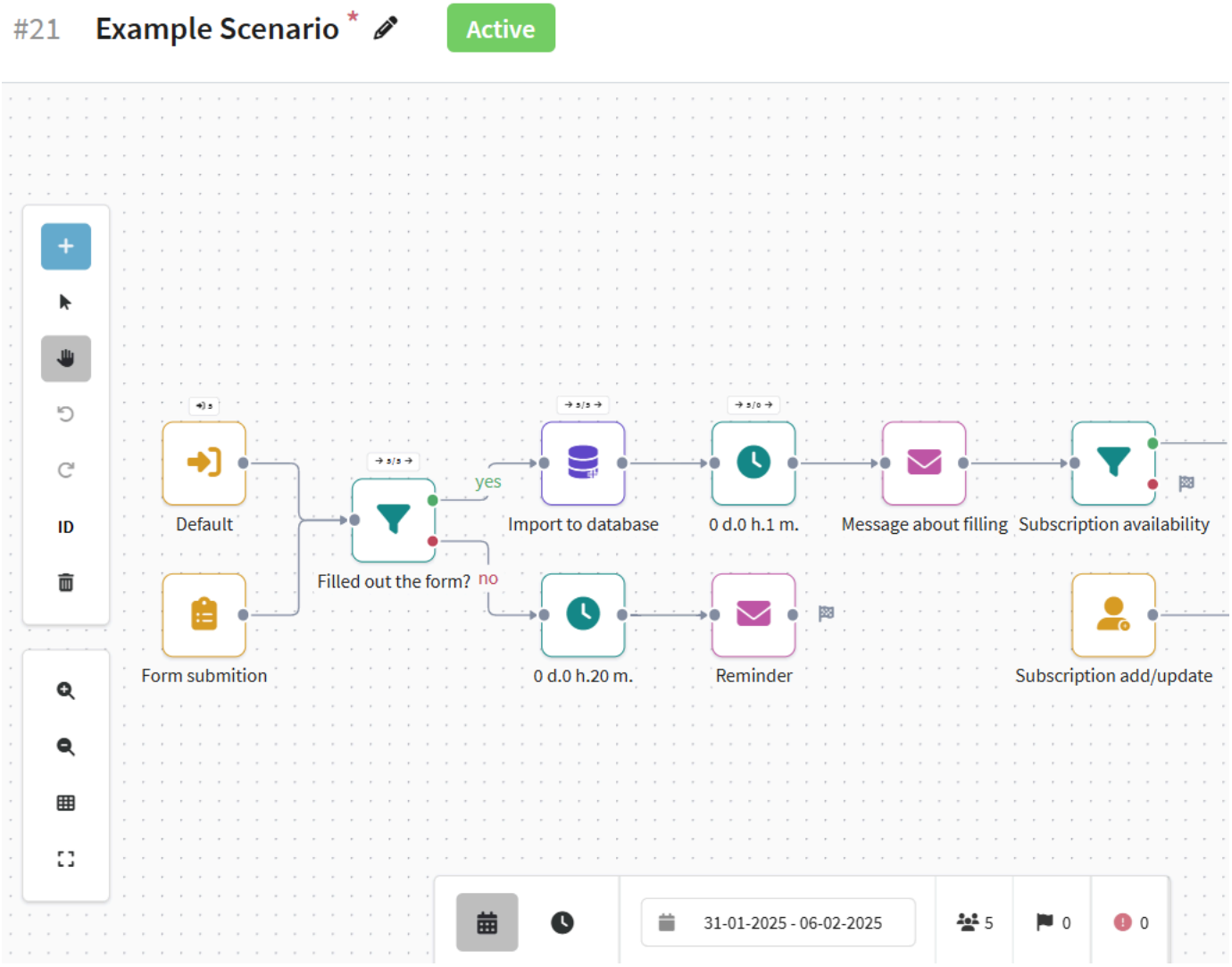
4. Via the "jump to scenario" node from another scenario:



Activating and deactivating a scenario

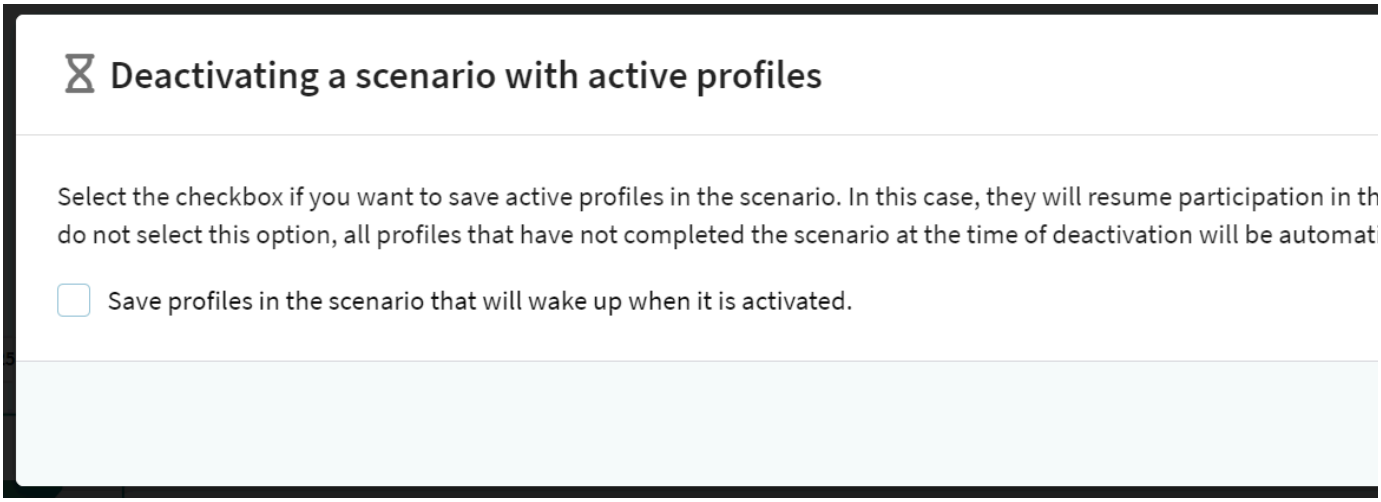
To activate the scenario, click

and then save it. Once activated, the scenario will start capturing profiles according to the specified cor



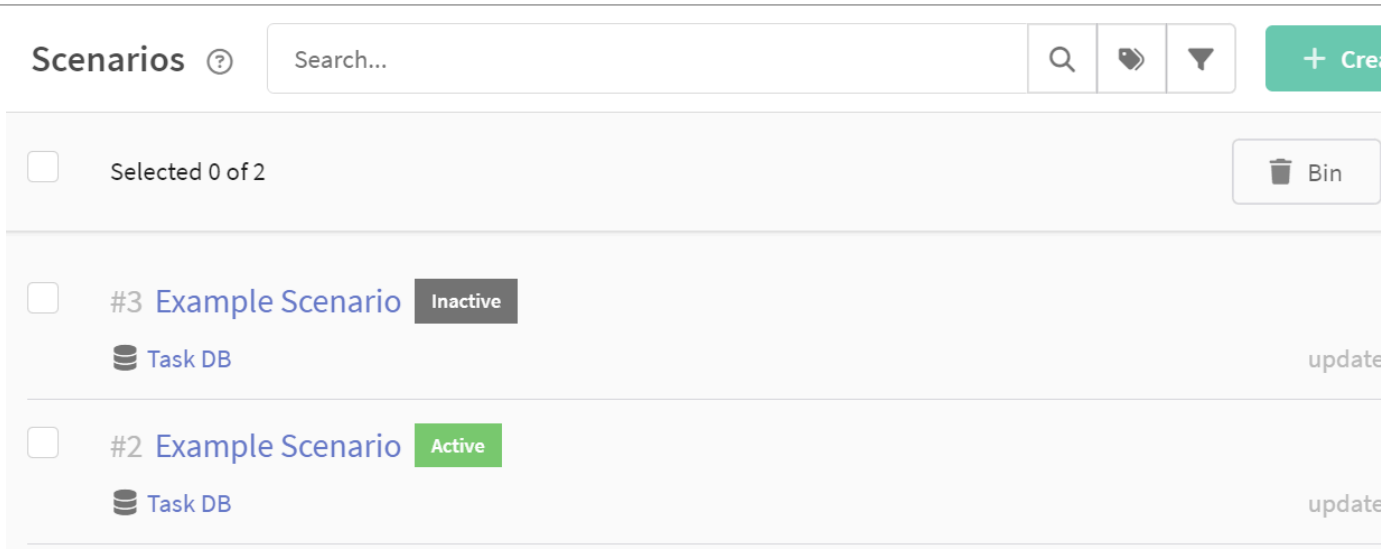
To stop the scenario, click

and save it. If there are profiles in the scenario at the moment of deactivation, you will see a warning wi



By default, after deactivating a scenario, all profiles will exit it. To avoid this, enable the setting Keep pr up when the scenario is activated. In this case all profiles in the scenario will be frozen, and after reacti continue to follow it.

You can also activate and deactivate a scenario from the general list:

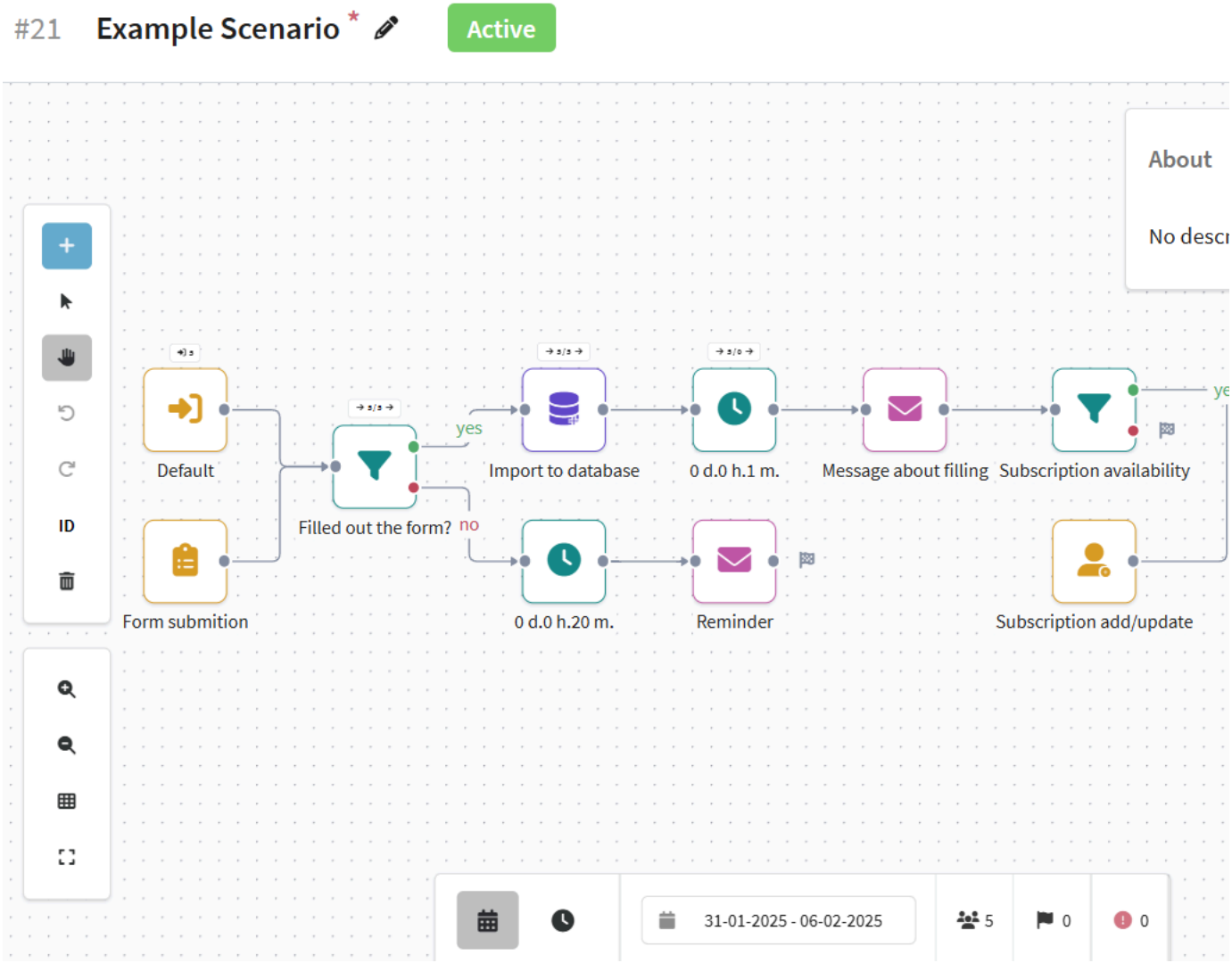


Scenario analytics

You can analyze the effectiveness of your scenarios directly in the scenario editor, and you can see deta the chain in the [channel report](#).

Client Scenario Completion Statistics

The statistics panel is located at the bottom of the editor. This panel contains information about comple



Viewing statistics is available in two modes:

— period statistics

— real time statistics

period statistics

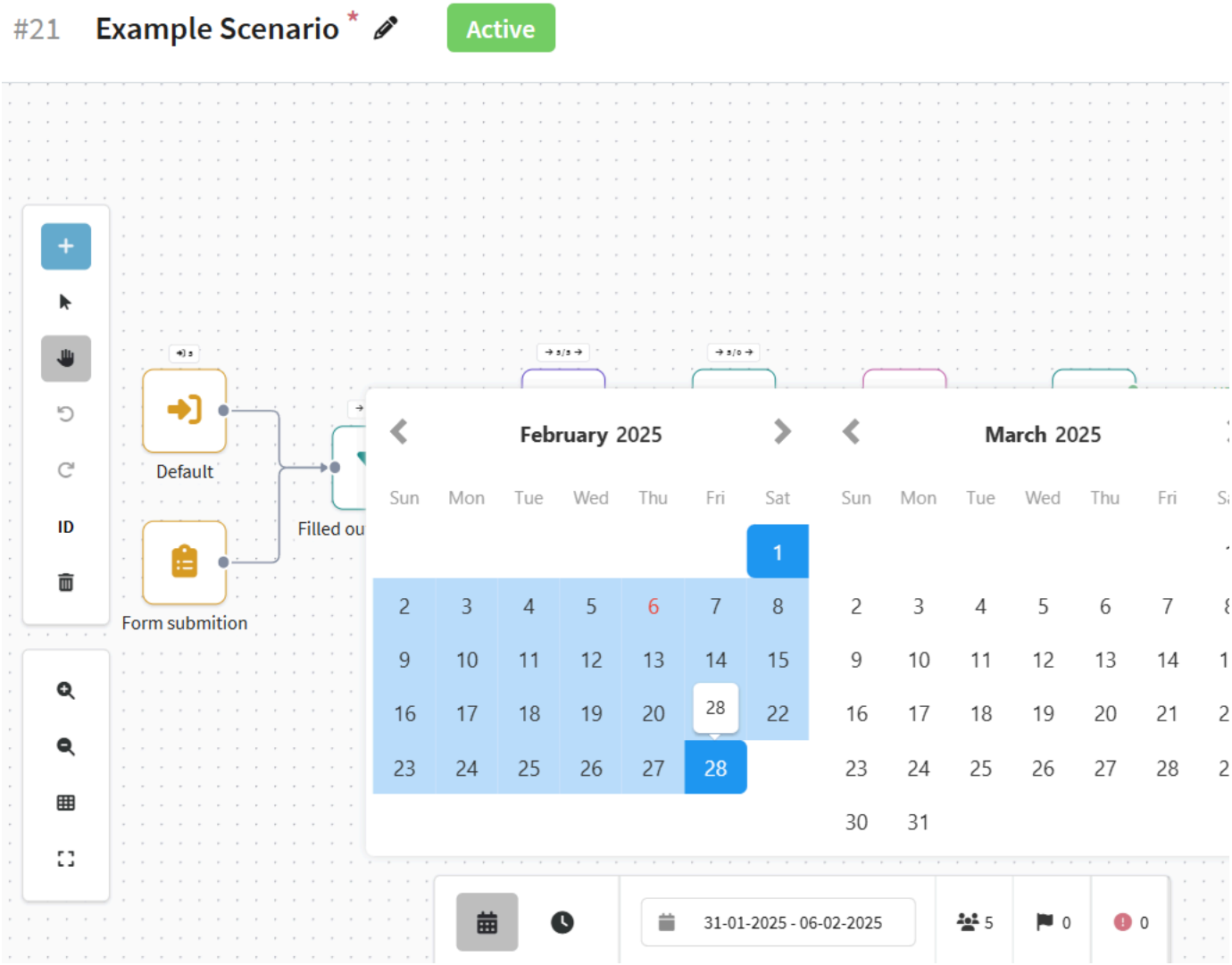
Here you can view scenario data for a specific time period:

— total number of profiles included in the scenario.

— number of profiles that have completed the scenario.

— errors that occurred during the scenario process.

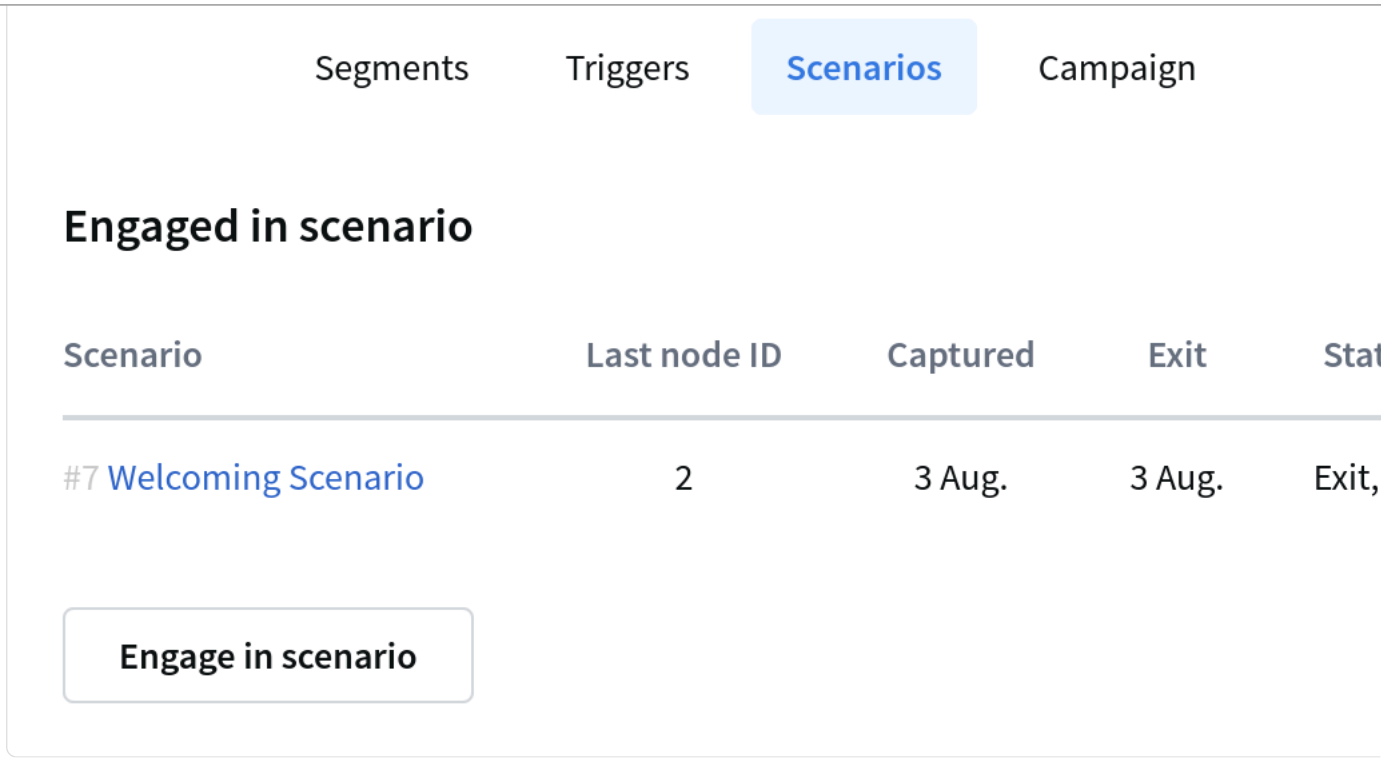
By default, the panel displays data for the last 7 days. To change the period, enter the desired dates man calendar:



Also above each scenario element it is shown how many profiles entered the element and how many exit

TIP

To find out on which element the profile left the scenario, open the profile card, go to the Interactions – "Exit" stat. The item ID will be displayed:

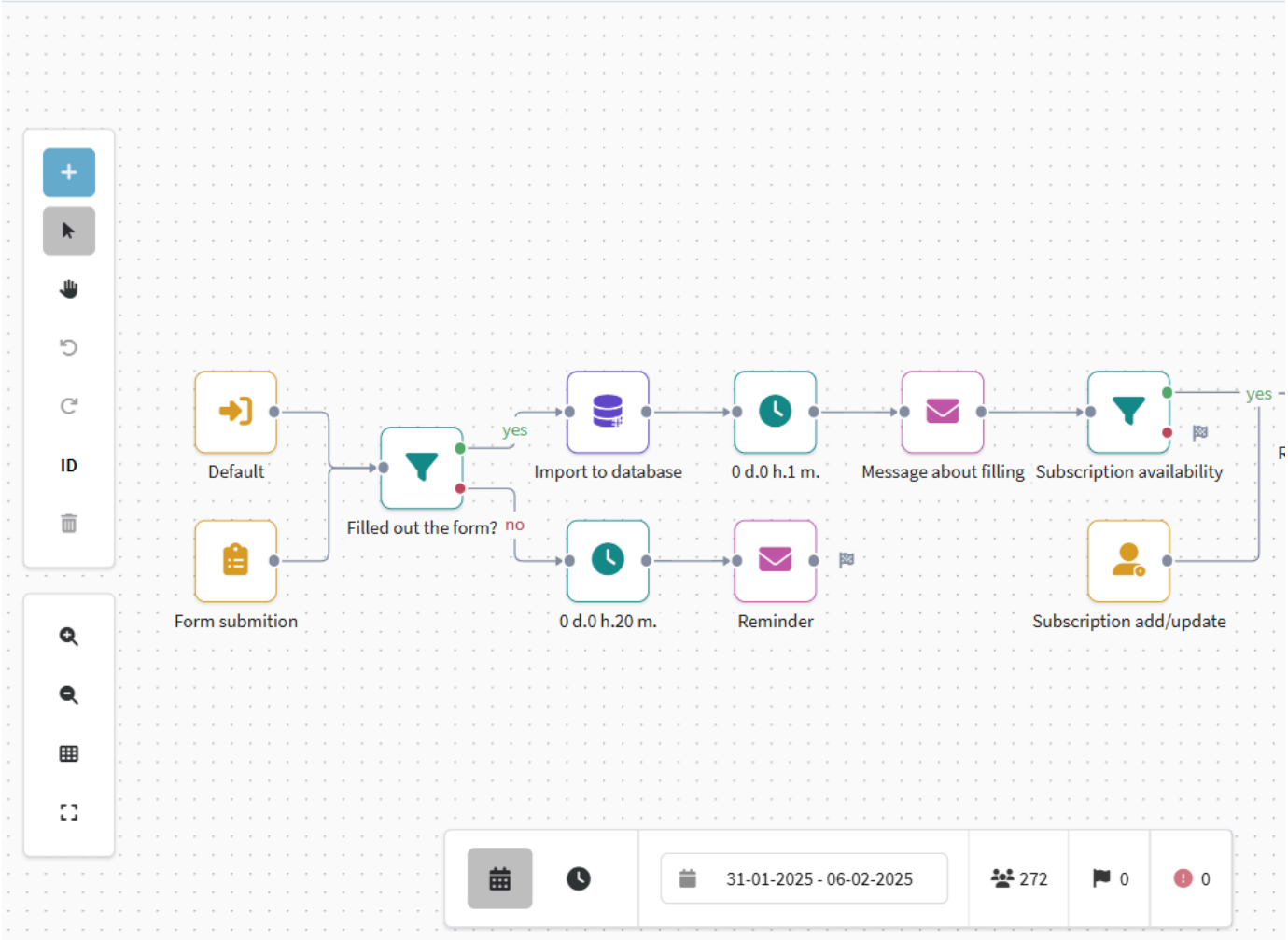


Real-time statistics

Here you can get information about the profiles that are currently in the scenario.

— the number of profiles that have entered the scenario but have not yet completed it.

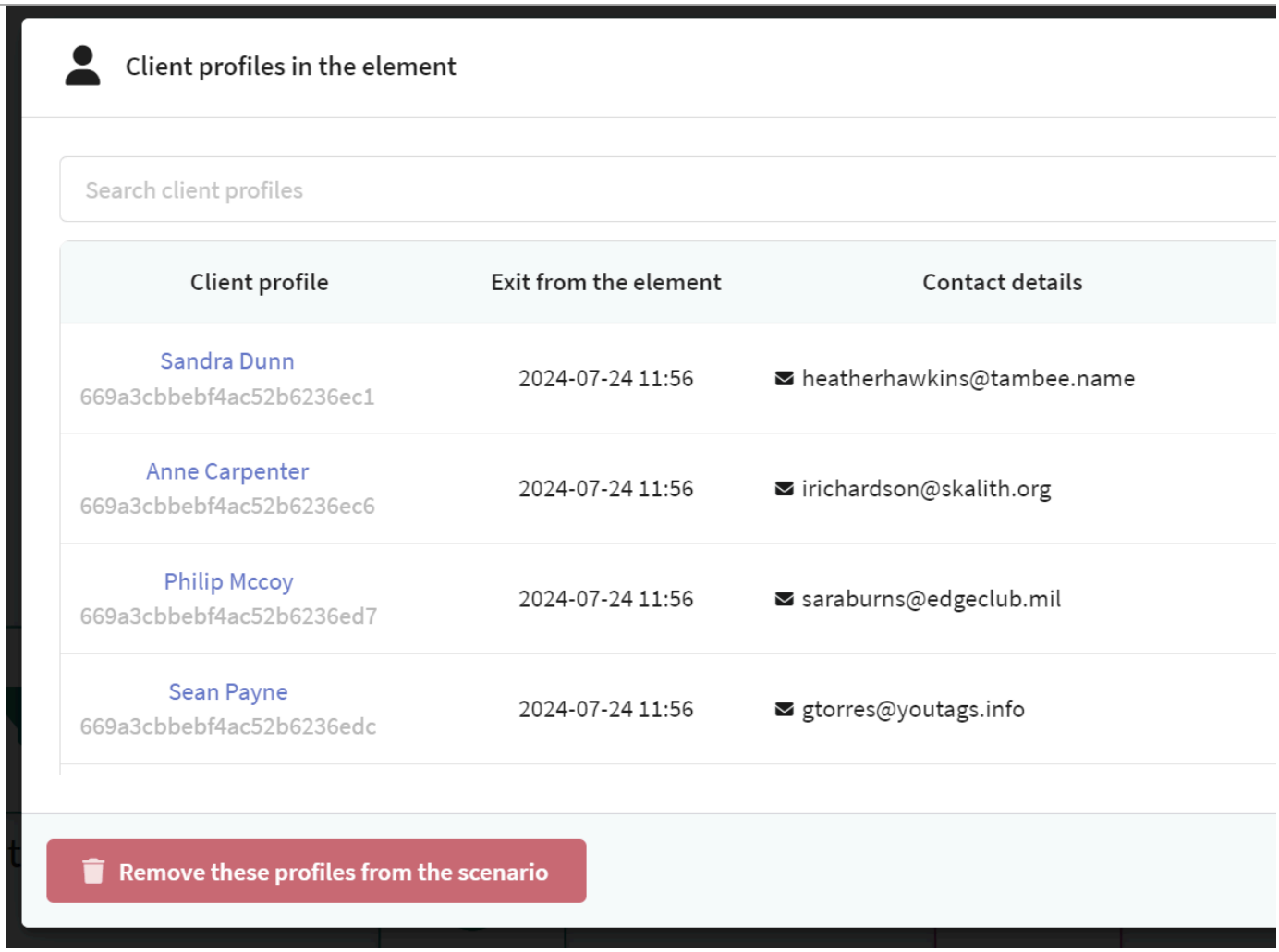
#22 Example Scenario * Not active



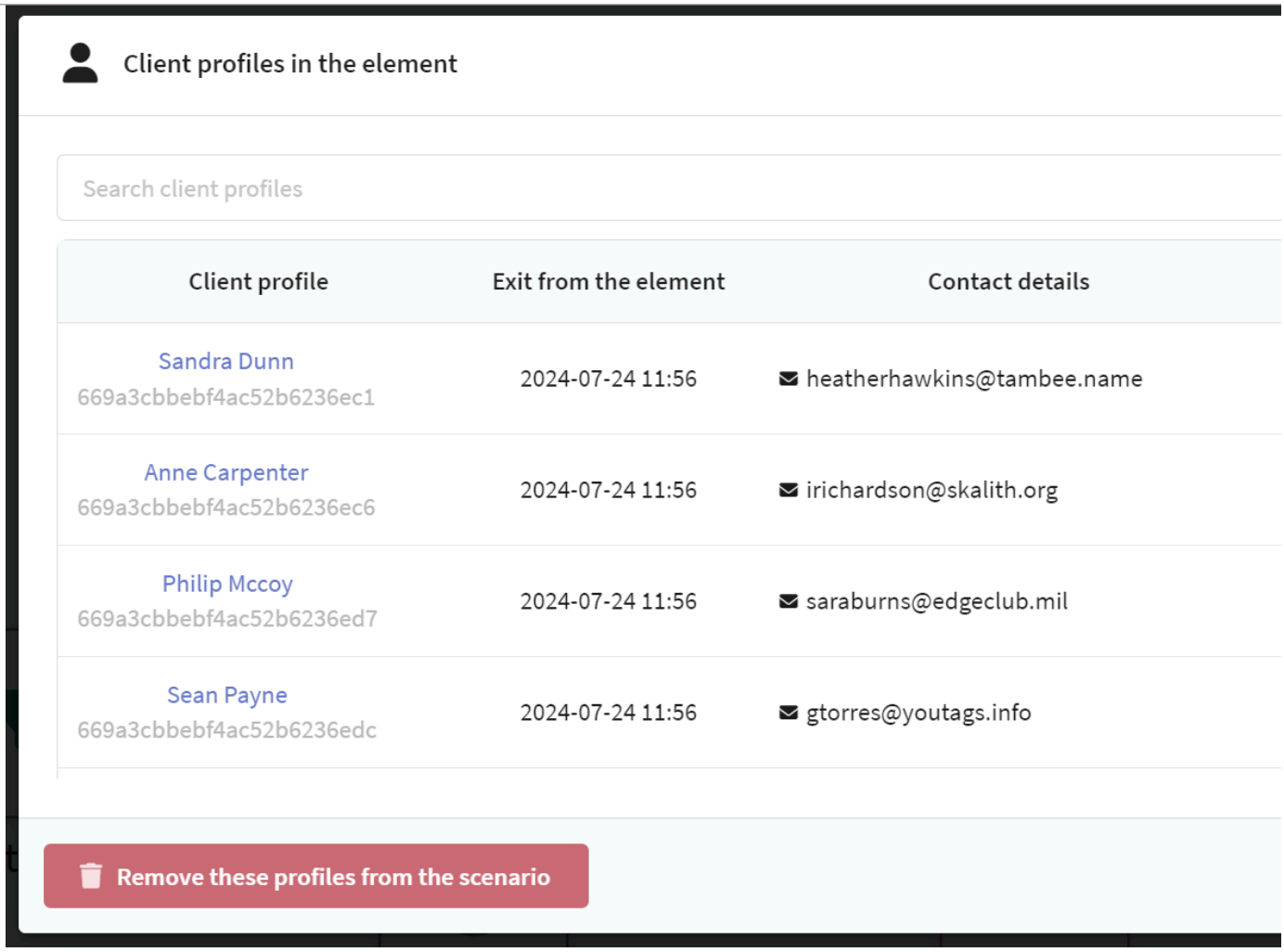
If there are profiles in the scenario at the moment, you will see a number above the element that contain number of profiles in the element. There can be several such elements. In most cases, they are waiting e or Delayed Condition).

To find out information about the profiles in the element, click on this number. A modal window opens wh

- **go to the client's profile in the database;**
- **find out the exact time when the profile will leave the element;**
- **download a file with API content, if the corresponding parameter was passed in the API-request.**



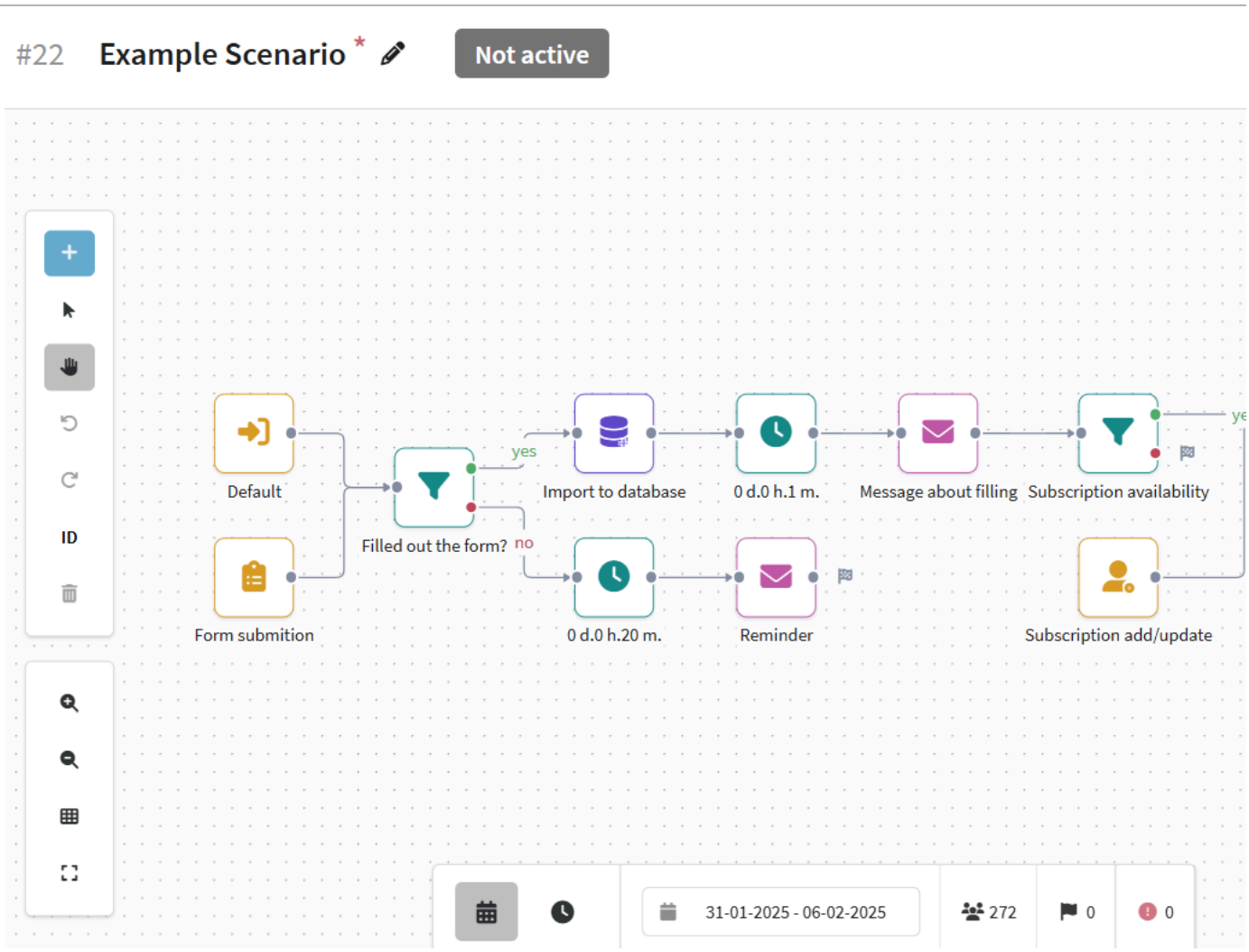
It is also available to delete profiles from the scenario. If you want to delete a specific profile from the s next to its name:



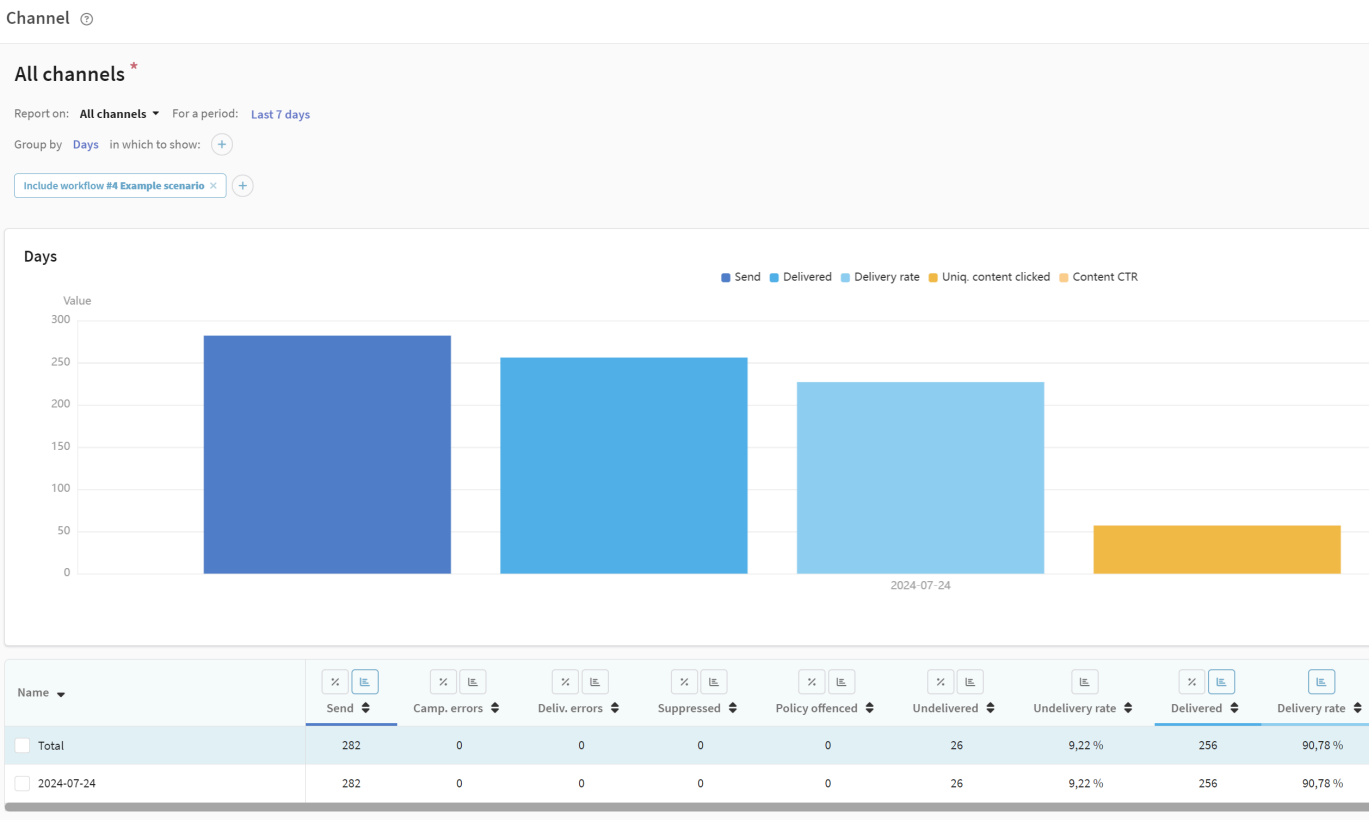
To remove from the scenario all profiles that are currently in this element, click Remove these profiles fr

Mailing analytics in the scenario

To view a report of the mailings that the scenario runs, open Analytics:



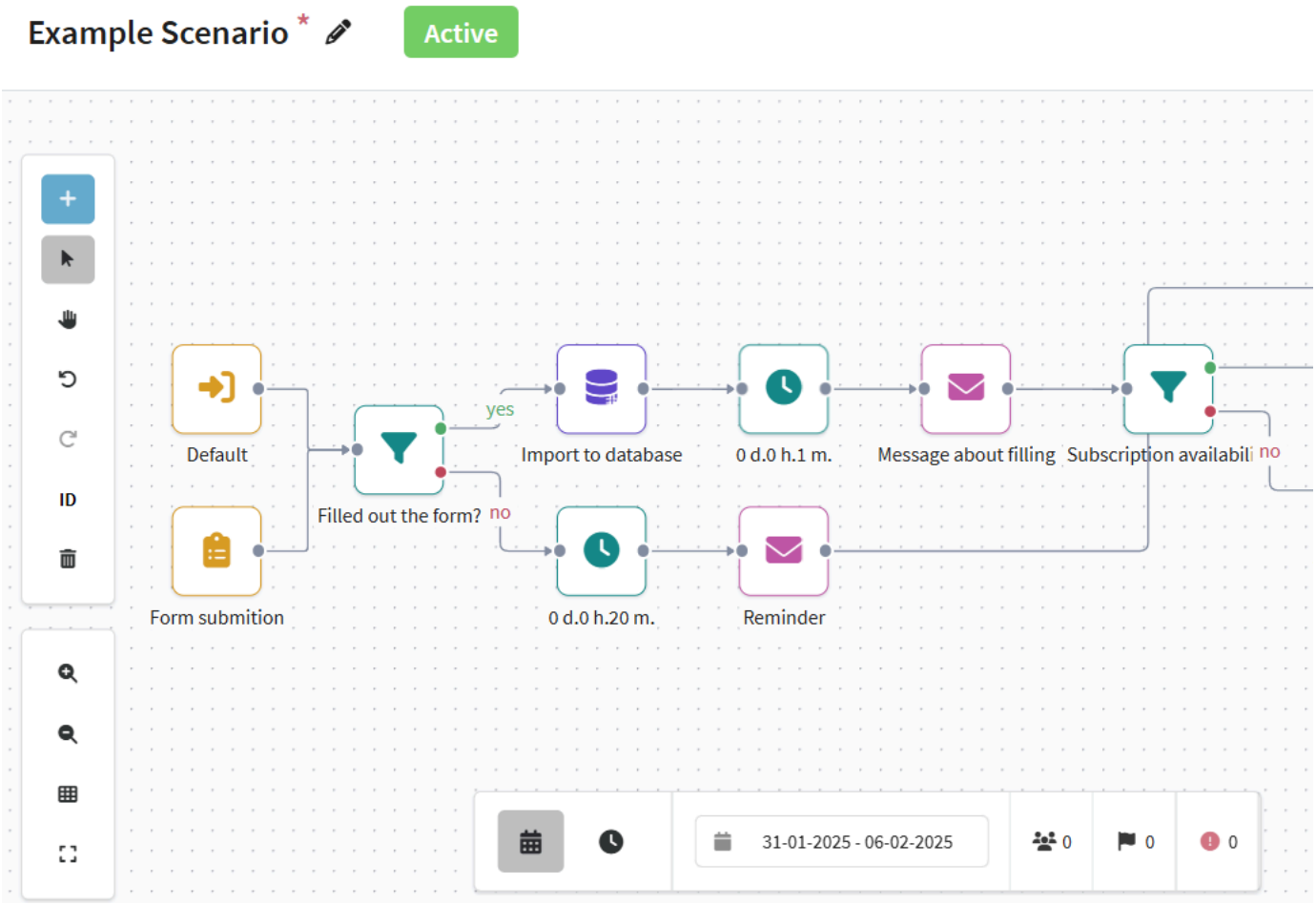
Use groupings and filters to display the data you need. For example, if you want to see the statistics of t you can use this setting:



Name	Send	Camp. errors	Deliv. errors	Suppressed	Policy offenced	Undelivered	Undelivery rate	Delivered	Delivery rate
Total	282	0	0	0	0	26	9,22 %	256	90,78 %
2024-07-24	282	0	0	0	0	26	9,22 %	256	90,78 %

Nodes of the scenario

A scenario consists of elements which are called nodes. Each node performs one action and then passes

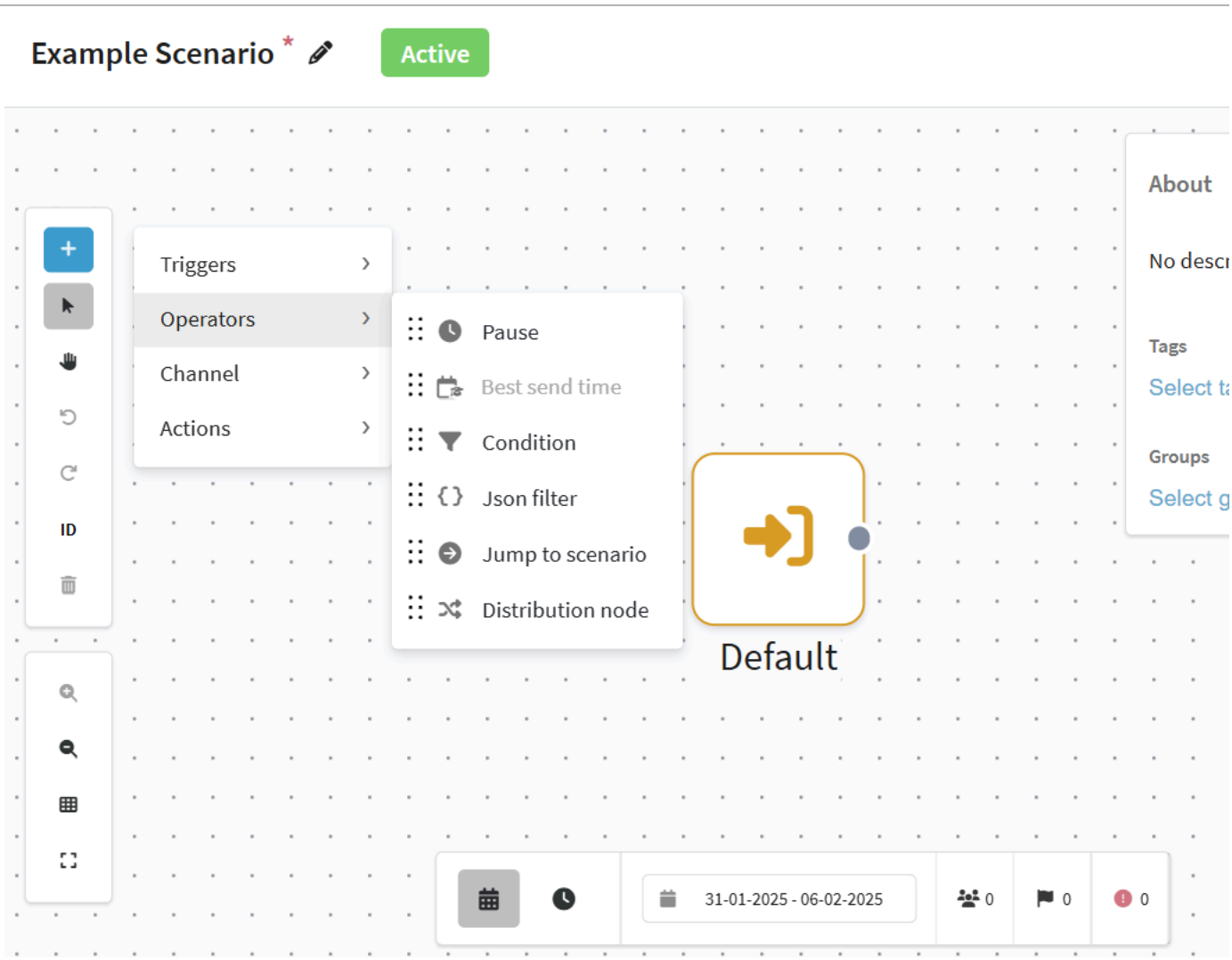


Any scenario must contain at least 2 nodes: a starting node and a connected node.

Node management

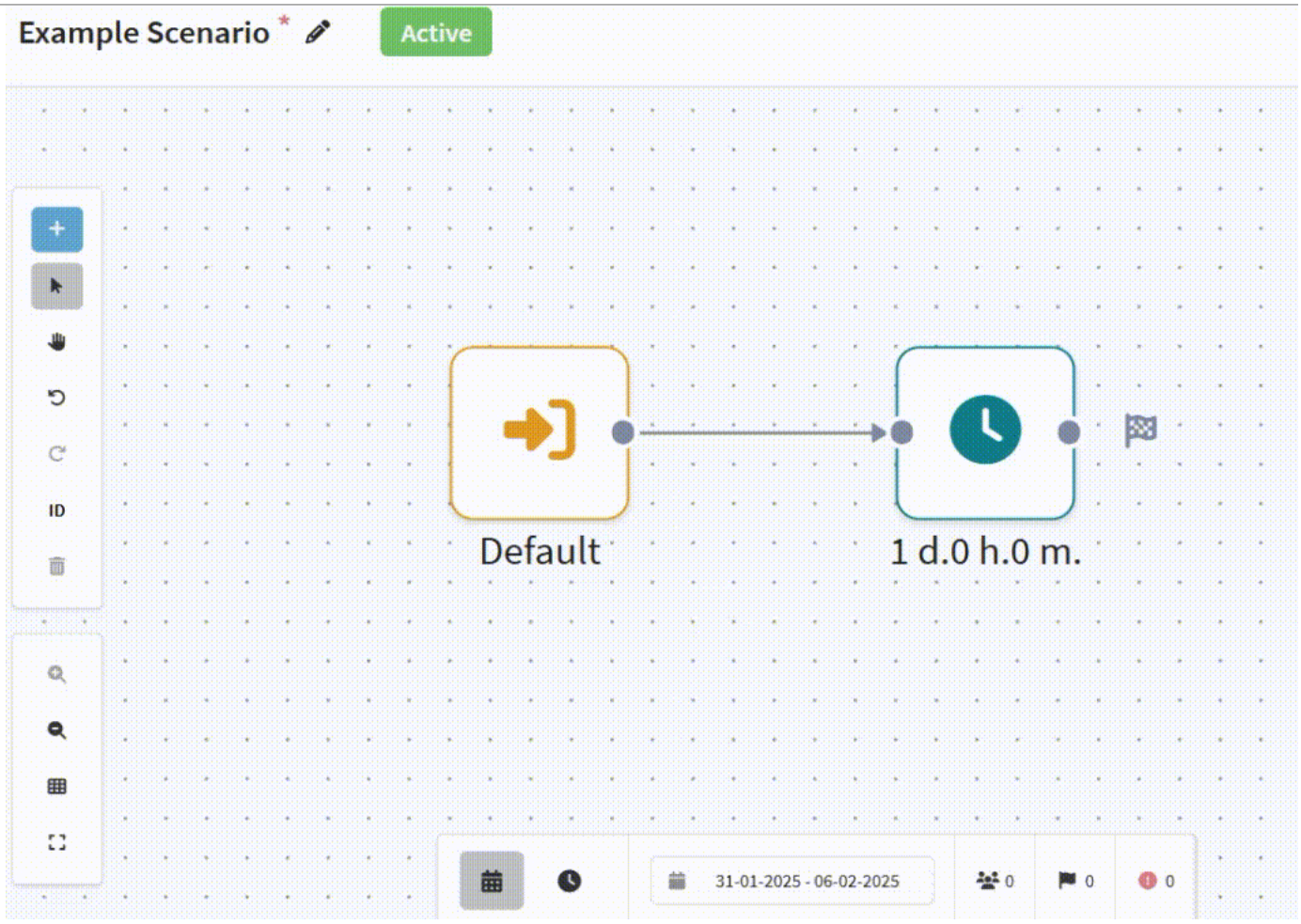
Adding a node

To add a node to the scenario, click the + Element button on the toolbar. In the opened list, click on the r workspace holding the left mouse button.



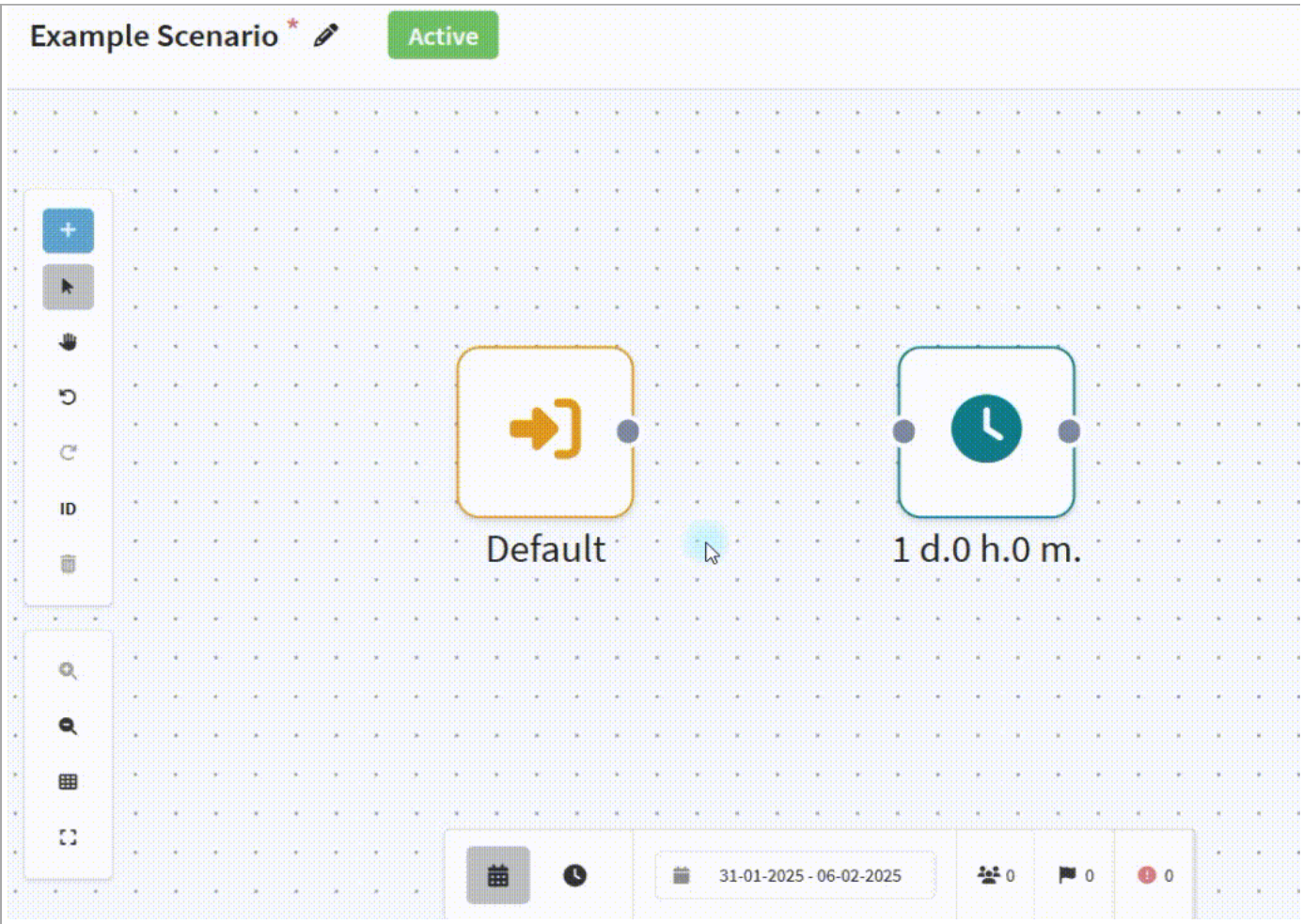
Setting up an node

Settings are set for each node of the scenario. To open the settings window, double-click on the desired “Apply”.

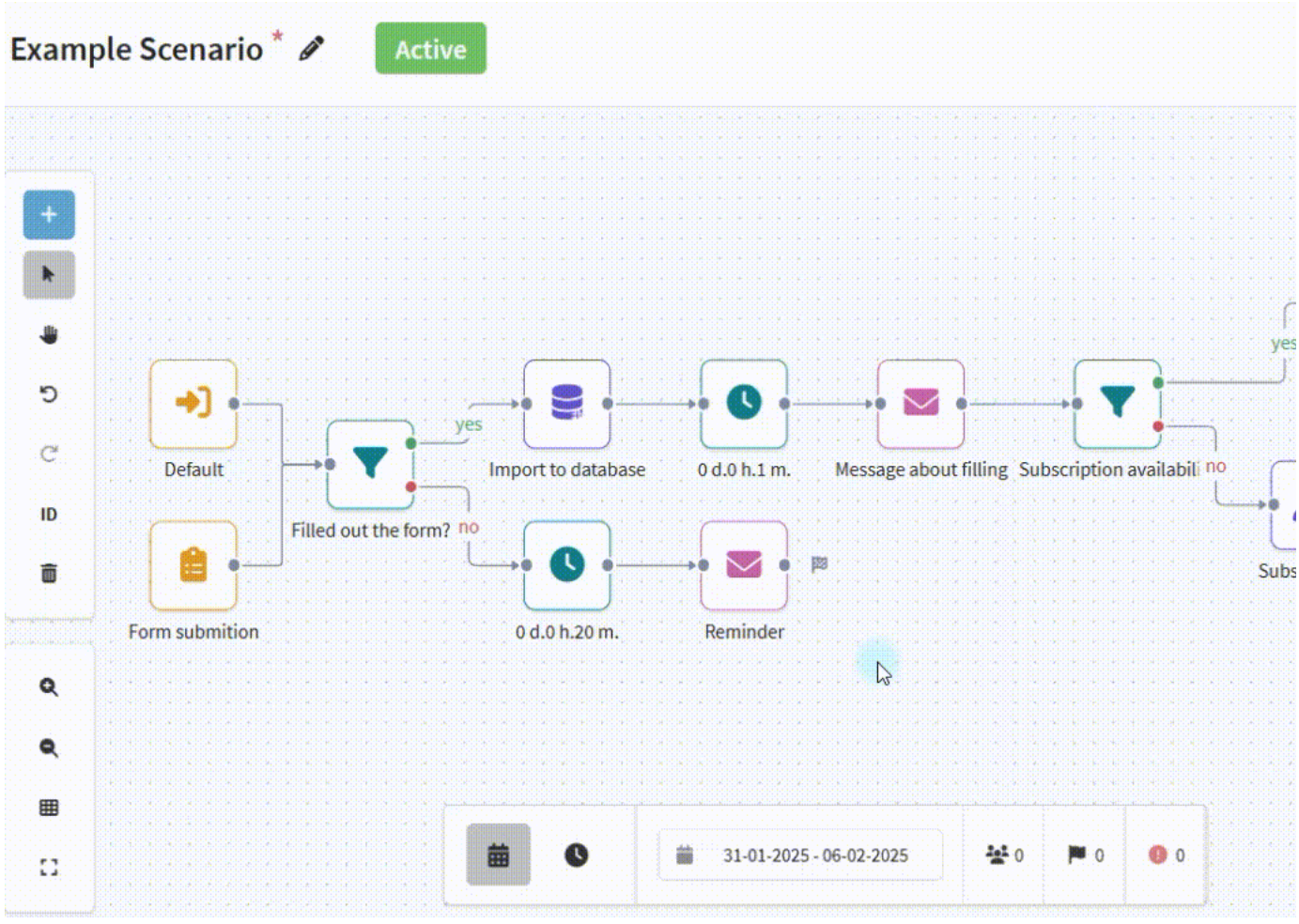


Adding a link between elements

Transition of a profile from one scenario node to another is defined using links. To link one node to anoth elements:

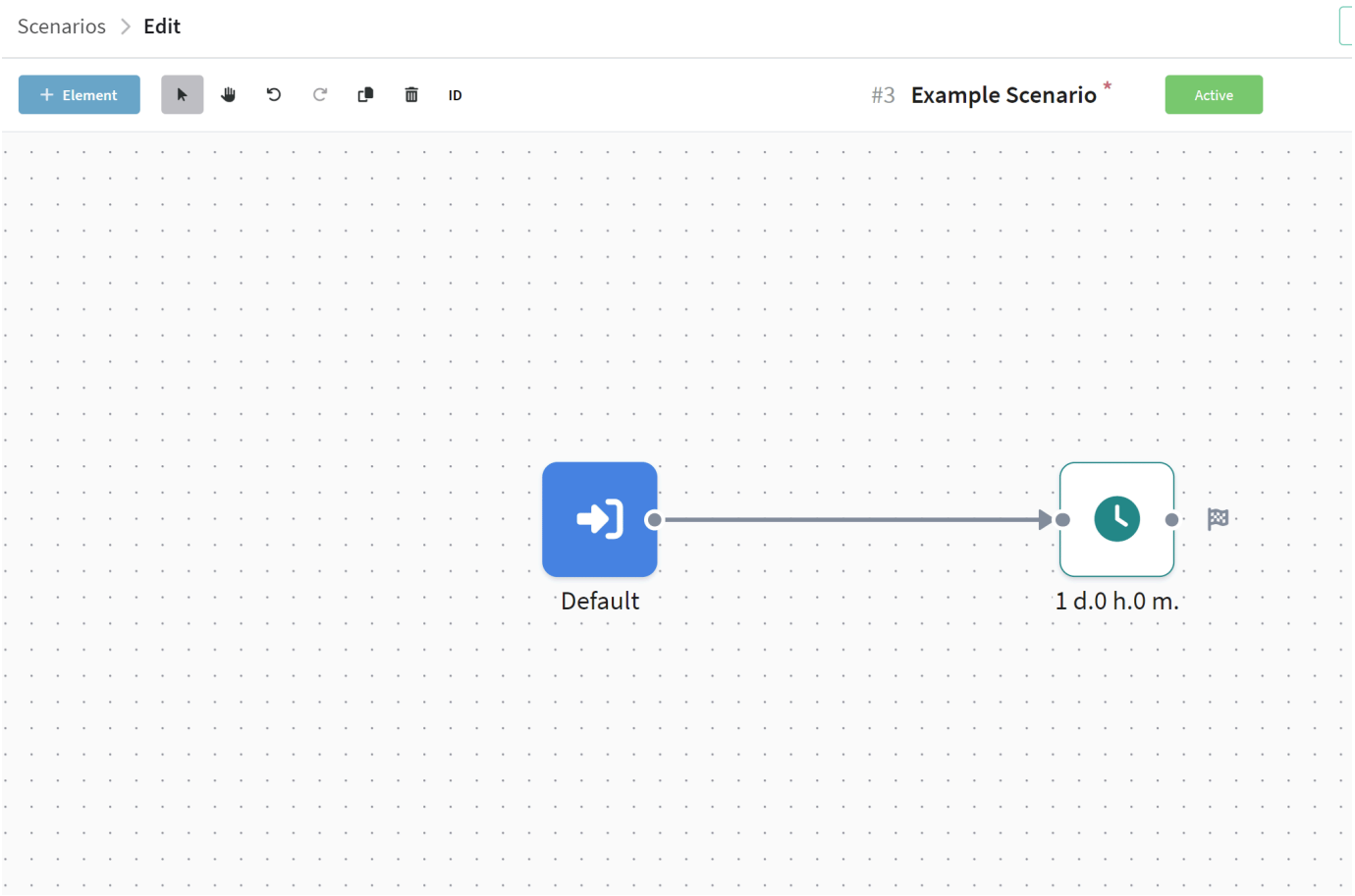


Elements can be linked not only sequentially. With the help of links it is possible to organize multiple pas section by a profile or switching to an alternative sequence of actions:

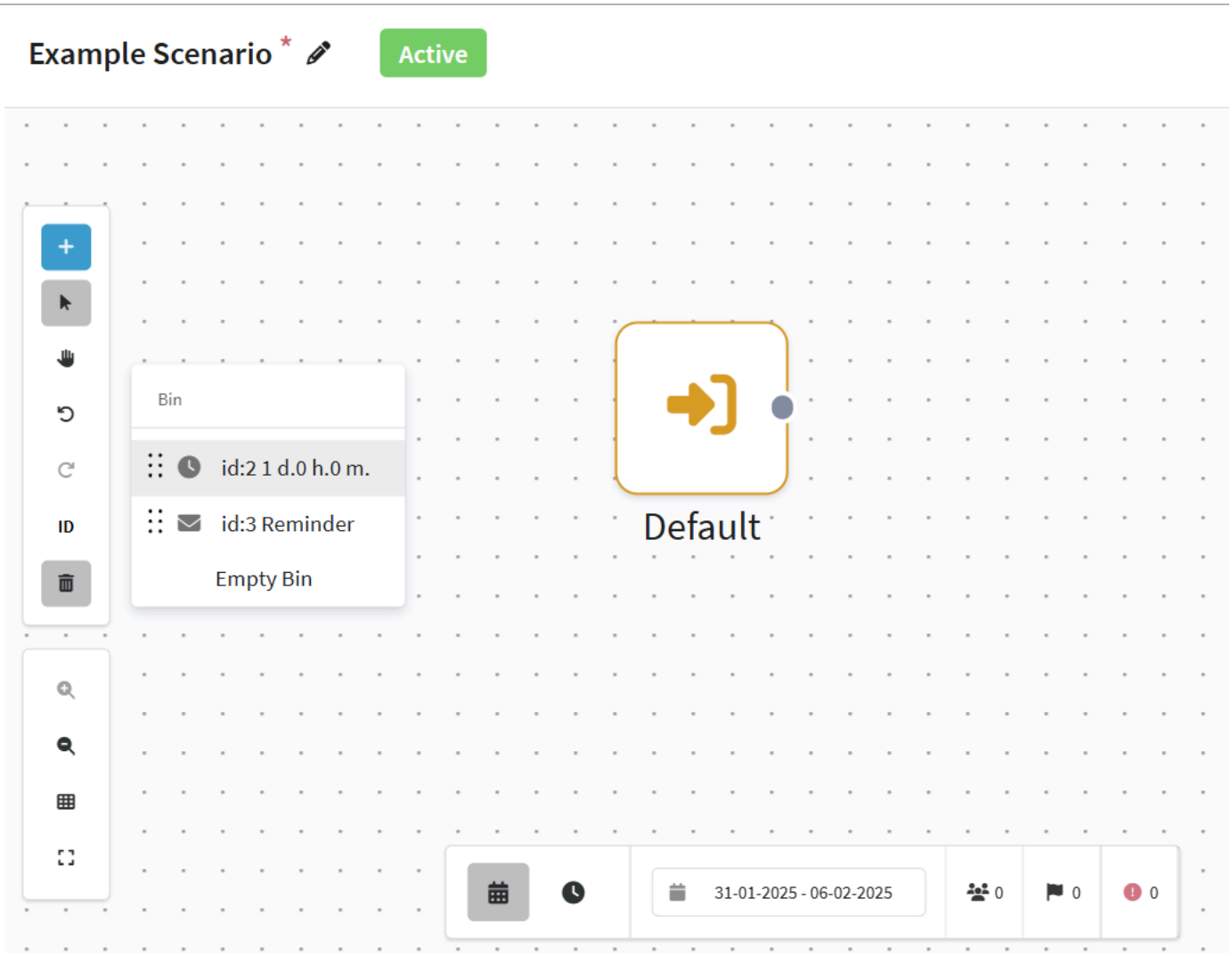


Deleting an node

To delete an node or a link between elements, select an object by clicking on it, and then click the trash possible to delete multiple elements, including links between them.



If you have deleted an node from the scenario, you can restore it. To do this, click the Bin button in the l deleted elements will appear. Select the node you want to return to the scenario. After restoring the nod



You can also irrevocably remove all elements from the bin by clicking the Empty bin button.

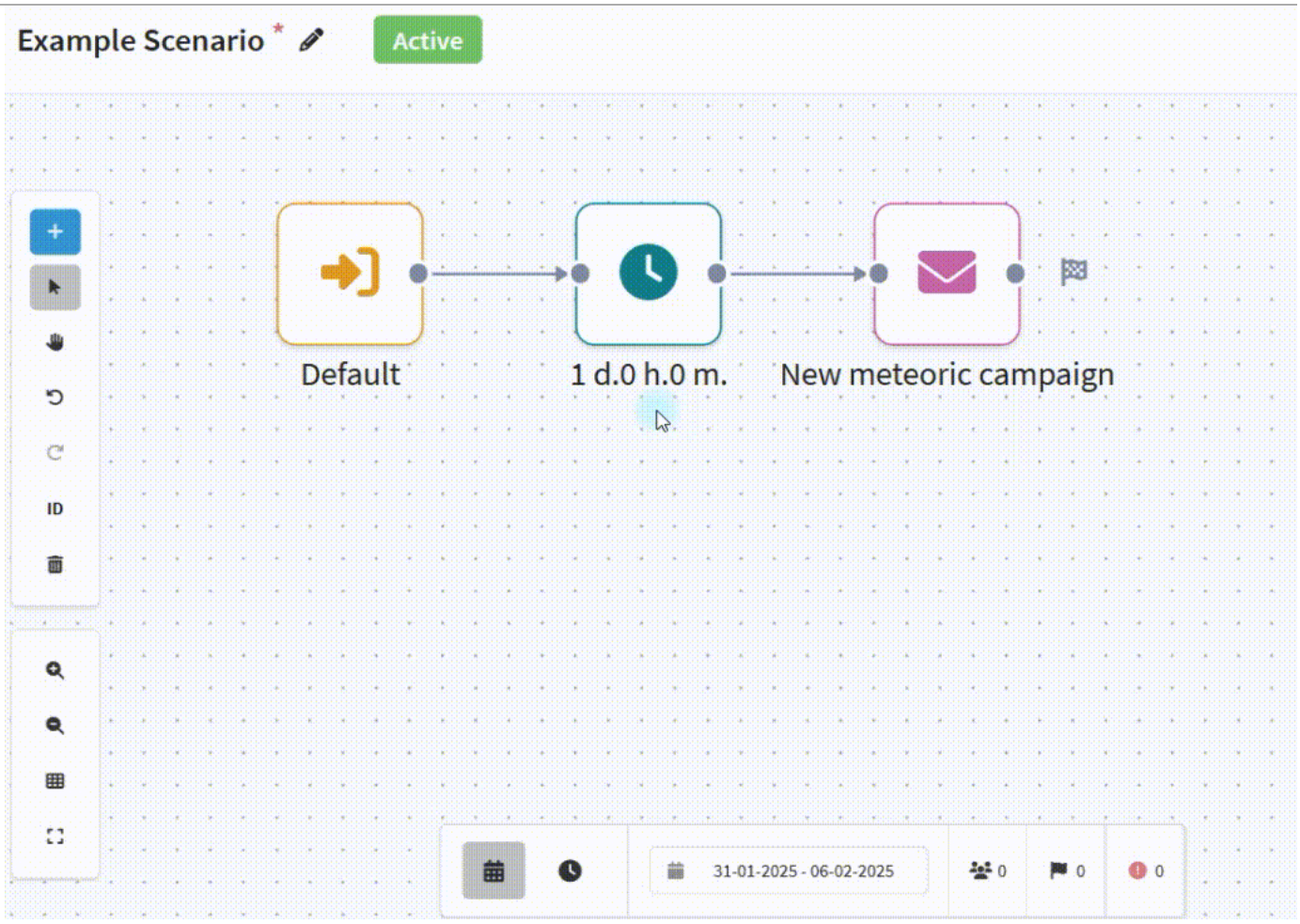
CAUTION

If you remove a node that has profiles, those profiles will be taken out of the scenario. If there are no pr following the updated scenario.

Cloning of a node

To clone a node, select it and left-click

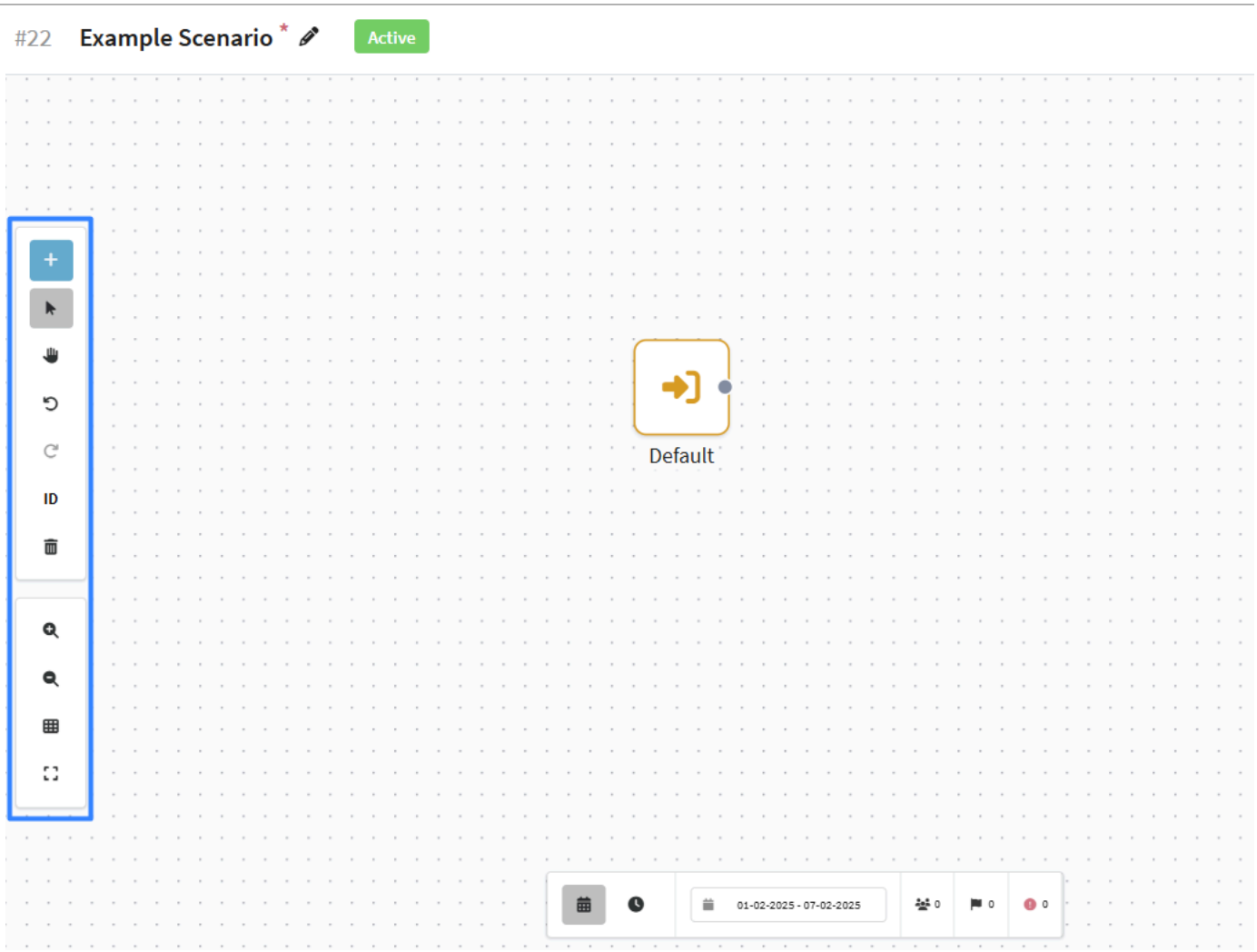
or use the `Ctrl + C` key combination. You can also clone multiple elements, including the links between t



TIP

To select several elements of the scenario, activate the Select tool. Then hold down the left mouse but desired objects or hold down the `Crtl` key and successively select the desired objects with the left mou

Toolbar



Tools for managing elements

— activate the “Select” tool. It is used to control elements of the scenario: move, connect, select, etc.

— activate the “Hand” tool (Space + drag). It is used to move around the scenario workspace.

— undo the last action (Ctrl + Z).

— revert the last canceled action (Ctrl + Shift + Z).

— open the “Bin”. The “Bin” contains all items deleted from the scenario.

ID

— display the ID of each node instead of the name.

Tools for setting up the scenario

— show scenario information (description, tags and groups for the scenario).

— show scenario settings (mode of profile re-entry into the scenario, etc.).

— to activate the scenario.

— to deactivate the scenario.

— view scenario analytics in a channel report.

Tools for workspace management

— zoom into the workspace (Ctrl +).

— downscale the workspace (Ctrl -).

— apply alignment. The elements are aligned relative to each other within the workspace.

— adjust the view. The scale changes to the optimal scale, allowing you to see all elements within the w

List of elements

All elements of the scenario are divided into four categories:

- **Triggers - elements that define the way of capturing a profile into a scenario.**
- **Operators - elements affecting the profile movement through the scenario.**
- **Channels - elements, by means of which a message is sent to the profile.**
- **Actions - elements that trigger a certain action in the platform at the moment of profile entry.**

Triggers

A trigger is a node that starts a scenario. You can configure one or several triggers. The maximum numb
One trigger type can be used multiple times in one scenario.

> NOTE
>
> **In this article, triggers are used in scenarios to start profiles within them. Do not confuse them with trig**

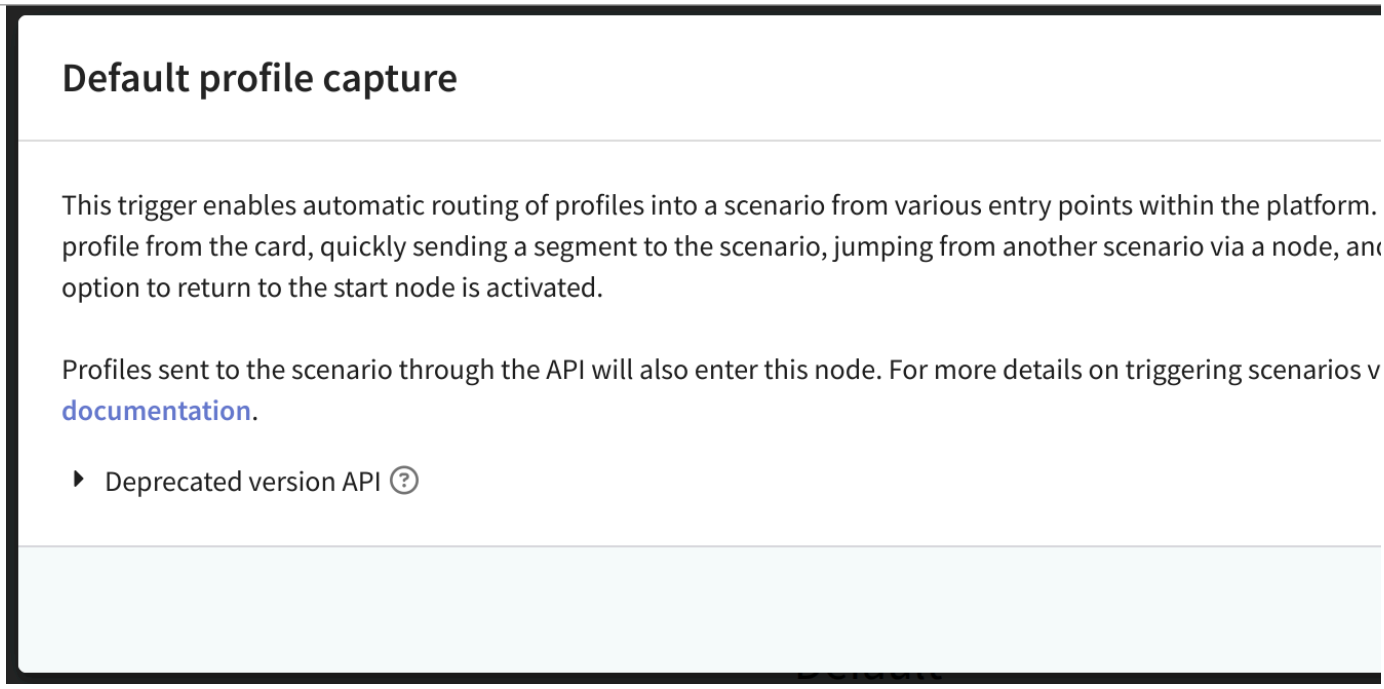
Default profile capture

![img] (/img/attachments/9407058/default_node.png)	This node is a built-in feature of the scenario and can't be removed. With the "Default" node, profiles can enter the scenario fr

Read More

This node captures profiles after following actions:

- **Sending a profile to the scenario via API. When the scenario is saved, the item window will display a**
  However, it is recommended to use a newer version of the API. Go to the documentation for more inf
- **Adding a profile to a scenario manually.**
- **Sending a segment to a scenario through quick actions panel.**
- **Jumping through the node from another scenario.**

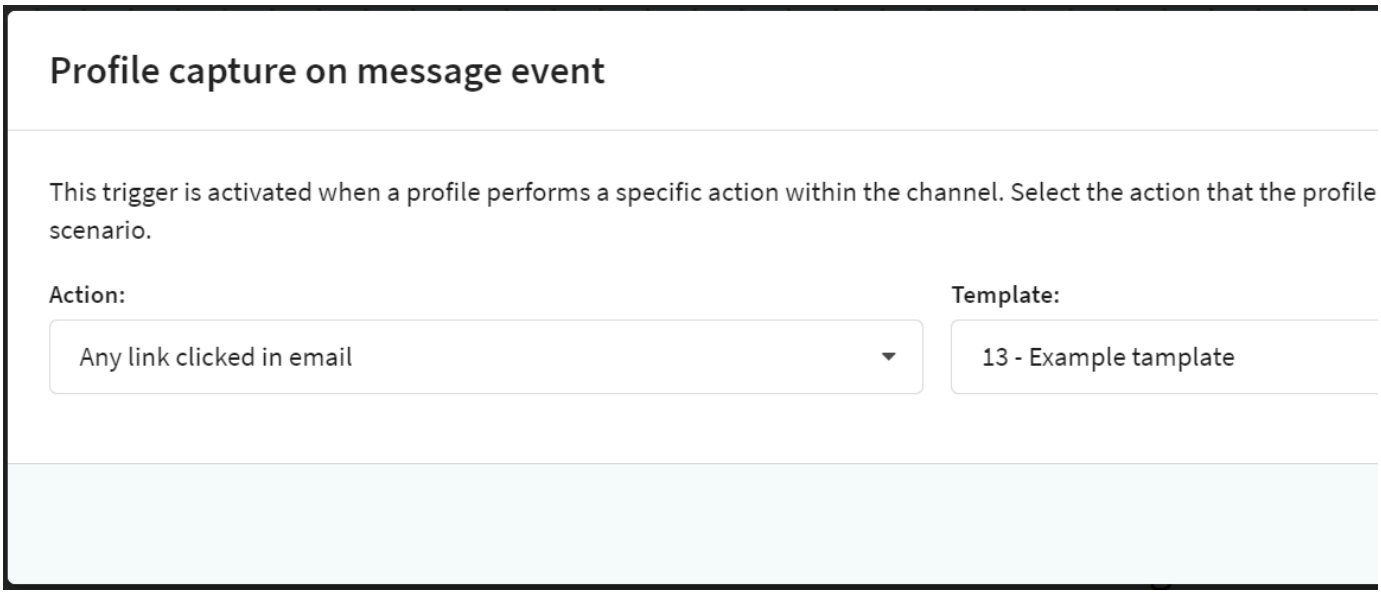


Action execution

![img] (/img/attachments/9407058/132556.png)	**Activated when the subscriber performs the desired action in the message.**

Read More

From the drop-down list, select action in one of the submission channels, and specify the template in wh



Action	Description
Any link clicked in email	**The recipient clicked any link in the selected email template.**
Specific link clicked in email	**The recipient clicked a specific link in the selected email template.**
Email opened	**The tracking pixel recorded the opening of the selected email template.**
Email read	**The tracking pixel recorded that the selected email template was read.**
Email confirmed	**The recipient clicked the subscription confirmation link in the selected email template.**
Any link clicked in SMS	**The platform recorded a click on any link in the selected SMS template.**
Specific link clicked in SMS	**The platform recorded the first click on a specific link in the selected SMS template.**
Push opened	**The platform recorded the opening of the selected push notification. Push notifications are usually opened auto**
Push read	**The platform recorded the reading of the selected push notification in the notification center.**
Any link clicked in push	**The recipient clicked any link in the selected push notification template.**
Pop-up opened	**The platform recorded the display of the selected pop-up template to the user.**
Click on pop-up	**The user clicked the selected pop-up template.**
Pop-up closed	**The platform recorded a click on the close button of the selected pop-up template.**
Pop-up custom event	**The platform recorded a custom event from the pop-up.**

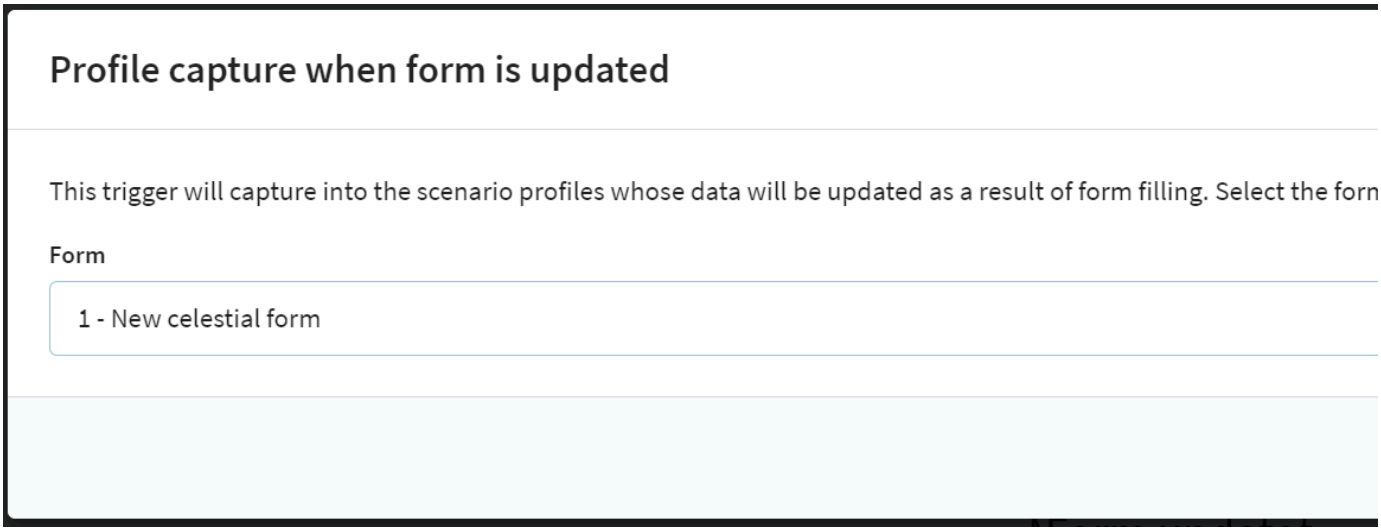
Pop-up filling	**The platform recorded the user filling in one or more fields in the selected pop-up template.**
Notify click	**The platform recorded a click on any link in the selected Notify template. The trigger fires on every click and re**
Notify unique click	**The platform recorded the first click on any link in the selected Notify template.**
Notify specific link click	**The platform recorded the first click on a specific link in the selected Notify template.**
Notify read	**The platform recorded the reading of the selected Notify template.**
MAX bot read	**The platform registered a read event for the MAX bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot click	**The platform recorded a click on any link in the selected WhatsApp*/Viber™/Telegram bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot unique click	**The platform recorded the first click on any link in the selected WhatsApp*/Viber™/Telegram bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot specific link click	**The platform recorded the first click on a specific link in the selected WhatsApp*/Viber™/Telegram bot templat**
WhatsApp*/Viber™/Telegram bot subscription confirmation	**The platform recorded a click on the subscription confirmation link in the selected WhatsApp*/Viber™/Telegram**

Form update

![img] (/img/attachments/9407058/134040.png)	**The trigger captures the profiles whose data will be updated as a result of filling out the form into the scenario. It is triggered only** **form via a transition from an email.**

Read more

In the drop-down box, specify the form to which the trigger will be bound:



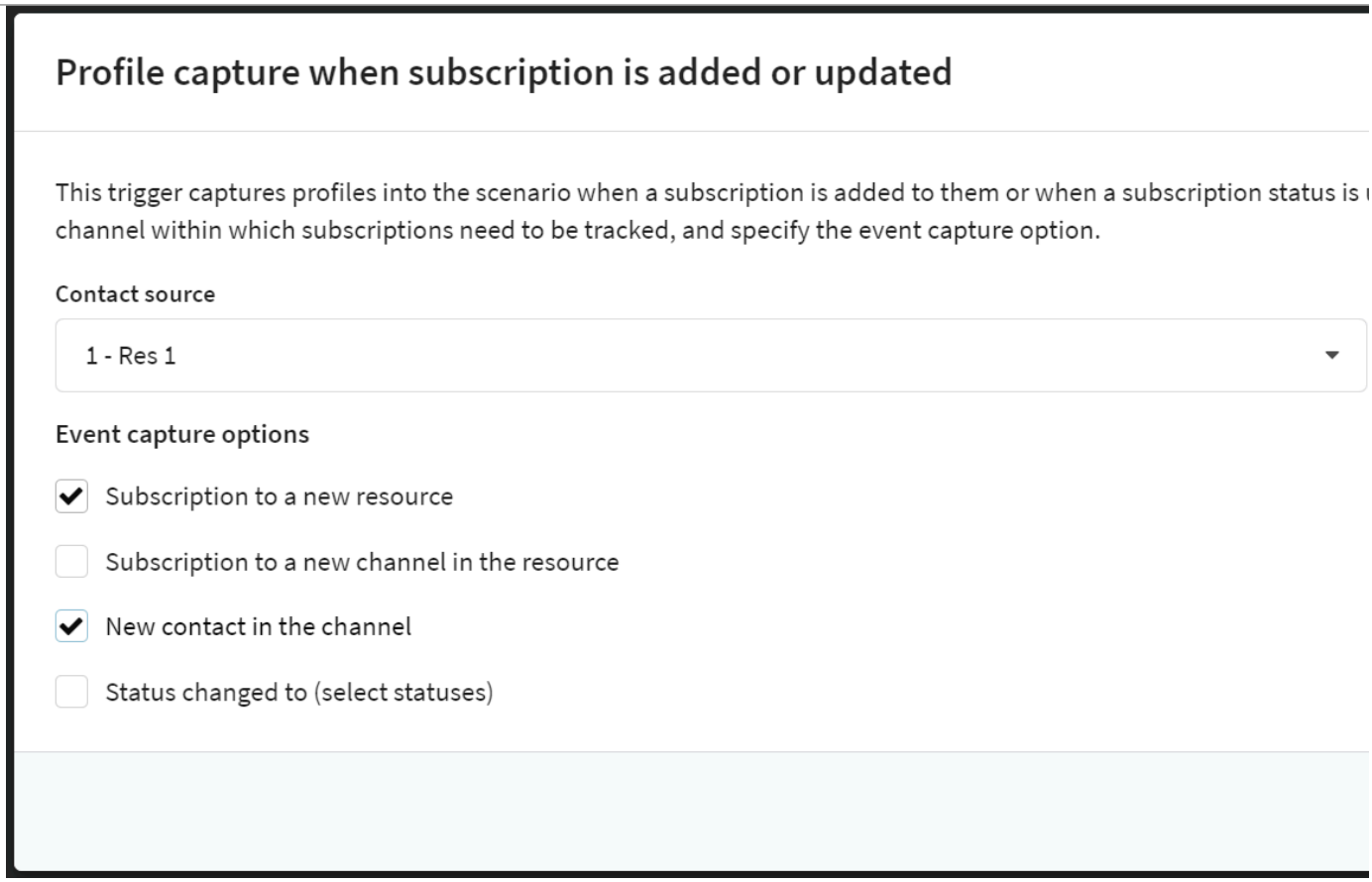
TIP

For more information on how to send a profile a link to a form, see [this article](#).

Subscription add/update

![img] (/img/attachments/9407058/134533.png)	**Trigger is activated when a new subscription is added to a profile or the status of an existing subscription is changed. For the trigg** **for which changes will be tracked. Otherwise, the mailing will expect changes for all channels of the selected resource, for all reso** **profile.**

Read more



- **Subscription to a new resource — condition will work only for the first subscription to the resource s subscriptions to the same resource will not activate the scenario.**
- **Subscription to a new channel in the resource — subscription to a new (for the profile) channel in the there was a subscription to email newsletters, now there is also a subscription to SMS or Push.**
- **New contact in the channel — activates the trigger each time a new subscription appears. The first considered a "new contact in the channel".**
- **Status changed to (select status) — will run when the subscription status changes to those selected**
- **The first three items are selected — the trigger will fire with each new subscription.**
- **All the boxes are unchecked — nothing will happen.**

NOTE

Subscriptions that are added with a status other than "Subscribed" will not be activated by the trigger.

Scheduled segment

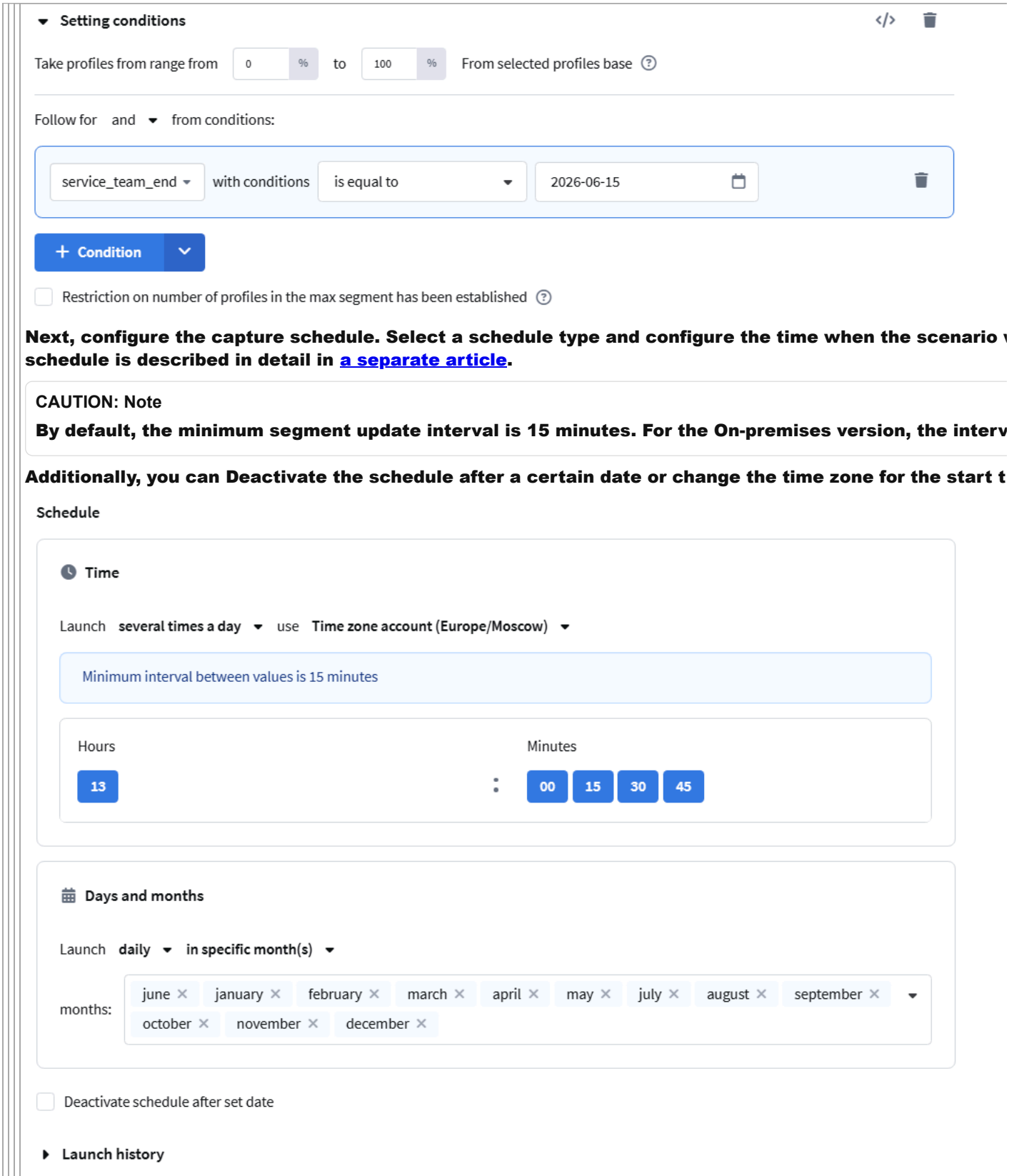
![img] (/img/attachments/9407058/140259.png)	A group of profiles that meet certain conditions is put into the scenario on a set schedule.

Read more

First, configure profile selection. Build a segmentation query against the database by adding rules or gro by:

- **profile data**
- **entity interaction**
- **communication channel activity**
- **and other parameters.**

Segmentation is described in more detail in the relevant documentation section. You can also set a limit percentage:



Next, configure the capture schedule. Select a schedule type and configure the time when the scenario v schedule is described in detail in [a separate article](#).

CAUTION: Note

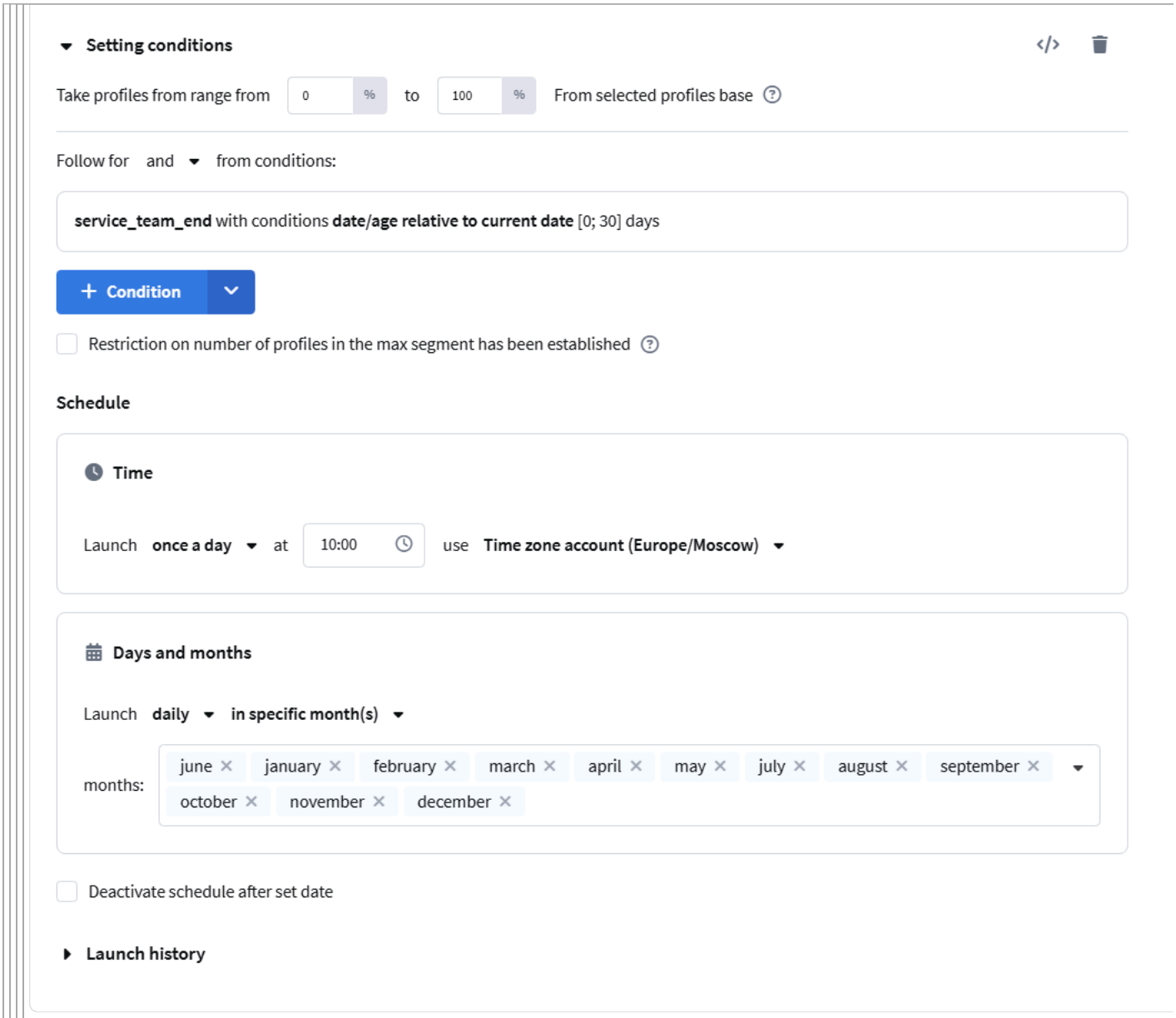
By default, the minimum segment update interval is 15 minutes. For the On-premises version, the interv

Additionally, you can Deactivate the schedule after a certain date or change the time zone for the start t

Schedule

TIP

Each scheduled run recalculates the segment — only those customers who meet the selection criteria a This allows you to build communication scenarios tied to changes in profile data. For example, the scen select every day customers who have exactly 1 month left until their service expires:

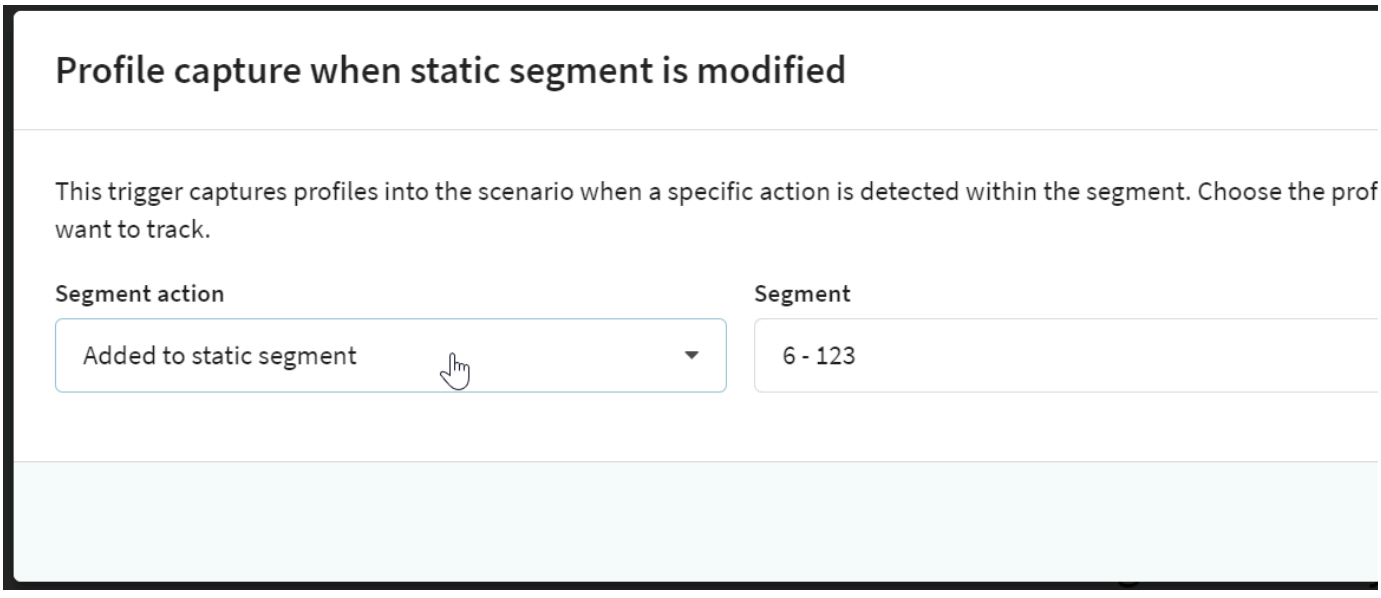


Static/updatable segment modify

![img] (/img/attachments/9407058/141514.png)	**Trigger is fired when there is a change in [**segment**](/en/user-guide/segmentation/): a profile can both enter and exit the specifie**

Read more

In the drop-down box, select the action (entering or leaving a segment) and the segment you want to mo



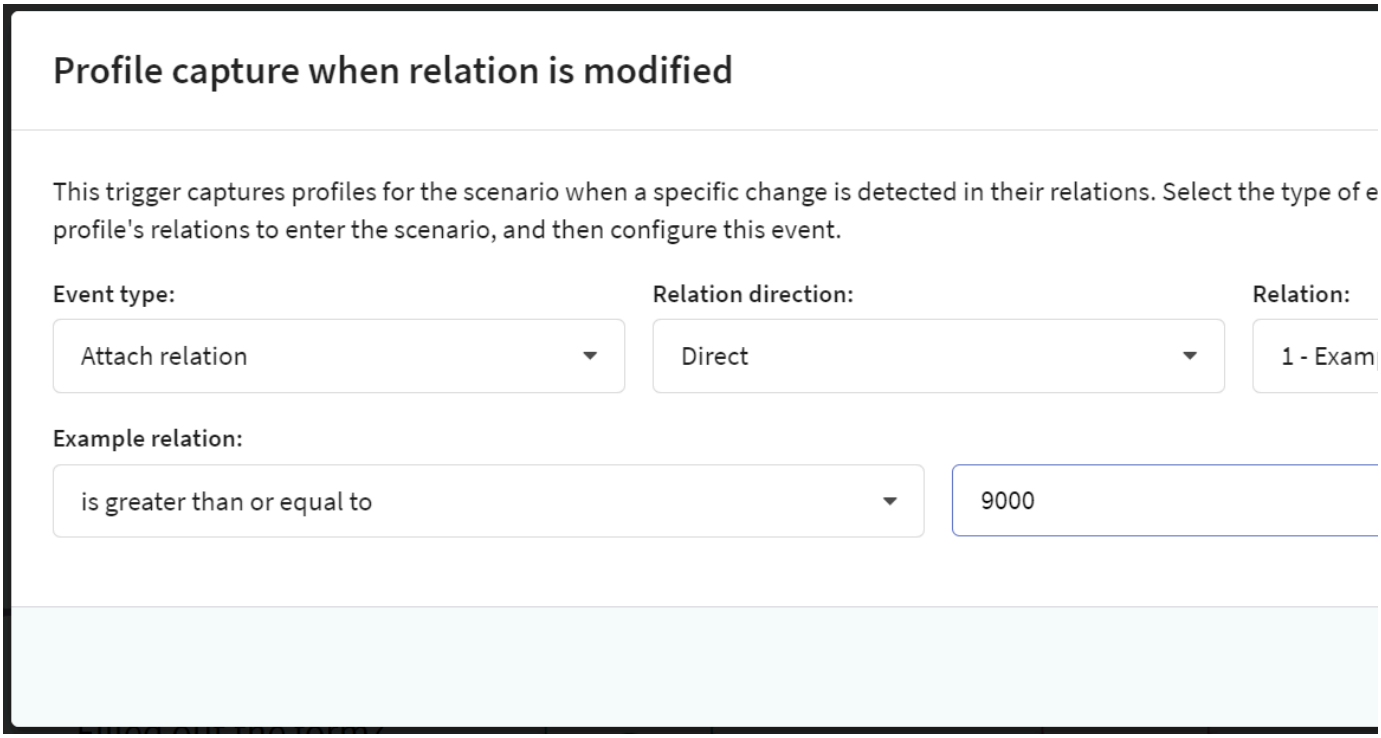
Relation modify

![img] (/img/attachments/9407058/142004.png)	Trigger is started when [**link**](/en/user-guide/profiles-and-databases/20480122) between profiles changes: both forward and back taken into account, as well as the link property reaching the value set in the trigger.Delete/Backspace

Read more

To configure a trigger, specify:

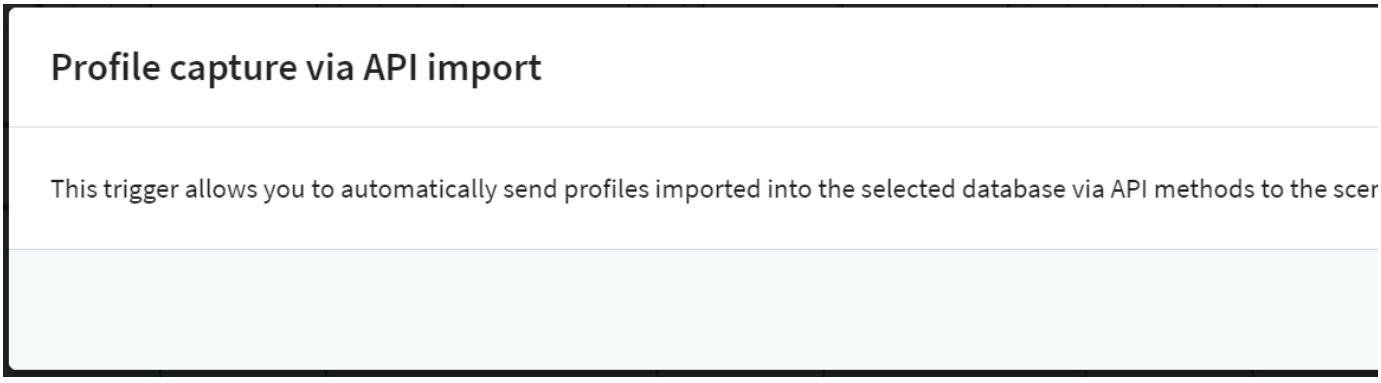
- **relation,**
- **the type of event attached to it,**
- **relation direction,**
- **link property,**
- **property value.**



API import

![img] (/img/attachments/9407058/142355.png)	The trigger is activated when a new profile is added to the database via [**API query**](/en/developer-guide/profiles/2195477). The

Read more



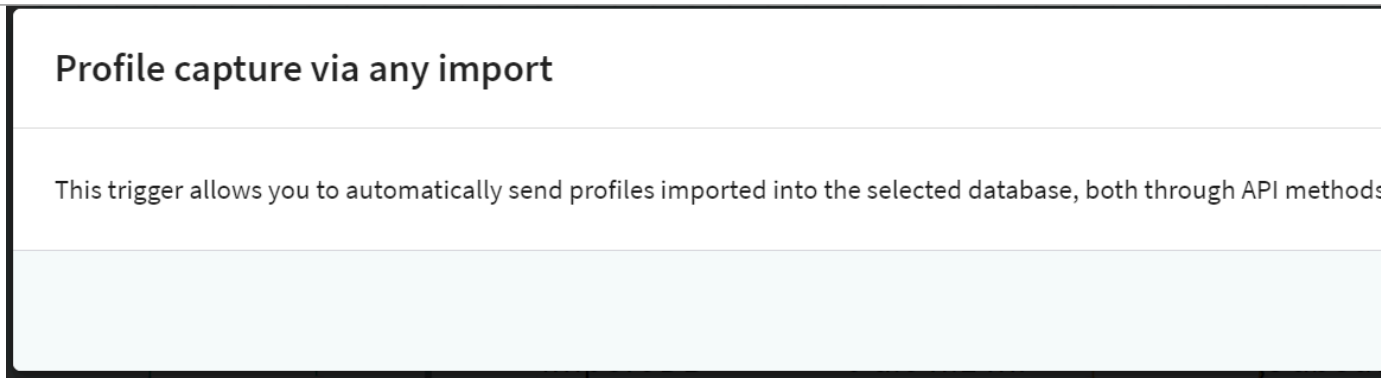
CAUTION

If you select the Any Import method, this trigger becomes unavailable.

Any import

![img] (/img/attachments/9407058/143017.png)	The trigger is activated when a new profile is added to the database in any way: via [**API request**](/en/developer-guide/profiles/2 databases/9405665#Subscriberprofile-Scenarios), from the profile card.

Read more



CAUTION

If you select the **Any Import ** method, the other import triggers become unavailable.

Manual import

![img] (/img/attachments/9407058/143329.png)	The trigger is activated when a new customer profile is [**manually added**](/en/user-guide/profiles-and-databases/9405665) to the

Read more

CAUTION

If you select the Any Import method, this trigger becomes unavailable.

Pop-up import

![img] (/img/attachments/9407058/134040.png)	The trigger will work if a new profile is created in the database as a result of a user filling out and submitting a form in a pop-up wi

Read more

Select the pop-up you need to track the completion of. Pop-up window can be created in Weblayer —> Po importing profiles to the database in the pop-up editor in the Actions.

CAUTION

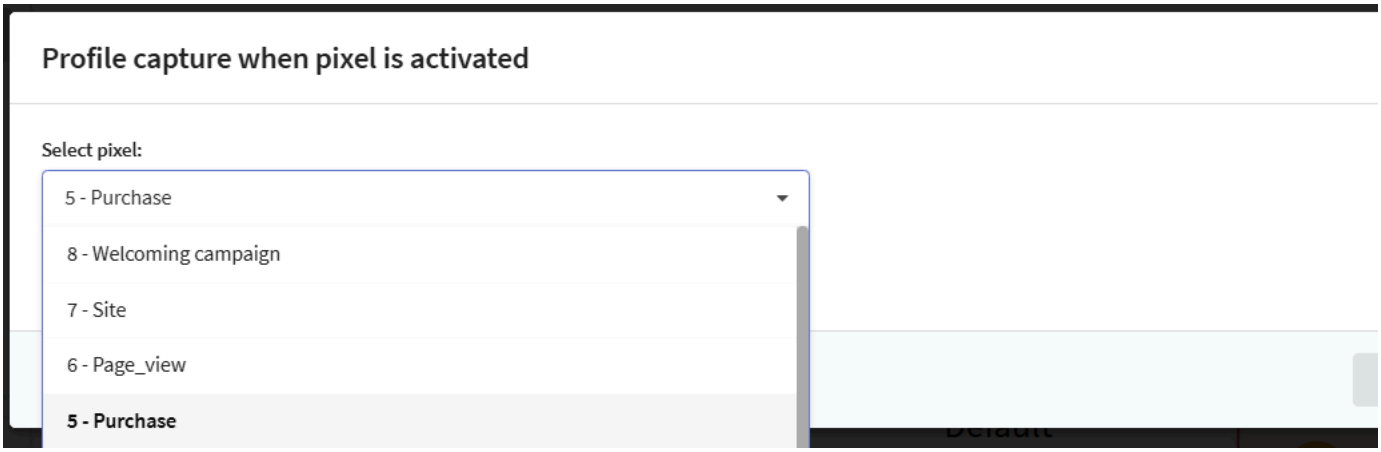
If you select the Any Import method, this trigger becomes unavailable.

Pixel activation

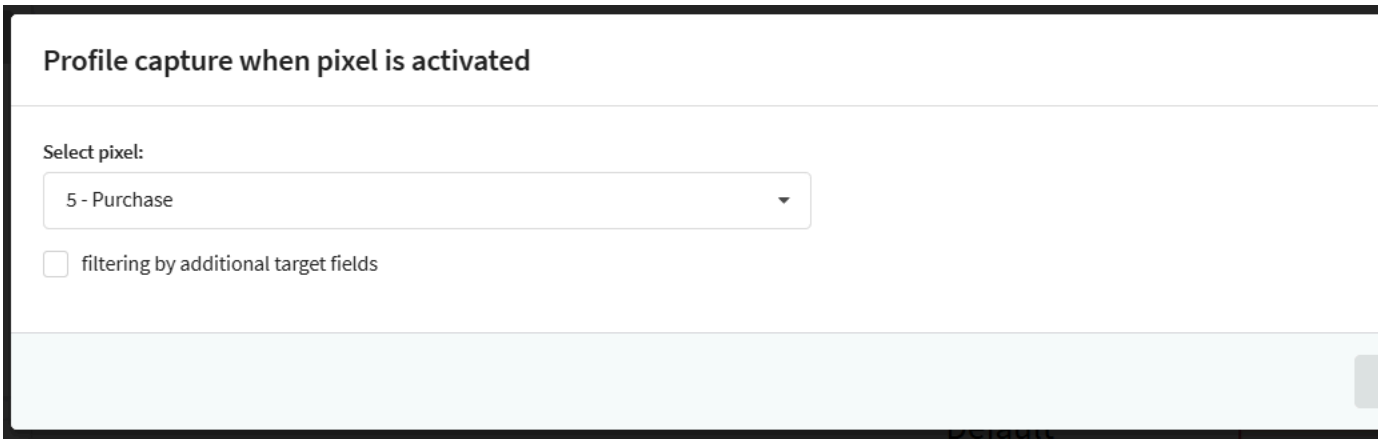
![img] (/img/attachments/9407058/153021.png)	**The profile enters the scenario when the pixel is activated.**

Read more

Select the required pixel in the drop-down box:

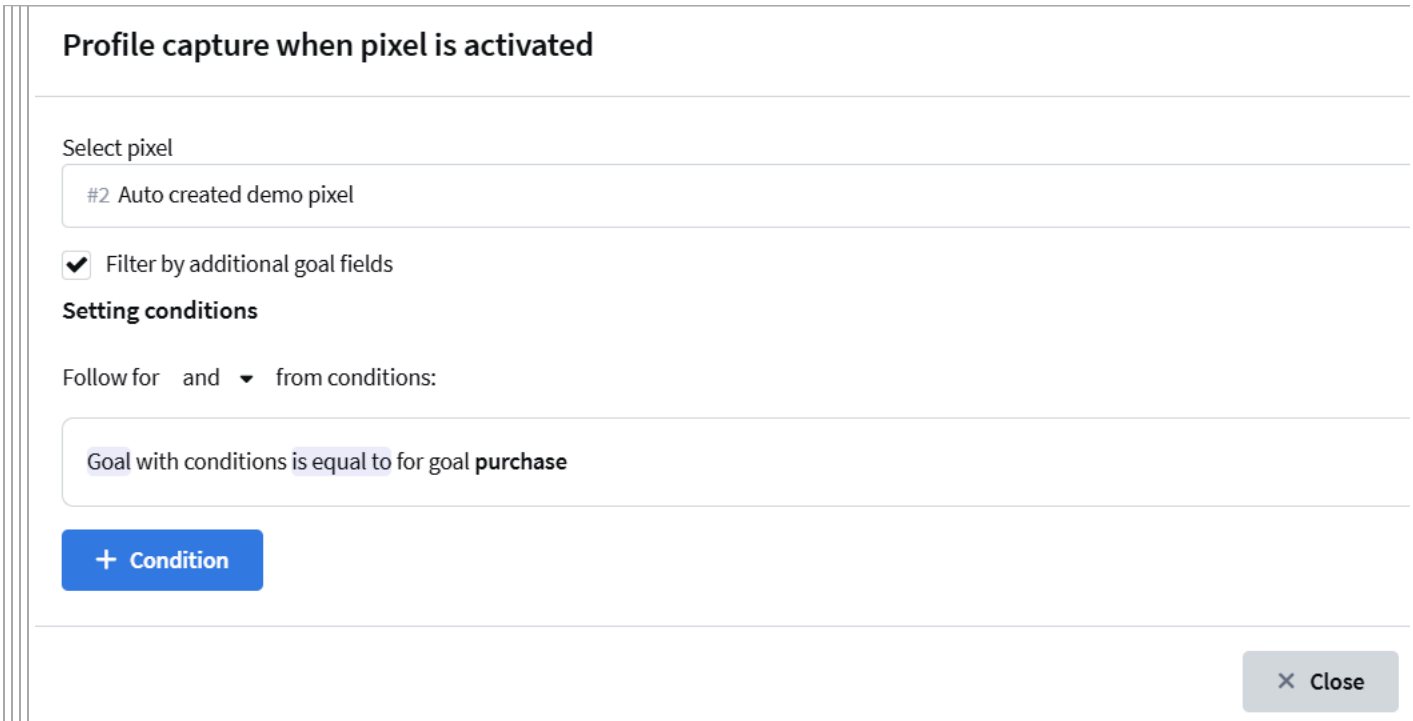


This trigger can keep track of the default pixel triggering:



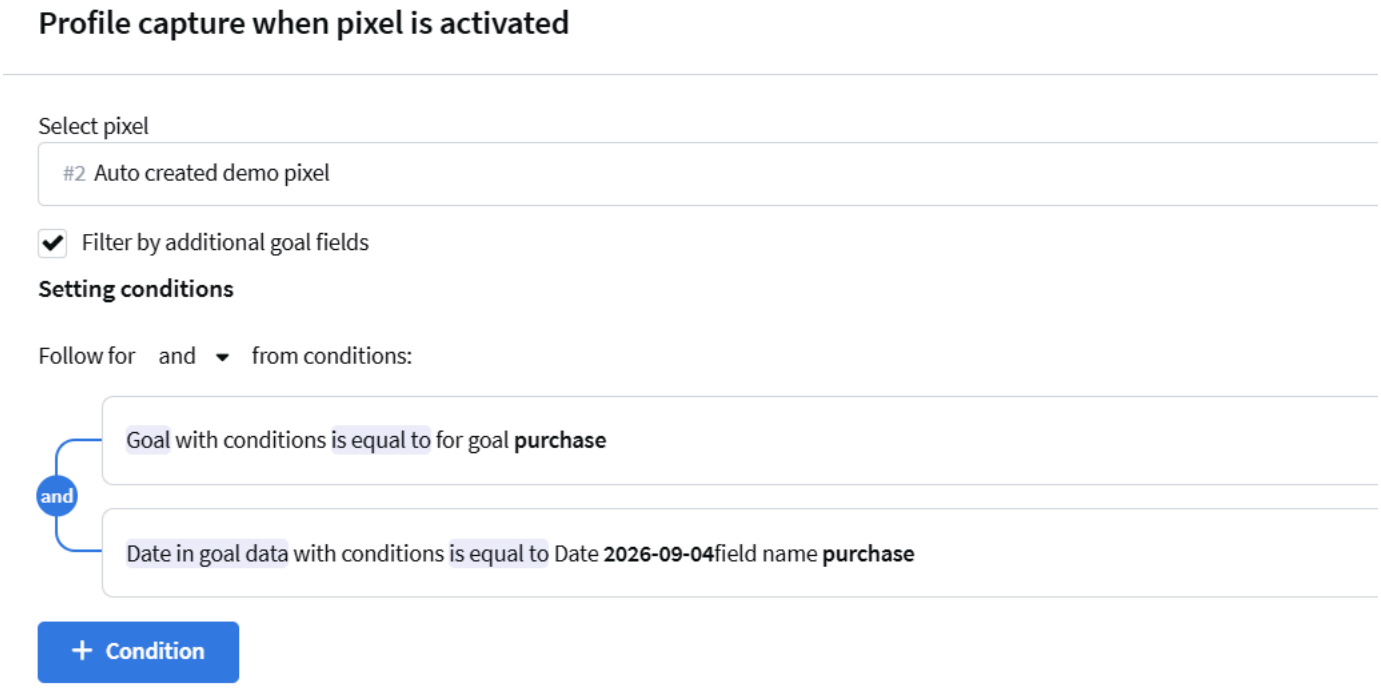
If you are required to track only certain events, you can set up a trigger to register a specific target. To
additional target fields box and configure the required selection parameters.

With the help of such a trigger, the pixel will record only those events that are related to a specific targe
filtering so that only profile registrations, purchases or other key actions are tracked.



For more precise customization, you can use a combination of several selection rules. This is required by target fields box, clicking Add rule and configuring two or more selection rules.

Use AND when you want an event to strictly fit both conditions. Use OR when you want the event to fit a



You can also filter profiles by additional fields. Values of additional fields transferred via API request.

TIP

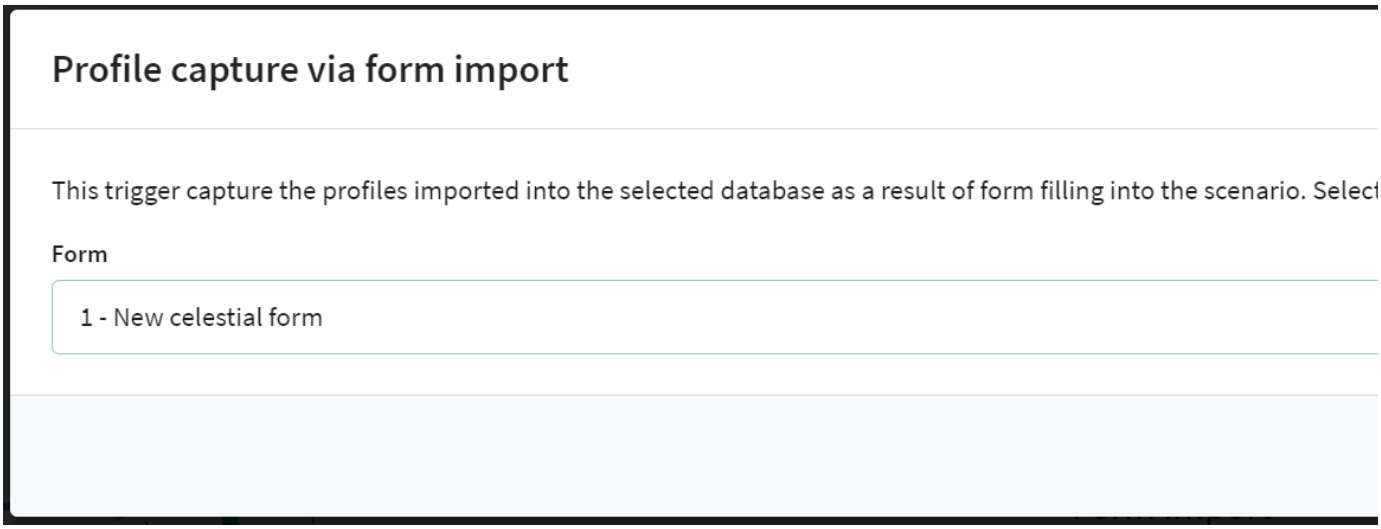
If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is re interval.

Form import

![img] (/img/attachments/9407058/154204.png)	**The trigger will work if a new profile is created in the database as a result of a user filling out and submitting a form.**

Read more

To use this trigger, you need to activate the Profiles import option on the Actions tab when configuring tl in the same database that is used in the scenario.



CAUTION

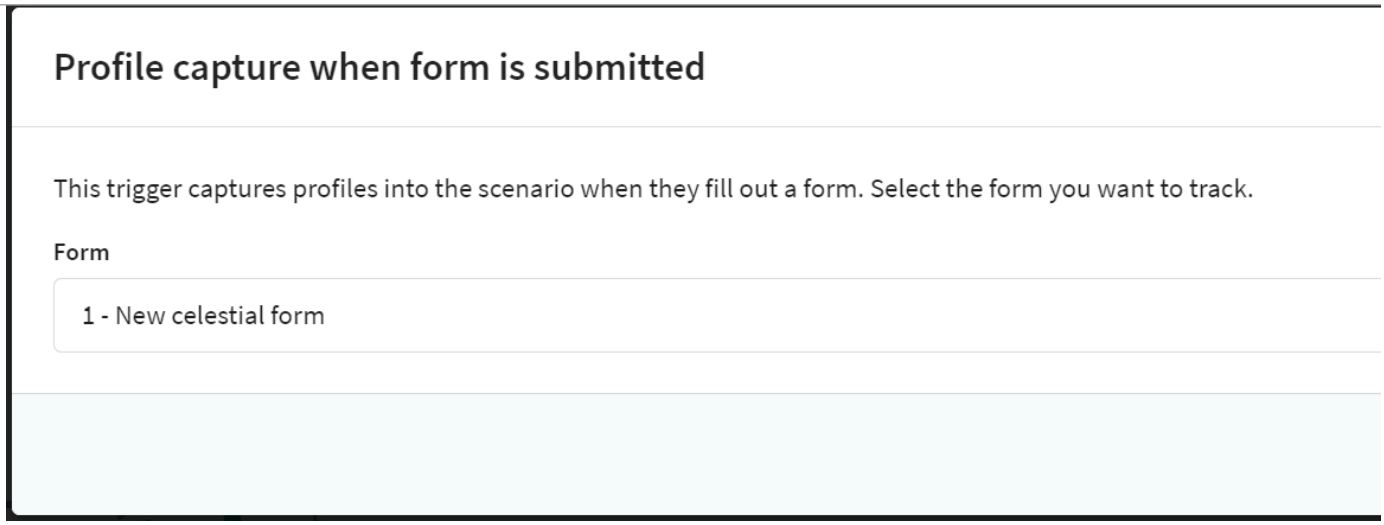
If you select the Any Import method, this trigger becomes unavailable.

Form submition

![img] (/img/attachments/9407058/123456.png)	**Trigger will be activated only if the form was filled out on the transition from the email.**

Read more

In the drop-down list, specify the form whose completion you want to track:



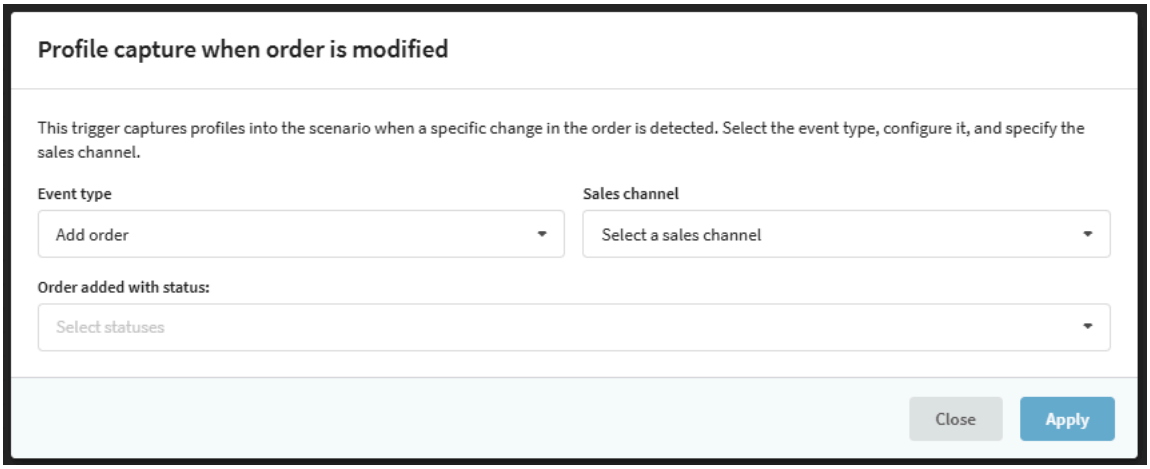
TIP

Read more about how to send a profile a link to the form in [this article](#).

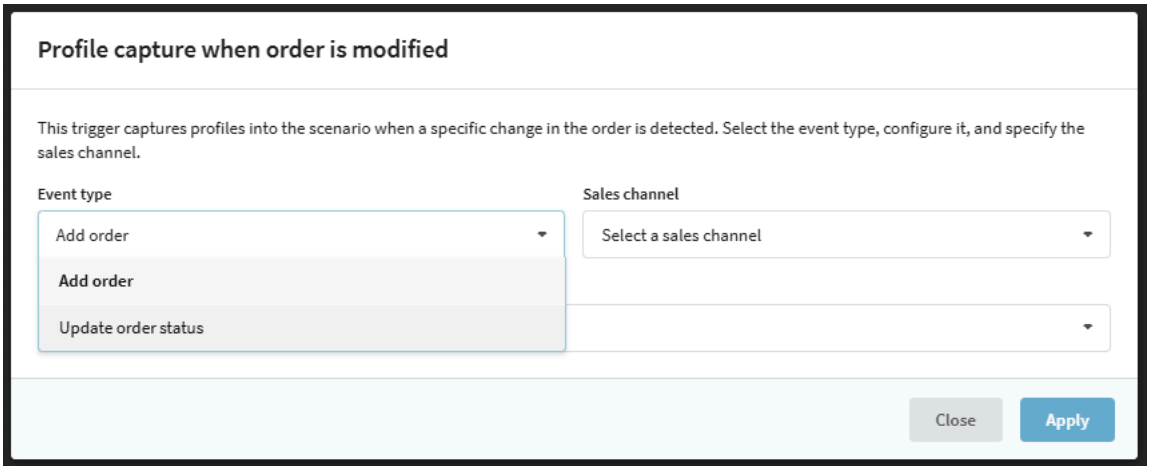
Order modify

![img] (/img/attachments/9407058/123457.png)	**This trigger captures profiles into the scenario when a specific change in the order is detected. The trigger is activated whena sale configured.**

Read more

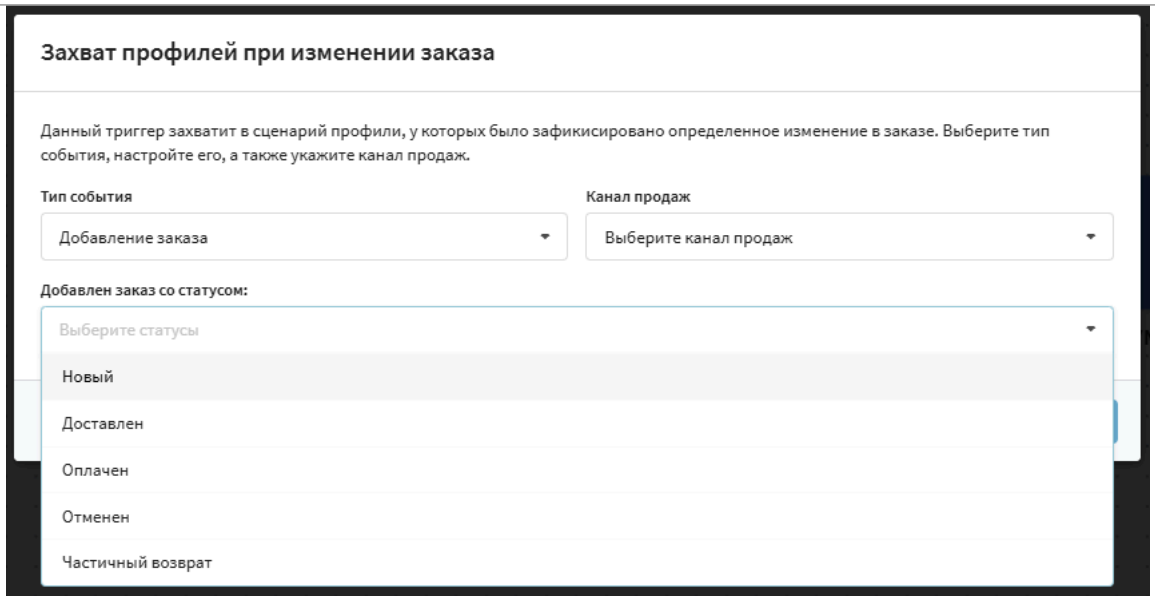


In the drop-down list slect the event type, sales channel and order status.
The following variants of event types are available:



- **Add order — the trigger will activate when an order with one of the specified statuses is added to th**
- **Update order status — the trigger will activate if the status of an existing order changes to one of th**

You can select one or more order statuses that result in a trigger:



TIP

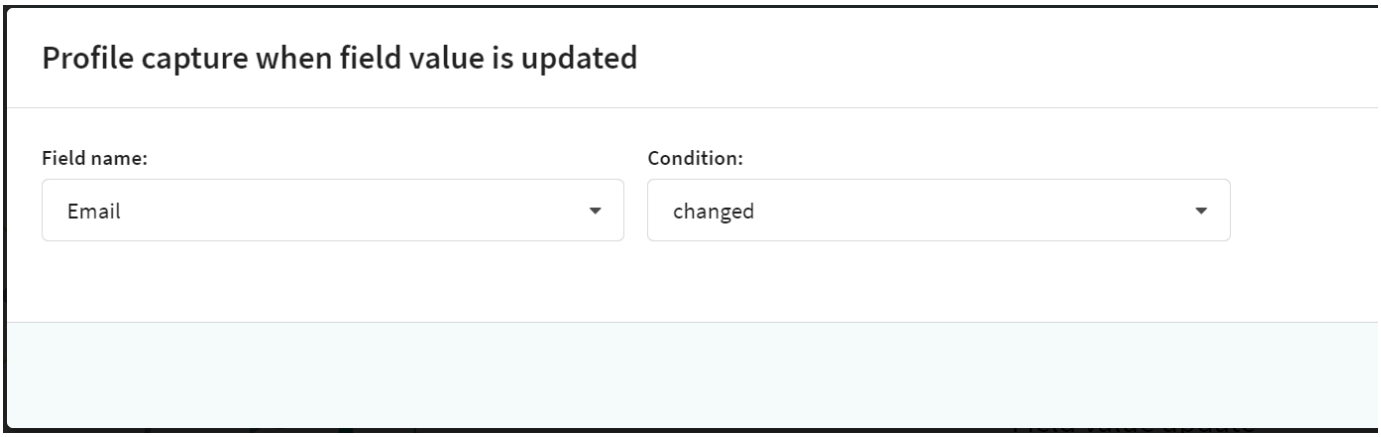
Read more about how to customize orders in [this article](#).

Field value update

![img] (/img/attachments/9407058/154838.png)	**The trigger is activated when the specified [**profile field**](/en/user-guide/profiles-and-databases/66420#default-fields) takes the d**

Read more

Select the profile field, condition, and expected value:



Conditions vary depending on the type of field:

Condition	Field type	Description
changed	**all**	**Any change in the profile field**
equal	**all, except *tags*, *object*, *array of objects***	**The new value is equal to the expected value**
not equal	**all, except *tags*, *object*, *array of oblects***	**The new value is not equal to the expected value**
not empty	**all, except *tags***	**The field received any non-empty value**
empty	**all, except *tags***	**All values are removed for the profile field**
added any tag from the list	***tags***	**The list is defined in the trigger**
removed any tag from the list	***tags***	**The list is defined in the trigger**
property changed	***object***	**The path to the object key is specified. If the value of the key changes, the trigger will be activated.**
property is equal	***object***	**The object key path and expected value are specified. If the value of this key takes the expected value, the trigger will be activated.**
property is not equal	***object***	**The object key path and expected value are specified. If the value of this key does not take the expected value, a trigger will occur.**

Condition	Field type	Description
element property has changed	***array of oblects***	**Specifies the path to an array element. If the property of this element changes, the trigger will fire.** **When selecting the path to the element property, you also select the data type to which the compared value should be converted. The fol float, and bool. Comparison of complex data types (object, array) is not possible. In the example below, you will not be able to track chan** `[ { "key_1": [1,2,3,4,5], "key_2": 2 }, { "key_3":"value", "key_4": [1,2,3,4,5] } ]` **Note that the drop-down list contains available paths to the field property, but the index of any element is 0 by default -** `$[0].key_3`**. To set jsonpath format -** `$[1].key_3`**.**
element property is equal	***array of oblects***	**The path to the array element and the expected value are specified. If the property of this element takes the expected value, the trigger**
element property is not equal	***array of oblects***	**The path to the array element and the expected value are specified. If the property of this element does not take the expected value, a tr**
element added	***array of oblects***	**A new element is added to the array.**
element removed	***array of oblects***	**An element is removed from the array.**

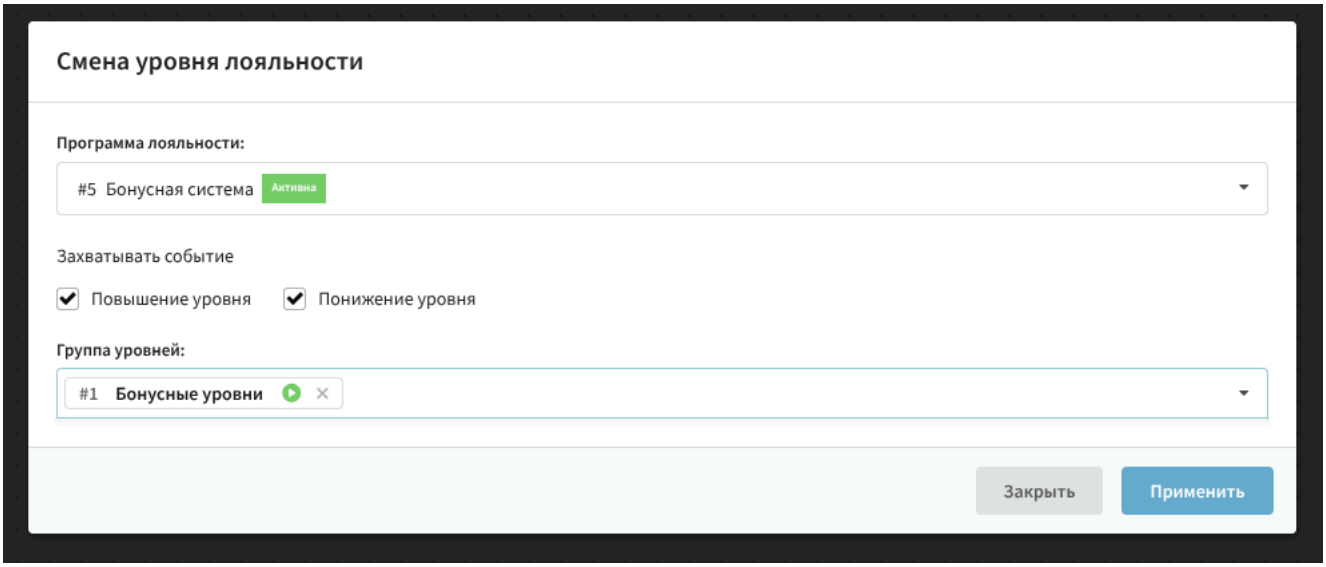
CAUTION

This type of trigger can be used with links in templates that are set to Update profile fields. Note that if link jumps in addition to the field update, they will also be triggered.

Loyalty level change

![img] (/img/attachments/9407058/loyalty-trigger.png)	**The trigger is activated when a profile moves to a different level in a loyalty program.**

Read more

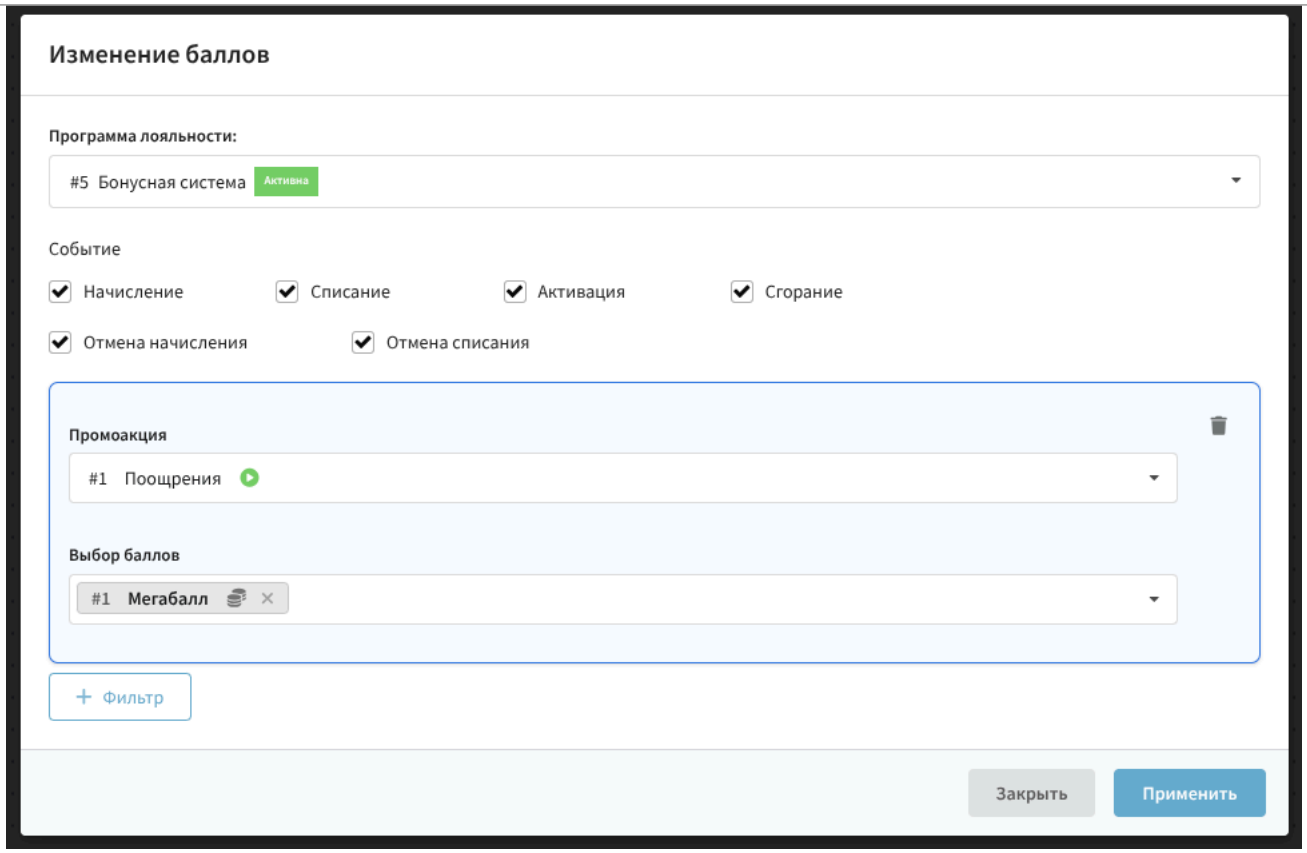


To configure the trigger, select a pre-created loyalty program and tiers group. You can configure the trig downgrade, or any level change.

Points change

![img] (/img/attachments/9407058/loyalty-points-trigger.png)	**The trigger is activated when the number of points on a profile changes in a loyalty program.**

Read more



To configure the trigger, select a pre-created loyalty program and the actions that will activate the trigge

- **Accrual**
- **Redemption**
- **Activation**
- **Expiration**
- **Accrual cancellation**
- **Redemption cancellation**

You can also add a filter to select the promotion and points type for which this trigger will fire.

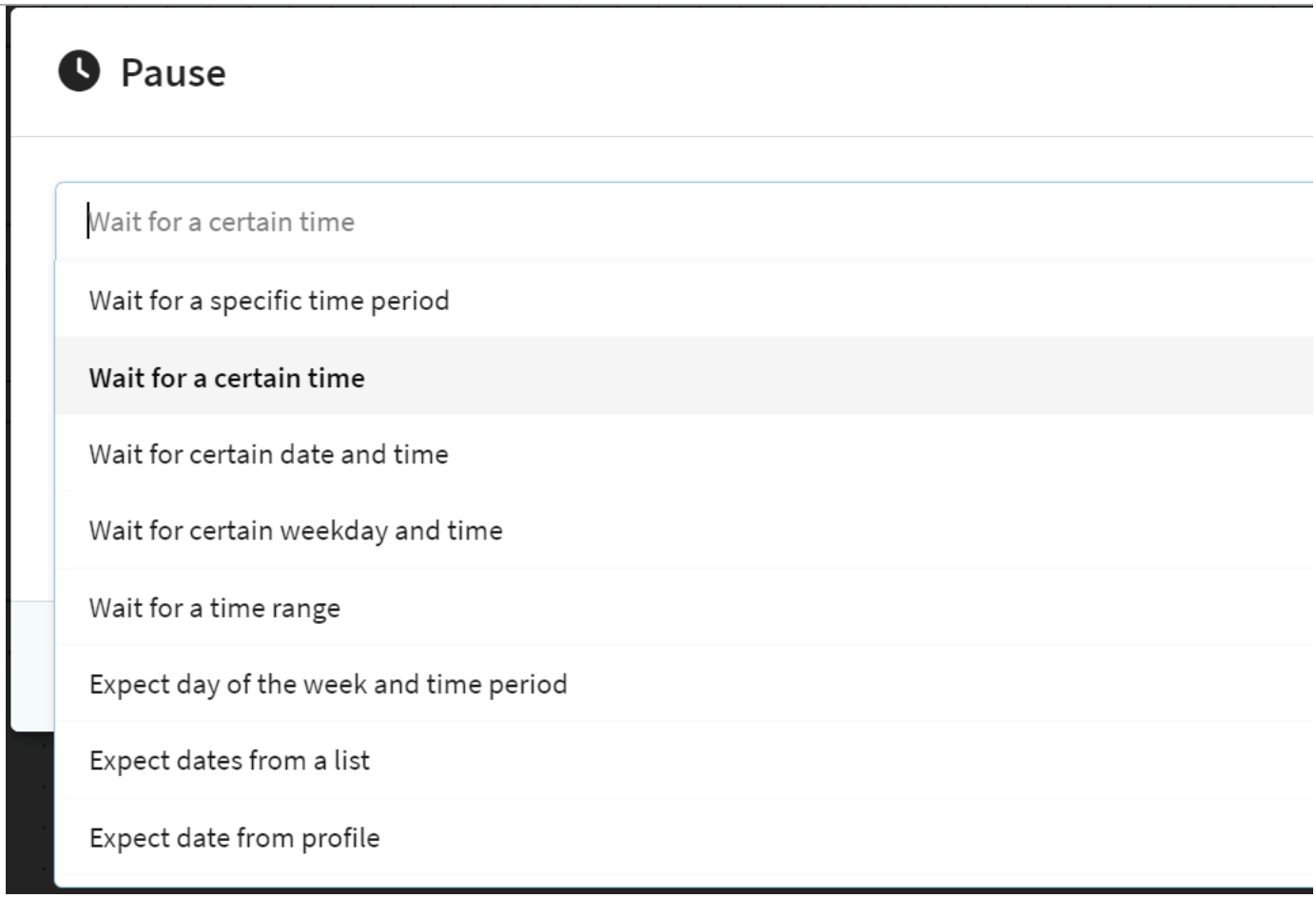
Operators

Pause

![img] (/img/attachments/9407058/pause_node.png)	**Sets the intervals between scenario actions. With this node you can configure the start time of the node that follows. This allo communication time with the client. The node can also wait for certain events: the start of a promotion, a holiday, etc.**

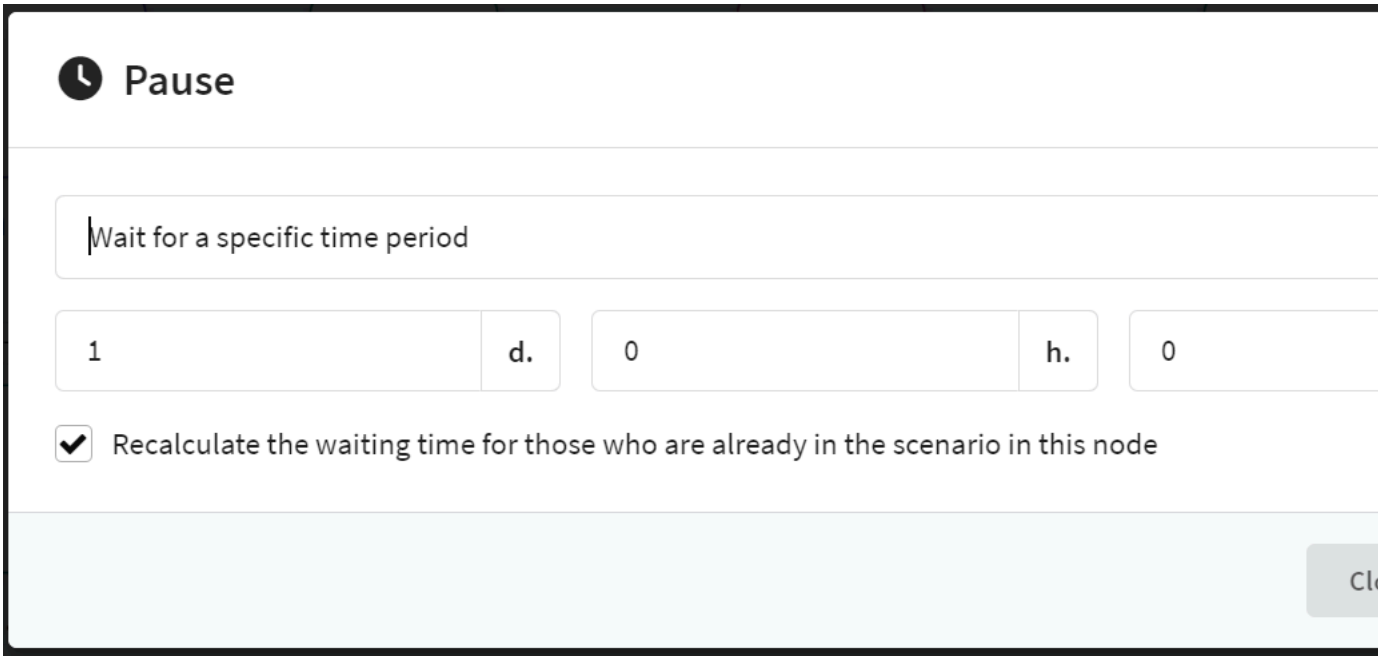
Read more

The following options are available:



Pause relative to the previous action

To wait a specific amount of time after a scenario has captured a customer profile or after a previous ac
“Wait for a specific time period” option:



TIP: Recalculate waiting time

When using the waiting time recalculation feature, the platform will recalculate the pause duration for t Scenario node. For example, you initially set a pause with a duration of 10 days and then increase the v a profile that is captured in the scenario after you make the change will wait 20 days. As for a profile th days before the change was made, the wait time would be another 12 days, not 20.

If the feature is disabled, changing the “Pause” node will not recalculate the waiting time.

Pause until a certain time

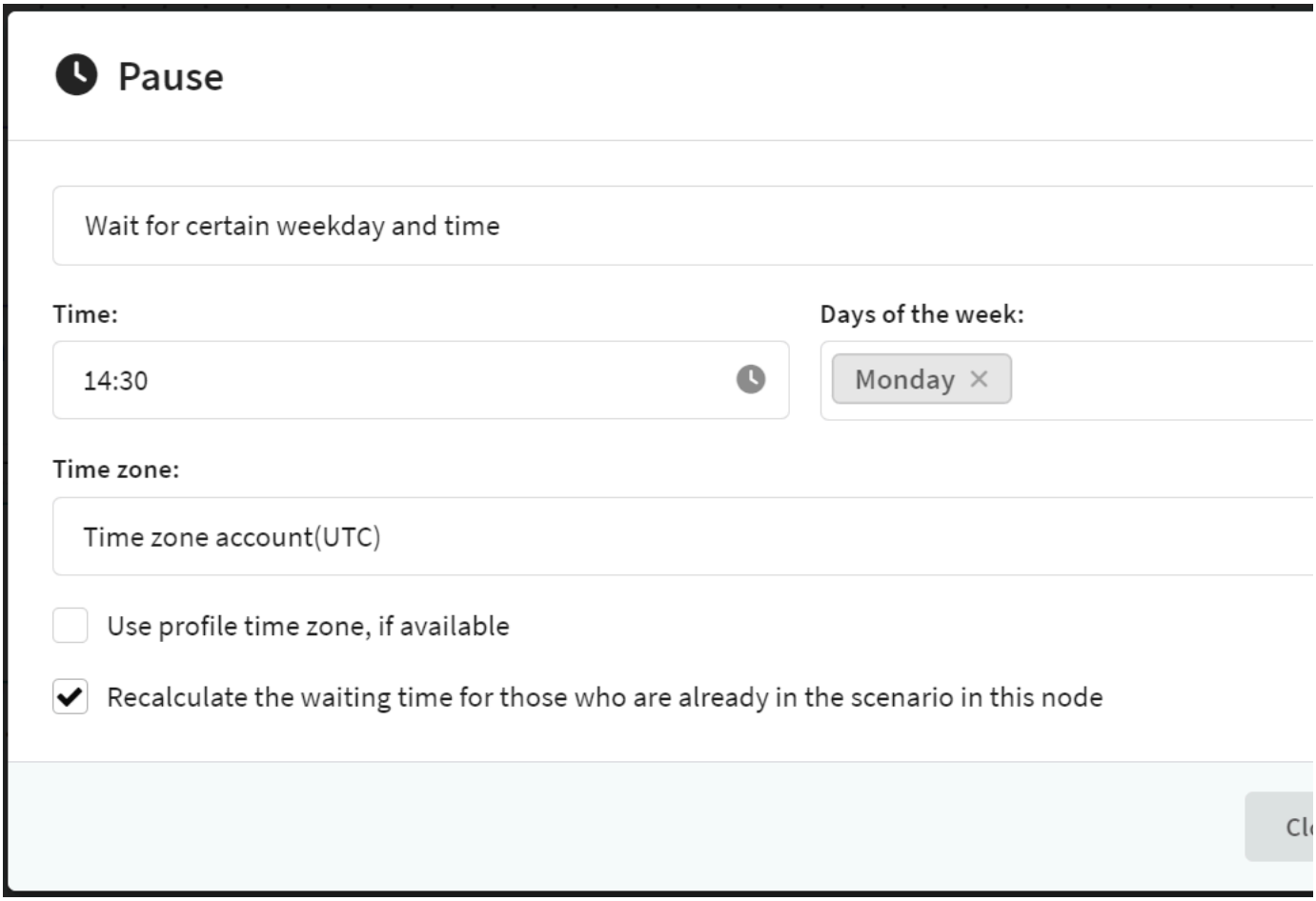
These options pause the scenario until a certain time of day - at the nearest selected time, on a specific week. And if the time zone of the profile is known, it can also be taken into account.

- **Wait for a certain time — the exact time of profile transition to the next node is specified.**
- **Wait for a certain date and time — indicates the exact time and date of the transition to the next noc**
- **Wait for certain weekday and time — the exact time and one or more days of the week are set up**
- **Expect days from a list — selects the exact time and multiple dates when the profile moves to the ne "stops" until the nearest date from the list.**

TIP

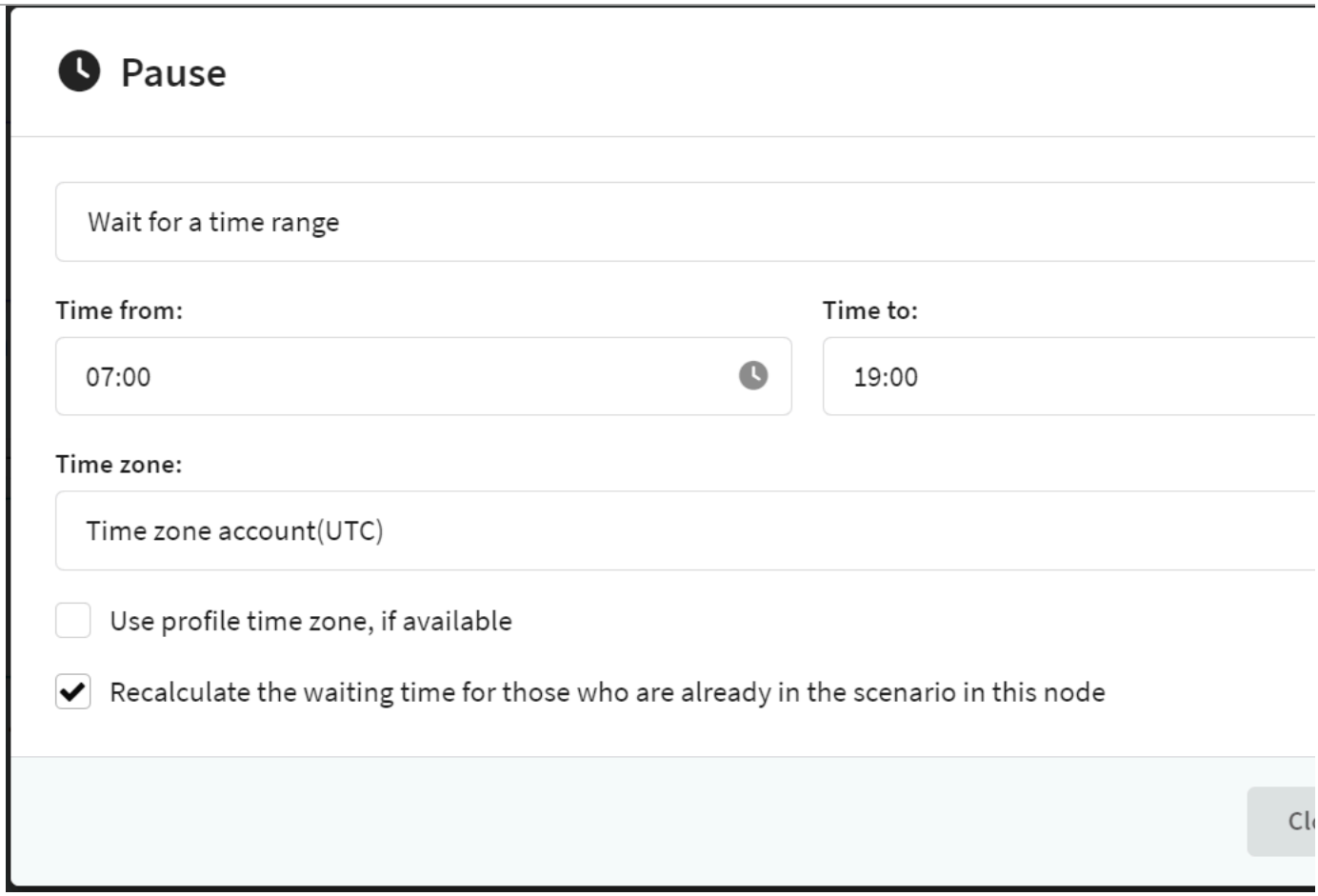
Here's an example of how the date list works in the "Pause" node.

Suppose you set the following dates: "2023-02-01", "2023-07-01", and "2023-10-01". If a profile reaches t continue through the Scenario on February 1. If a profile reaches the node on March 1, 2023, it will proc the node in August, it will wait until October 1. Each profile waits only once until the nearest date relati

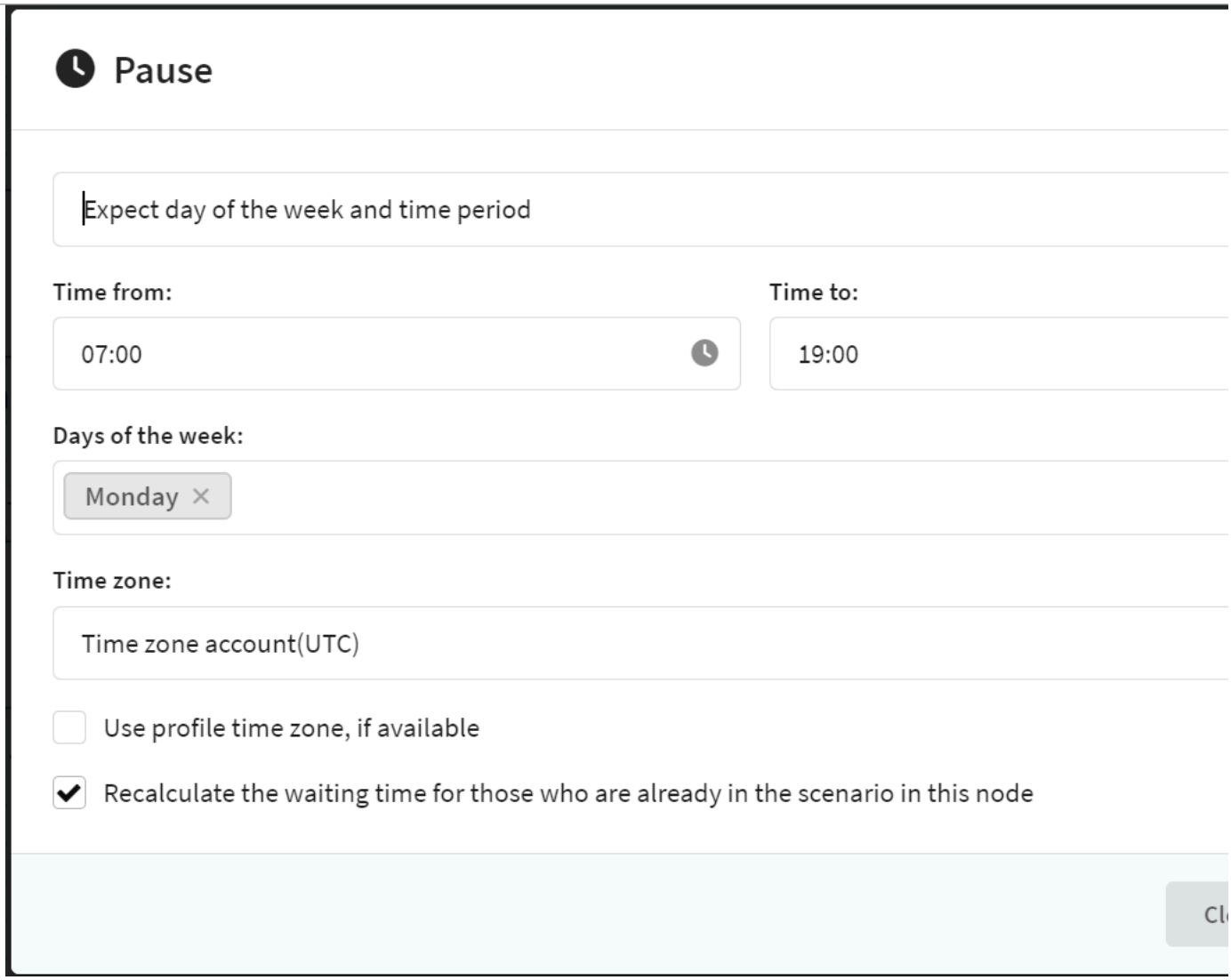


Pause until a certain period of time has elapsed

The Wait for a time range option sets the activity hours for the next node of the scenario. For example, if from 9am to 9pm:



The Expect day of the week and time period option sets the days and hours of activity for the next node want a certain mailing to run only on Saturdays and Sundays from 10 to 12.



Expect date and time from profile

These settings will pause the scenario until the date or time specified in one of the profile fields:

- **Expect date from profile — the profile will move to the next node when the date specified in the sele the "Pause" node settings occur.**

NOTE

The date from the profile is fixed at the moment the profile enters the "Pause" node. If the value in the profile will continue waiting for the date that was set when it entered the node.

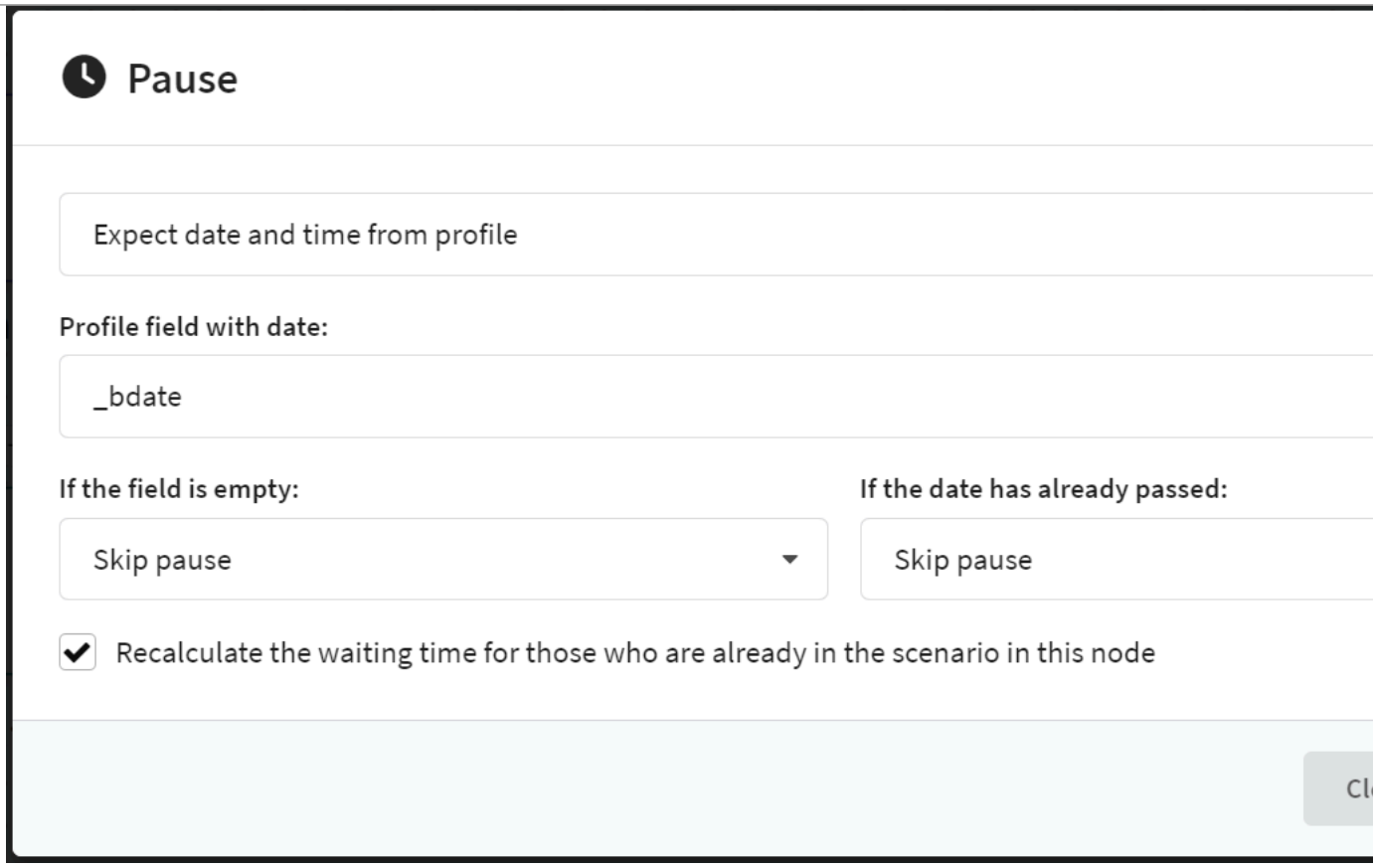
- **Expect time from profile (date-independent) — profile will move to the next node when the time spec date may not be the same as the current date.**
- **Expect date and time from profile — the profile will move to the next node when the dates and times**

Select a field in the database with the required data. It can be date of birth (`_bdate`), date of registration custom field with the "Date" type (for example, date of a scheduled event).

Next, configure the scenario to work if the selected field is not filled in for the profile:

- **Skip pause — the profile will immediately move to the next node without waiting for a certain time.**
- **Abort a script for a profile — the profile will leave the scenario.**

If the date specified in the profile field has already passed, you can either drop the profile immediately t ignoring the pause, interrupt the scenario for the profile, or move the date to the next year.



Best send time

![img] (/img/attachments/9407058/bst_node.png)	**Sets a pause before the next node of the scenario, calculated on the basis of ML service data. The profile will not move to the nex** The optimal way to work is to add a node from the "Channels" category after the "Waiting for best send time" node. The mailing will then send the profile a message at a time that is most convenient for them. </td>

Read more

INFO

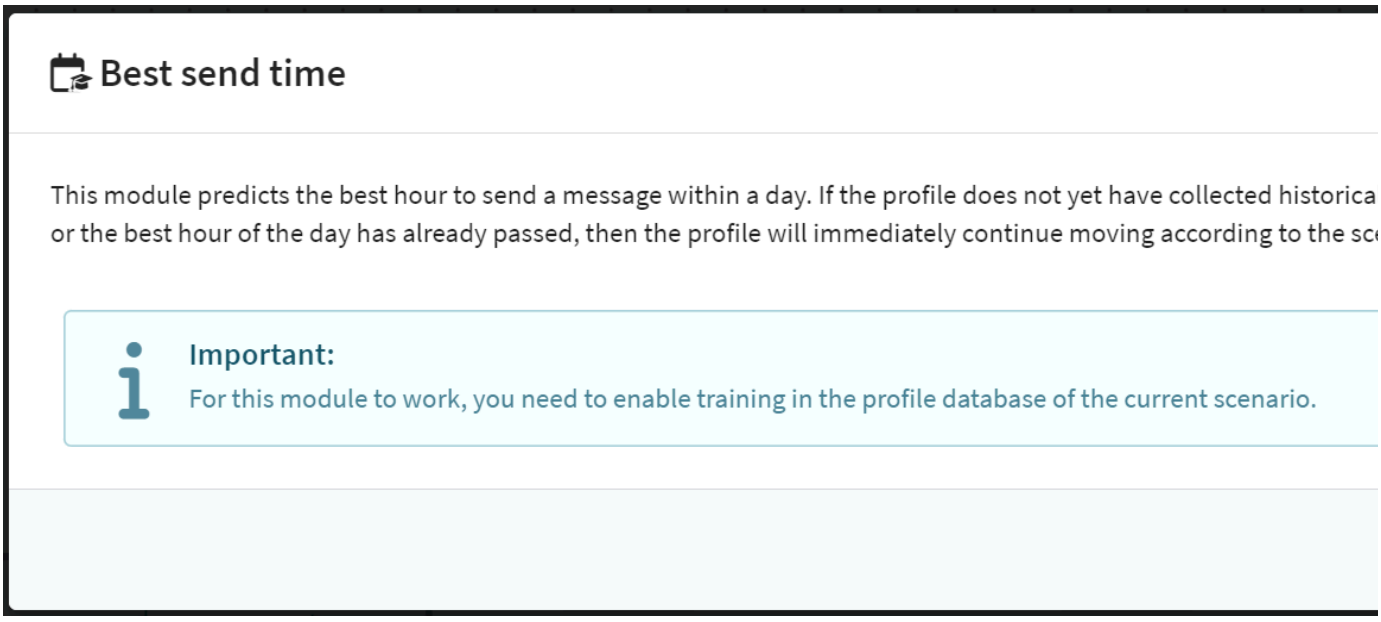
The calculation of the best time is done through machine learning algorithms. ML service determines tl a profile opening a message is the highest, and selects an array with several good sending hours.

The node is available if the profiles captured in the scenario come from a [database] for which machine l

The profile will move to the next node when its best sending hour arrives.

If the best hour has already passed when the profile hits the node, the node will wait until the next good also passed, the profile will immediately move to the next node.

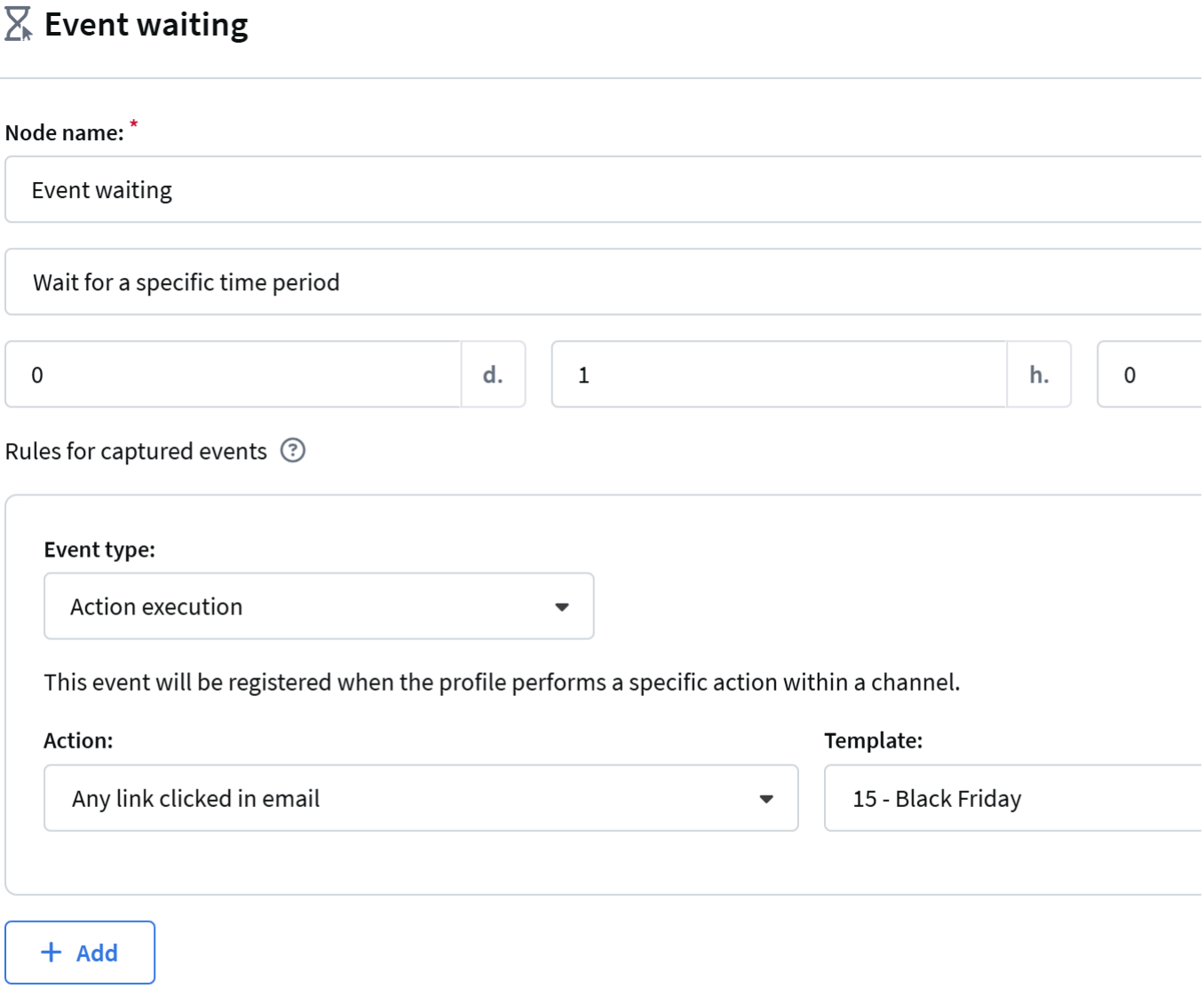
If a profile that has no data on the best sending hour enters the node, it will also be immediately forward



Event waiting

![img] (/img/attachments/9407058/event_waiting.png)	Stops a profile's progress along the scenario **until a specific action is performed by the profile**. Once **any** of the speci proceed to the next node along the upper (green) branch. If the waiting time exceeds the defined value, the profile will move

Read more



Enter the node name for display in the scenario. Then configure the event waiting time. The profile will r
period. If none of the specified events occurs during this time, the profile will proceed along the scenari

Wait time settings:

- **Wait for a specific time period — sets a fixed duration for waiting for the event. The profile will move does not occur within the specified time.**
  `![](/img/attachments/9407058/en/days-hours-minutes.png)`
- **Wait for a certain time — the profile will move along the lower branch if the event does not occur by**
  `![](/img/attachments/9407058/en/certain-time.png)`
- **Wait for a certain date and time — the profile will move along the lower branch if the event does not**
  `![](/img/attachments/9407058/en/date-and-time.png)`
- **Wait for a certain weekday and time — one or more days of the week and an exact time are configur lower branch if the event does not occur by the nearest selected day and time.**
  `![](/img/attachments/9407058/en/day-and-time.png)`
- **Expect dates from a list — several dates and a time are selected. The profile will move along the low by the nearest date from the list. Each profile waits until the date closest to the moment it entered t**
  `![](/img/attachments/9407058/en/date-list.png)`

- **Expect date from profile — the profile will move along the lower branch if the event does not occur b profile field.**

- **Expect time from profile (date-independent) — the profile will move along the lower branch if the eve specified in the selected profile field. The date may differ from the current date.**

- **Expect date and time from profile — the profile will move along the lower branch if the event does nc specified in the selected profile field.**

Condition

![img] (/img/attachments/9407058/condition_node.png)	**Realizes logical selection of the next step of the scenario for customer profiles. The selection condition is defined by Segm continue to move to the right in the scenario. If the condition is not met, the profile will move down the scenario and an alte can also wait for the condition to be met for a certain amount of time.**

TIP

If you link several Condition elements vertically, you can create several alternative chains in one - depe

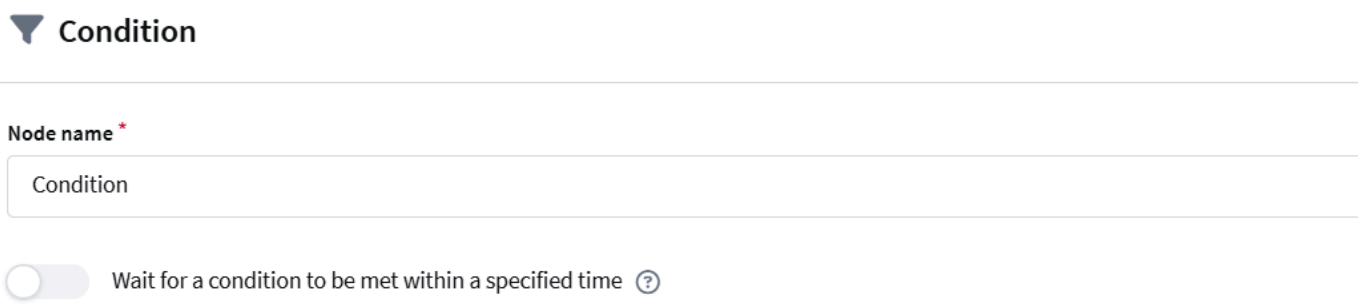
Read more

General settings

Enter the name of the node to be displayed in the scenario.

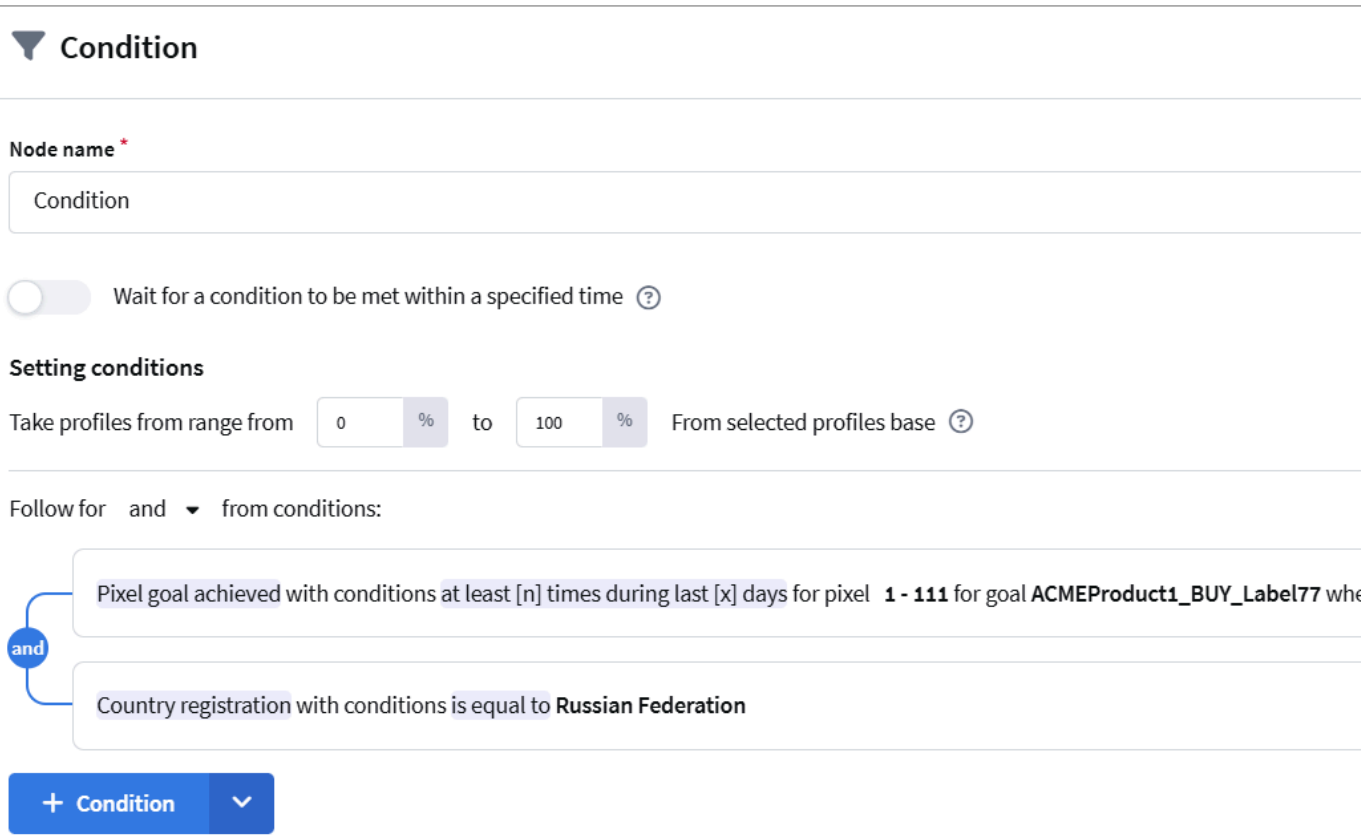
TIP

If you want to dynamically check if the condition is met, enable the “Expect condition matching for a ti check will be performed several times. If the waiting time is fixed, do not enable this option, but instea condition.



Rules for fulfilling the condition

The condition is composed by rules and rule groups similar to Segmentation rules. Subscribers can be c subscription data, action history:

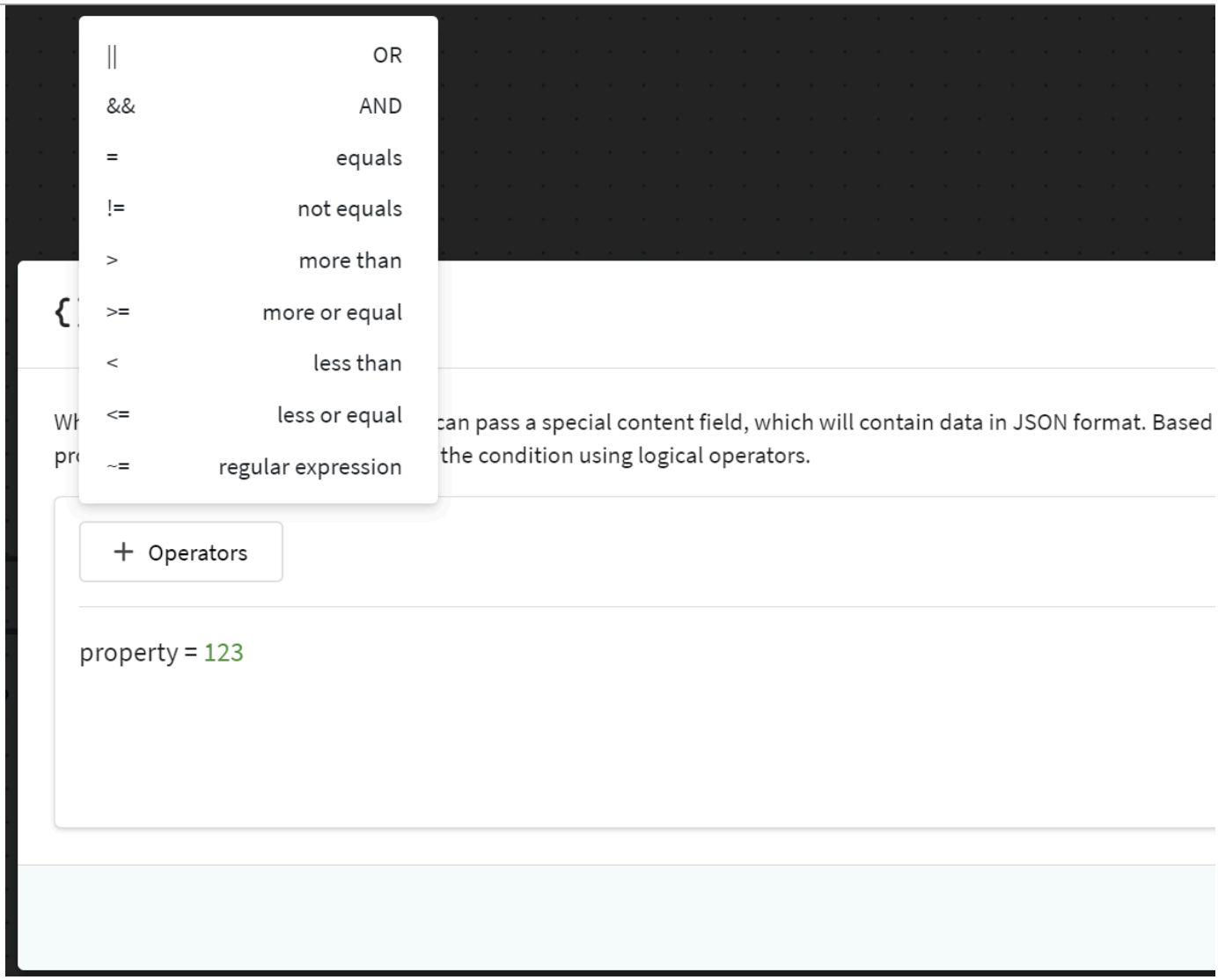


JSON filter

![img] (/img/attachments/9407058/json_filter_node.png)	If you call a scenario via API, you can pass a special `content` field with an undefined structure in JSON format (you define an Action Hook, or to build conditions in a scenario. For example, when a user takes an action, the website (or application) to it. The "JSON filter" node determines further actions for the profile based on the validation of this data.

Read more

JSON filter works in the same way as a normal condition. You can repeatedly use this node to build com
You can set conditions in the code editor using logical operators and expressions.



An example of a `"content"` field in an API request:

```
{
  "content": {
    "customer_id": "12345",
    "total_purchases": 5,
    "last_purchase_amount": 250,
    "average_purchase_amount": 100,
    "preferred_category": "electronics",
    "loyalty_points": 300,
    "membership_status": "premium",
    "discount_coupons_used": 2,
    "active_coupon": "win0513",
    "promo": null
  }
}
```

Note, the JSON object key is specified without quotes (e.g., `membership_status="premium"`).

Operator	Example	
`||` **Or**	`(preferred_category = "electronics") || (preferred_category = "computers")`	**The condition will be fulfilled if the c categories.**
`&&` **And**	`(total_purchases >= 5) && (average_purchase_amount >= 100)`	**The condition will be fulfilled if the c check exceeds 100.**
`=` **Equal**	`membership_status = 'premium'`	**The condition will be fulfilled if the c**
`!=` **Not equal**	`membership_status != 'standard'`	**The condition will be fulfilled if the c**
`>` **More than**	`last_purchase_amount > (2 * average_purchase_amount)`	**The condition will be fulfilled if the c purchase amount.**
`>=` **More than or equal**	`last_purchase_amount >= 250`	**The condition will be fulfilled if the a**
`<` **Less than**	`loyalty_points < 500`	**The condition will be fulfilled if the n**
`<=` **Less than or equal**	`loyalty_points <= 500`	**The condition will be fulfilled if the n**
`~=` **Regular value**	`active_coupon ~= '^win'`	**The condition will be fulfilled if the p**
`!~=` **Opposite regular value**	`active_coupon !~= '^win'`	**The condition will be fulfilled if the p**
`+` **Addition**	`(loyalty_points + bonus_points) < 200`	**Arithmetic addition operator.**
`-` **Subtraction**	`(loyalty_points - penalty_points) > 100`	**Arithmetic subtraction operator.**

Operator	Example	
`*` **Multiplication**	`(bonus_points * years) > 360`	**Arithmetic multiplication operator.**
`/` **Division**	`(total_purchase / 365) > 5`	**Arithmetic division operator.**
`%` **Modulo division**	`(total_sum % 15) = 5`	**Operator for calculating the remaind**
`^` **Exponentiation**	`(promotion_points ^ 3) > 360'`	**Operator for exponention a number t**
`is defined` **Checking the existence of a field**	`promo is defined`	**The condition will be fulfilled if the " specified for it or not.**
`isnot defined` **Field absence check**	`promo isnot defined`	**The condition will be fulfilled if the "**
`is null` **Checking empty field value**	`promo is null`	**The condition will be fulfilled if the "**
`isnot null` **Checking the field value**	`promo isnot null`	**The condition will be fulfilled if the " `null`).**
`contains` **Checking if the value on the right contains the value on the left**	**String:** `"hello" contains "he"` **Array:** `[1, 2, 3] contains 2`	**The condition will be fulfilled if the f**
`in` **Checking if the value on the left is contained in the field or array on the right**	**String:** `"he" in "hello"` **Array:** `2 in [1, 2, 3]`	**The condition will be fulfilled if the v**
`notin` **Checking if the string or array does NOT contain the value of the field on the left**	`"bye" notin "hello"`	**The condition will be fulfilled if the "**

To access a nested field of an object, use the following syntax: `object.field`.

To access elements of an array by index, use the following syntax: `array.[0].field`.

Examples of usage:

```
// Object:

{
  "content": {
    "customer_id": "12345",
    "total_purchases": 5,
    "promo_campaign": {
      "tickets": 115
    },
    "tickets_info": [
      {
        "ticket_id": "T001",
        "price": 50,
        "date": "2024-08-30"
      },
      {
        "ticket_id": "T002",
        "price": 75,
        "date": "2024-08-31"
      }
    ]
  }
}

// Accessing a nested field of the object:

promo_campaign.tickets >= 100

// Accessing the first element of the array:

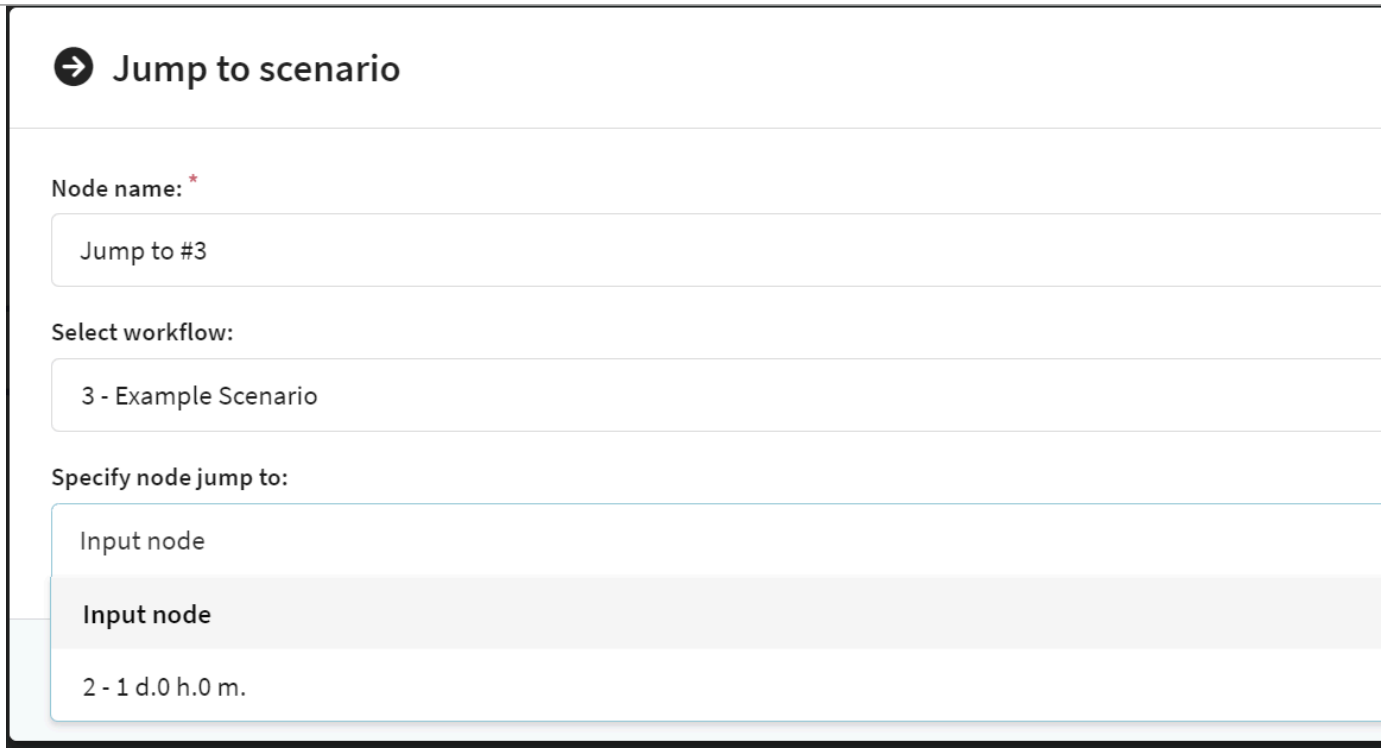
tickets_info.[0].ticket_id
```

Jump to scenario

![img] (/img/attachments/9407058/jump_node.png)	**Sends the profile to the beginning of another scenario or to a certain node of another scenario. In this case, the current scenari same database.If the chain in the scenario continues after the transition node, the profile moves to another scenario and at the**

Read more

Enter the name of the node. Select the scenario to which you want the profile to be sent. If you want the another scenario, select input node as the node to jump to. If you want the profile to go immediately to a select that node from the list.



Distribution node

![img] (/img/attachments/9407058/distribution_node.png)	Creates multiple elements that separate scenario options for customer profiles regardless of any conditions. Both brute f

Read more

General settings

Select the distribution principle and the number of parts into which the audience will be distributed. You editing the distribution elements.

Distribution by round-robin

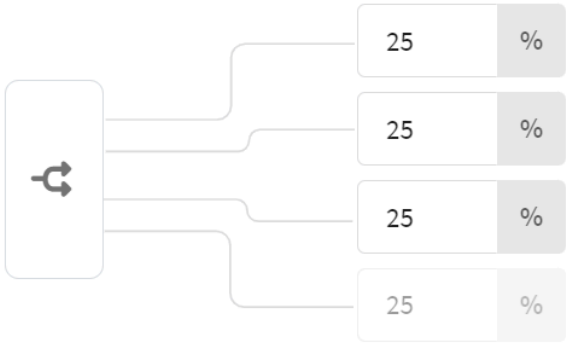
Profile distribution takes place at the moment of hitting the scenario node according to the specified pro mechanism of Smooth Weighted round-robin balancing.

Client distribution principle:

Round-robin

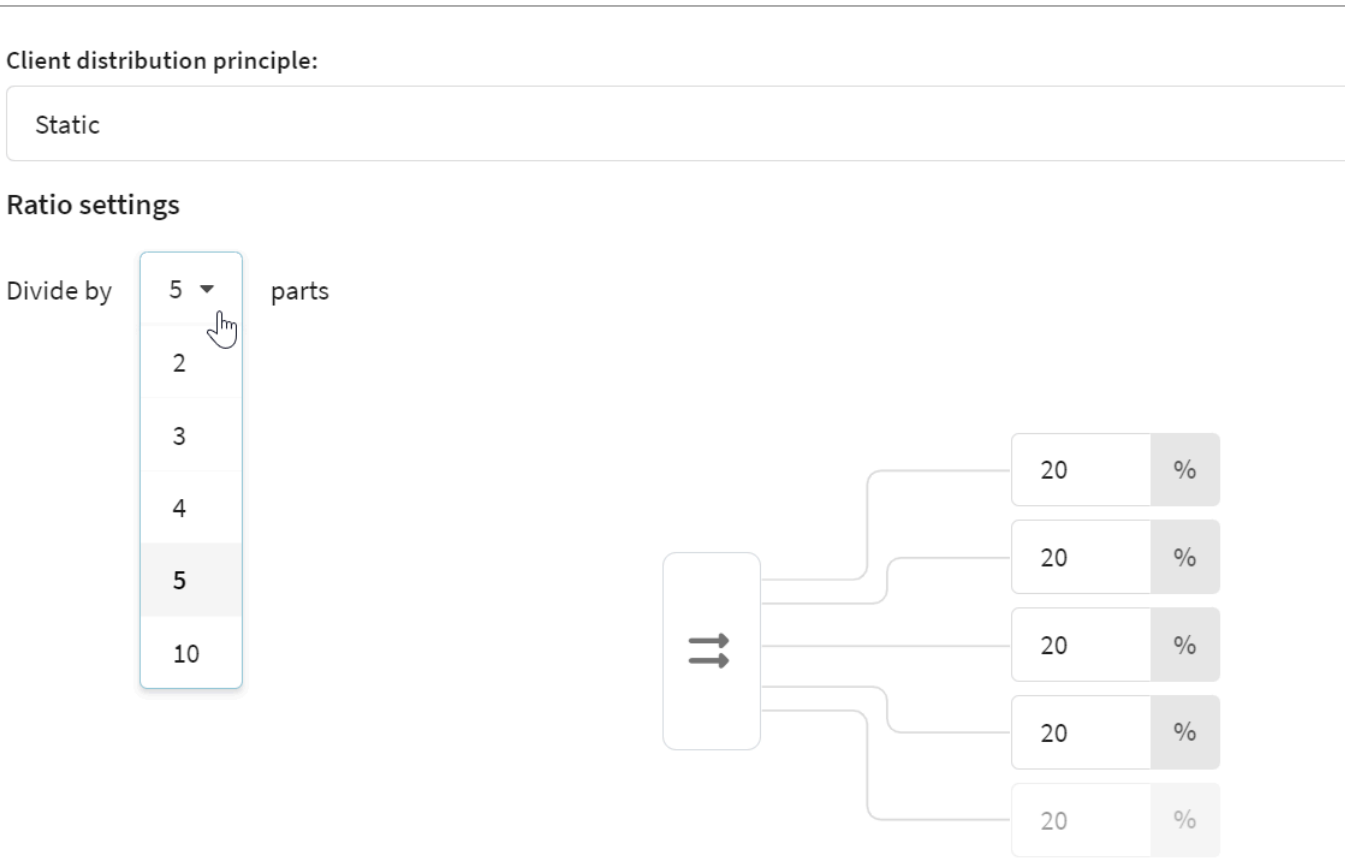
Ratio settings

Divide by 4 parts



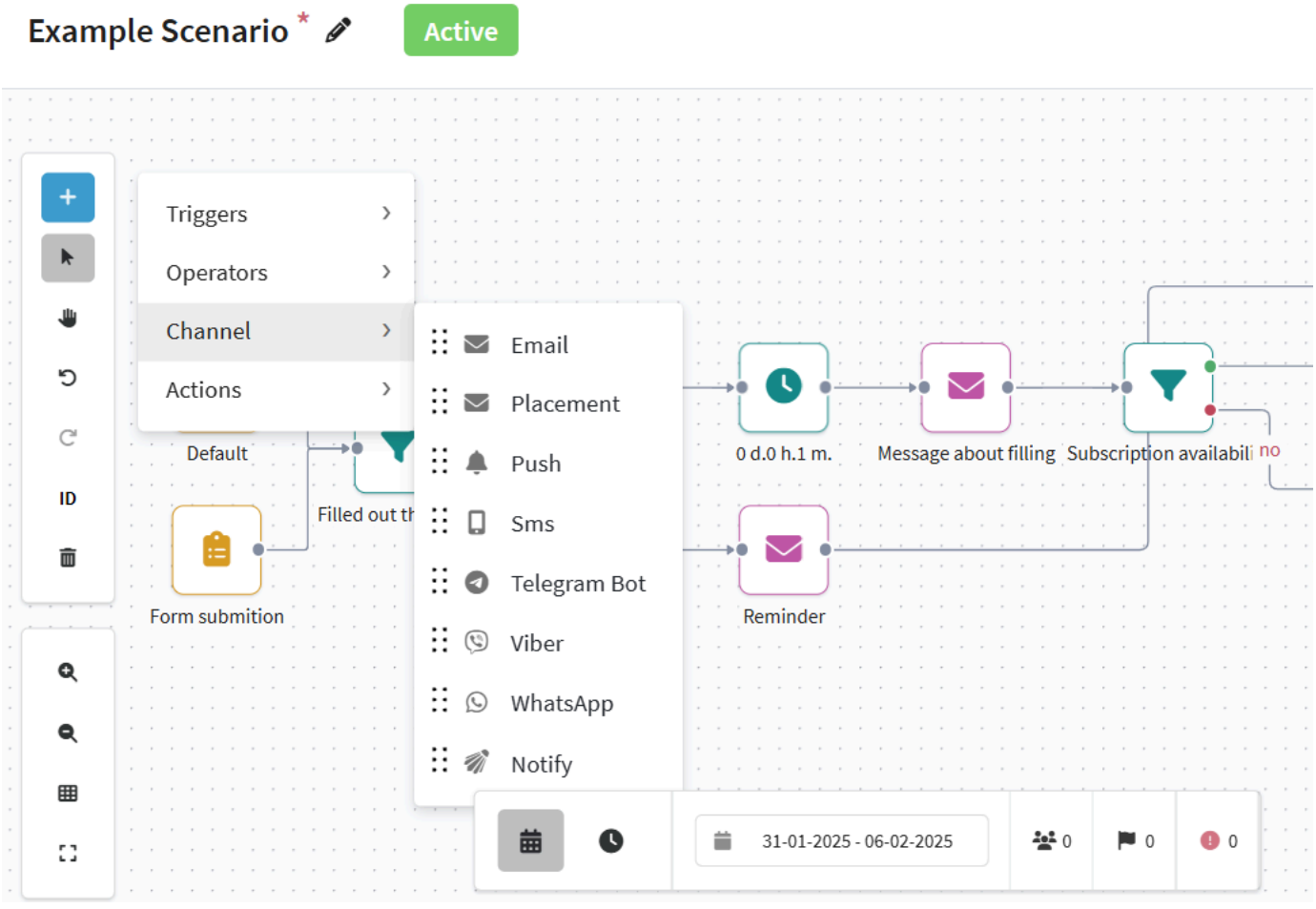
Static distribution

When using static distribution, the profile will always fall into the same distribution percentage, which c
is not recommended for a small number of profiles (up to 10,000).



Channels

Elements in this category launch [a trigger mailing]through a specific channel. A new mailing is created d



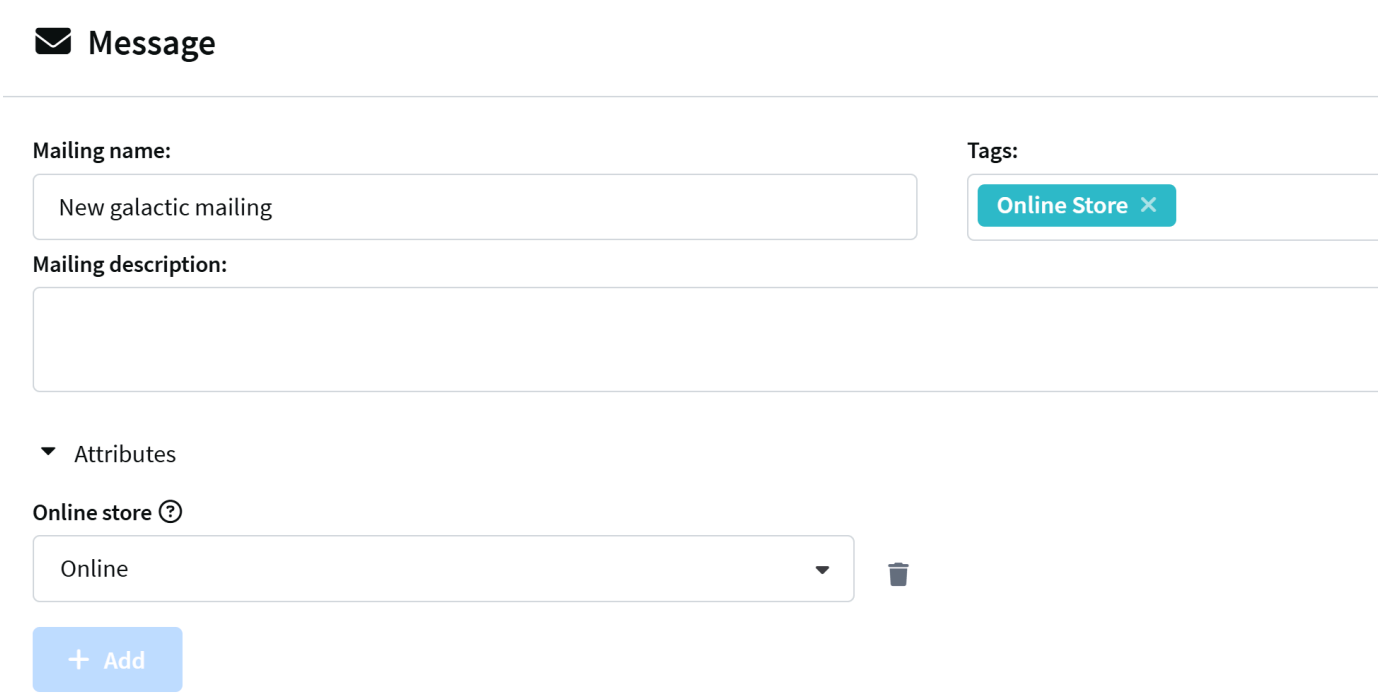
TIP

The Placement node allows you to launch a placement mailing in a scenario. Setting up such a mailing i

Read more

General settings

Enter a name for the new mailing, add a description if necessary. Here you can also set custom tags and

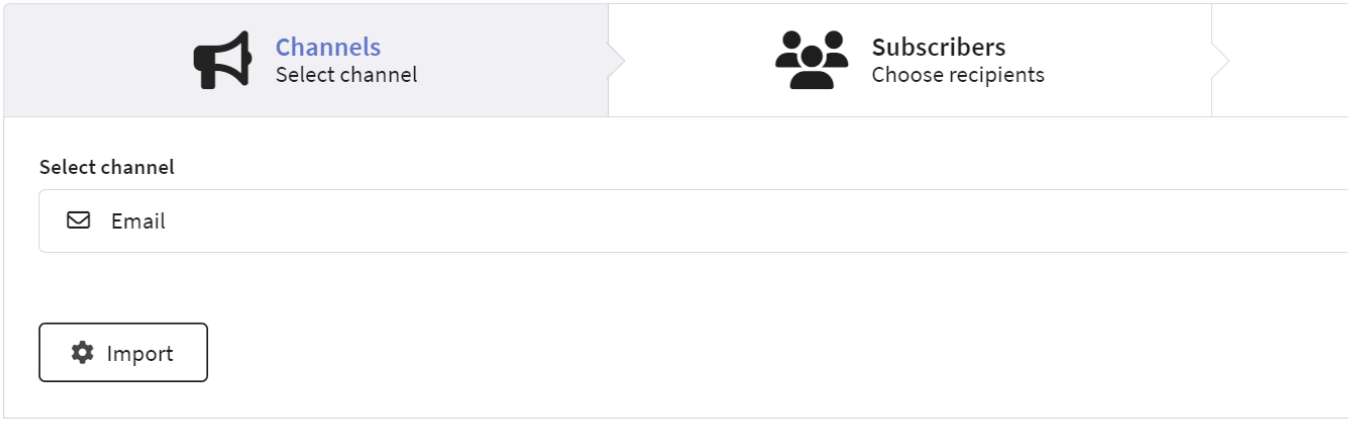


INFO

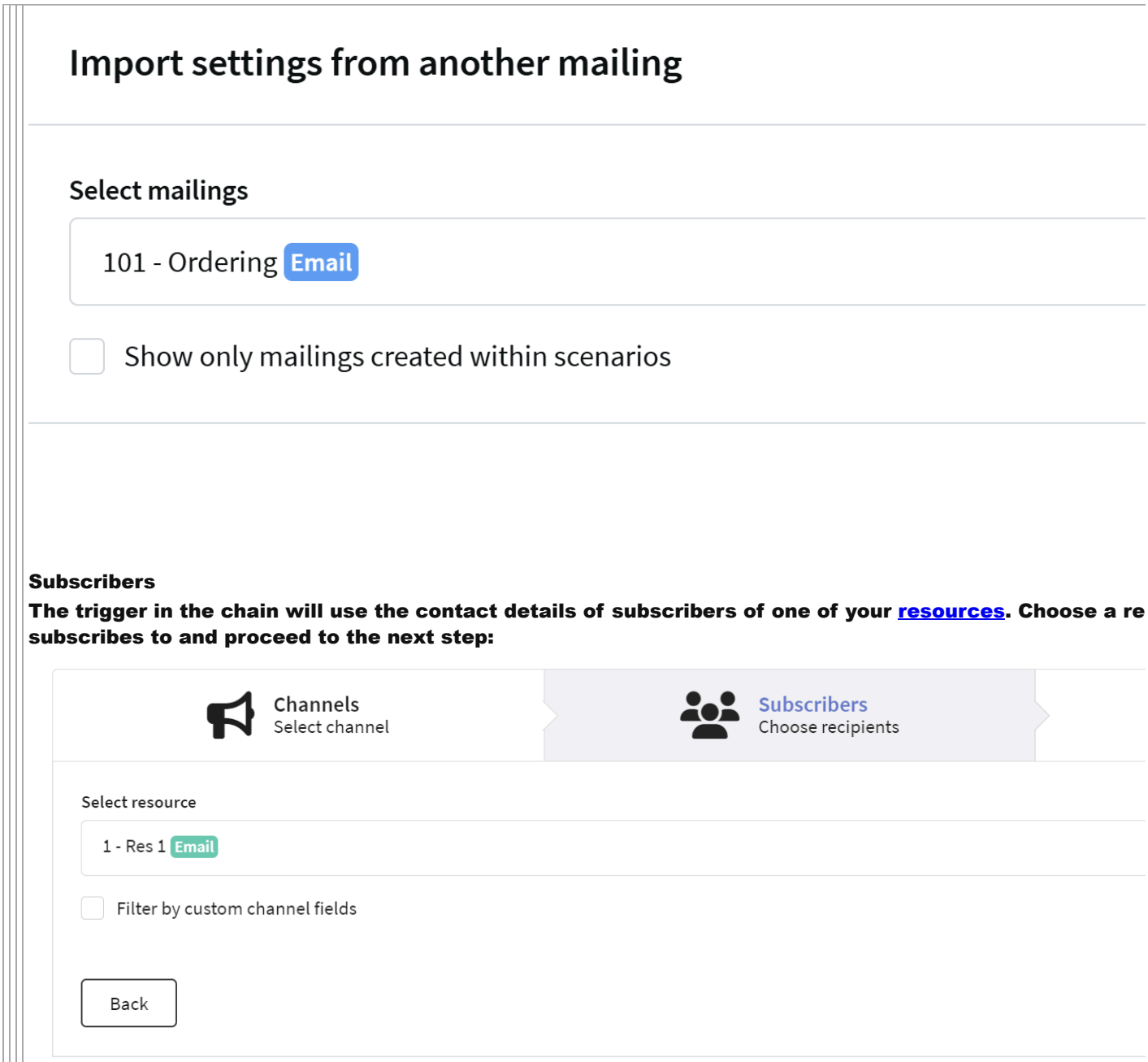
If the scenario is marked as high-priority, priority sending is automatically enabled in all of its nodes, ar turn it off manually. The toggle value is not overwritten: as soon as the scenario is no longer high-priorit restored.

Mailing settings

On the first settings tab, select a communication channel: each mailing can only be sent via one channel



On the same tab you can import an existing mailing into the node. This can be a single mailing or a maili



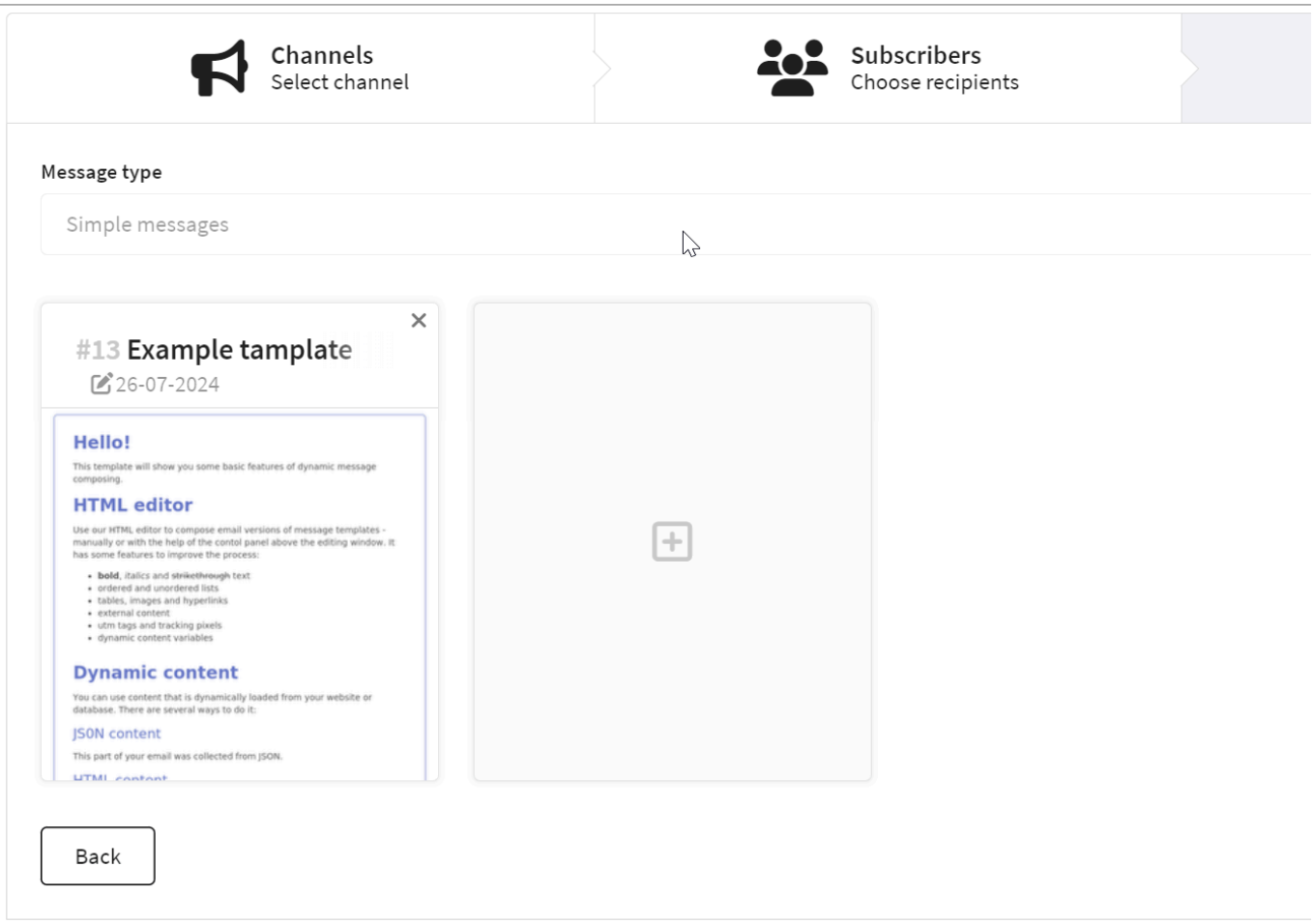
Subscribers

The trigger in the chain will use the contact details of subscribers of one of your resources. Choose a re subscribes to and proceed to the next step:

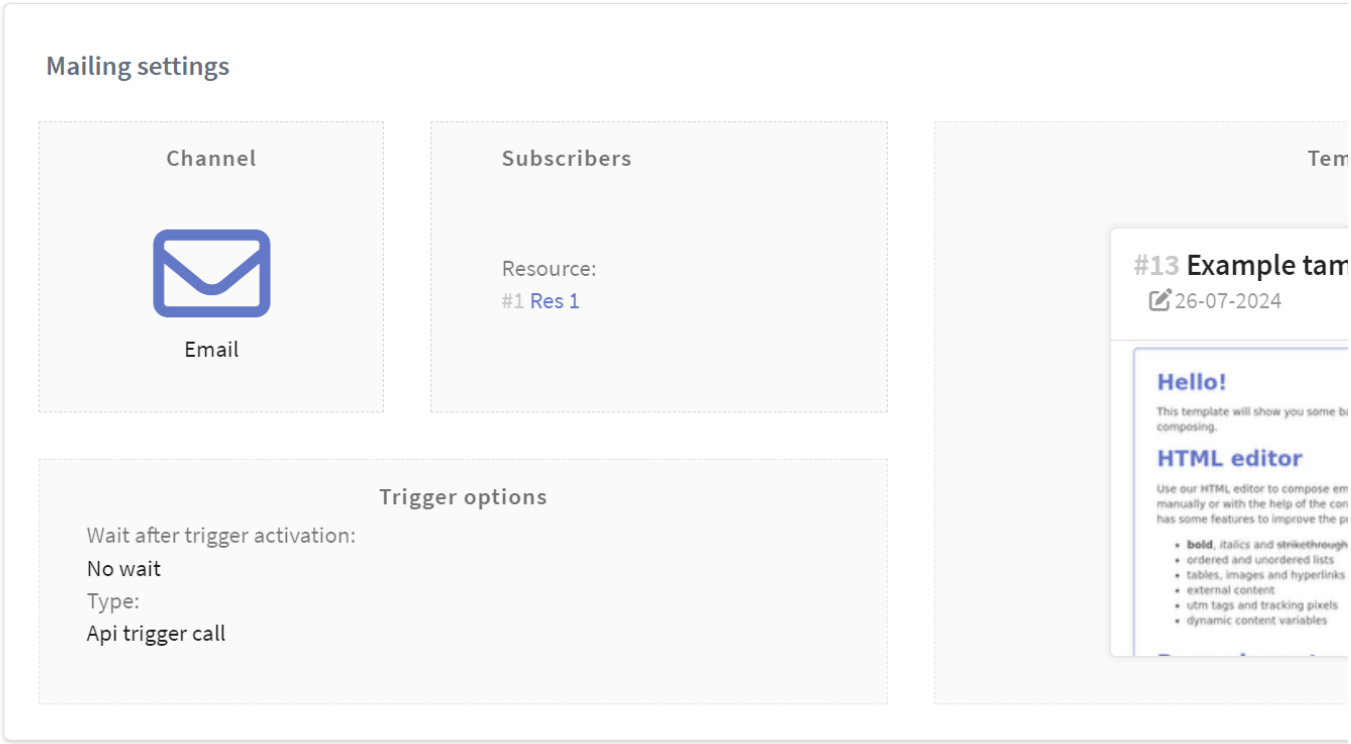
Additionally, you can filter by custom channel fields. To create a filter by fields, select the standard or op "Selection condition" field, select a logical operator and, if necessary, specify a value.

Content

On the third tab, select Message Template that you want to send to subscribers. It is possible to select s templates in one mailing). In this case they will be sent in random order. Just like when creating a mailin desired template and edit it, if necessary:

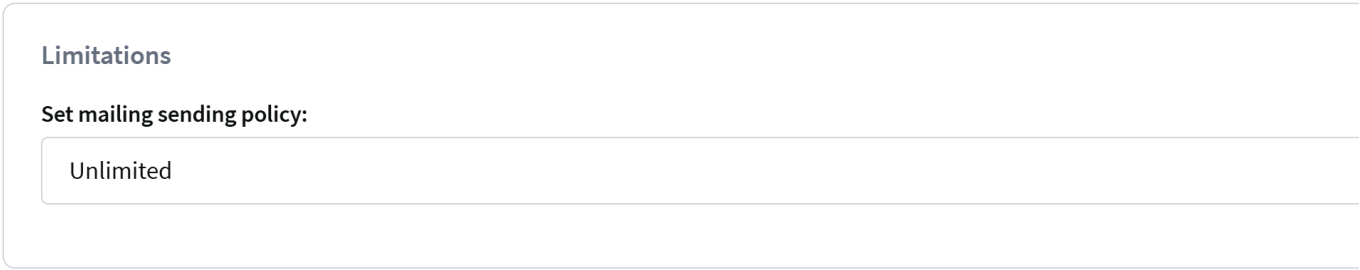


After selecting the template, click the “Finish” button. The mailing settings will be displayed in one tab, changes at any time by clicking the edit button:



Mailing Limitations

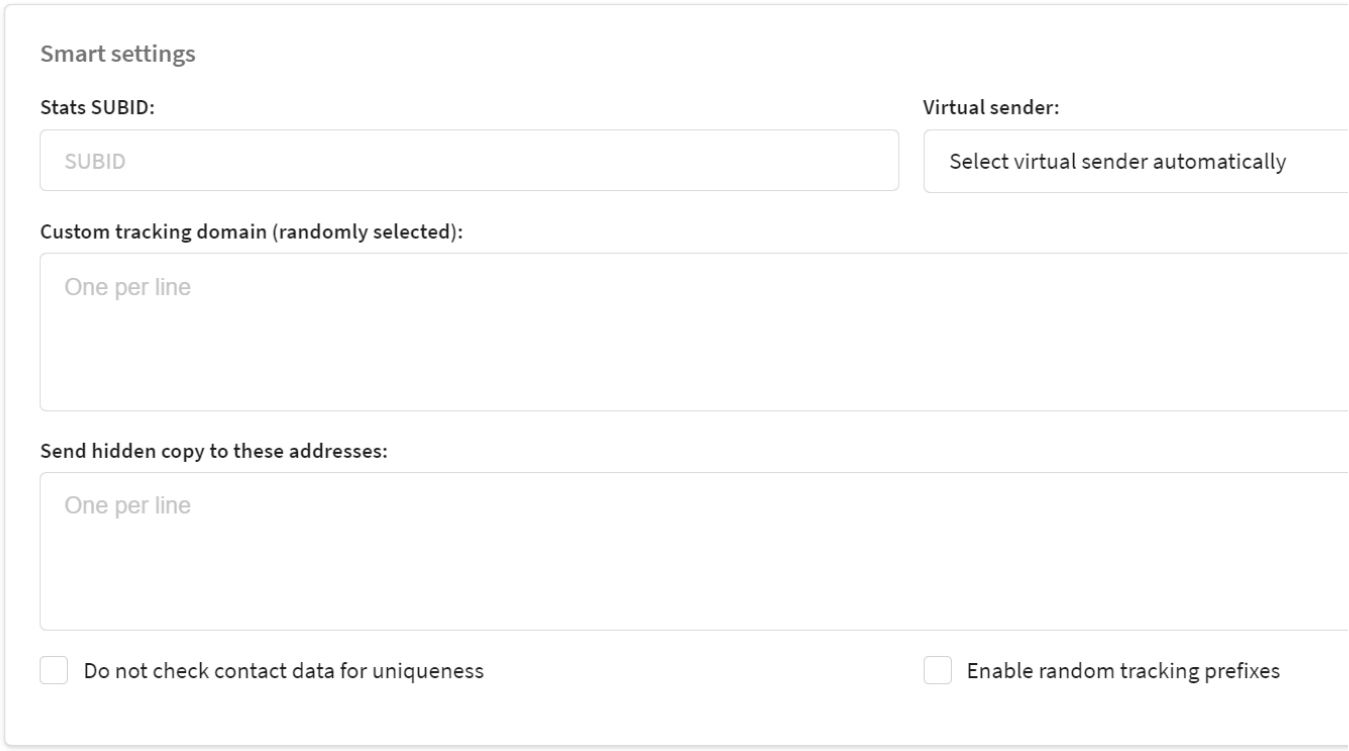
In this opening block, you can set send policy of the mailing. By default, the policy of the selected resou applied:



Smart settings

In smart settings, you can additionally customize the mailing:

- **SUBID statistics**
- **Selecting a virtual sender**
- **Custom domains for event tracking (in addition to the one configured in the resource)**
- **List of addresses for hidden copy to sent e-mails**
- **Other options depending on the platform configuration**



CAUTION

If there is an error when sending a mailing, the profile will continue to follow the scenario.

Actions

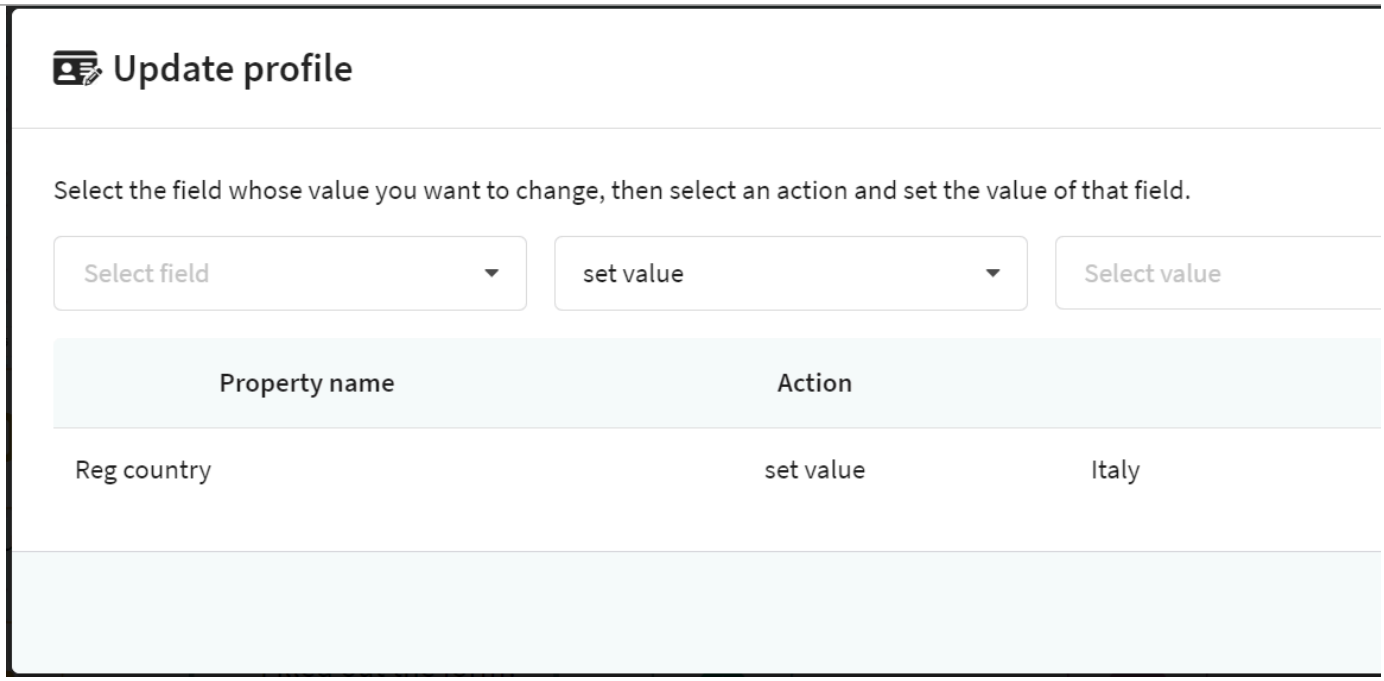
Update profile

![img] (/img/attachments/9407058/update_profile_node.png)	**Updates the data of selected fields of the subscriber profile or deletes values in these fields. Related actions are available for different field types. The node can add tags, change field values with selectable optio**

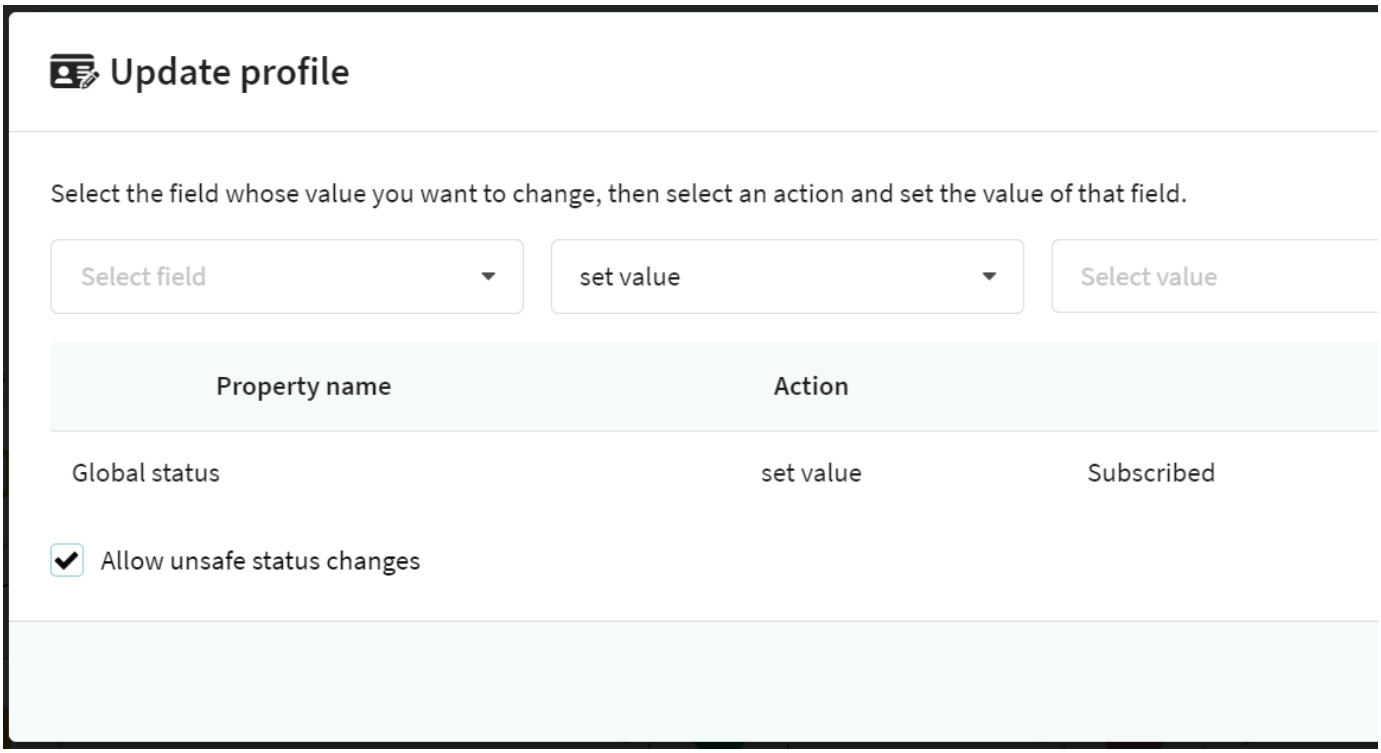
Read more

From the drop-down list, select the field you want to update. Then select the action and set the value of field can be edited by clicking on the pencil icon, as well as deleted from the node.

Action	Field type	Description
Set value	**All**	**The new value of the field is specified manually. For fields of the List (Enum) type, the value is changed to one of those specified in the l Boolean type the value is changed to Yes or No.**
Remove value	**All**	**Deletes the value from the selected field.**
Take the value from the field	**All**	**The value is taken from the `apicontent` field. It is used together with "API call" node, which is able to save the request result into `apiconte`**
Increase by	**Integer**	**Increases or decreases the value. To decrease a value, you must enter a negative number.**
Add element	**Tags**	**Write another tag in the field.**
Remove element	**Tags**	**Remove one of the existing tags from the field.**



With this node you can also change the global status of a profile. If you allow unsafe status changes, the statuses other than “Subscribed”.



INFO

When referring to the sequence number of an array element from `apicontent`, the % sign must be used. C zero. For example, we have the following object with an array:

```
{
    "results": [
        {
            "objectId": "12asd34",
            "name": "Company 1"
        }
    ]
}
```

In order to get the name of the first organization, we need to use the following query `res.results.%0.name`, response from the API Request node. Please note that queries do not support JSONPath.

Example of correct query structure: `resp.data.data.%0.profile._region` Example of incorrect query structur `"$.data[0].profile._region" "item")}`

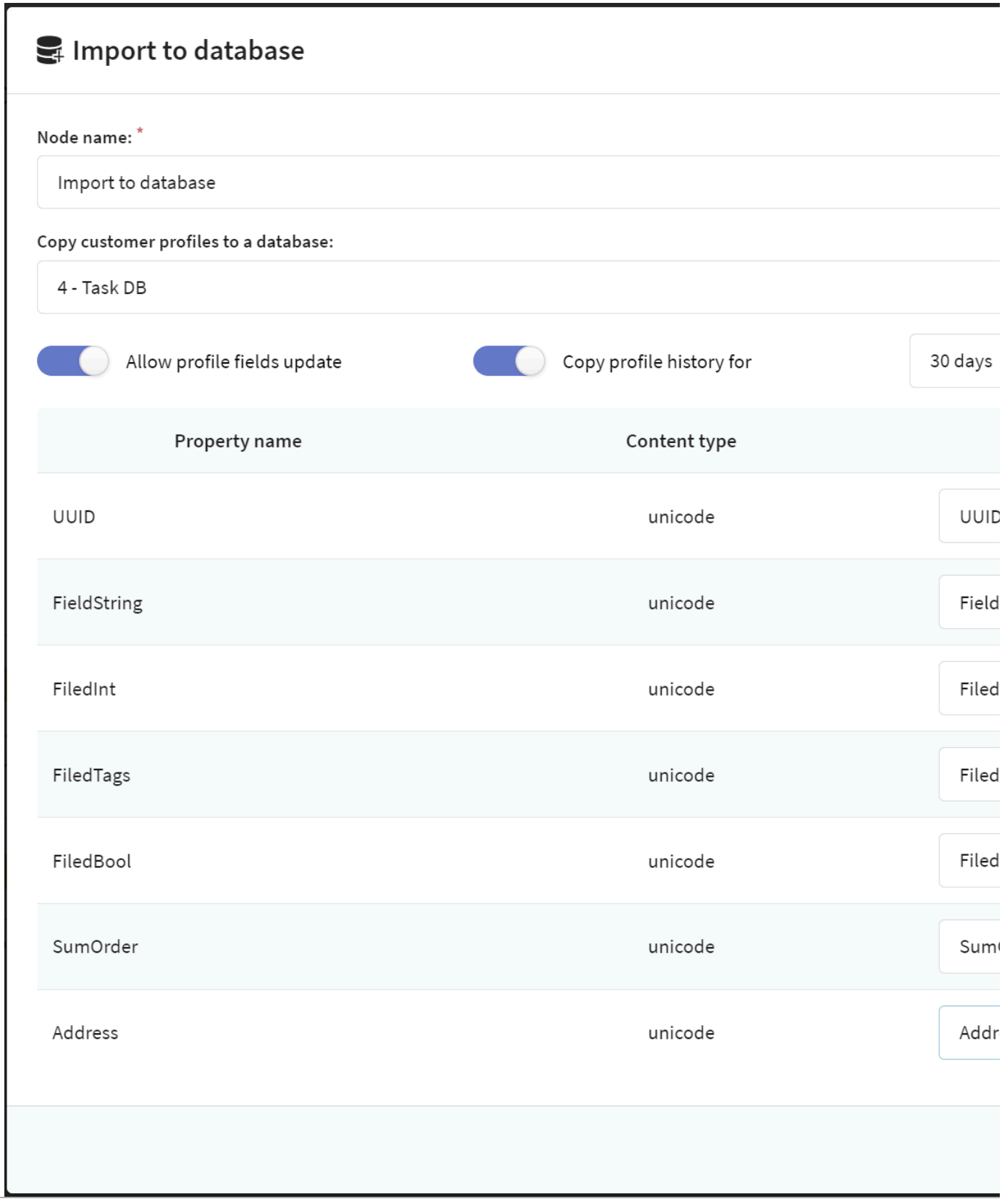
Import to database

![img] (/img/attachments/9407058/db_import_node.png)	**Copies a subscriber's profile from the profile database for which the scenario was created to another one. You can also cop profile fields.**

Read more

Enter the name of the scenario node and select the profile database to which you want to copy the custo the data of a profile that already exists in the target database - activate the "Update subscriber fields if additional fields in the source database, their data can be written to fields of the appropriate type of the customize the field matching.

You can also transfer the profile history from the source database to the target database. To do this, use and select the period of data you want to transfer. Profile history includes mailing actions (sends, opens, imports, pixels, promotion code assignments, and participation in static segments.



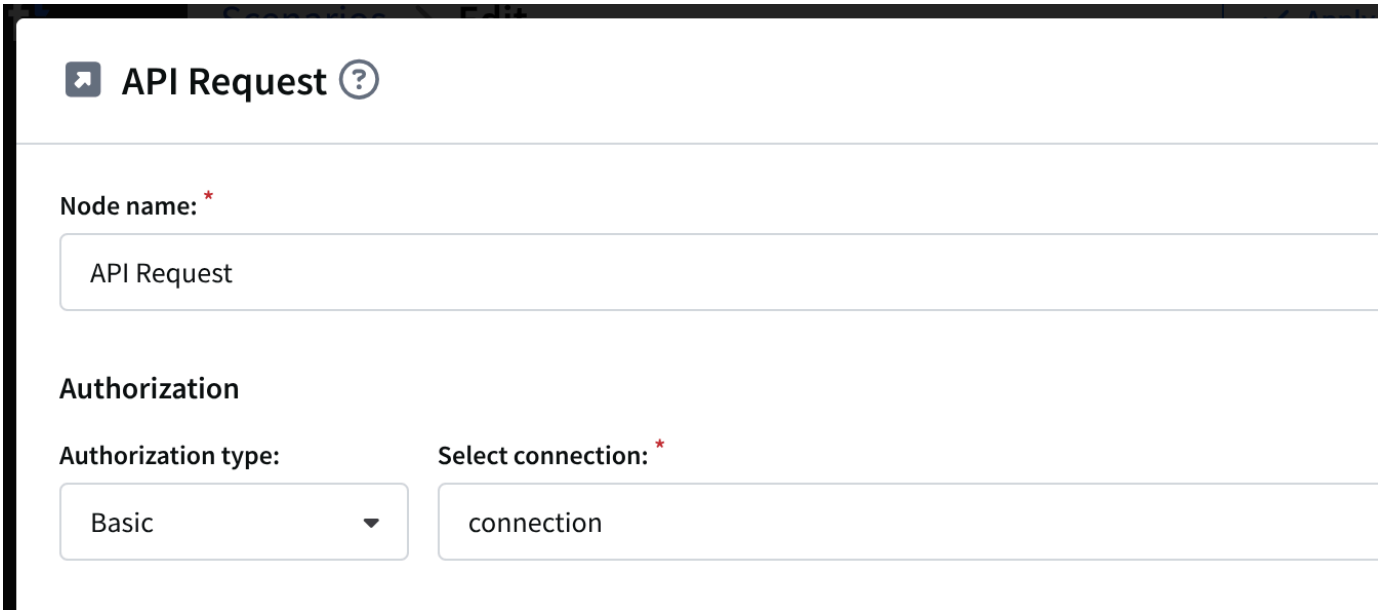
API request

![img] (/img/attachments/9407058/api_call_node.png)	Send a GET or POST request to another system. You can add parameters to the request with static data or taken from a subs

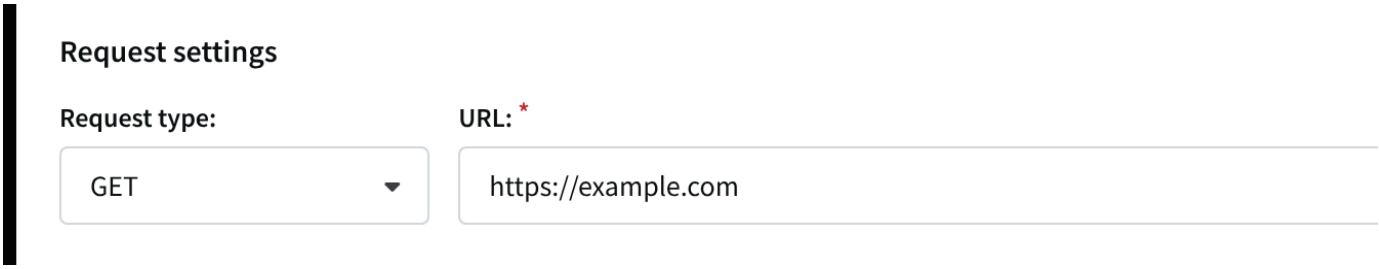
Read more

Enter the name of the node.

Select the required authorization type. This is necessary in order to transfer profile data to external serv authorization. You can first link the external service to the Altcraft platform via the connection settings, this menu:



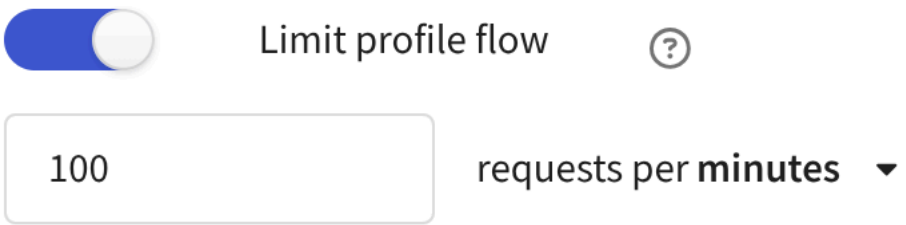
Select type of request to the external system. In the URL field enter the address of the service receiving



The request URL also uses variables. Click `</>` or enter a value manually. Example of using a promo cod

```
http://www.example.org/api/?Promocode={loyalty.black_friday.promocode}
```

The user can limit the number of profiles passing through this node within a specific time period. This is operation of the scenario without overloads. To do this, enable the corresponding toggle, specify the nur interval (minute, second, or hour):



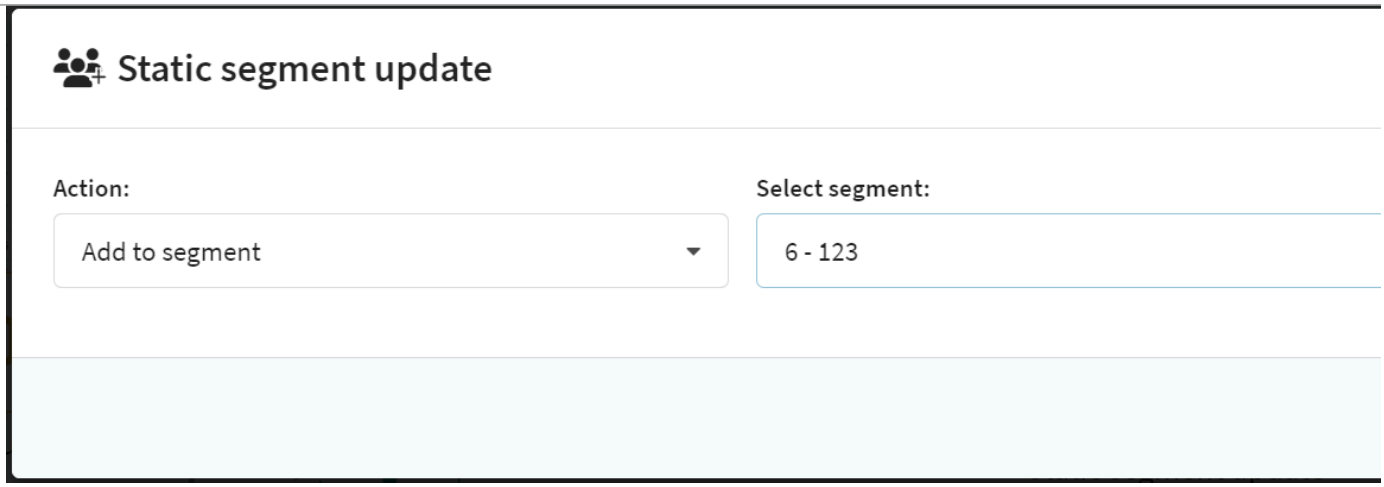
▶ ****Rate Limiter Mechanics****

Static segment update

![img] (/img/attachments/9407058/update_stat_segment_node.png)	Adds a customer profile to the static segment of the audience or removes it from the segment.

Read more

In the node settings, select add or delete and the available static segment:



Subscriptions

![img] (/img/attachments/9407058/add_subscription_node.png)	Subscribes a client to a specific resource or updates existing subscription data. Profile data or subscription data for

Read more

The system uses two types of subscription channels:

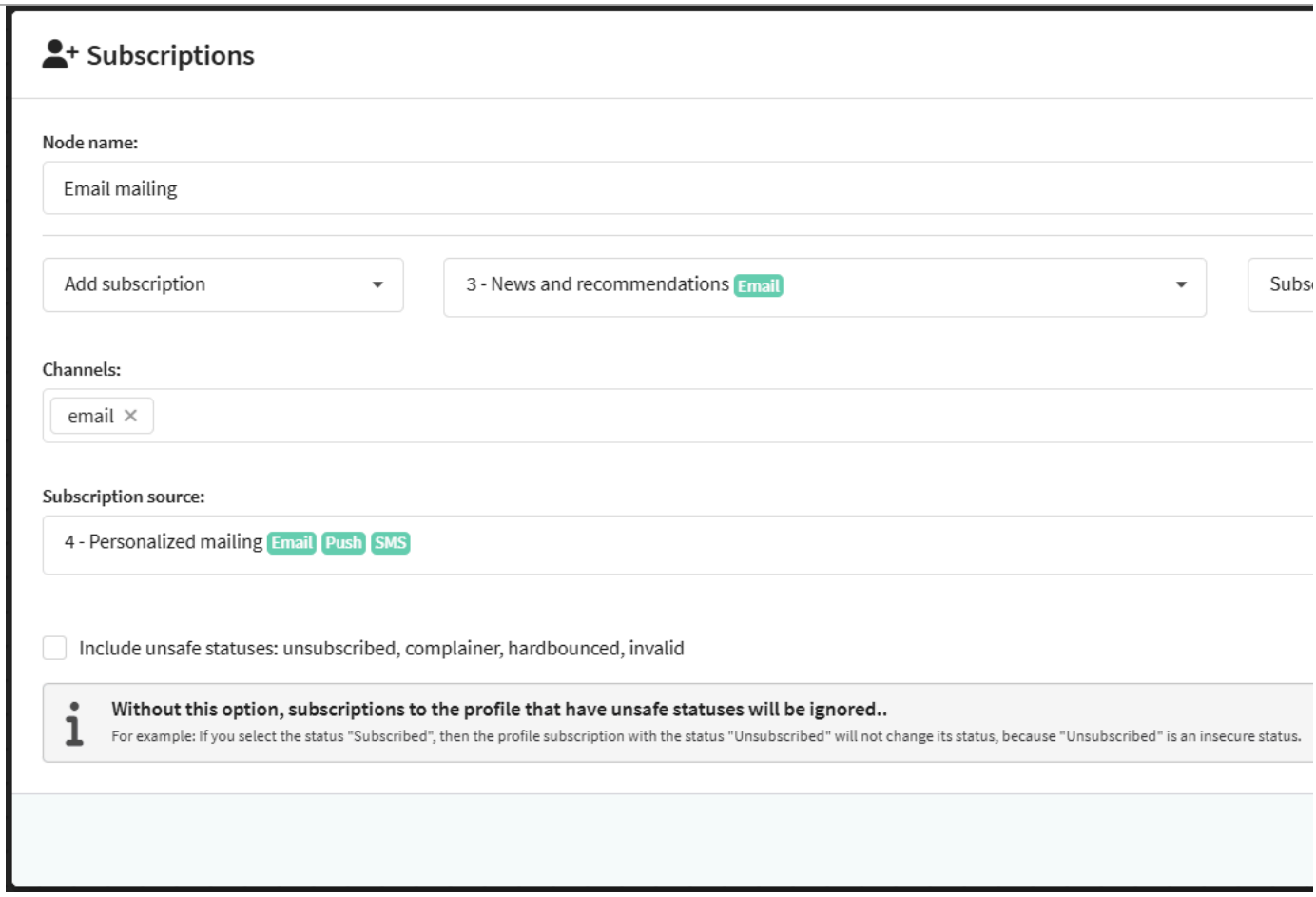
- ****Indirect channel** — sends a message to an unspecified number of users. It is used only for sendi Group-channel.**
- ****Direct channel** — sends a personalized message to a specific recipient.**

Add and update subscription

- **A subscription can be **added** if the client is not already subscribed to the resource.**
- **You can **update subscription**. For example, when you are confirming a subscription or while usin**

How to add or update subscription to a resource:

1. **Enter node name.**
2. **Click Add subscription or Update.**
3. **Specify the resource from the drop-down list.**
4. **Change the subscription status if it is required.**
5. **Select the available channels.**
6. **Use profile data (e.g. Email, Phone) or subscription data from another resource as the subscription s**



TIP

You can only subscribe to Push notifications if you select a resource for which you have already configu source.

Promocode

![img] (/img/attachments/9407058/promo_code_node.png)	Assigns a promocode from the selected loyalty program to the customer profile. The promocode can then be sent to the or used to calculate personalized offers.

Read more

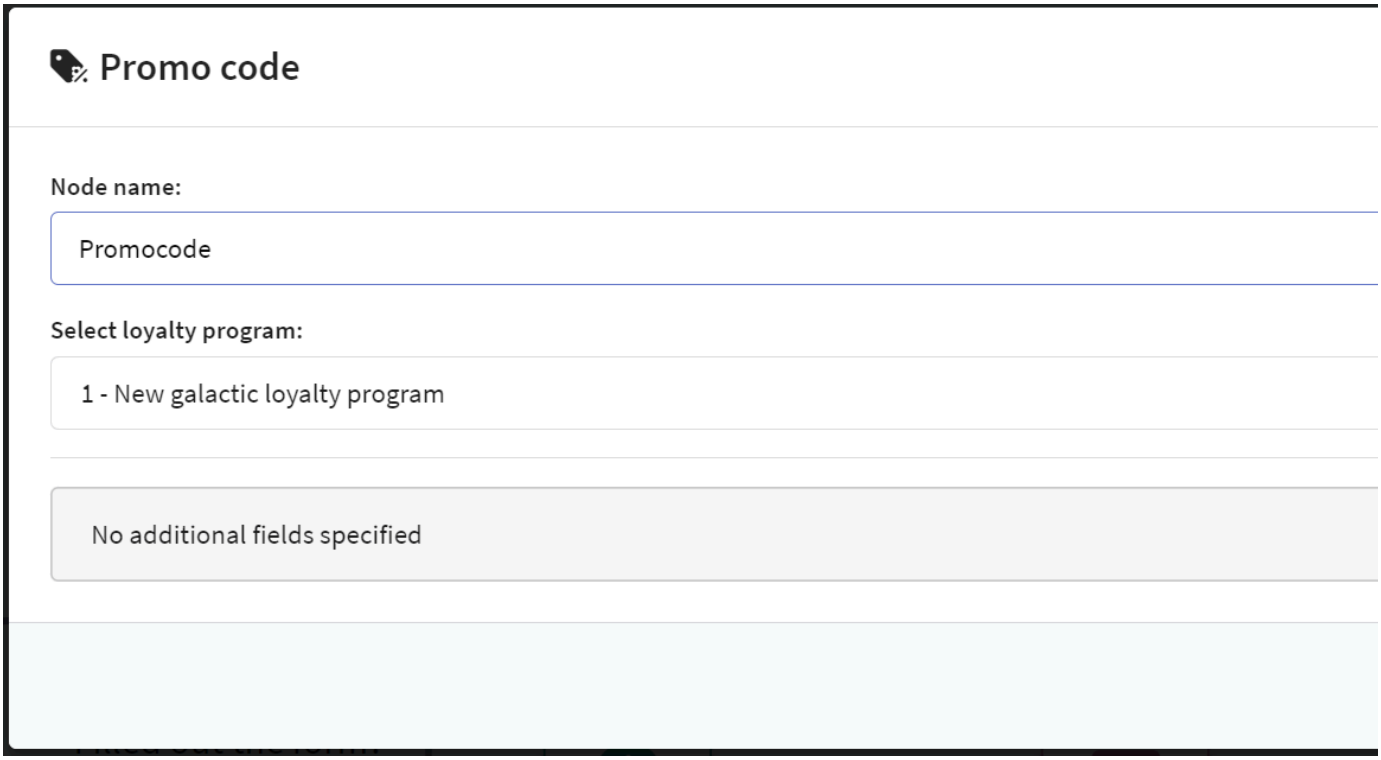
Enter the name of the scenario node and select the required loyalty program. The program must generat or the option of creating promocodes by scenario must be active.

CAUTION

If the profile already has a promocode for this loyalty program, the scenario will be fulfilled.

If a promocode cannot be assigned for one of the reasons below, the scenario will end on that node:

- **promocode limit has been reached**
- **loyalty program is not active**
- **the loyalty program has no free promocodes, and its option "The scenario can create promocodes i disabled**



If the loyalty program has custom fields configured, the default values specified when creating the prom scenario assigns the promocode.

For fields of type “Text” and “Number”, you can define a list of options. Each option consists of a Value ((unique identifier saved in the promocode data). When the scenario assigns a promocode, the value sele promocode is used. For more information on configuring custom fields and options, see Promotion codes

Goal registration

![img] (/img/attachments/9407058/goal_registration_node.png)	Records in the client profile the achievement of a certain goal and the associated cost of the action. Can be used fo on conversion and profit.

Read more

Enter name of node and pixel to which the target will be bound. Registration of identical targets for diffe programs or for A/B testing of the interface of your site or other resources.

Below, select existing targets or add new ones. Specify the value to be transferred for the targets.

TIP

The value for ALL targets will be written as a single value. If multiple targets with different values are t elements - one after the other, with different targets.

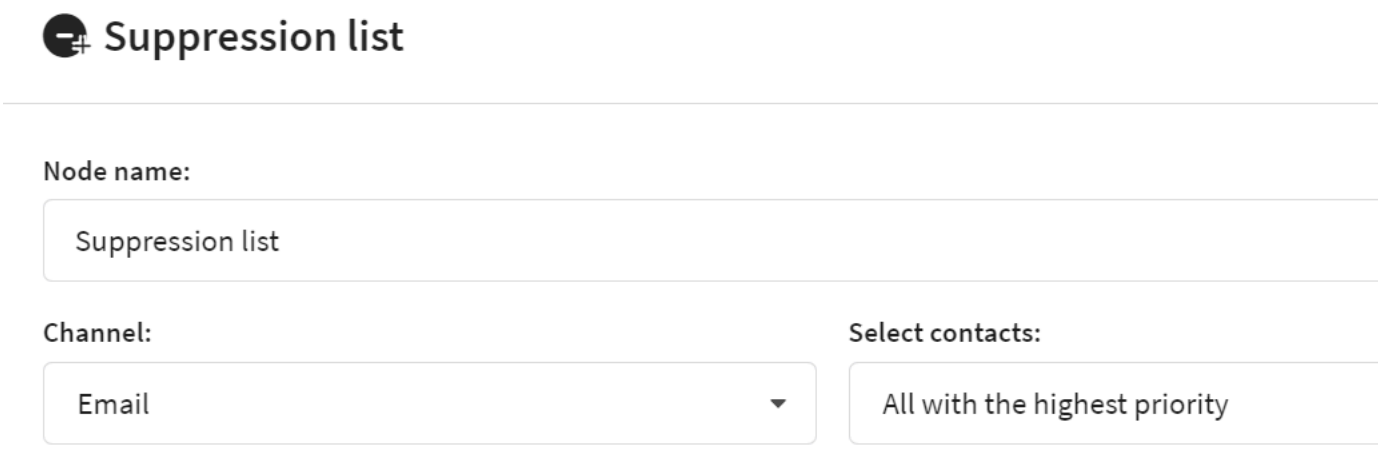
If necessary, you can pass variables related to pixel trigger data inside templates. You can learn more al

Suppression list

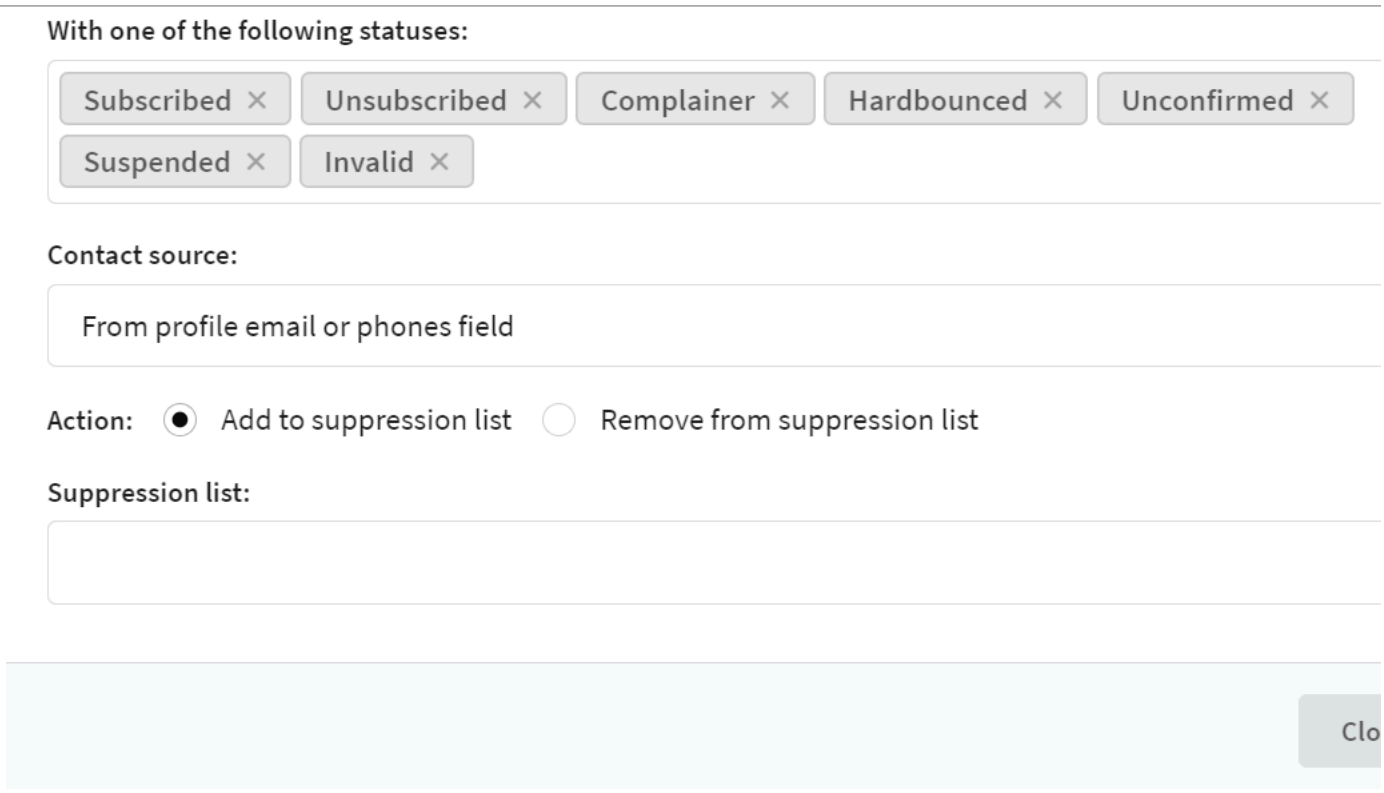
![img] (/img/attachments/9407058/suppress_list_node.png)	**Adds certain client contact details (email address and phone number) to the stop list of the platform. Communication w versa, removes contact details from the stop list, restoring the ability to communicate with the client.**

Read more

Enter the name of the node to be displayed in the scenario. Next, select which subscriptions contact inf restore. Select the communication channel and priority of subscriptions to change:



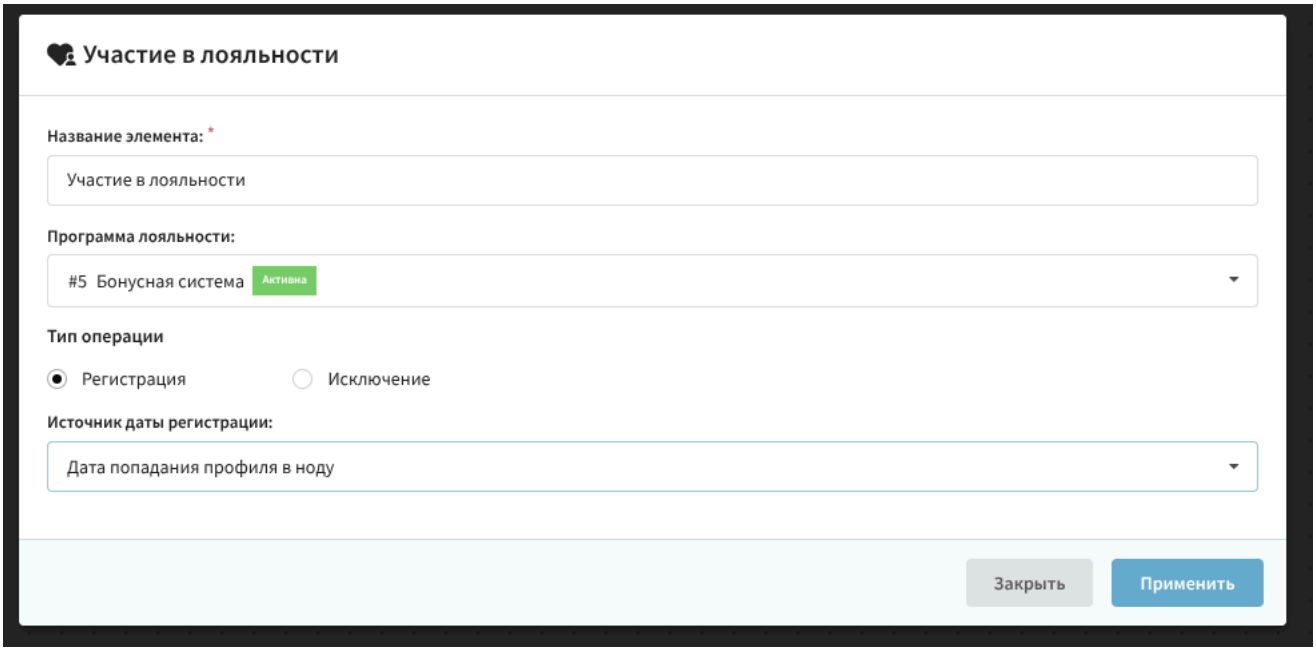
Select the resource and subscription statuses whose contact information you want to add to the suppre suppression list. At the bottom of the form, select the suppression to add or remove:



Loyalty participation

![img] (/img/attachments/9407058/loyalty-action.png)	**Registers or removes a user from a loyalty program.**

Read more



To configure the node, select a pre-created loyalty program and the operation to be performed on the pro the loyalty program.

When registering a profile, you can specify the source of the registration date in the loyalty program:

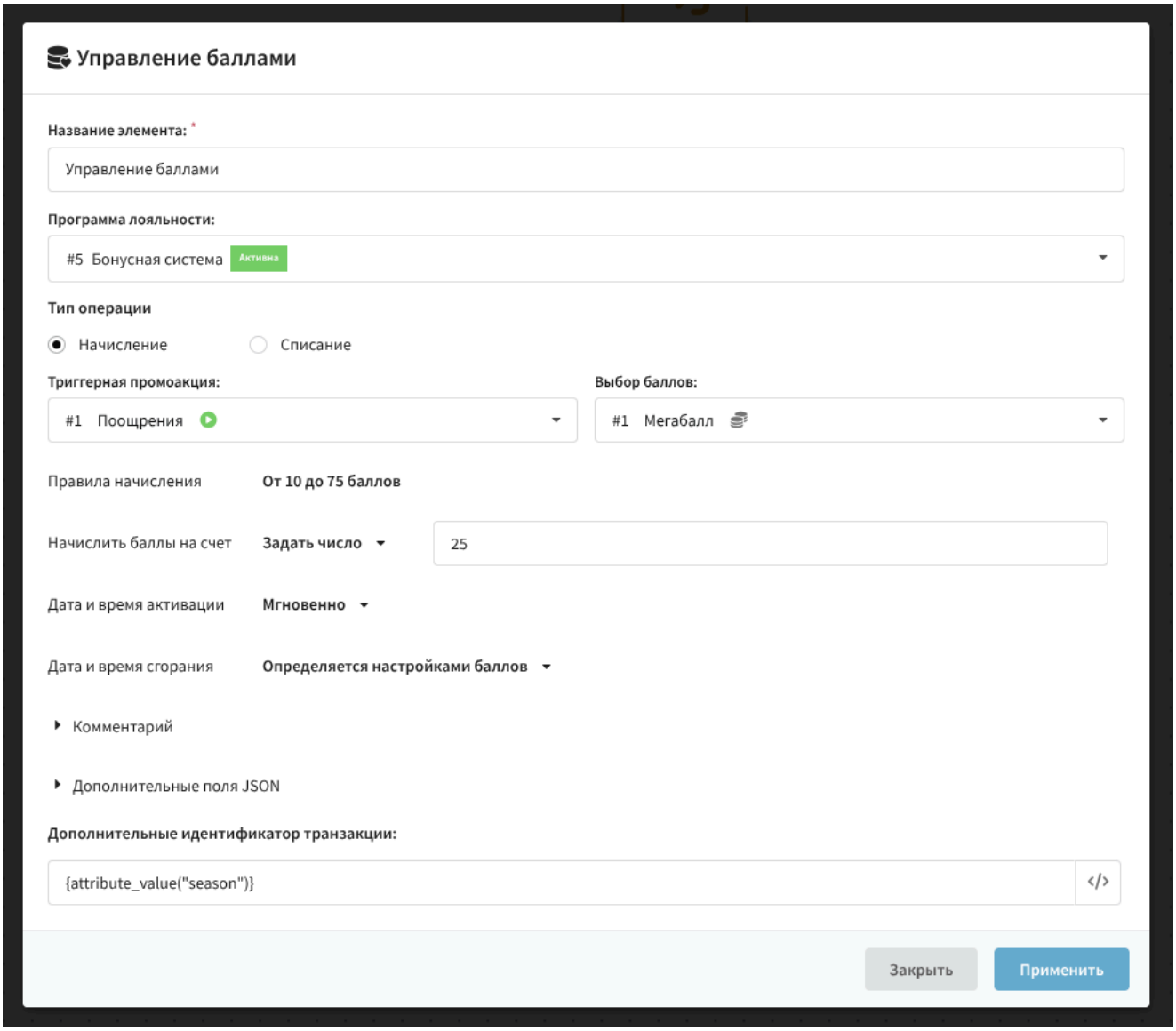
- **Profile entry into the node date — the time when the profile entered this scenario node;**
- **Date from profile field — the registration time is taken from the selected profile field with a date valu**
- **API content — the registration time is taken from the specified API field.**

For the last two options, two ways of handling invalid data are available: using the profile entry into the

Points management

![img](/img/attachments/9407058/loyalty-points-change.png)	**Changes the number of points on a profile in a loyalty program.**

Read more



To configure the node, select a pre-created loyalty program and the operation type — points accrual or p
Then specify the promotion under which the accrual is performed and the points type. The number of poi from a profile field, or passed via API content.
Additionally, you can specify:

- **Activation date and time (immediately, from a profile field, or from API content);**
- **Expiration date and time (according to points settings, from a profile field, or from API content);**
- **Transaction comment;**
- **Additional JSON fields;**
- **Additional transaction identifier.**

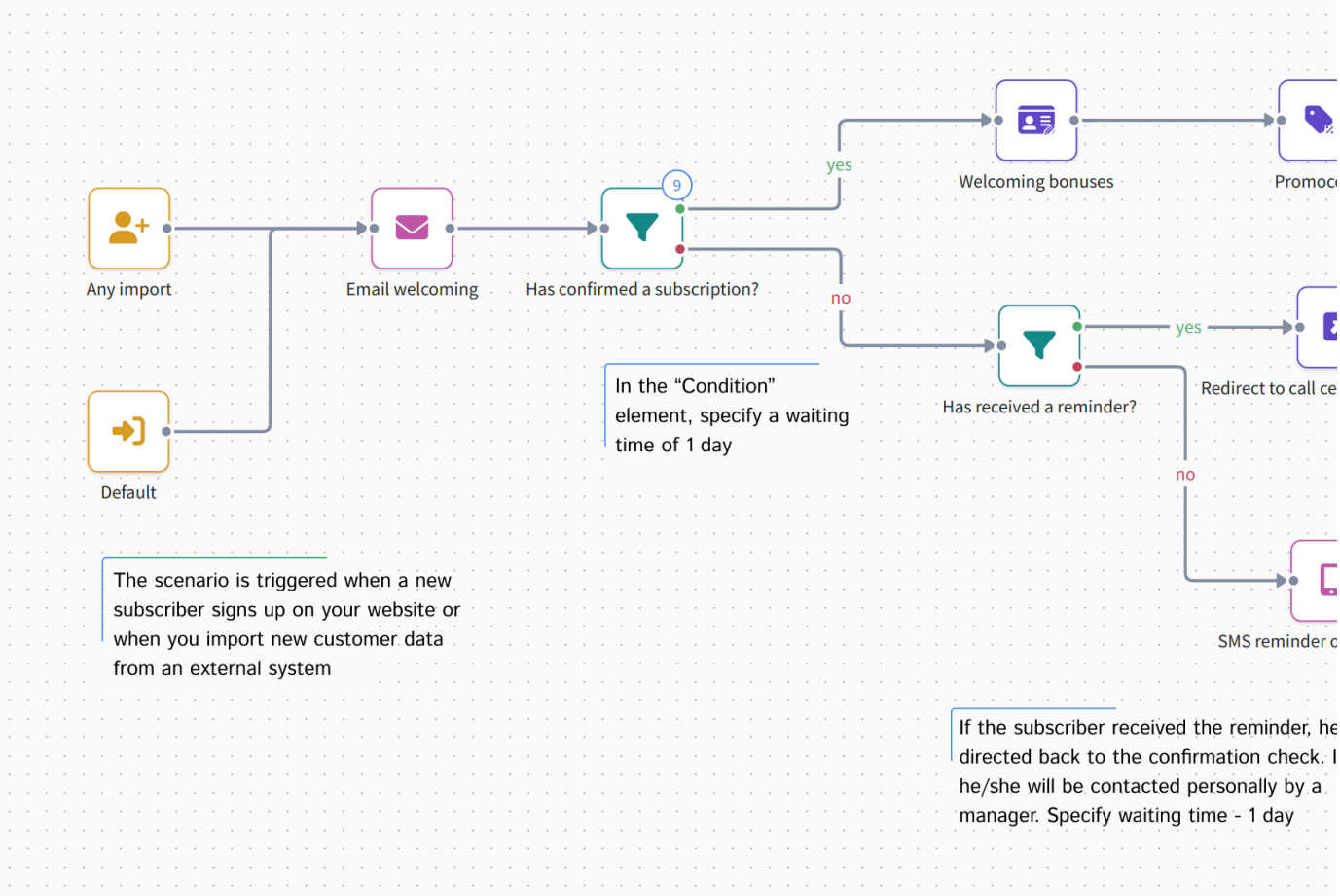
The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.

Classic marketing scenarios

Welcome workflow

Welcome workflows can start engaging your customers right after registration. A s displayed below will

- **automate opt-in process**
- **remind visitors of a dropped registration**
- **assign welcome bonuses and promotion codes**
- **redirect customers to your call centre or sales team for a direct call**



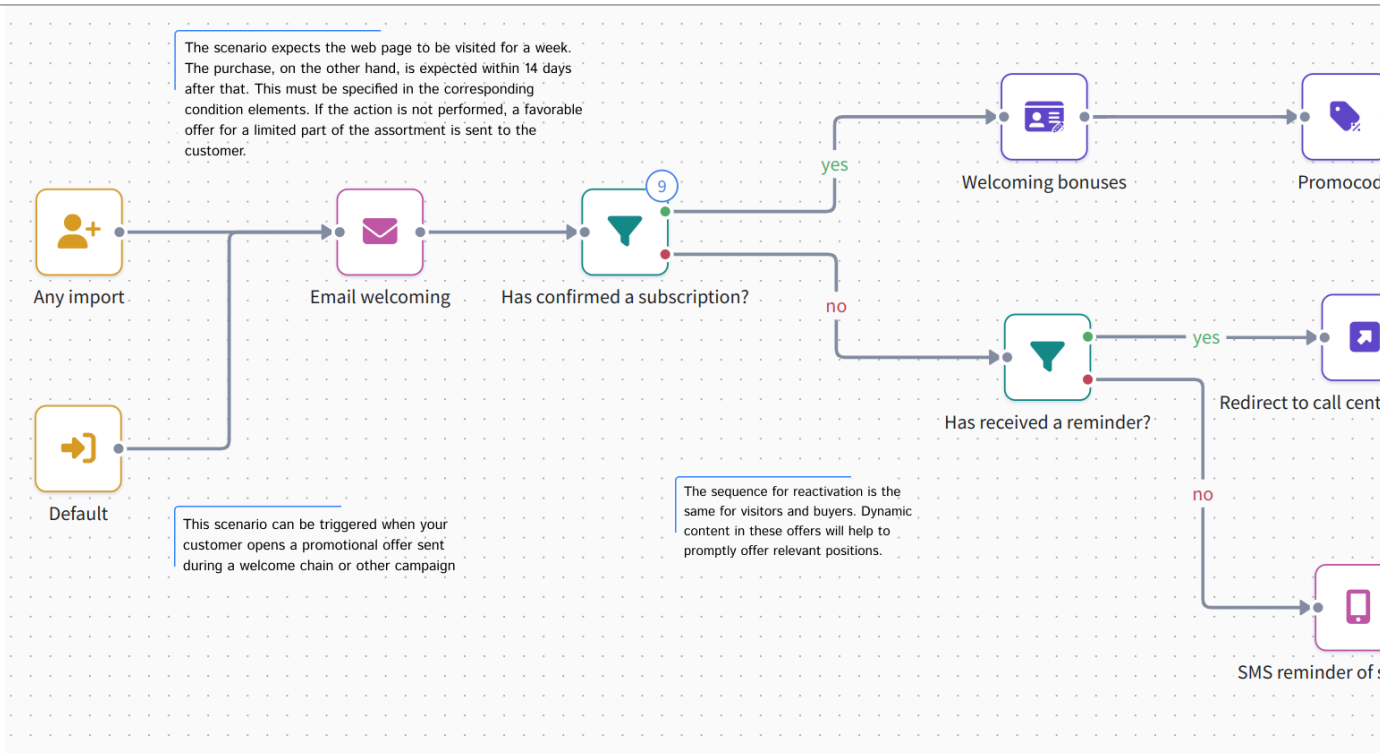
TIP

You will find a guide to creating a welcome scenario on this page.

On-boarding: first purchase

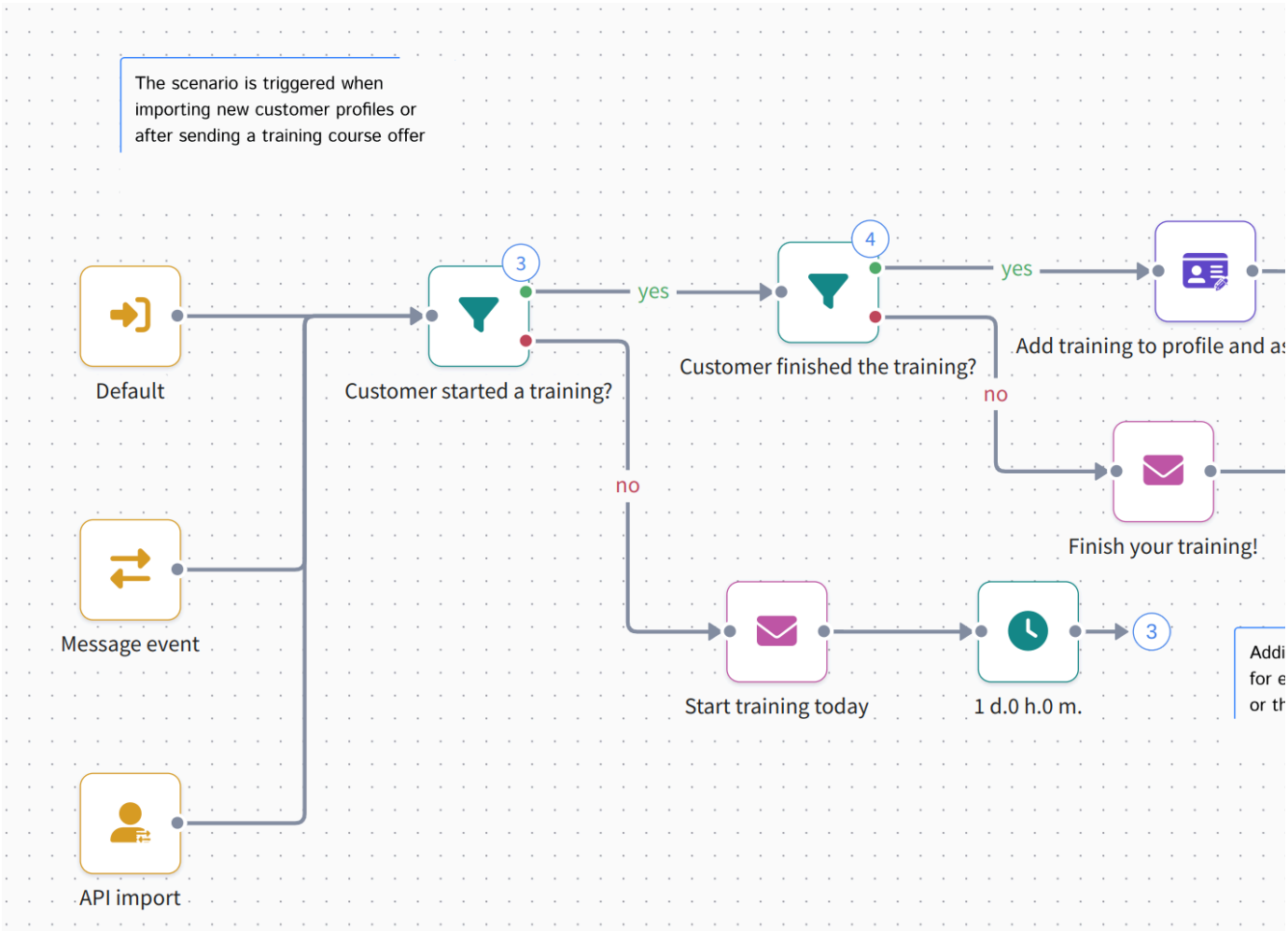
On-boarding workflows will help you engage customers from the moment of the first promotion they rece mailing about new product line you can encourage customer activity by a scenario similar to this one:

- **scenario launches when a customer interacts with your email**
- **reactivates customers who do not pay attention at first**
- **reactivates those who open store page and leave with no purchases**
- **assigns limited stock [promotion codes](/en/user-guide/loyalty-programs/) (for tripwires e.g.)**



On-boarding: customer education

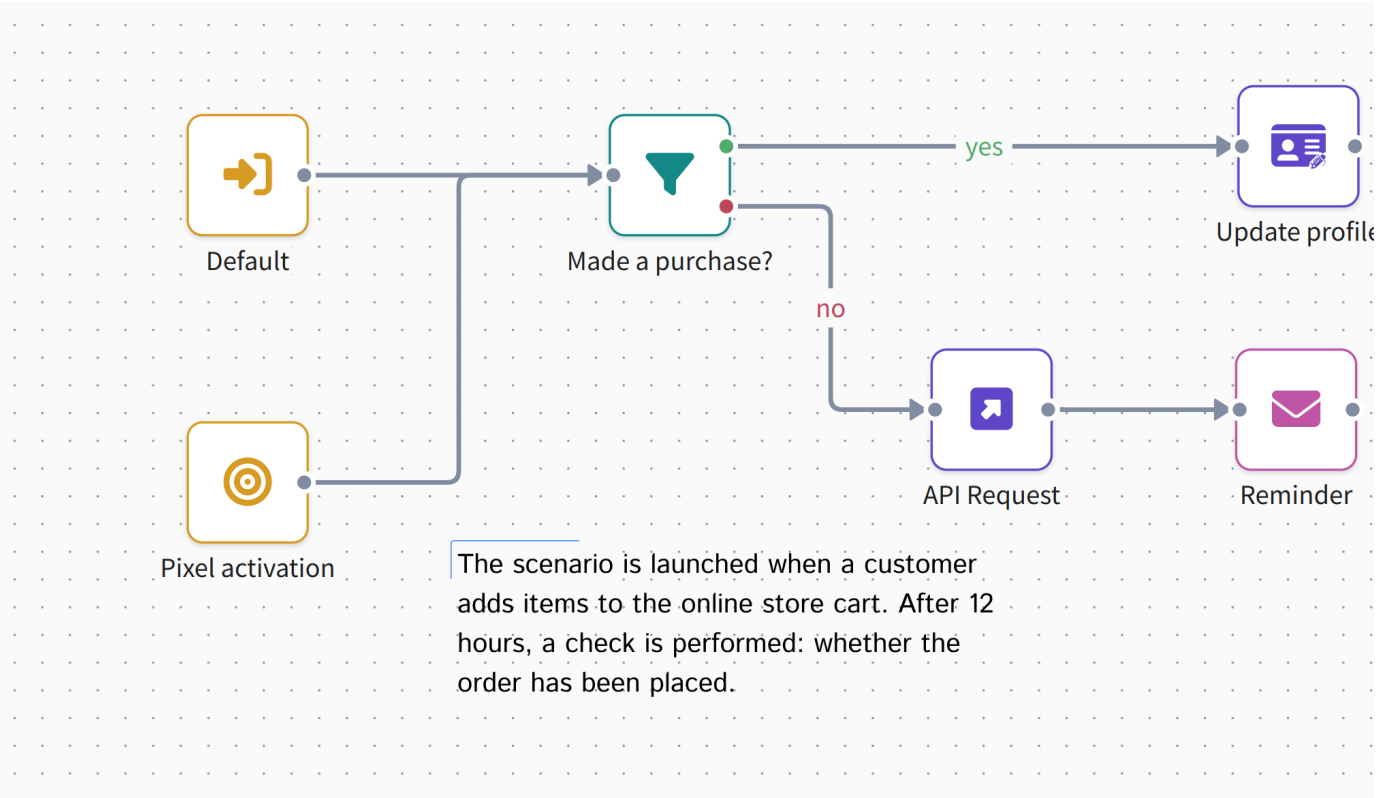
One of the best ways to gather a highly loyal audience is to make them educated in your field of busines speaking of financial sector where customers' trust is the key to success. And here you come as a tutor Customer learning can be automated and made interactive with Altcraft Platform scenarios, that will che relevant trainings:



Abandoned shopping cart

If your potential customer is interested in your offer as much as adding products to a shopping cart — th you can get a second chance by reminding of the abandoned cart. If you can sacrifice a discount in exch scenario can automatically assign promotion codes for a limited stock of products.

API request is used to launch this scenario:

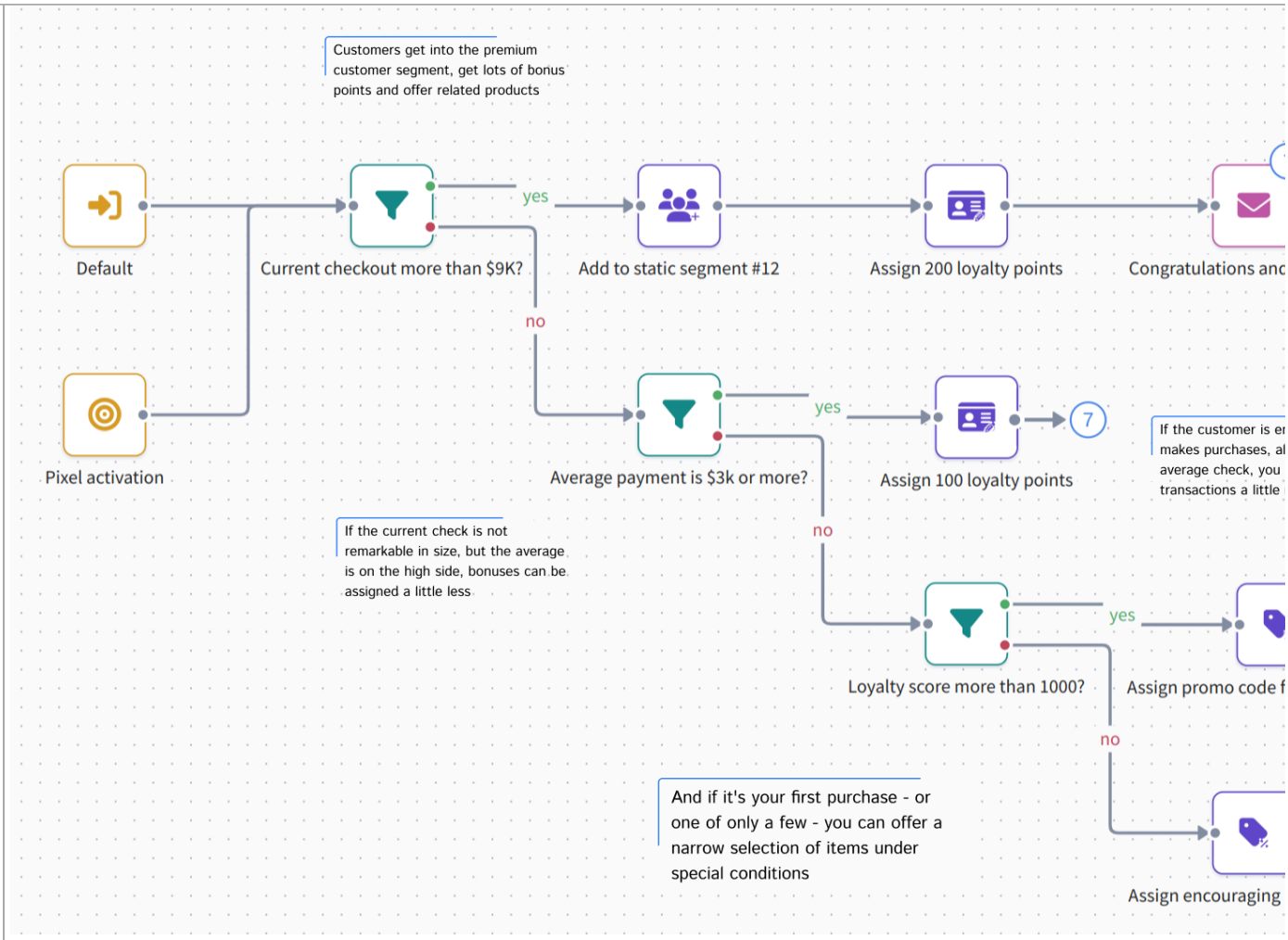


TIP

You can find an example of how to implement a shopping cart abandonment scenario in this article.

Customer loyalty management

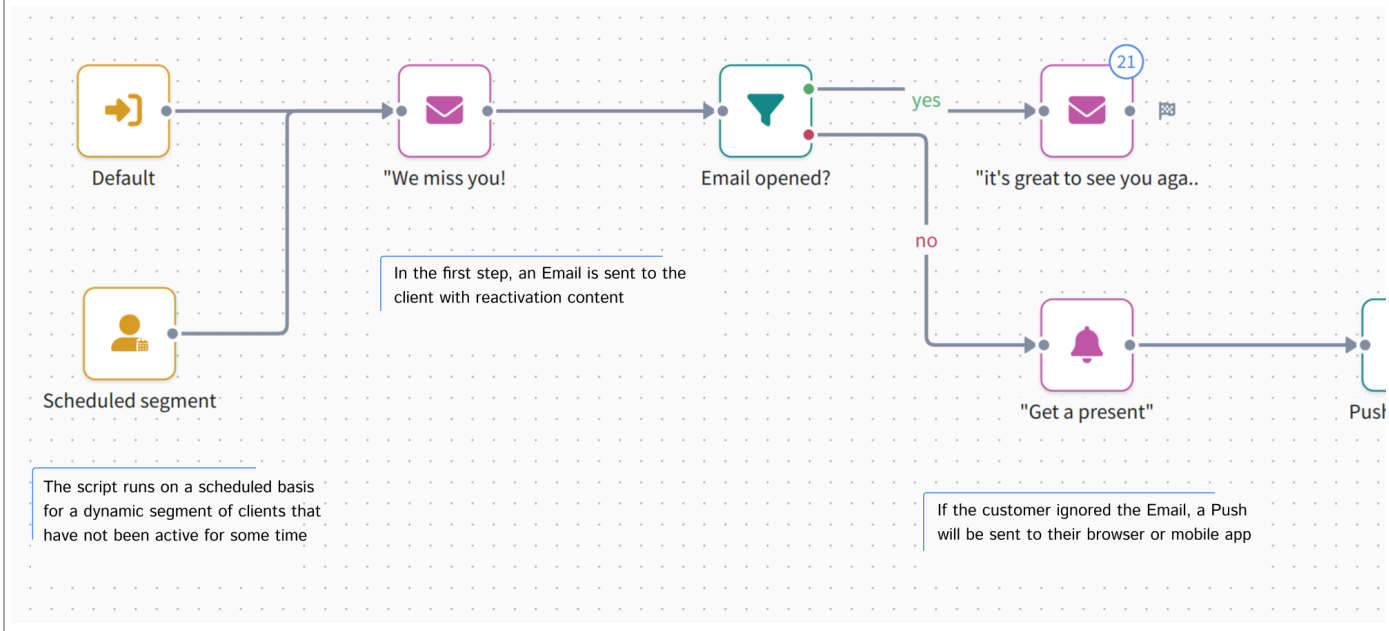
Customer loyalty is one of the key aspects of the modern market. It is a good way to implement gamifica Compose a scenario that will ispect customers' loyalty score, RFM metrics (average payment) and order customers' loyalty:



Omnichannel customer reactivation

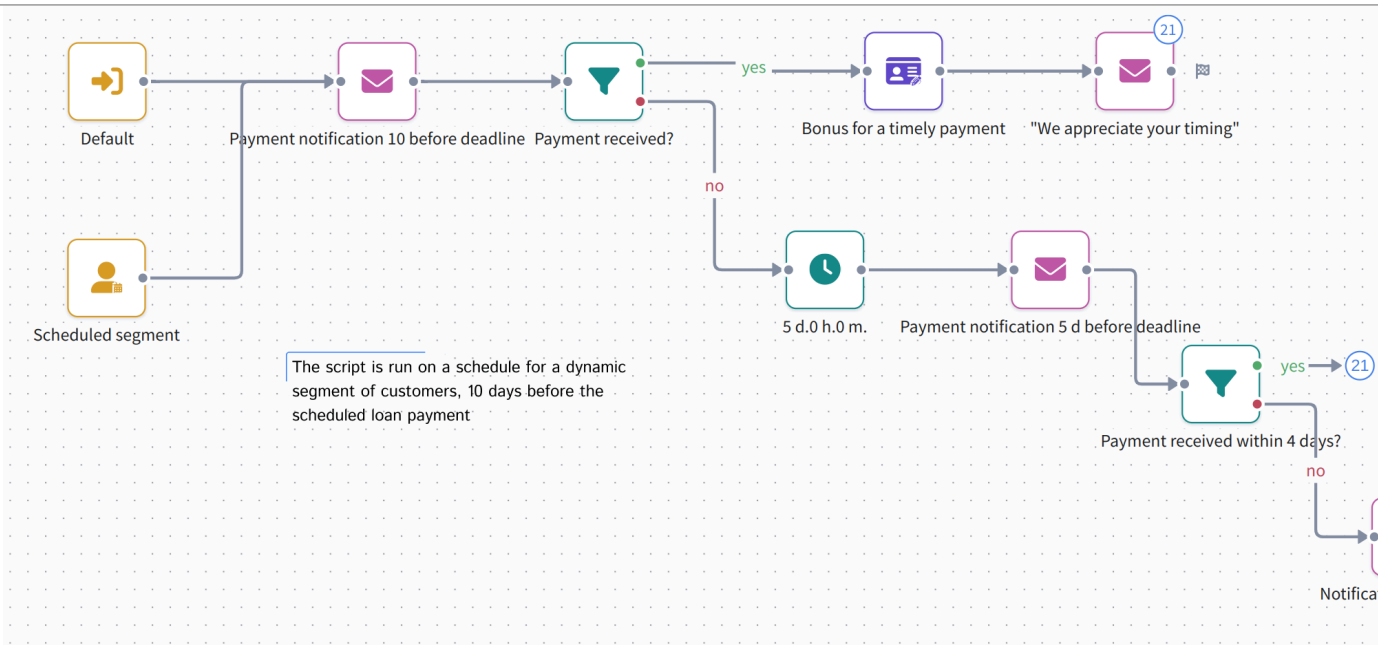
You can reactivate inactive customers automatically with the help of the scenario in this example. Scen as a regular mailing. Each launch a dynamic segment will provide customers who have ignored your mail of time.

The sequence itself is what is called Cascade mailing — messages are sent consequently until a custom use this type of omni-channel communication in a wide variety of cases:



Regular payment notification

This scenario will help you automate omnichannel notification about regular payments, service period er dynamic segment and schedule are used to launch a sequence of notification — via one or several chann notification scenario will check if the payment has been made:


Default
Payment notification 10 before deadline
Payment received?
yes
Bonus for a timely payment
21
"We appreciate your timing"
no
Scheduled segment
5 d.0 h.0 m.
Payment notification 5 d before deadline
The script is run on a schedule for a dynamic segment of customers, 10 days before the scheduled loan payment
Payment received within 4 days?
yes
21
no
Notifica

Step-by-step welcome scenario guide

This guide will help you set up an automation scenario, managing interactions with new customers - fro mailings and purchases. We give you some basics on how to choose scenario nodes based on the situati Welcome workflow is one of the [classic marketing automation scenarios](). You can implement it effective

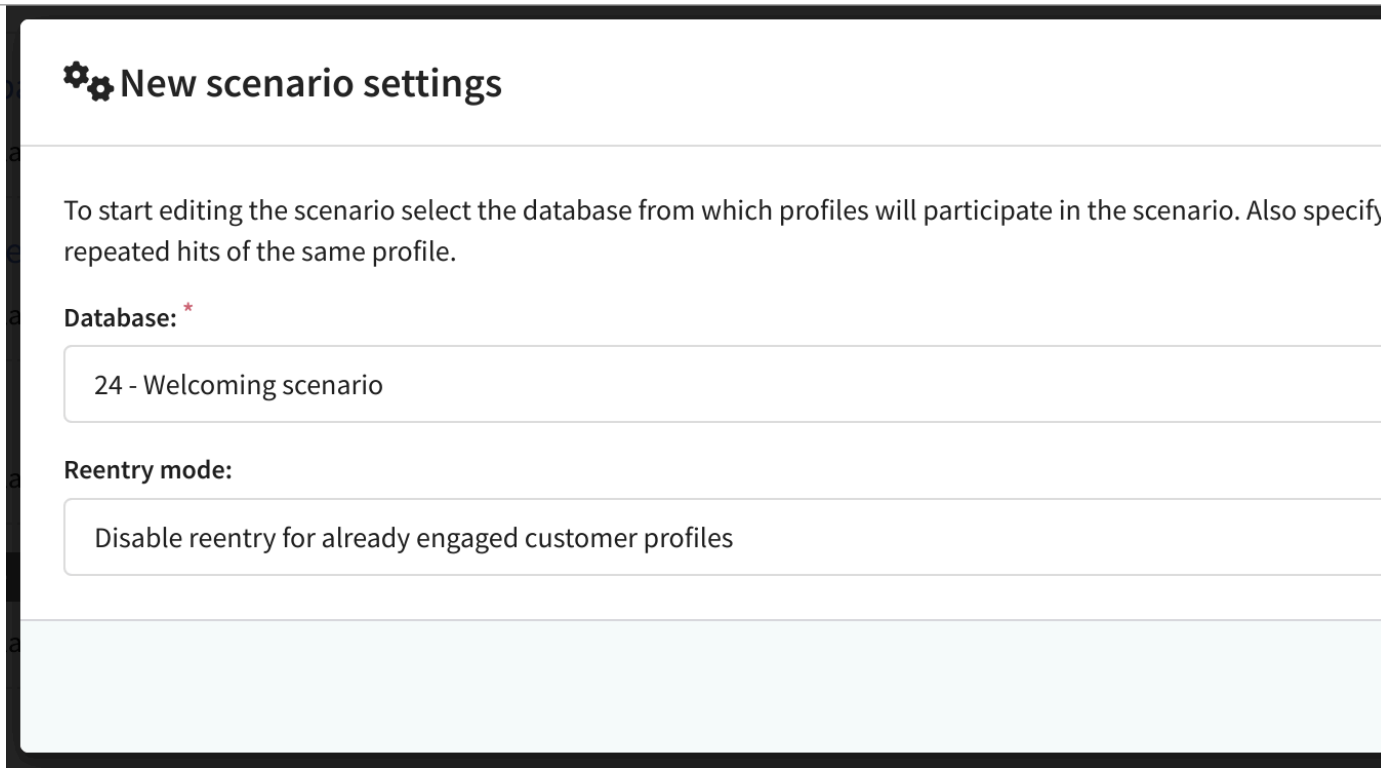
TIP

Automation scenario basics are described [in this article]().

[This page]() will provide you with information about scenario steps - nodes.

Step 0: Create a new scenario

In the Scenarios section, create a new scenario by clicking the appropriate button at the top of the scre disable re-entry for the profiles involved in the scenario, as the greeting is a one-time event:



Step 1: Configure customer capturing mode

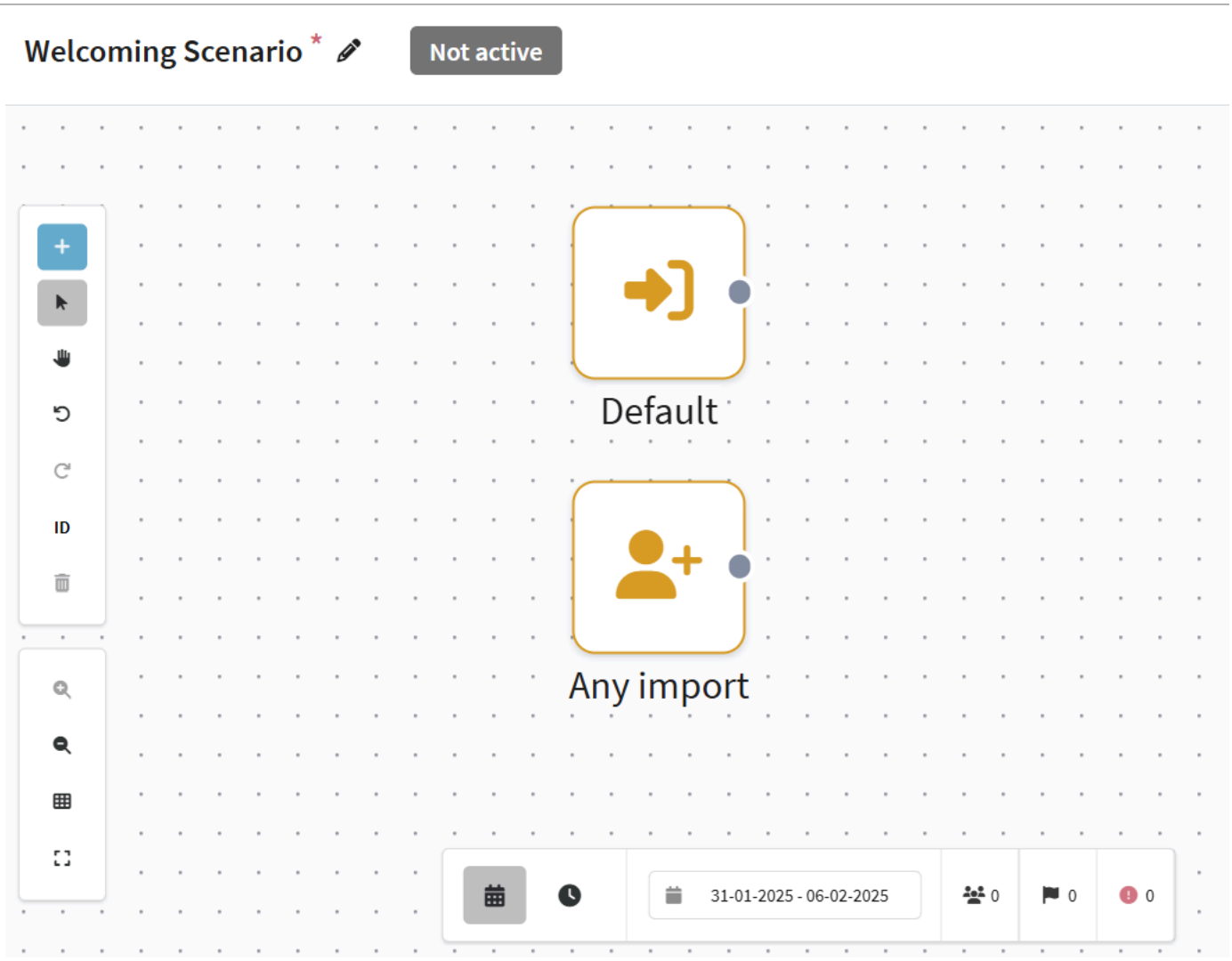
Customers will get engaged in the scenario when their subscriber profiles are created within the platfor how you import your customers' data into the platform:

- **Any profile import - launches any time a new customer profile is created.**
- **Manual profile import - launches when you create a customer profile manually**
- **API profile import - launches when you import a profile via platforms' web API from your website or i**
- **Form import - launches when filling in form fields. To learn how to create a form in the platform, read**
- **Popup import - launches when filling out a form in a popup. You can learn more about creating and c**

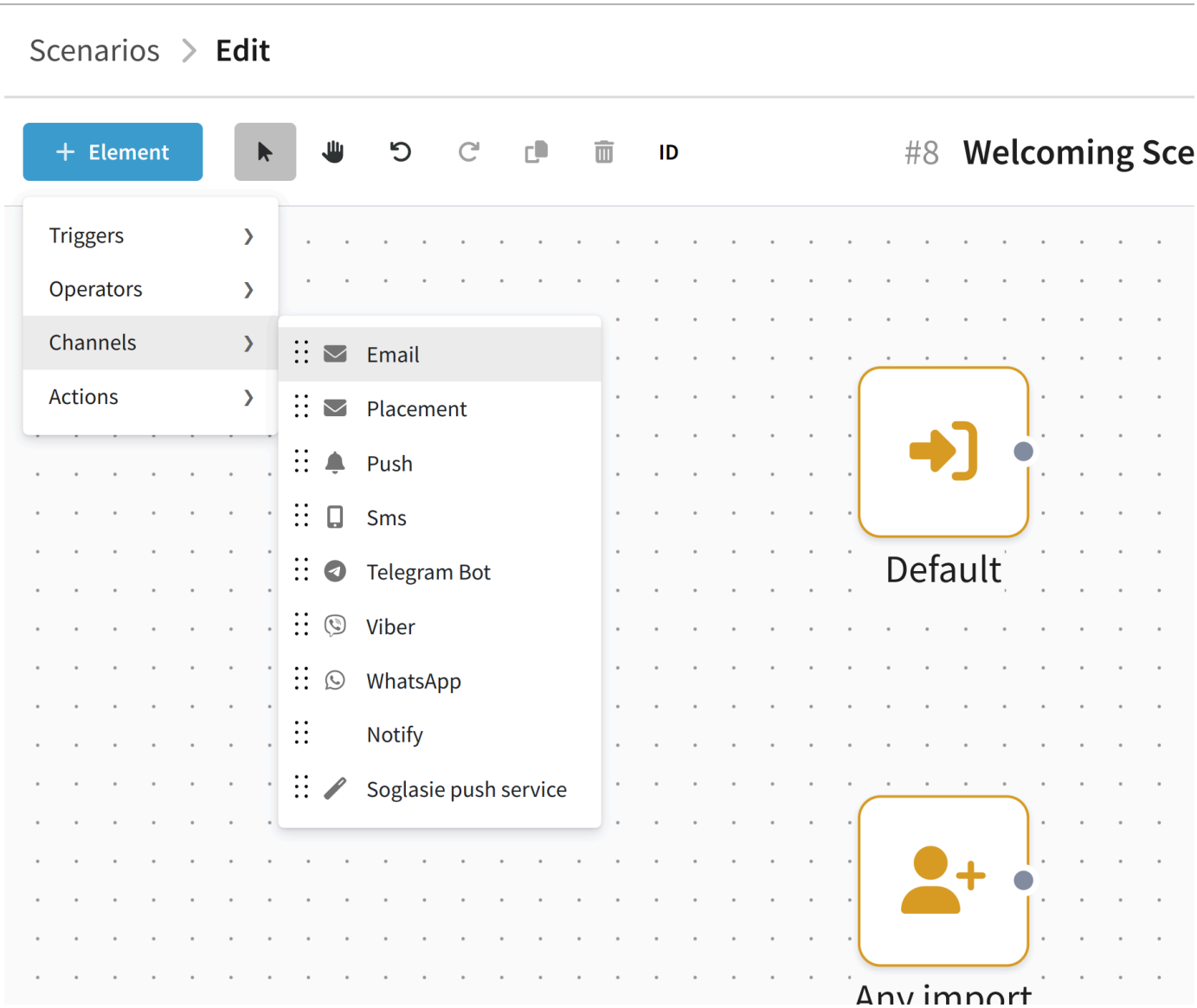
There is also an element capture default profiles in any scenario. It is required to capture profiles added another script.

> TIP
>
> **Various customer data import and update options are described in this article.**
> **This article describes scheduled customer profile import from an SQL storage.**
> **API requests considering customer profiles are described in this section of API docs.**

In order for all new profiles from the database to get into the script, we use the trigger Any profile impor



Step 2: Create an Email welcoming message



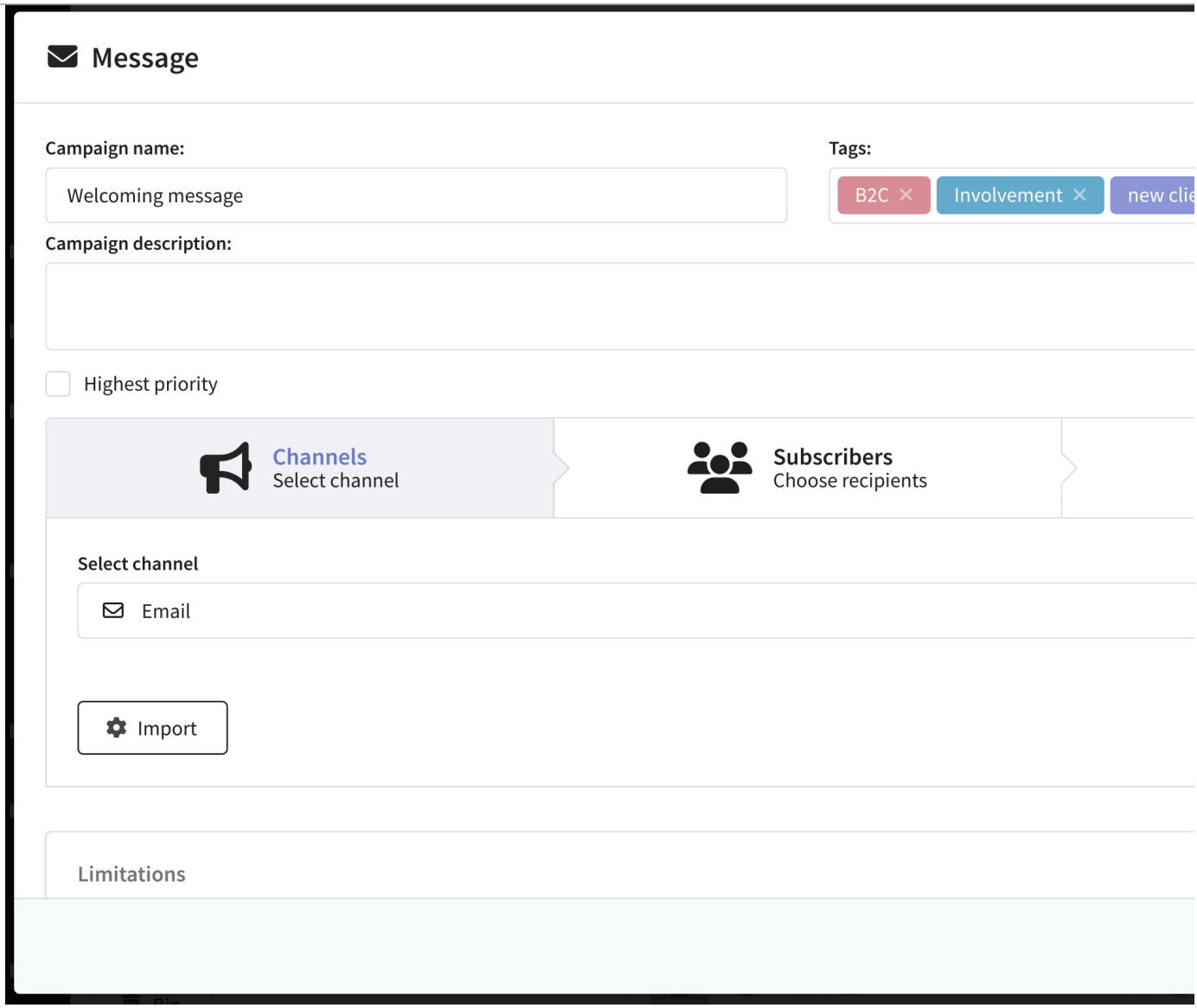
Add the following element to the script - email distribution. To do this, click on + Element - Channels - Em customization window. Here you can specify channel, resource and message template.

TIP

Subscription resources for your audience are described in this article.

This section will tell you all you should know about message templates.

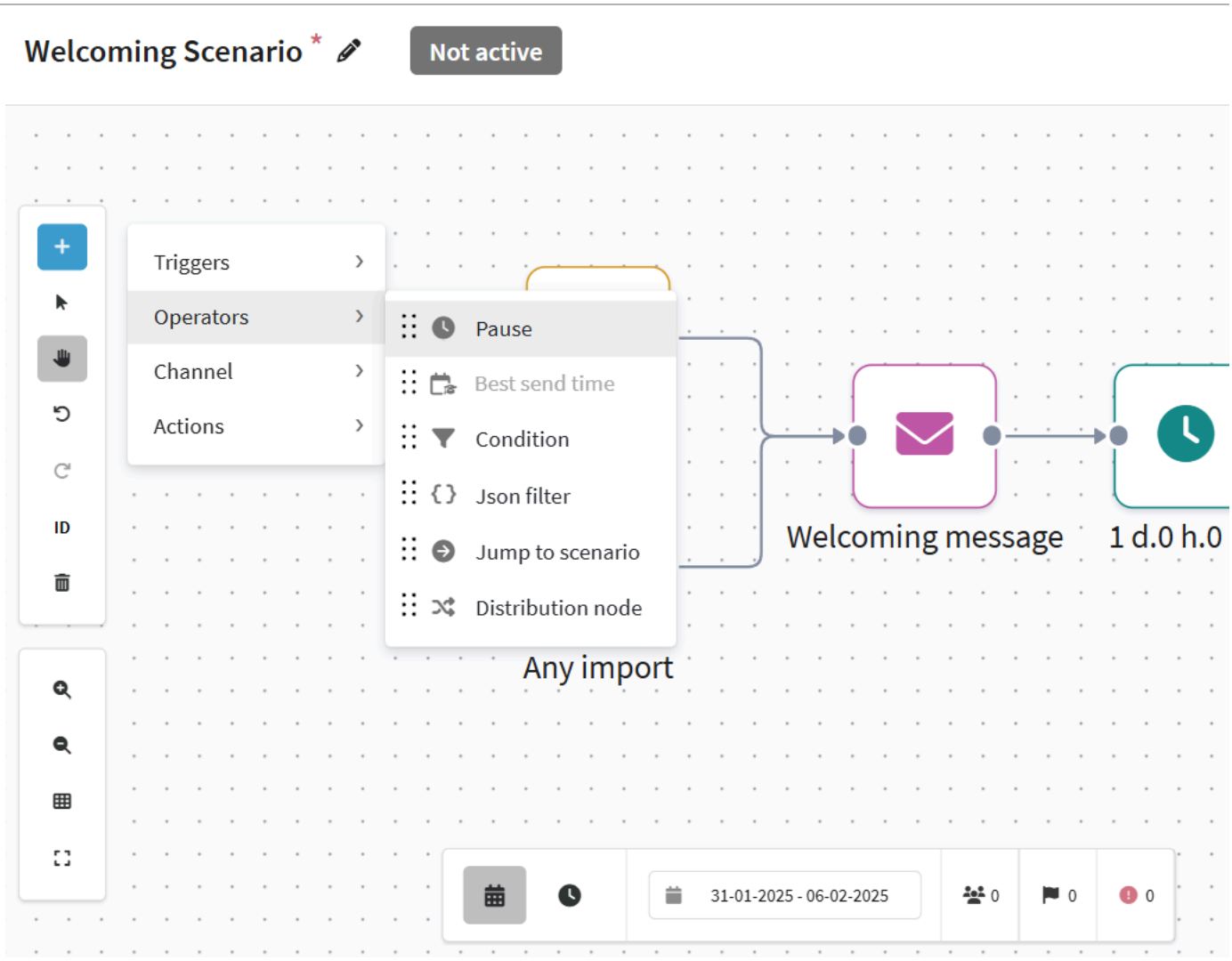
You can find a step-by-step Email mailing guide here.



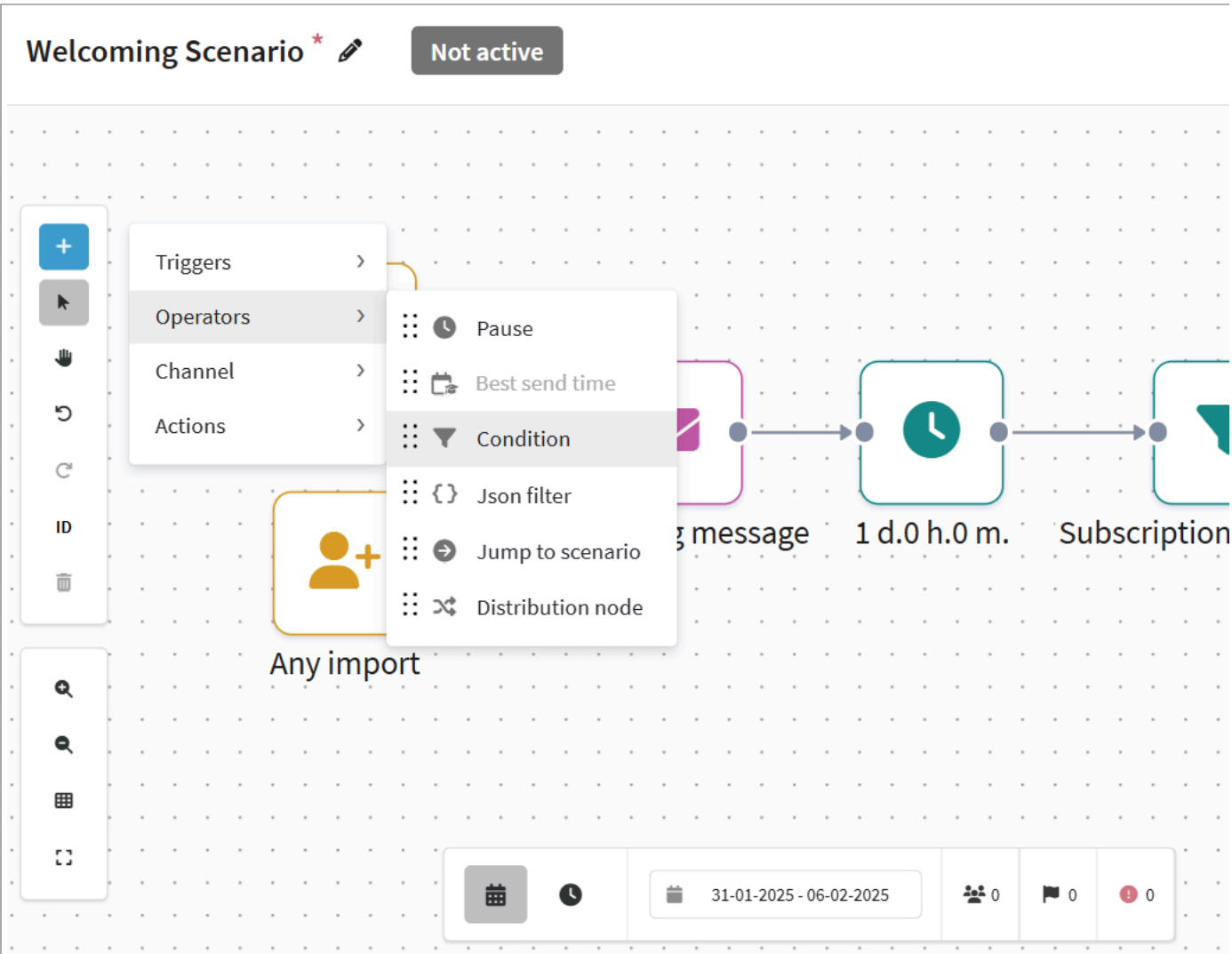
Step 3: Set up a sequence for active recipients

After you send an Email welcome, a part of your customers will open, read it, click a link to the purchase of those who do is called your marketing conversion. Now, to increase welcome workflow conversion let sequence. Various scenario nodes will help you a lot.

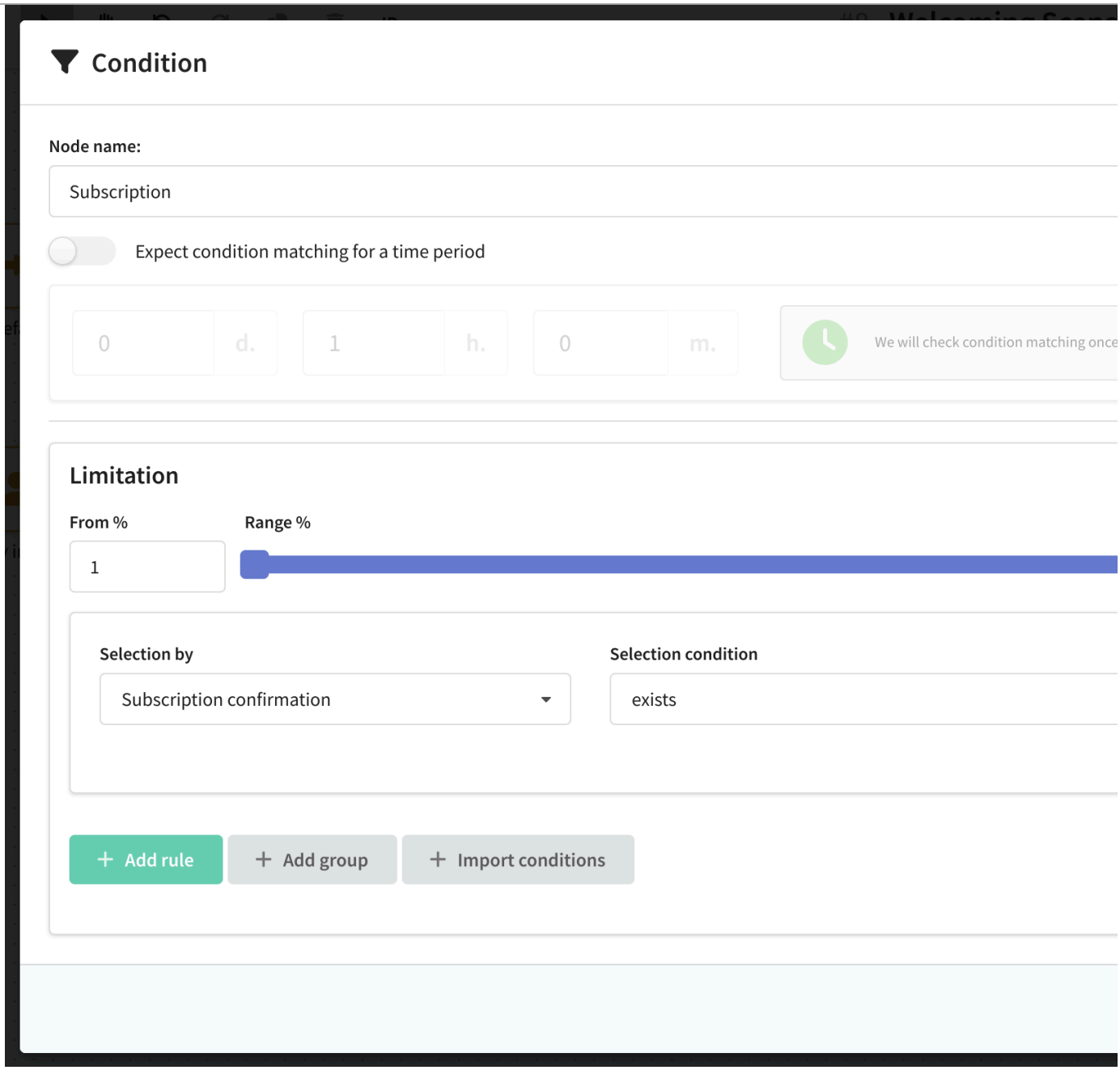
The next element will be the pause. With its help we will give customers time to read the email. The Pau Operators category:



Next, using the Condition element, let's separate out those subscribers who opened the welcome messa on the first day:

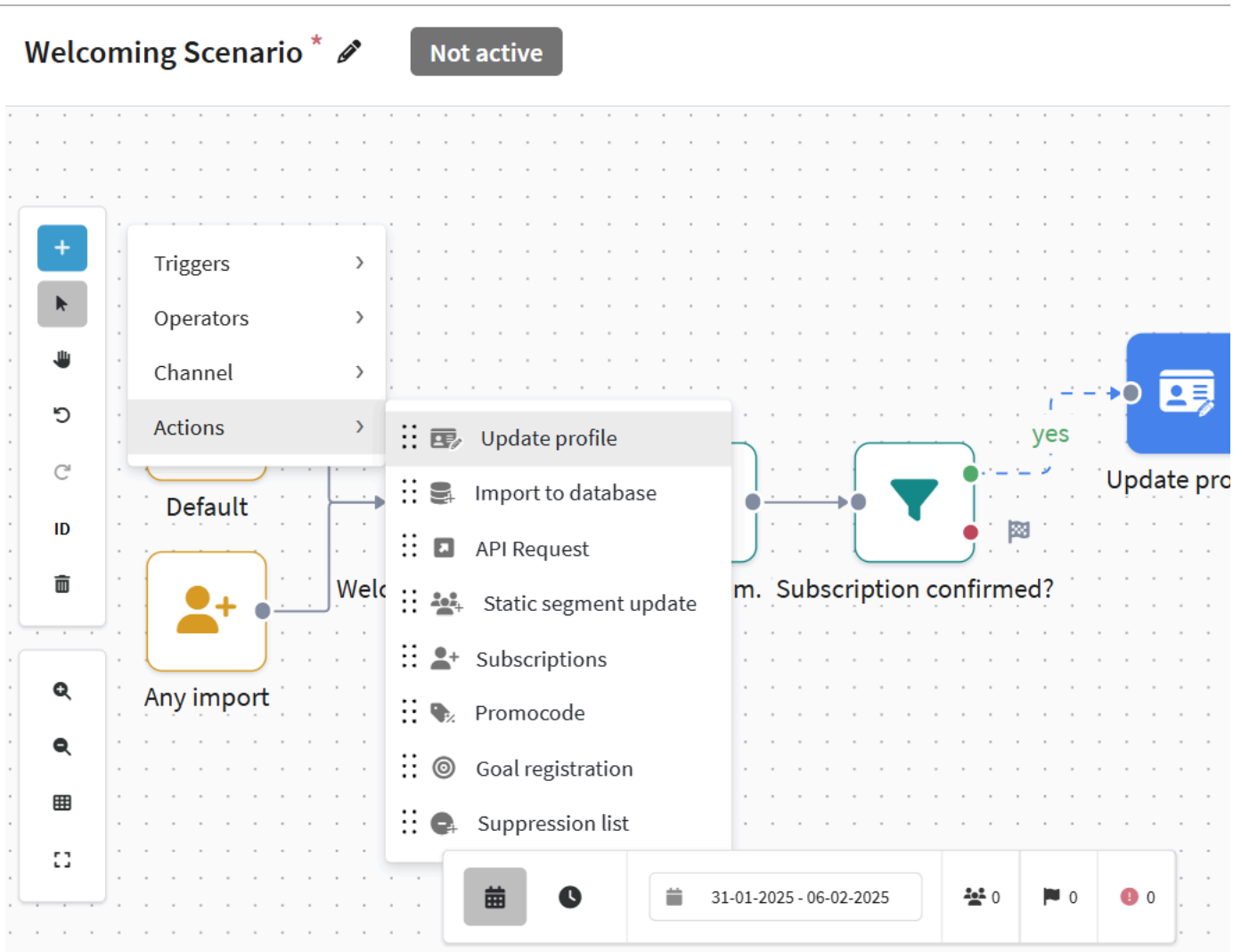


You can use the “Email opened” or “Click on link” rule instead of the registration confirmation. The exam strategy subscription confirmation:

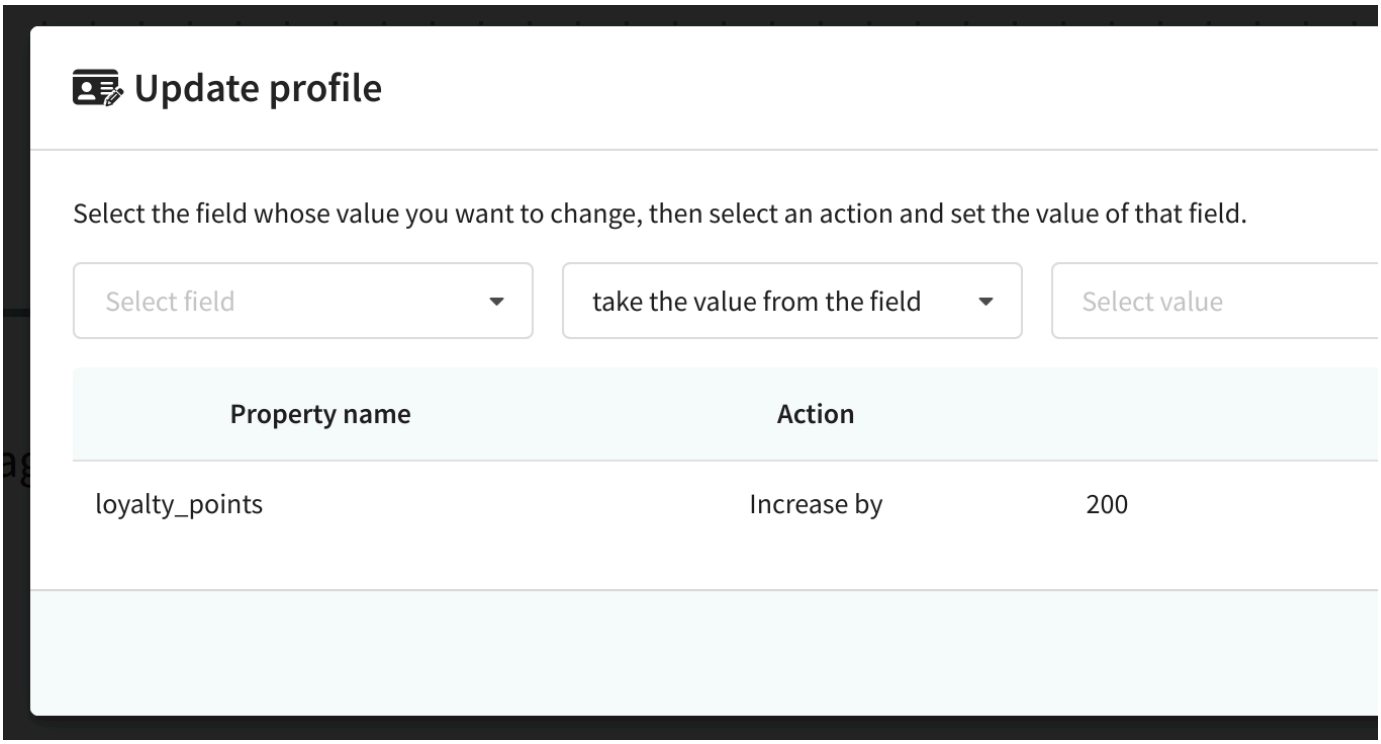


These customers are likely to be highly motivated to buy. It's a good idea to reward these customers, for points right away. These can then be used to assign promo codes, discounts or gifts.

Such actions in marketing are called scoring. Scoring is one of the elements of sales gamification that c rates. To implement it in the platform, you must first create a numeric field in the profile database for the update these points in the script using the Update Profile element:



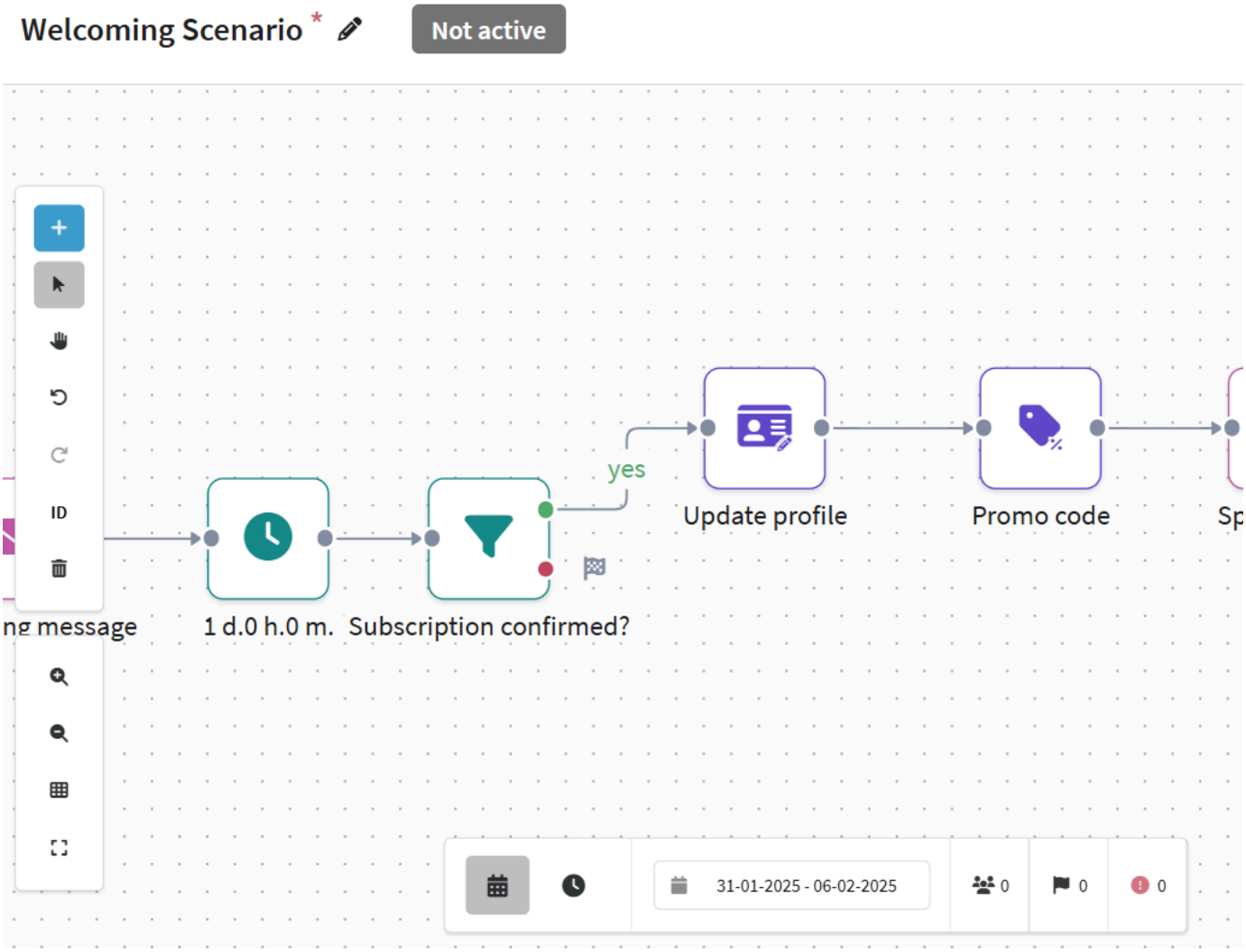
Add the element to the top (green) branch coming from the condition. In the element settings, specify th and increase its value by the required amount:



Next, you can assign promo codes to new customers (for example - for a part of a product serving specif promo offers to customers. To do this, add a Promo Code element and an email channel message.

TIP

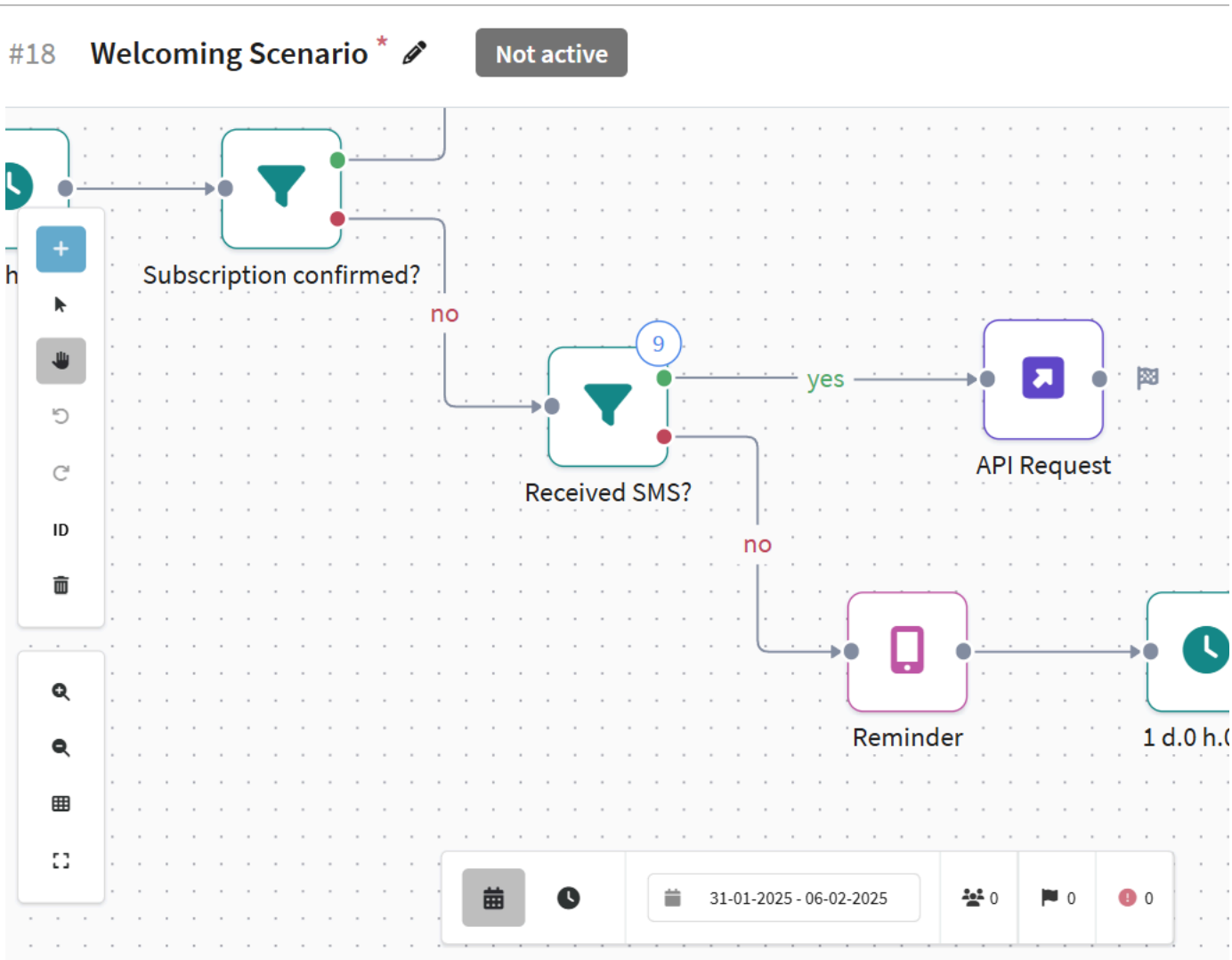
Promo codes are created by Altcraft Platform loyalty programs. You can read more [in this article](#).



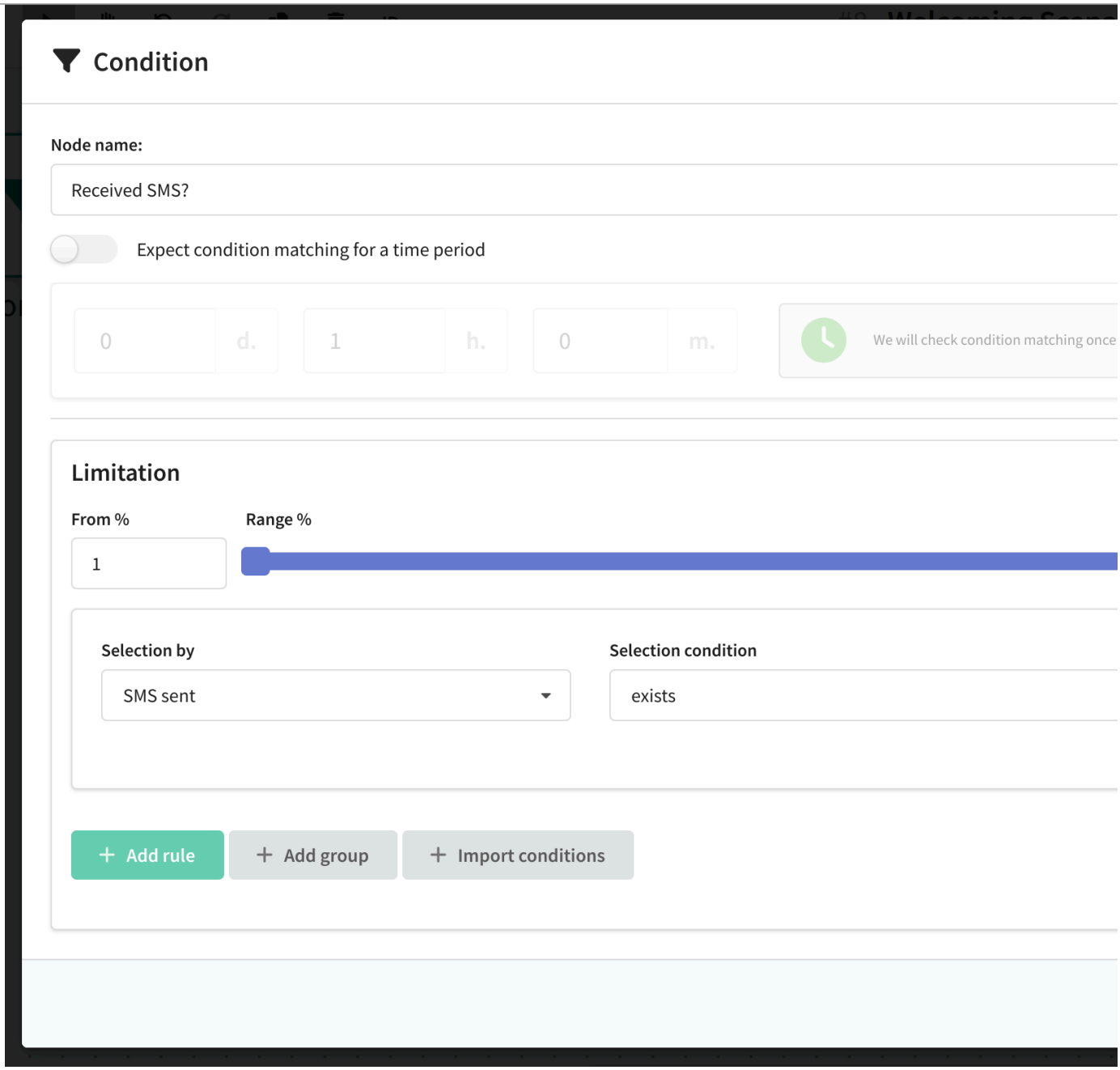
Step 4: Set up a sequence for not-so-active recipients

Some of the customers to whom you sent a welcome message may have forgotten about it, not received interested in the offer. In this case you can remind them about the incomplete registration. It's best to us phone and it's always at hand.

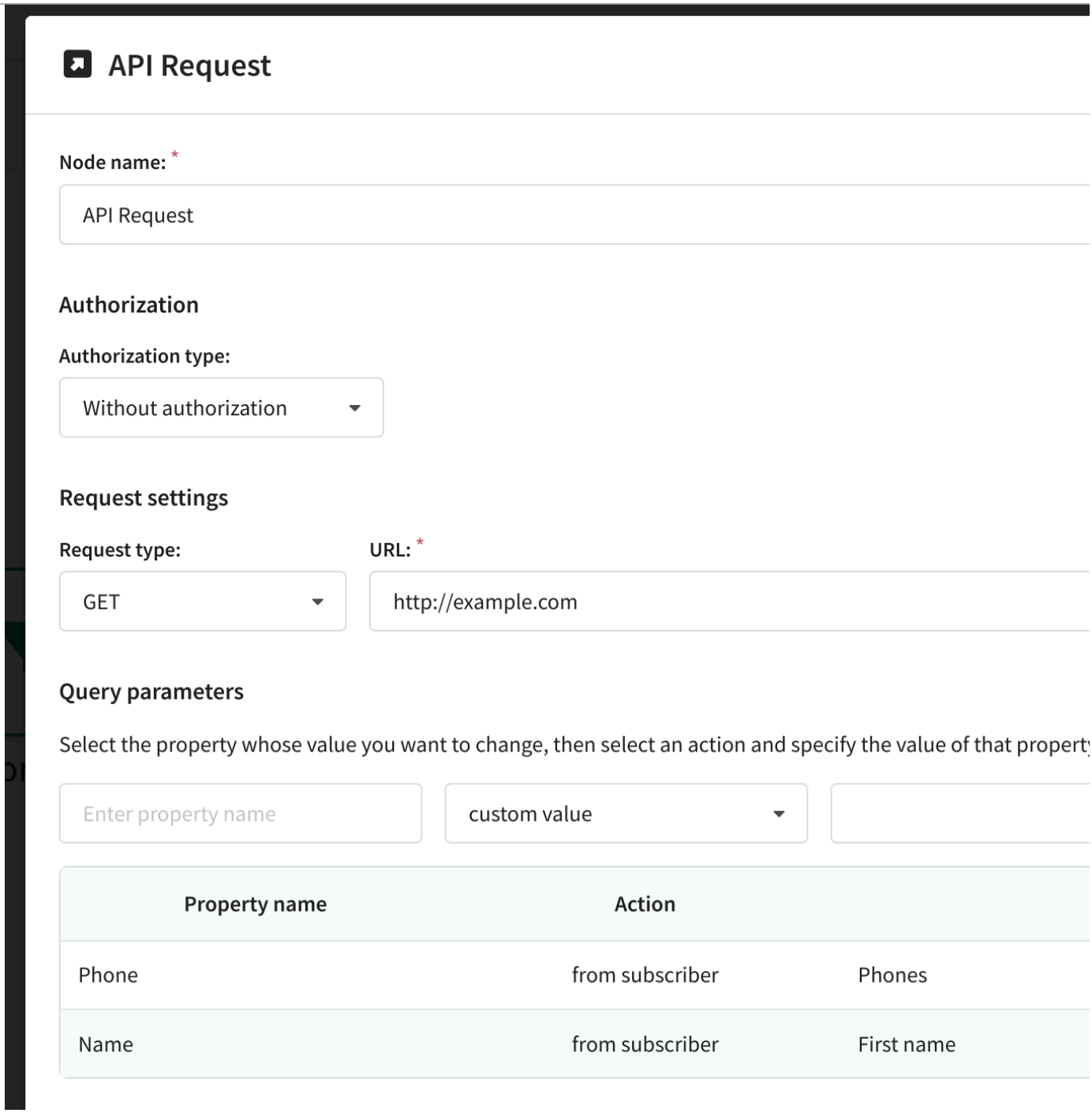
However, it is usually not always advisable to send the same reminder more than once. It is much more e tactics. In the example, let's use two: first, a SMS is sent, and if there is no response to it, a notification profile is sent to managers or call center operators so that an employee can personally communicate wi inactive customers would look like this:



To apply two tactics at the same time, add a Condition element for those who have been sent an SMS re
There is an existing SMS sending:



From the top (along the green branch), add the API Request element. In it, specify the network address, t want to pass to the call center. Contact your company's system administrators for details:

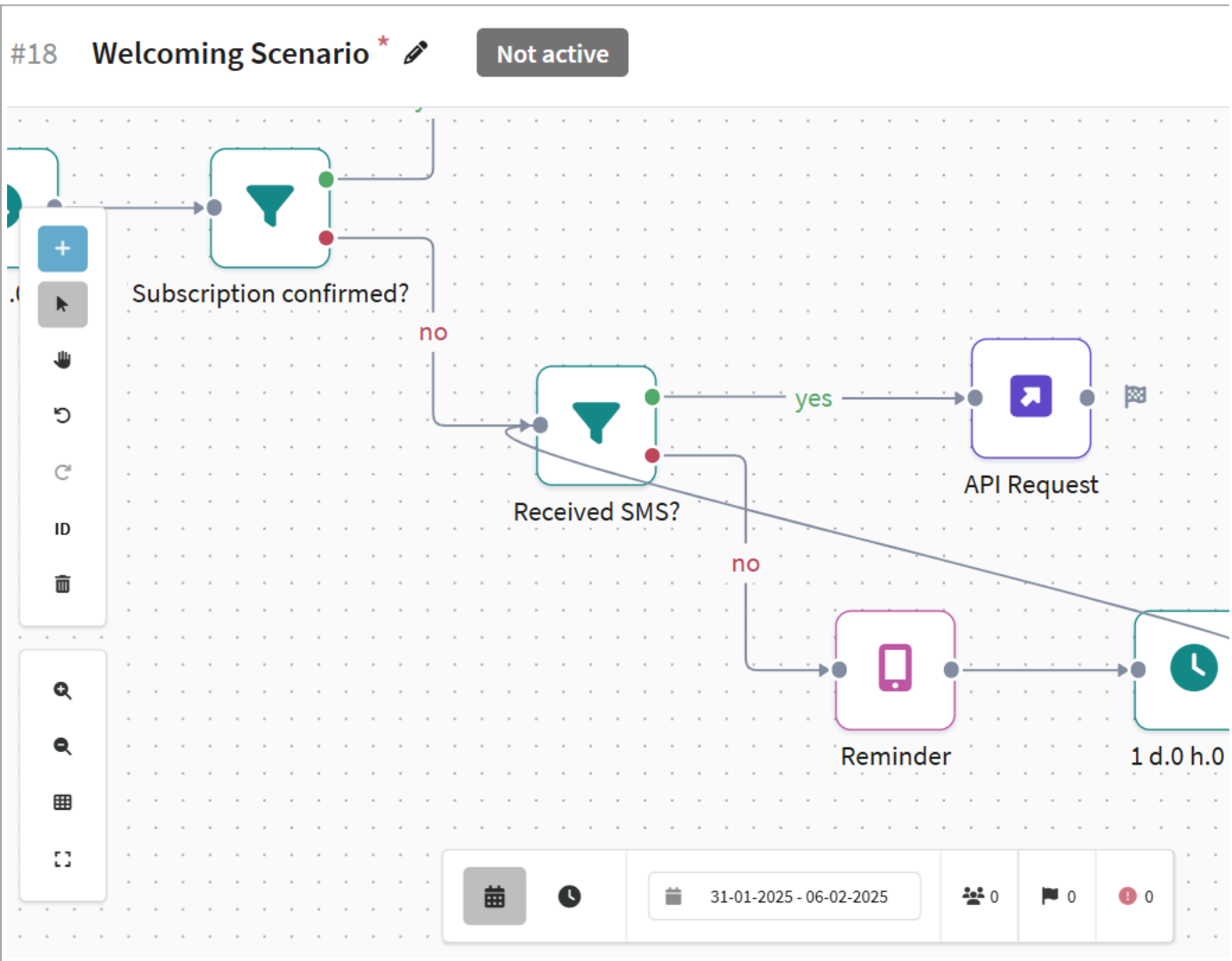


Down from the condition (along the red branch) add an SMS element where you create a new SMS mailin

TIP

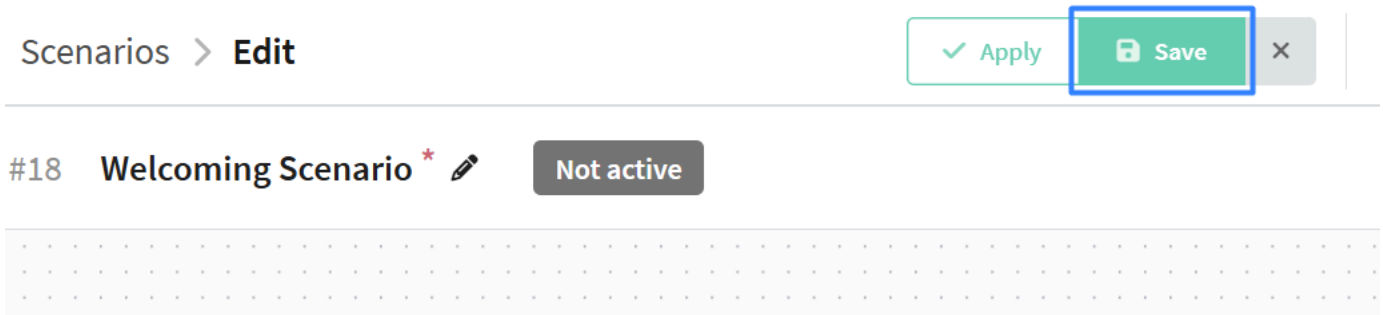
You can find a step-by-step guide on how to create an SMS newsletter right there.

After the message, add a Pause - just like you would after an Email Welcoming. When time runs out after should be sent back to the beginning of the script, to recheck the subscription confirmation in the Condi cursor from the right point of the Pause element to the SMS sending condition:

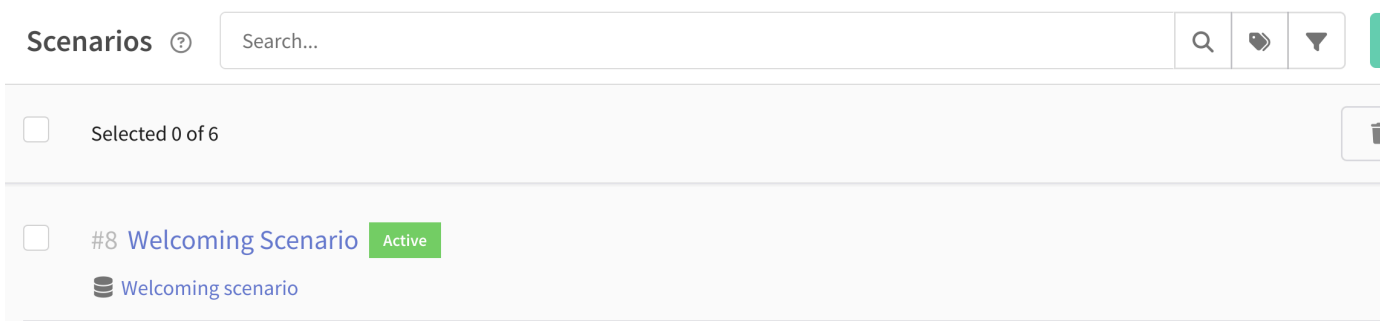


Step 5: Save and activate the scenario

Once you have made the necessary settings and configured the script steps, save and activate it by clicl top of the screen:

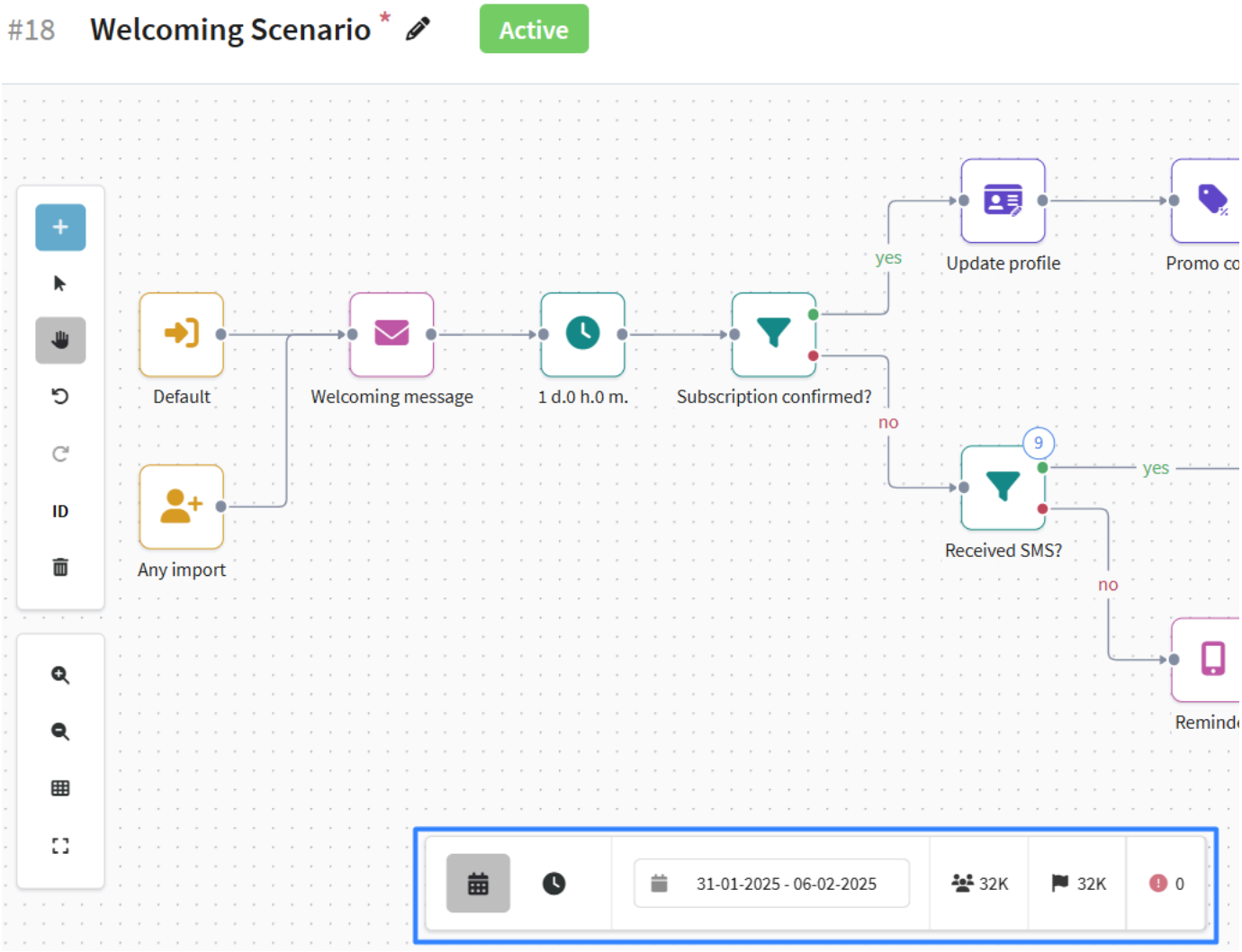


The welcome sequence will be a great start to effective long-term communication with your customers. and editing by clicking on its title:

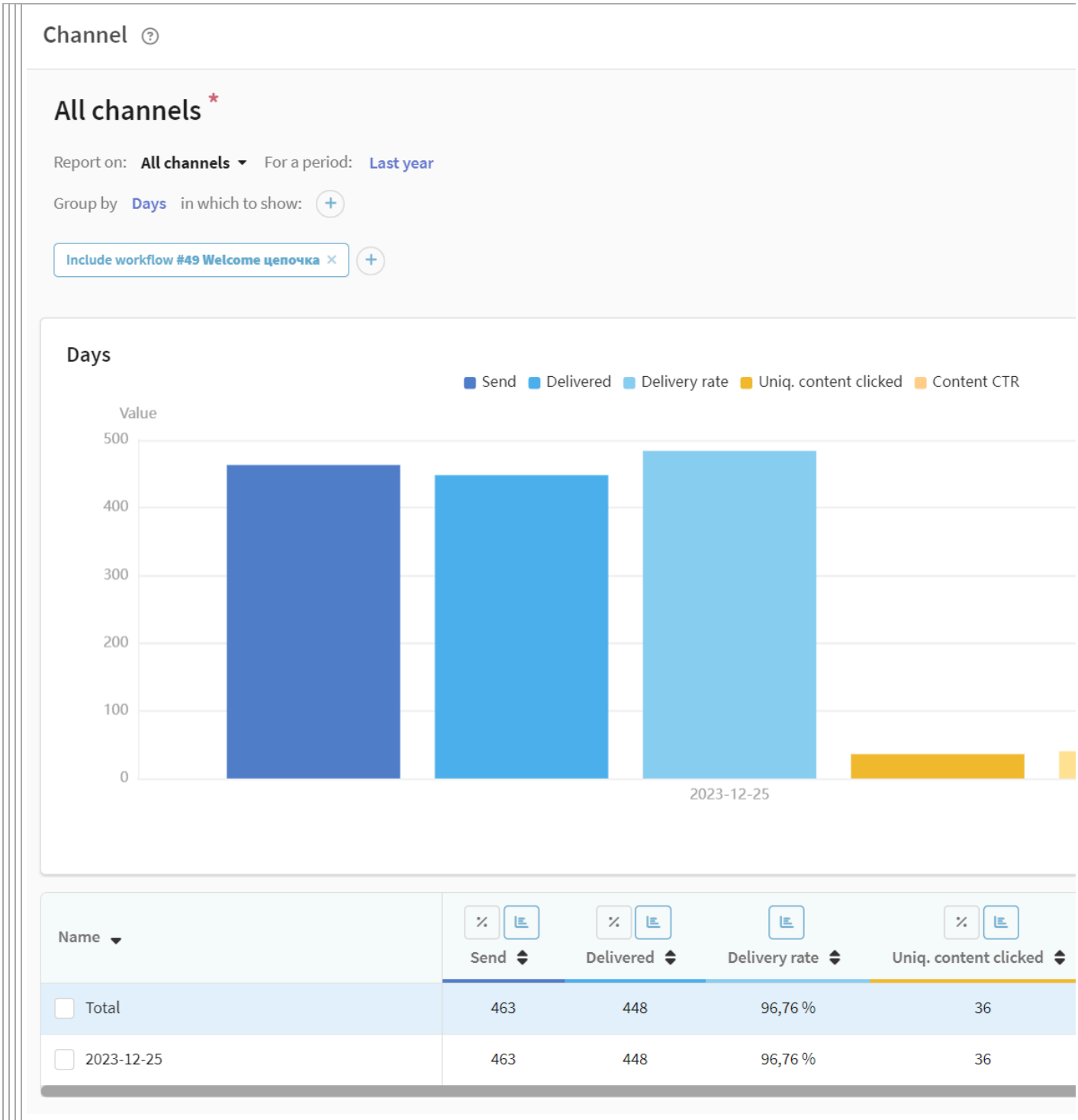


Step 6: Analyze performance

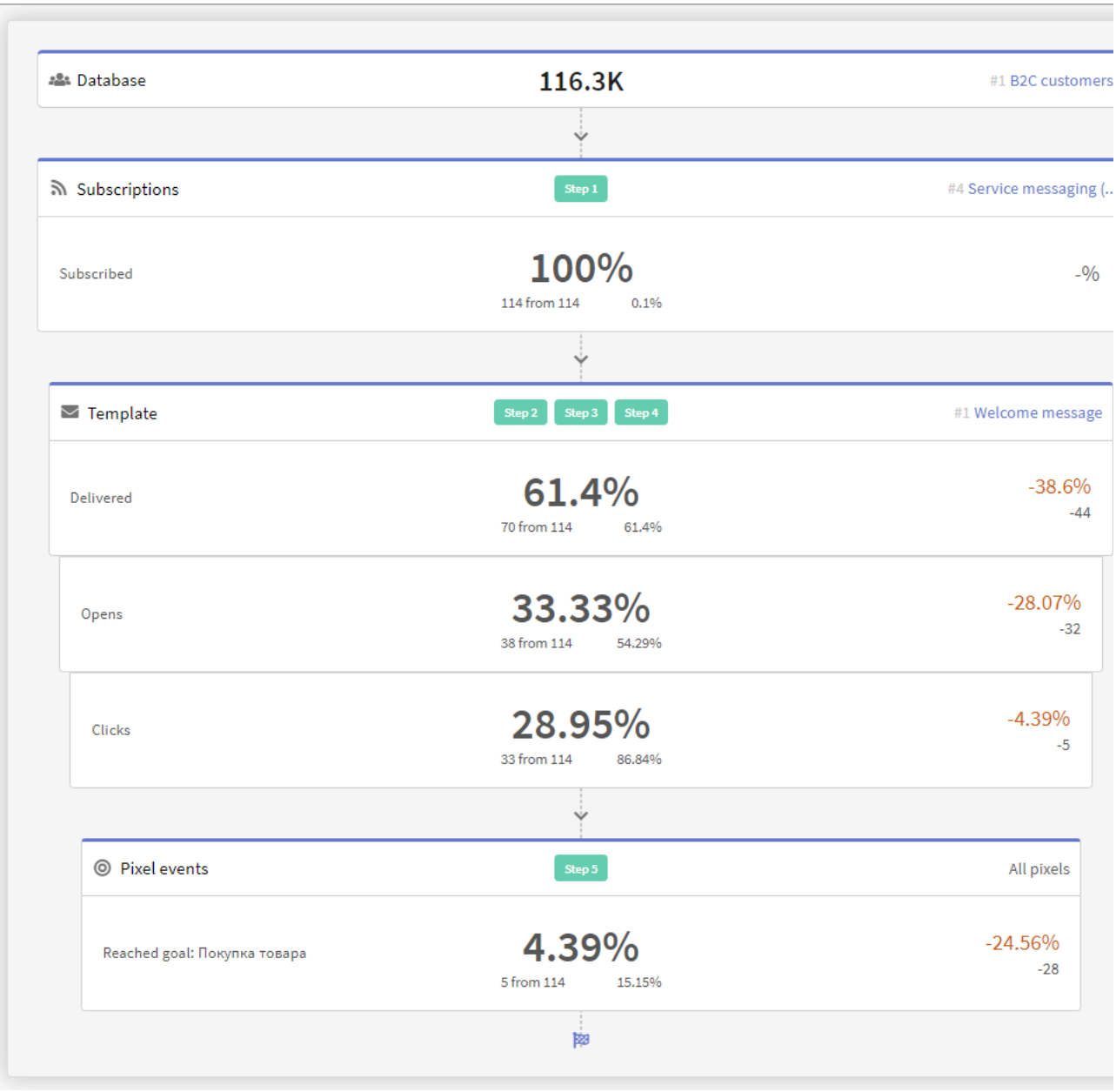
You can evaluate the effectiveness of a scenario using several analytical tools.
The first is statistics within the scenario. Select the required period and view the pass rates:



The second is [channel report] of the scenario. It will provide detailed information on the mailings within t



Finally, you can add mailings and other scenario events to [funnel-report] to monitor conversion rates ever funnel can be whether a customer falls within the sample conditions or reaches a certain goal (Purchase

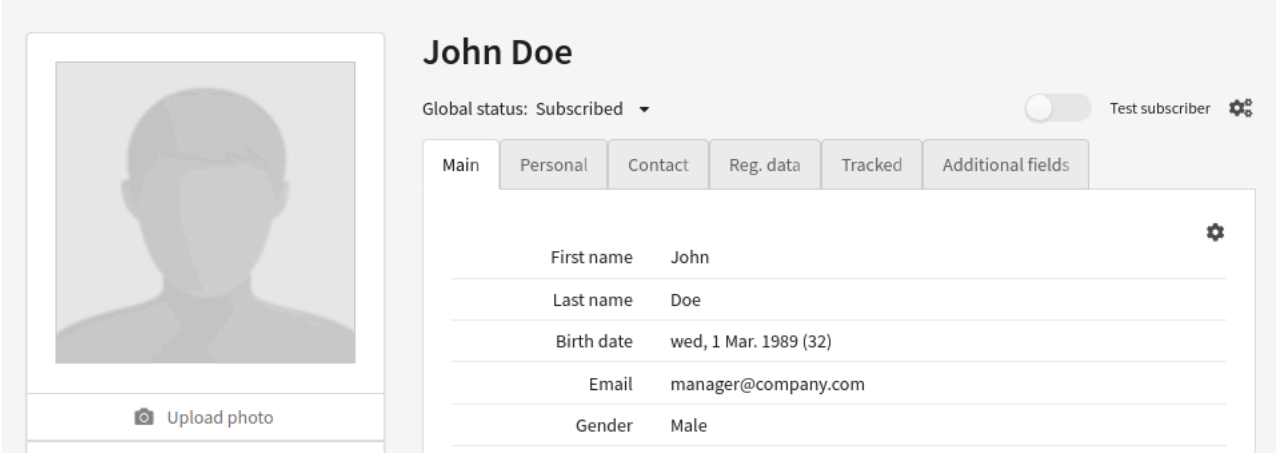

Database
116.3K
#1 B2C customers
Subscriptions
Step 1
#4 Service messaging (..
Subscribed
100%
114 from 114
0.1%
-%
Template
Step 2
Step 3
Step 4
#1 Welcome message
Delivered
61.4%
70 from 114
61.4%
-38.6%
-44
Opens
33.33%
38 from 114
54.29%
-28.07%
-32
Clicks
28.95%
33 from 114
86.84%
-4.39%
-5
Pixel events
Step 5
All pixels
Reached goal: Покупка товара
4.39%
5 from 114
15.15%
-24.56%
-28

Scenario for automatic notification of the manager

Let's look at a typical case that can be implemented using scenarios in the Altcraft Platform platform. Here is the task. Every time the user performs any target action, the manager must receive a letter with Let's consider the implementation of this functionality through an API call from a scenario. The lead data filled.

Step 1. Create a manager database

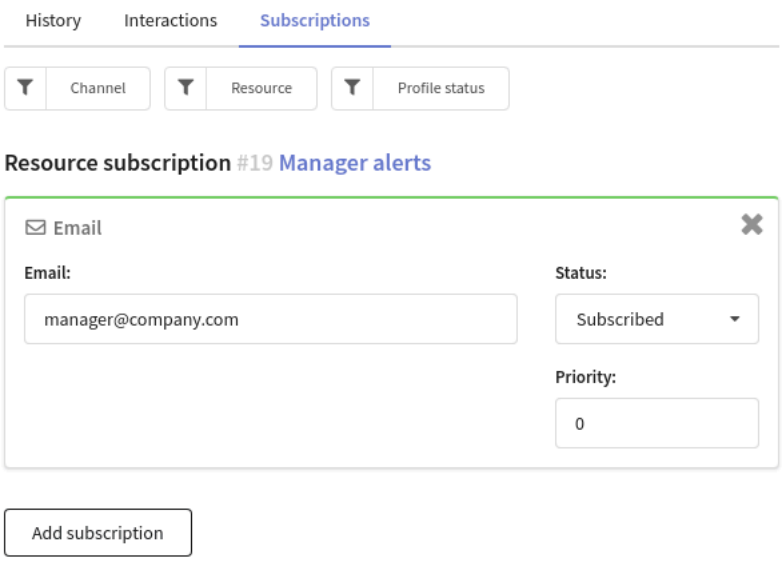
First of all, it's necessary to create a database. Let's call it "Managers". You need to set up the database you need to create a new profile.



You can learn more about profile bases right there.

Step 2. Create a resource and a subscription

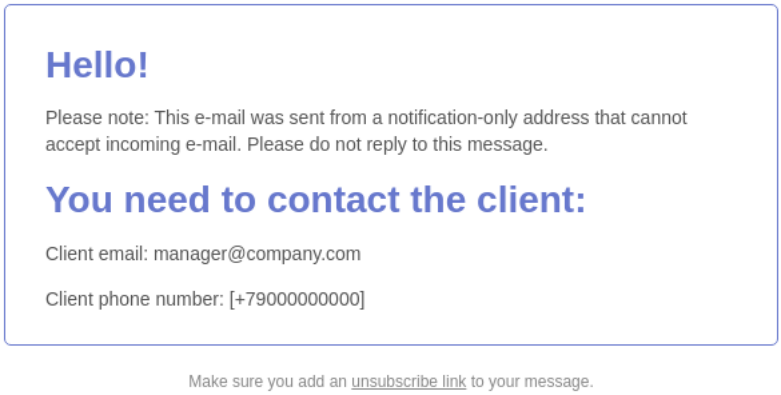
Only when the manager database is created and the profiles are added to it, it is necessary to create a r alerts"). When you create a resource, specify the Managers database in its settings. After the resource h subscribe a manager profile to it. In the example, let's add a subscription manually:



You can learn more about resources in this article.

Step 3. Create a template

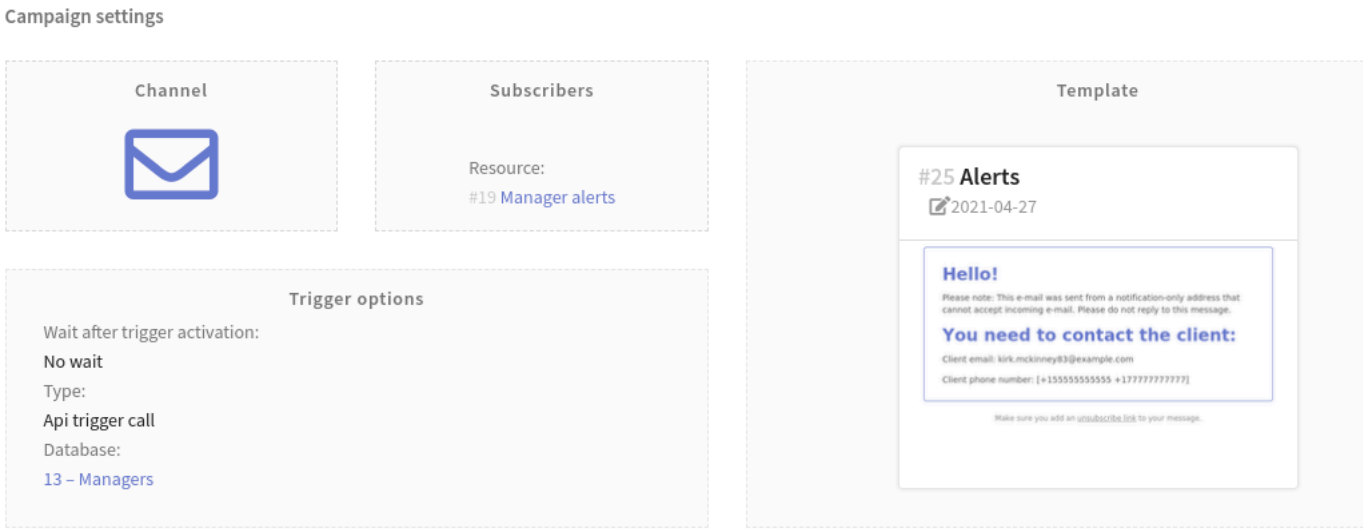
Now it's time to create the template of the message that will be sent to a manager. Let's call it "Alerts". apicontent to personalize messages.
Client email: {apicontent.mail}
Client phone number: {apicontent.ph_number}
The example of the template for notification of a manager:



You can learn more about templates in this article.

Step 4. Create a trigger

We're going to use a trigger mailing as a tool of notifying a manager. When creating it, you need to selec template "Alerts", the condition for triggering - API trigger call and the database "Managers".



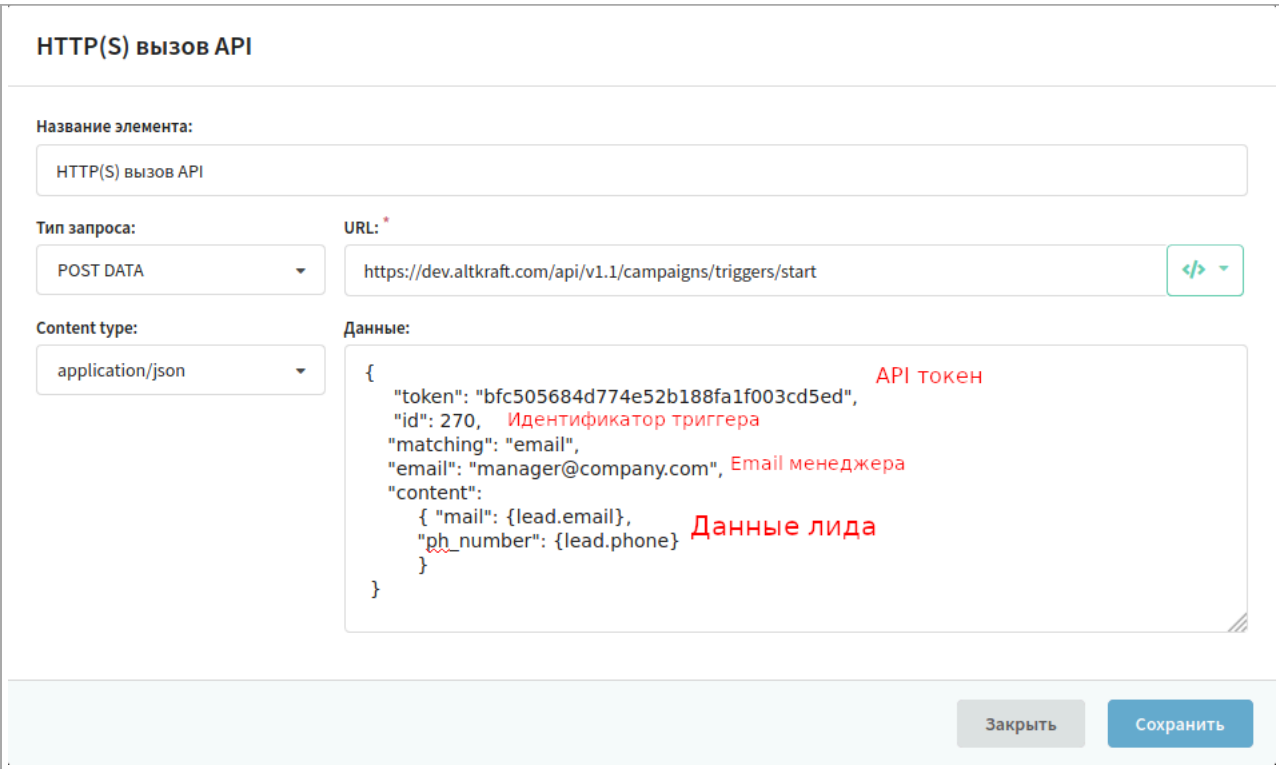
You can learn more about trigger mailings right there.

Step 5. Create a scenario

The key element that will launch a trigger call and then send a notification to a manager is the HTTP(S) A the request type, Content type - JSON or XML (in our example, it's JSON). The URL contains the address (`api/v1.1/triggers/start`). The body of the API request is placed in the Data section, according to which the (you can learn more about the API right there).

```
{
  "token": "bfc505684d774e52b188fa1f003cd5ed",
  "id": 270,
  "matching": "email",
  "email": "manager@company.com",
  "content": {
    "mail": "{lead.email}",
    "phones": "{lead.phones}"
  }
}
```

This request specifies the API token for authorization, the trigger identifier, the email of the manager, as of the lead, about which the manager will be notified.



INFO

When sending a request, an API token is required for authorization. The token must be created in the pl
our article.

Well, you've just completed the configuration of the manager's notification automation scenario!

Abandoned cart scenario

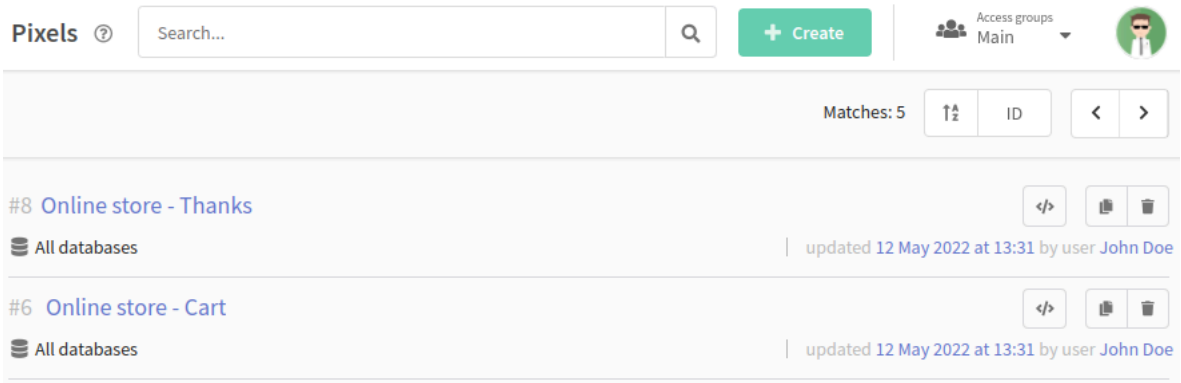
Visitors to an online store often add items to their carts but do not place an order. However, if you remin after a while, the chances of completing the purchase will increase. This guide is here to help you set up automatically send reminder emails to your customers.

Step 1: Place tracking pixels on website pages

Let's say that a user adds items to the cart, goes to the cart, pays for the purchase and then is redirecte will receive a reminder about the abandoned cart if they visit the cart page, but do not get to a thank-yo complete the checkout. Pixels will help you track visits to certain website pages.

In the Pixels section, create 2 pixels:

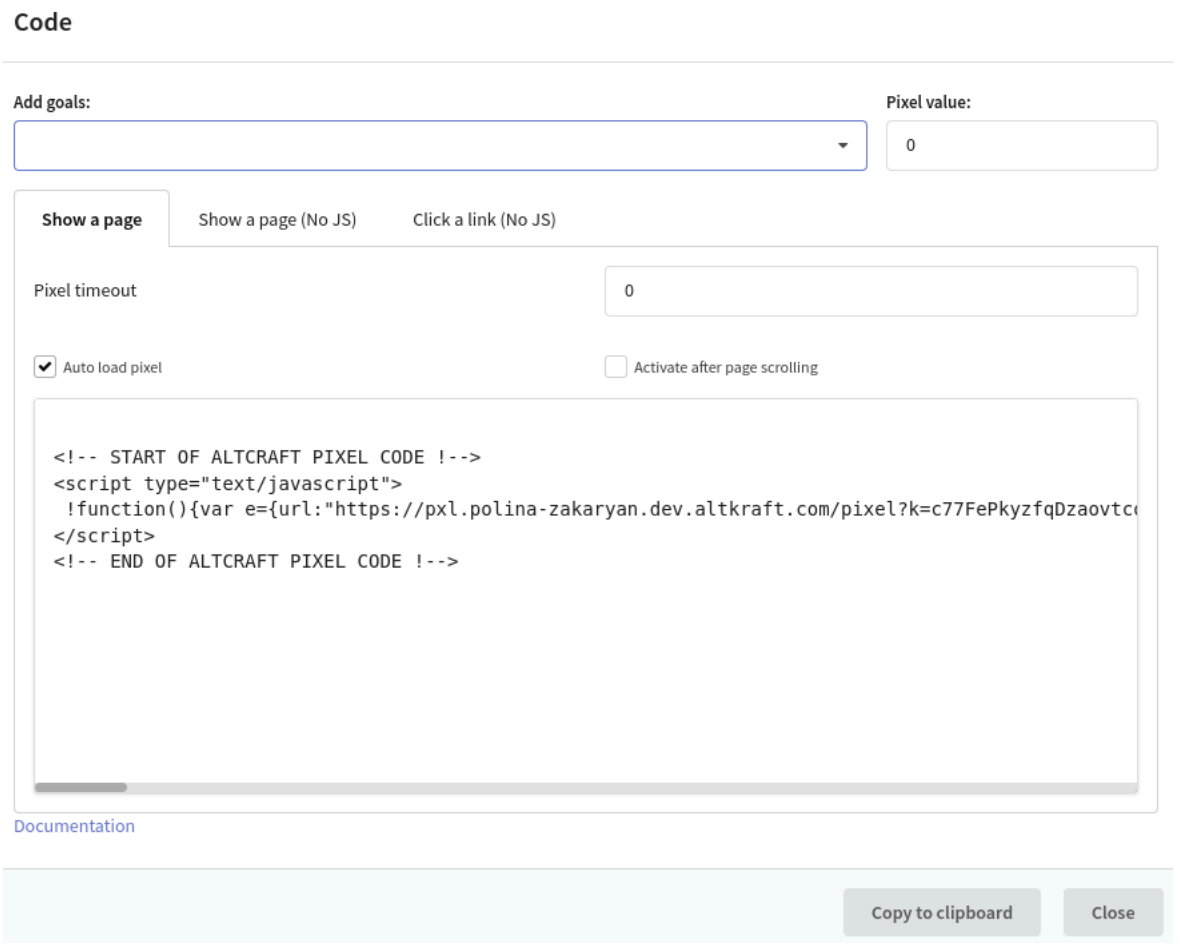
- **Online store - Cart: it tracks visits to the cart page**
- **Online store - Thanks: it tracks visits to a thank-you page**



TIP

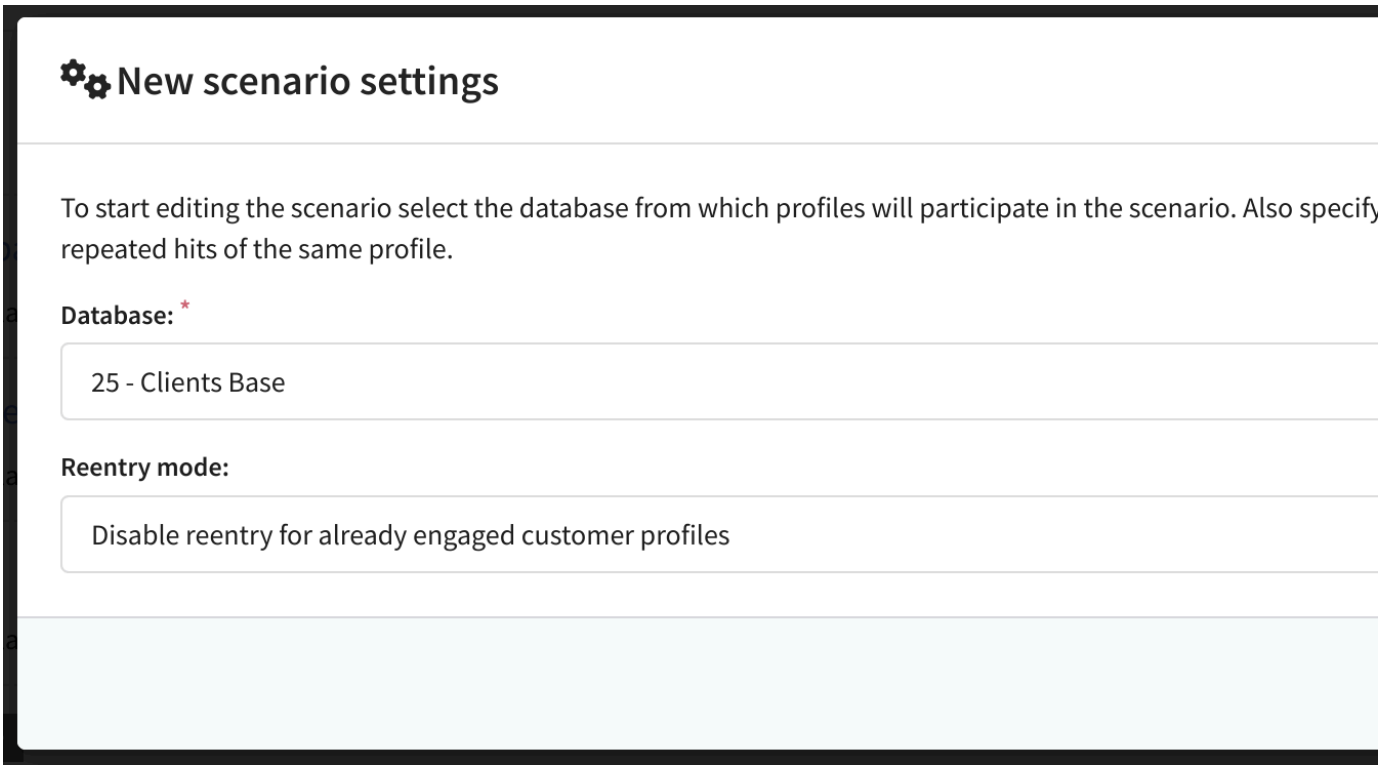
In some situations, a website visitor may open the cart several times. If you don't want to spam your cli reminders in one day, block reactivation of the pixel for the cart for 7200 minutes (5 days).

To view the pixel code, click "Get pixel code". Copy the code from the window and place it on the websit

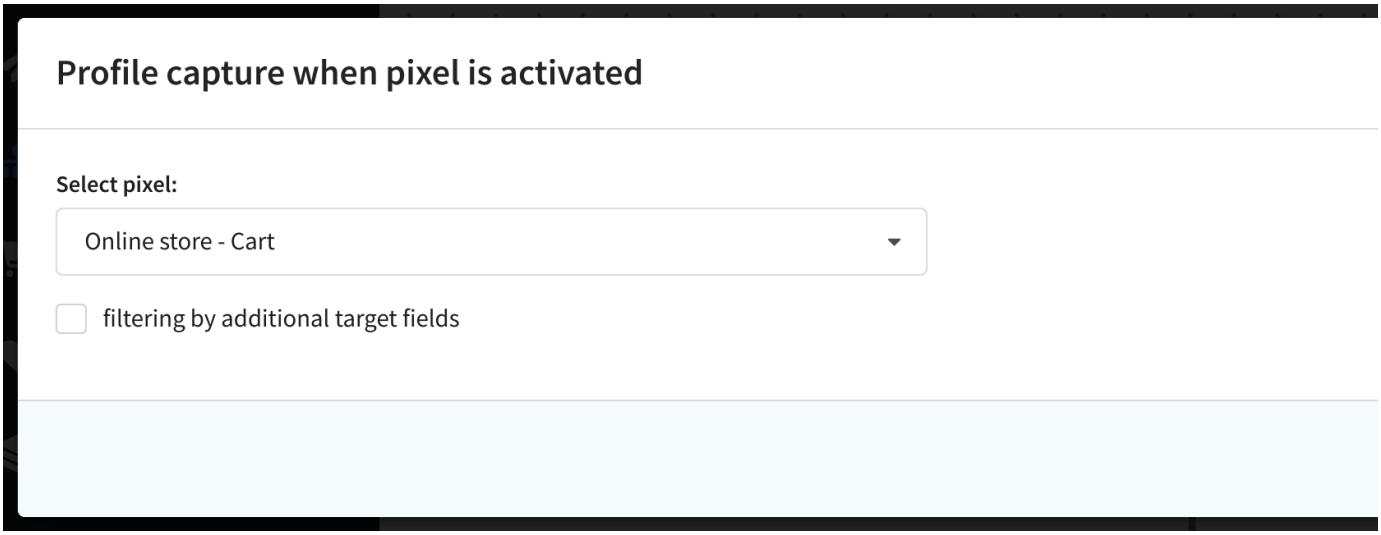


Step 2: Configure customer profile capture

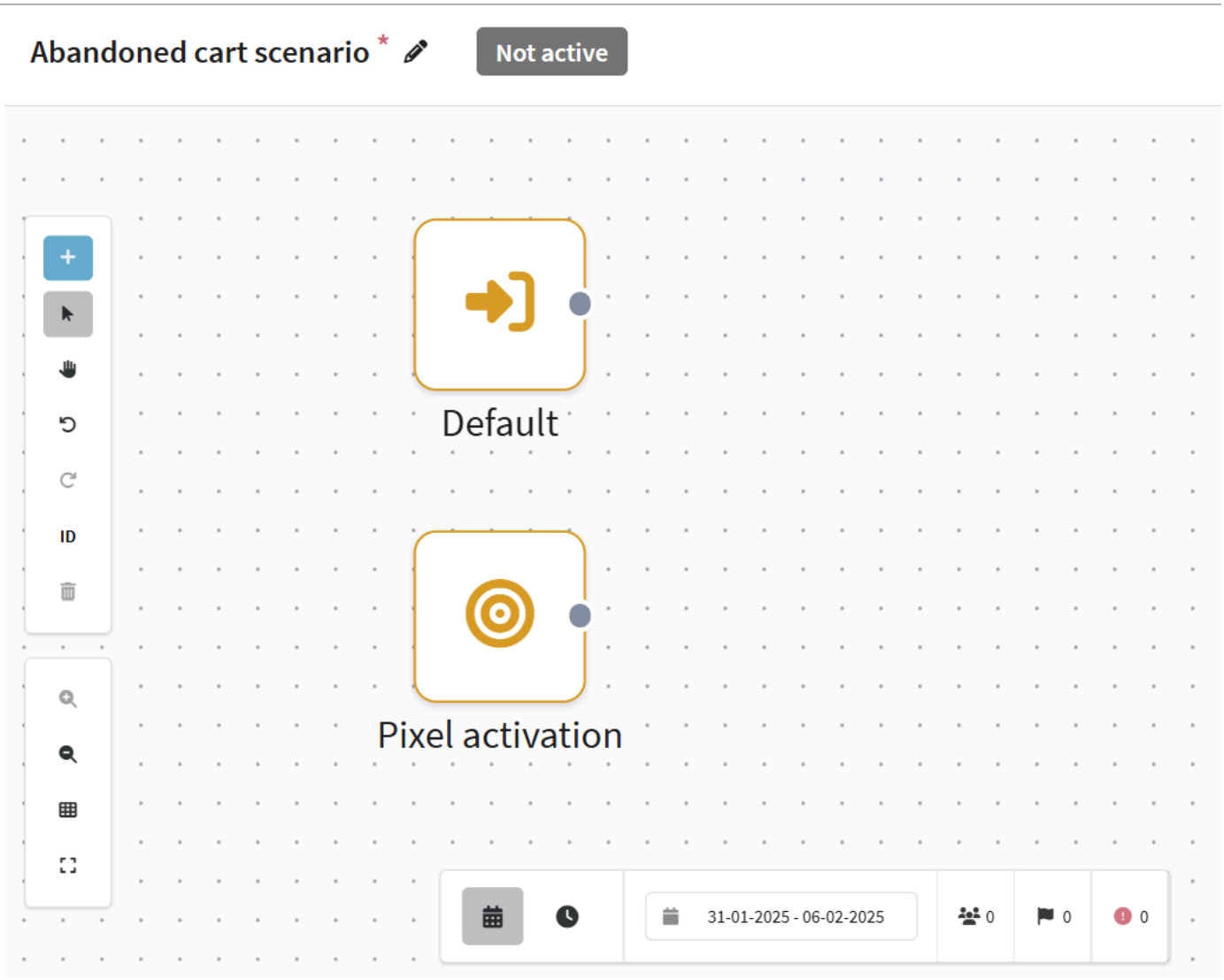
Create a new scenario. Specify the base - the profiles of these clients will be hit in the scenario if the ca to do with the profile in case of repeated hits. The option Disable reentry for already engaged customer customer will be able to get into it again, but only when he is going to make the next purchase after som



To get users into the scenario, add a trigger element When a pixel is activated. The pixel we placed on t (Online store - Cart) should be activated.

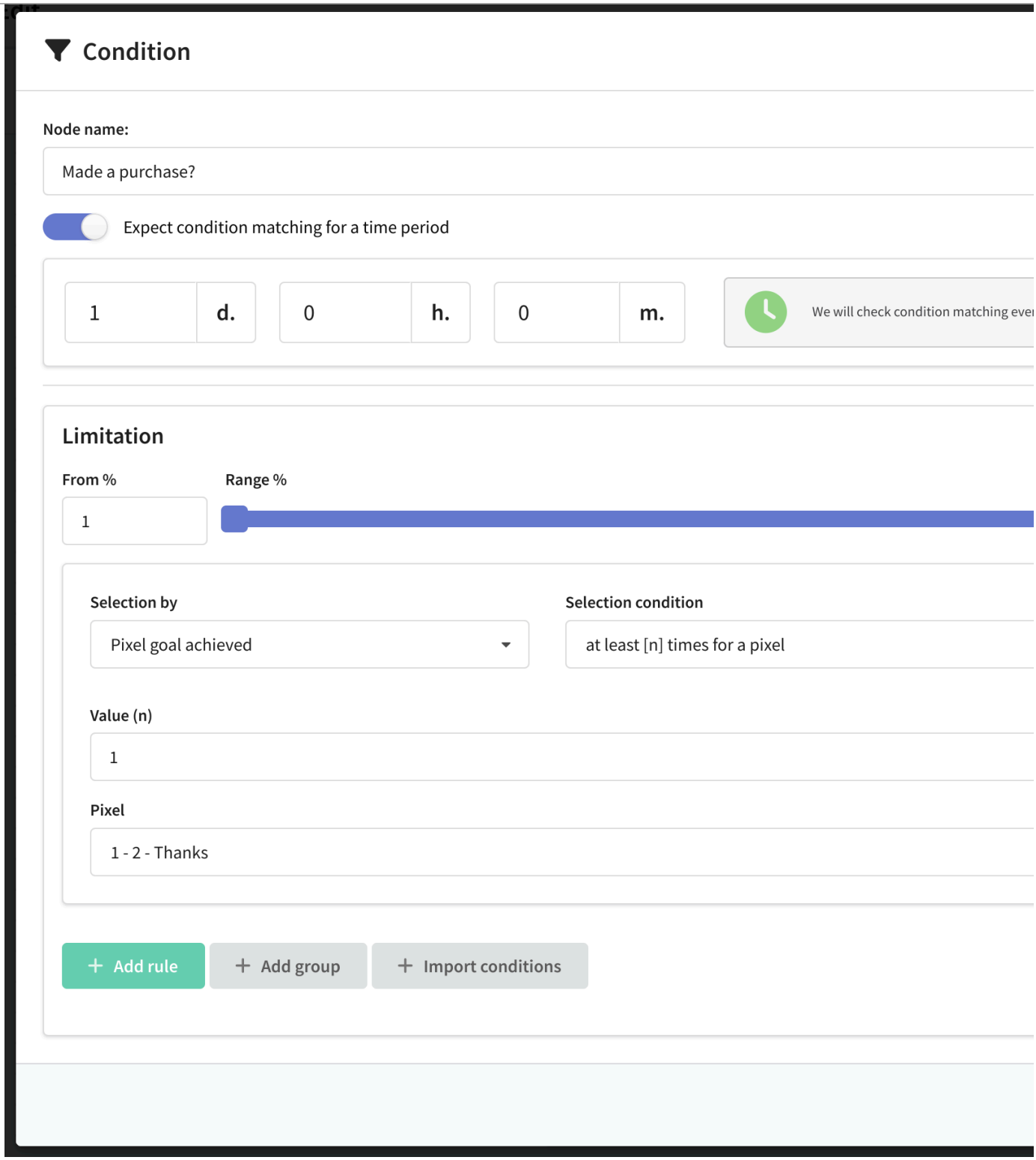


In any script, an Default Import element is automatically created. It is required for importing profiles ma from another script. This is a system element, it is impossible to delete it:

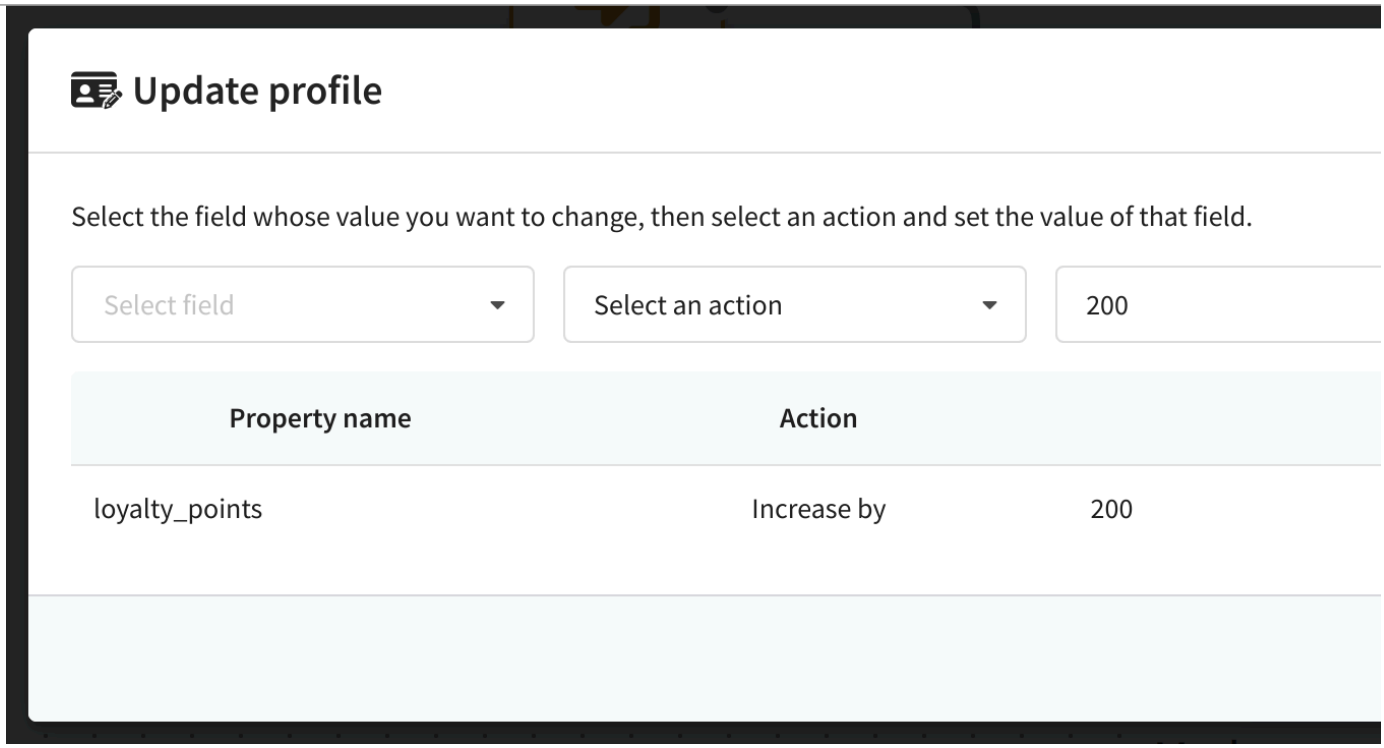


Step 3. Set up a workflow for the profiles that have completed their orders

Next, add the Condition element to check whether the user has made a purchase or not. The condition w thank you page, i.e. the Online store - Thanks pixel was activated. When customizing the element, toggle specified time and specify the waiting time - 1 day. The platform will check if the customer has made a p



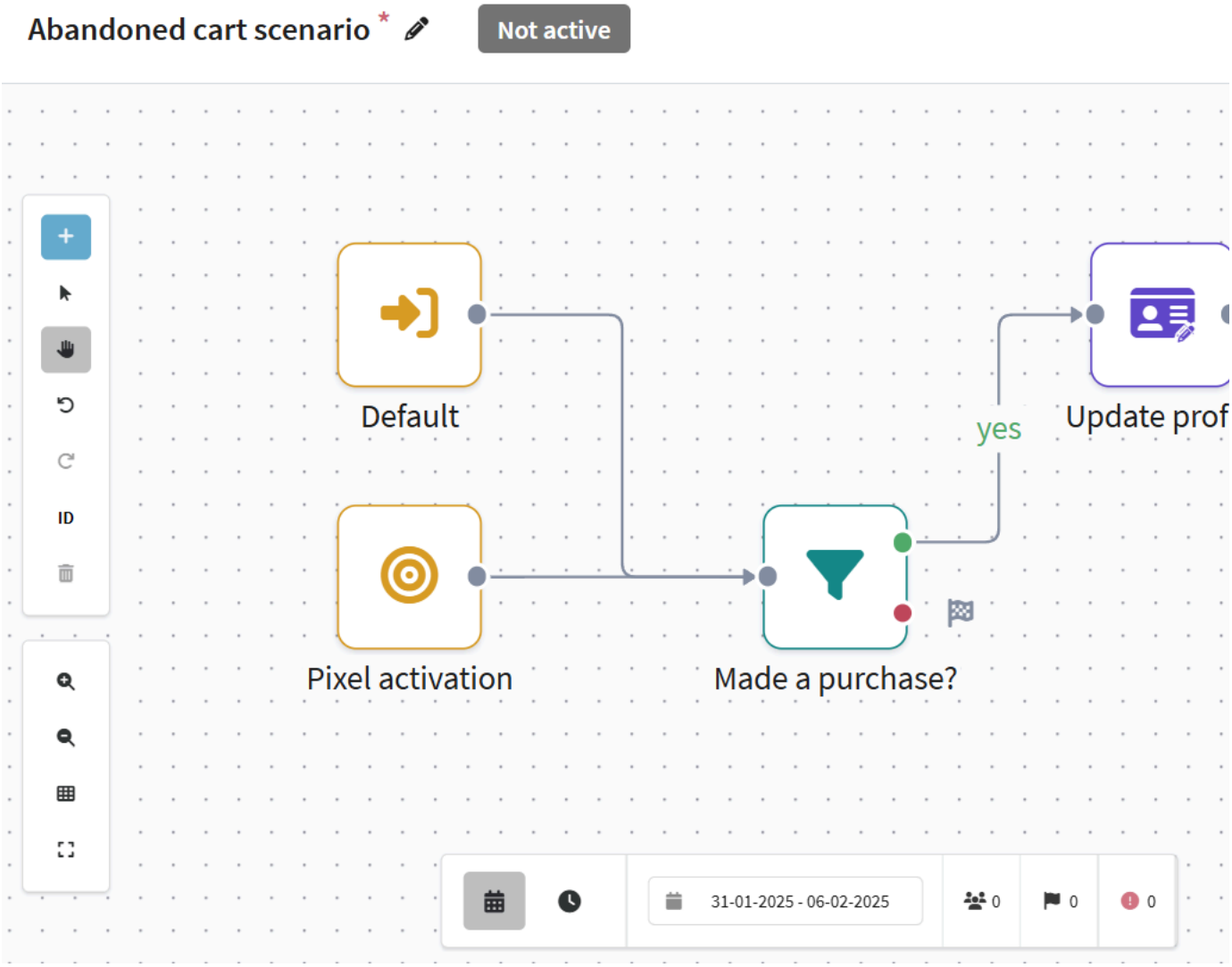
If the customer has made a purchase, award him bonus points. To do this, there should be an [additional] such as loyalty_points. Add a Update Data element to the scenario and increment the field value:



TIP

If you want to change the number of bonus points depending on the purchase amount, use the "Take va instead of “Increase by”. To do this, you need to add the API Request element in the previous step and s required data.

So, the positive branch of scenario is ready:



Step 4. Set up a workflow for the profiles that have left the website

Let's go back to the customers who left the site without completing the checkout. After the condition ele them to the red branch.

Using the API call, we need to pass the data about the products abandoned in the cart to the platform, a email template. Add an API Request element. Specify the URL where the data about the items in the cart

Next, activate the field "Save the http response in {apicontent} field" and specify the name of the field w name of the field can be arbitrary, for example, `cart_items`. It is this field that will later be used when sub template. Add the necessary query parameters, such as id or profile email, which will be used to find the

Node name: *

API Request

Authorization

Authorization type:

Without authorization

Request settings

Request type:

GET

URL: *

http://api.example.org/data/cart

Query parameters

Select the property whose value you want to change, then select an action and specify the value of that prope

Enter property name

custom value

Property name	Action	
custom_id	from subscriber	custom_id

Retry until success maximum 2 times every 15 minutes

Response

Result type:

Status 200

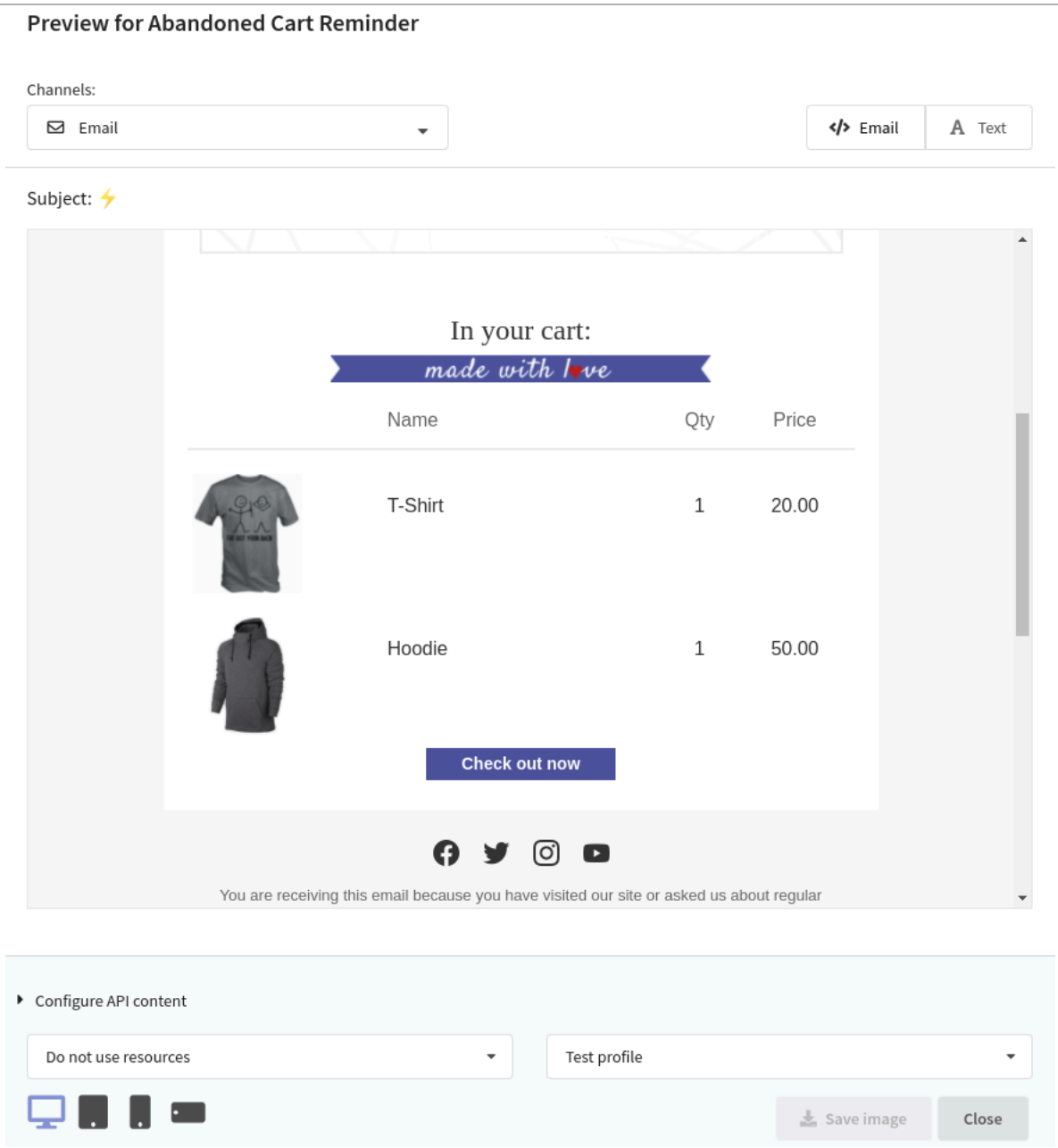
Save the http response in {apicontent} field

cart_items

Save the scenario at this point. All that remains is to create a message template and add the distributior

Step 5: Create an email template

Go to the Templates section and create a new template for the email sending channel:



The content of the email should be dynamically updated depending on the recipient. The `{apicontent}` var data that you receive using an API call node. For example, a JSON object with data about products in the

Example of a JSON cart

```
[
  {
    "product_id": 1,
    "product_name": "T-Shirt",
    "product_price": "20.00",
    "product_quantity": 1,
    "product_image_url": "https://example.com/pic/jersey.png"
  },
  {
    "product_id": 2,
    "product_name": "Hoodie",
    "product_price": "50.00",
    "product_quantity": 1,
    "product_image_url": "https://example.com/pic/hoodie.png"
  }
]
```

Once you've saved a JSON object in the apicontent field, you'll have access to the elements of this objec `{apicontent.cart_items.field_name1}`. To add a variable,select from the list (`</>`) Services —> API content.

If the product data is in an array, use loops:

```
In your cart:

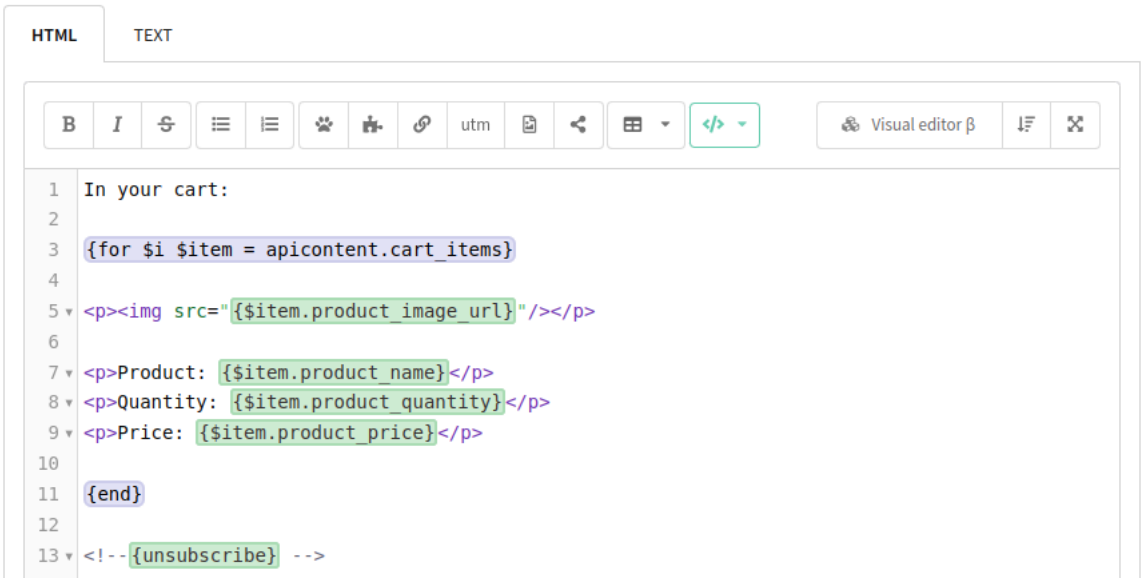
{for $i $item = apicontent.cart_items}

<p><img class="img" src="{$item.product_image_url}"/></p>

<p>Name: {$item.product_name}</p>
<p>Quantity: {$item.product_quantity}</p>
<p>Price: {$item.product_price}</p>

{end}

<!--{unsubscribe} -->
```



Step 6: Check the preview of the apicontent variable

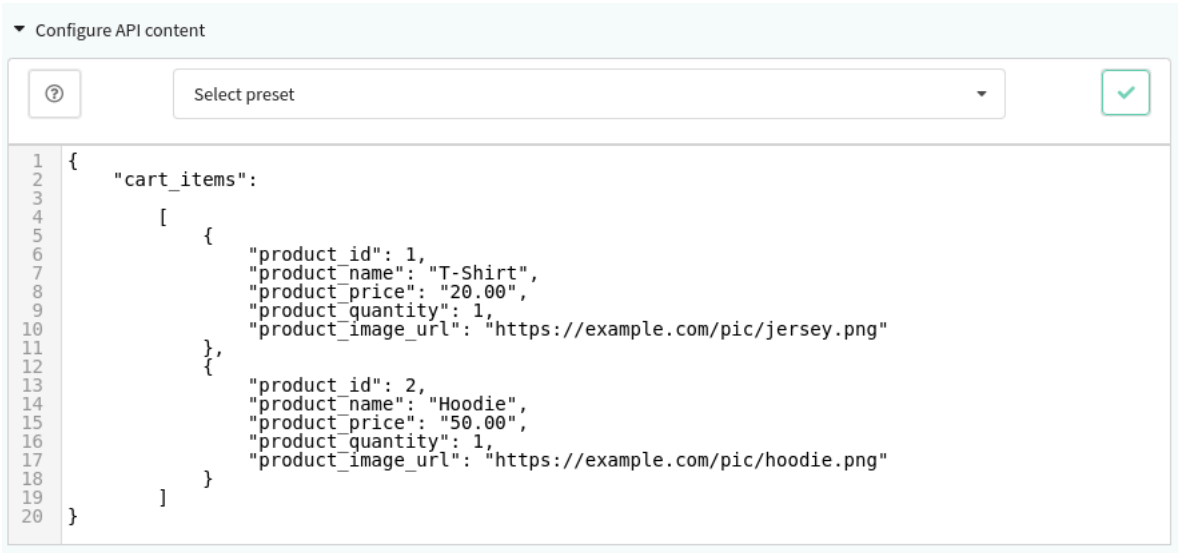
When adding the apicontent variable, the message preview will not work because the variable doesn't c know what message will look like, set up the preview window.

Open the preview mode

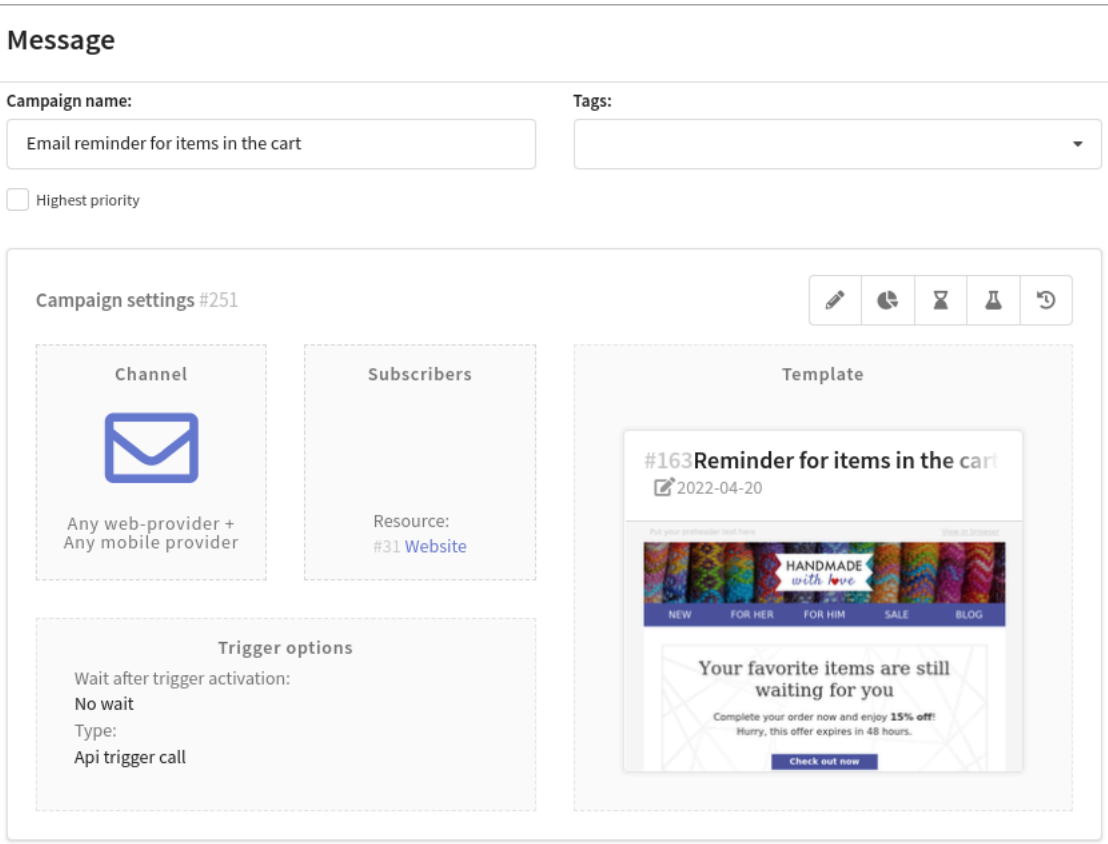
and find "Configure API content" below. Add a JSON object similar to what you'll receive after the API re

Please note that the test array of products must be stored in the `cart_items` field.

Click on the check mark. The values of the field will be added to the template:



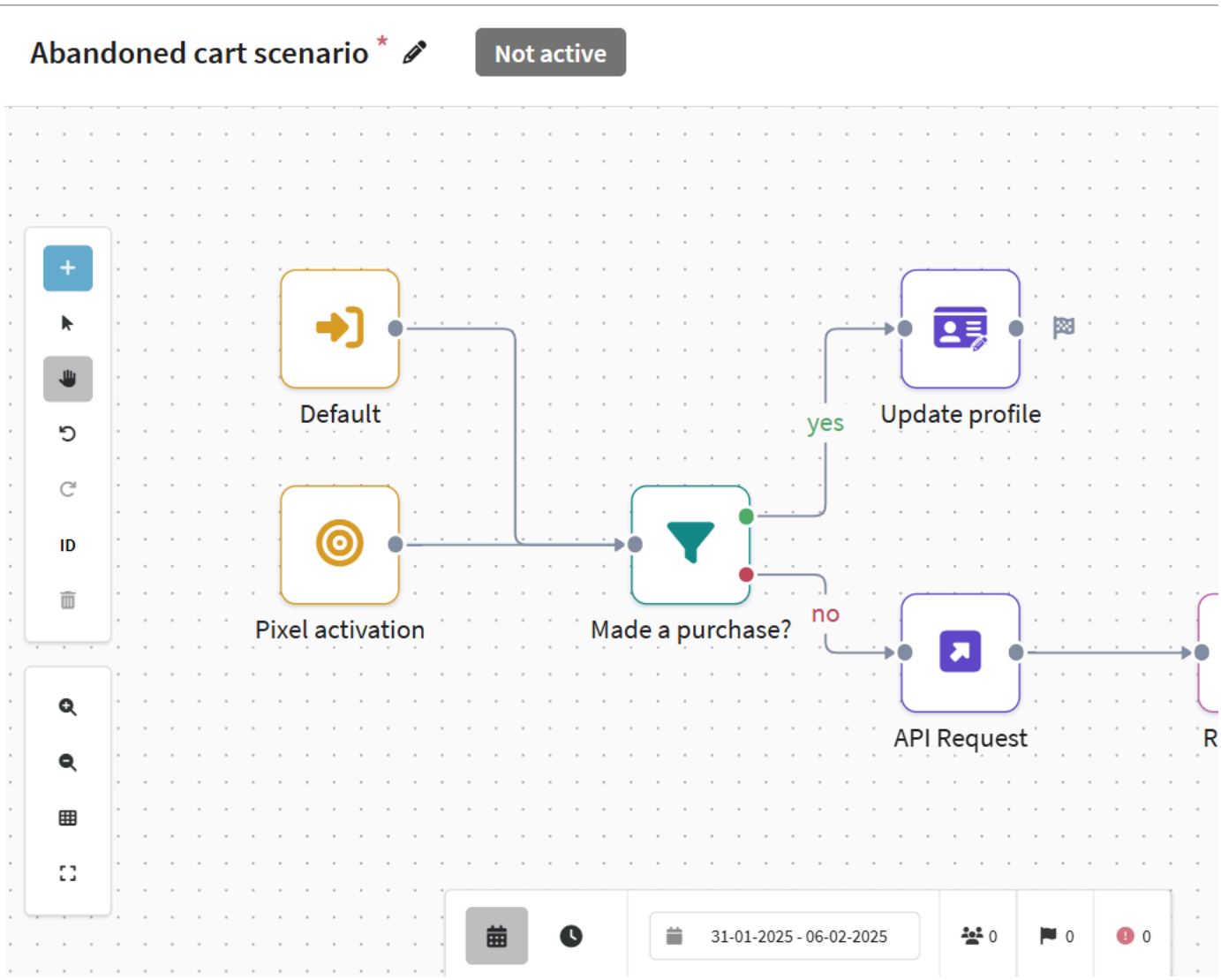
Go back to the scenario and add an Email element from Channels category. Select email channel, the re template in the settings:



Save the scenario by clicking the appropriate button at the top of the screen.

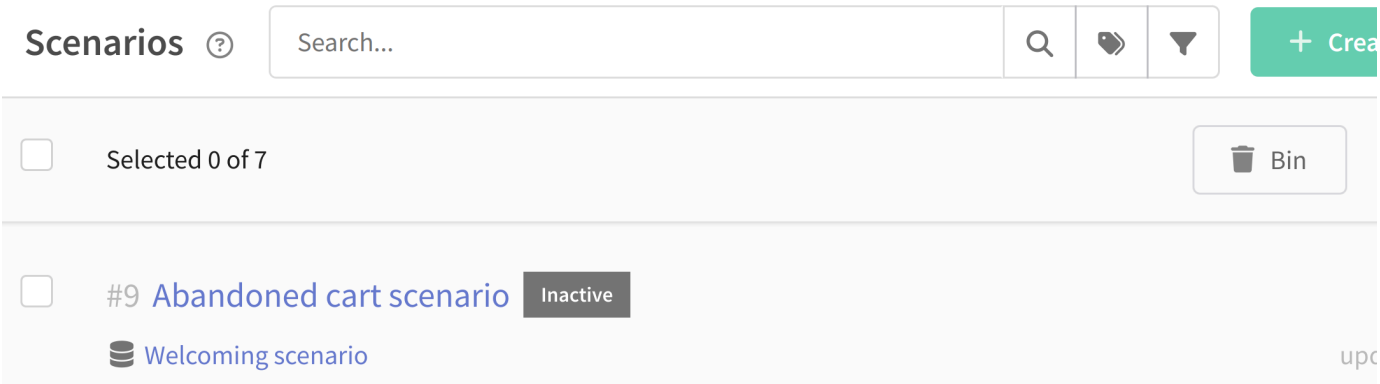
Step 7. Activate the scenario and evaluate its effectiveness

The finished scenario looks like this:



The scenario can be expanded. For example, if the subscriber has not completed the purchase, give ther of related products.

After saving, activate the scenario in the quick actions menu:



You can evaluate the effectiveness of the scenario by clicking on the corresponding icon in the quick act

Cycle Handling in Automation Scenarios

When building automation scenarios, profiles may pass through the same sequence of nodes multiple tin back to previous nodes or using the Jump to Scenario node, which sends a profile to another scenario. The platform automatically detects such cycles and protects against infinite scenario execution.

What Is a Cycle in a Scenario

A cycle occurs when a profile passes through the same sequence of nodes repeatedly. This can happen visited nodes, through conditional transitions, or when using the Jump to Scenario node if the profile ret example, a profile follows the path A — B — C — A and starts the cycle again.

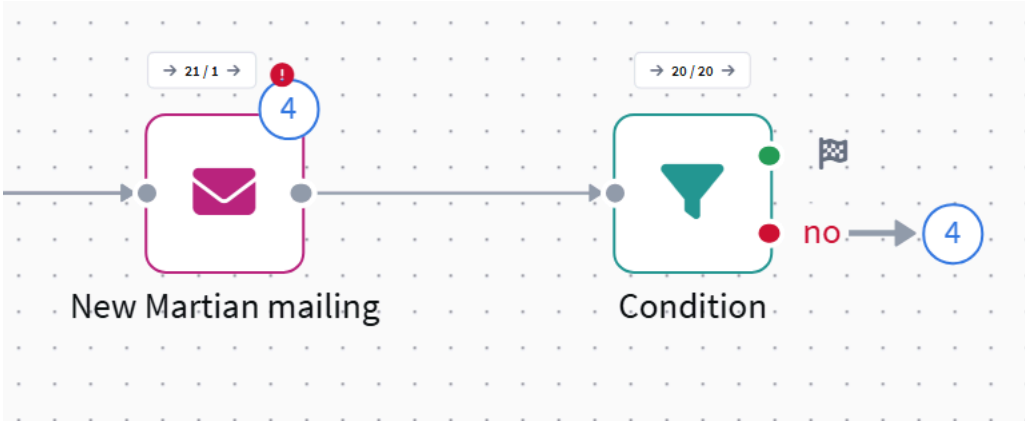
> INFO
>
> **Waiting on a single node is not considered a cycle. A profile can stay on the Condition node with a delay needed — this will not be counted as a cycle repetition.**

Cycle Limit

By default, the platform limits the number of repetitions of a single cycle to 20. The platform administrat platform configuration file.

When the limit is reached:

- **Task execution for the given profile stops**
- **An error with the message `Cycle limit of 20 has been reached` will appear on the last node of the cycle w detection**
- **The profile is no longer processed by this scenario**



> CAUTION
>
> **If a profile has hit the cycle limit, check the scenario for incorrect conditions that prevent the profile fr**

> INFO: Please note
>
> **When a cycle is repeated (before reaching the limit), the platform automatically adds a delay before re- profile has completed cycle A — B — A, it will wait 60 seconds before the next pass (before node A). Th the delay value.**

Cross-Scenario Protection

When transitioning between different scenarios through the Jump to Scenario node, the platform counts visited the target scenario. If a profile has visited the same scenario more than 20 times, further transiti This prevents a situation where two scenarios "pass" a profile back and forth infinitely.

Recommendations

- **Build exit logic into cycles. When designing scenarios with cycles and backward transitions, make su cycle when certain conditions are met.**
- **Check conditions when the limit error occurs. If a profile has hit the cycle limit, check whether the s incorrect node conditions.**
- **Avoid mutual transitions between scenarios. If two scenarios pass profiles back and forth through th the profile has a condition to exit this chain.**

TIP

When designing scenarios with cycles, it is useful to test them on a small segment of profiles to ensure the cycle correctly.

High-priority scenarios

By default, tasks of all scenarios are processed by shared workers in the order they arrive. If the platfor high, tasks of a critical scenario may wait for processing along with the rest.

To have a scenario's tasks processed before regular ones, you can mark the scenario as high-priority. A s cluster is allocated for it: these workers drain the priority queue first and only then pick up everything el

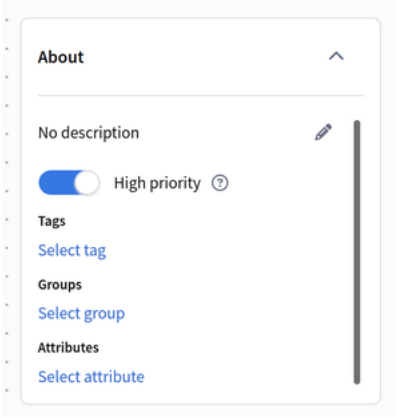
> INFO
>
> **Scenario priority is a separate mechanism from priority sending in mailings. Scenario priority speeds up execution), while priority sending speeds up message delivery in channels. They work independently ar**

When to use

Assign high priority to scenarios where processing delay is critical for the business — for example, scen immediately after a user action.

How to enable

To activate the option, open the scenario settings and enable the High priority toggle:



Then save the scenario. No service restart is required — the new value takes effect immediately.

You can change the priority at any time: turn the toggle off and save the scenario. The change affects or already in the scenario continue processing with the previous priority.

What happens when enabled

- **Dedicated processing. The scenario's tasks are processed by dedicated workers on the hosts that th for priority tasks. While high-priority tasks are present, these workers do not pick up regular ones. W the workers help with regular ones — the throughput of regular scenarios is not reduced.**
- **Priority sending in mailing nodes. In all Channels nodes of the scenario, the High priority toggle is au you cannot turn it off manually. In regular scenarios, the toggle is available for manual control. Togg does not overwrite the value the user set in a campaign: it is restored as soon as the scenario becon**

Limitations

- **Priority does not guarantee processing or delivery time. It sets the order in the queue but does not o limits: sending limits, channel queues, and mailing speed settings work as usual.**
- **Priority message sending is not available in all channels. Details — in Trigger mailing.**
- **Campaign processing inside a scenario is synchronous: a worker is occupied with it entirely, from re queue. Priority speeds up how quickly a worker gets to the campaign, but within the sending itself, t run in the general order.**
- **A node failure does not lead to loss or reprocessing of profiles: unfinished tasks are automatically pa**

Campaigns

INFO: Terminology update

Previously, all types of marketing Messages in the platform were called Campaigns. In the new platform Messages, while the new Campaigns represent a broader tool.

Campaigns are a tool for comprehensive management of marketing activities in a single workspace.

They allow you to configure audiences, segment databases, manage user profiles, control groups, and sc

Within campaigns, you can both create and transfer existing marketing activities: scenarios, broadcasts, Messages, and placements.

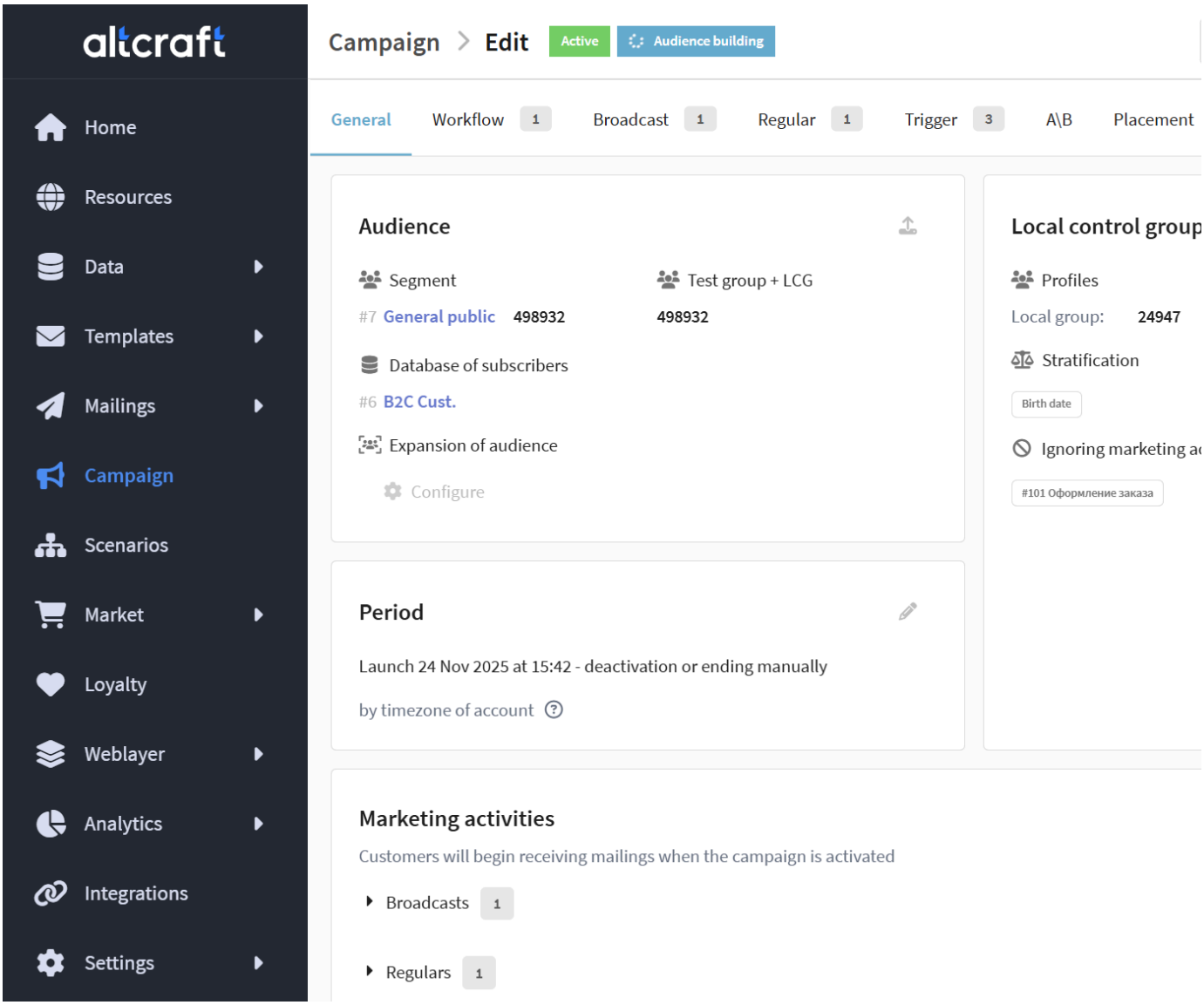
Working with Campaigns

Description

INFO: Terminology update

Previously, all types of marketing Messages in the platform were called Campaigns. In the new platform Messages, while the new Campaigns represent a broader tool.

Campaigns are a tool for uniting marketing activities into a single strategy and managing them from one broadcasts, regular and triggered Messages, scenarios, A/B tests, and placements. You can configure te marketing goals, track strategy performance, and manage the lifecycle of activities from a single interfa



Creating a campaign

Before creating a campaign, prepare a static or updatable segment for the target audience. It will be nee and determines who will participate in the activities.

The campaign audience is the current composition of the segment at the time of campaign activation an

- **Profile not in segment = profile not in campaign audience**

For example:

The campaign is configured with an updatable segment.

The segment is set to weekly update on Mondays with the condition “purchase within the last 7 days”:

A profile makes a purchase on April 9 (Sunday).

- **April 10 (Monday) — the nearest update. The profile enters the segment and therefore the campaign, last 7 days.**
- **April 17 (Monday) — the next update. Since there were no purchases from April 10 to April 17, the pr and is removed from the segment and, accordingly, from the campaign.**

For a static segment
A profile will also be removed from the campaign audience if it ceases to be a member of the static segr

- **manually deleted from the segment;**
- **deleted by conditions;**
- **deleted via API;**
- **deleted via a scenario;**
- **any other way of excluding a profile from a static segment.**

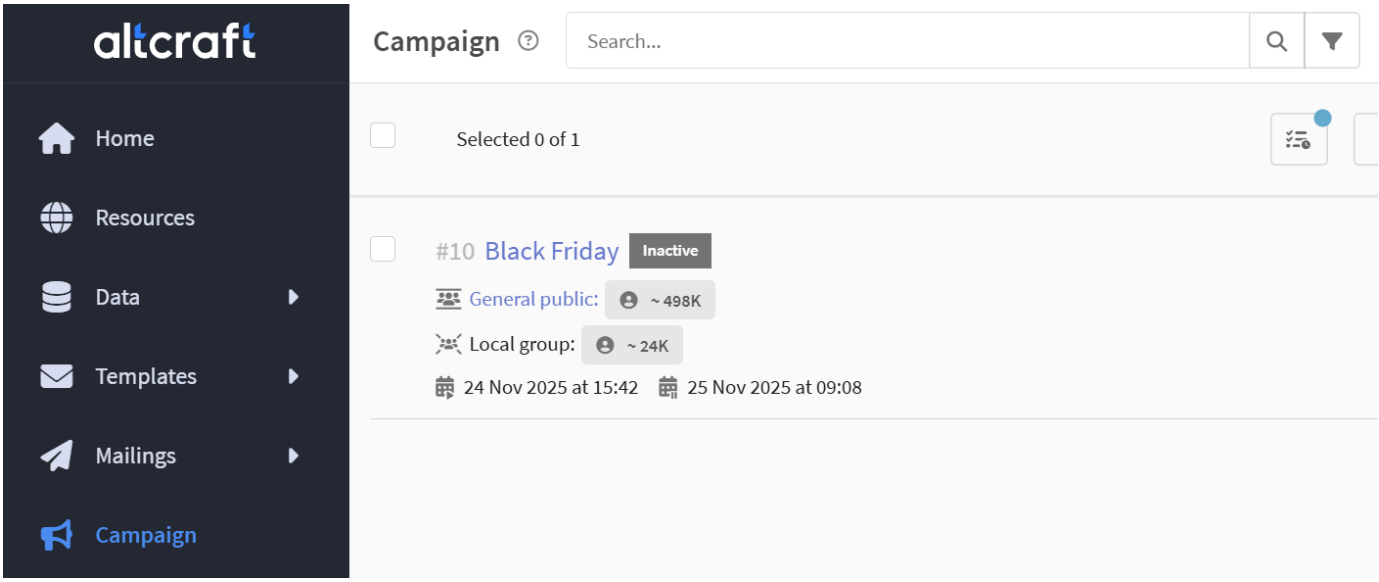
Profile exit from campaign If a profile leaves the segment audience, it will also leave the campaign audie

- **The profile receives a “Campaign exit” event;**
- **All active campaign scenarios for this profile are deactivated;**
- **If the profile later re-enters the segment, it will be added to the campaign as a new one and will und**

INFO: Please note:

When using an updatable segment as the main campaign segment, the audience expansion function is

Open the Campaigns section in the main menu and click `+ Create` in the campaign list:



On the first step, specify the name, select the profile database and a static or updatable segment that d
If mandatory attributes are configured in the system, you must also select them at this stage.

NOTE: Please note

The selected segment defines the campaign audience. Upon activation, the system splits it into two pa

- **Test group — profiles that will receive campaign activities;**
- **Local control group (LCG) — profiles that do not receive activities, if LCG is enabled in the settings.**

![img](/img/attachments/super-campaigns/eng/scampaign_create.png)

CAUTION

After creating a campaign, you cannot change the segment or profile database.

Managing marketing activities

In the Marketing activities section, you can manage all activities that are part of the campaign.
Activities can be created directly inside the campaign or existing ones can be added.

NOTE: Please note

When selecting an activity, the following are not included in the list:

- **activities from another profile database;**
- **Messages and scenarios with status “Active” or “Completed”;**
- **scenarios with active profiles;**
- **activities already used in other campaigns;**
- **broadcasts and regular Messages that have been activated at least once.**

![img](/img/attachments/super-campaigns/eng/marketing-activities.png)

The following activity types are available:

- **Broadcasts — one-time mass sends;**
- **Regular Messages — scheduled sends;**
- **Triggered Messages — automated sends based on events;**
- **A/B tests — variant comparison to achieve marketing goals;**
- **Scenarios — multi-step automations;**
- **Placements — content display to subscribers on a selected resource.**

When creating an activity, you can select any mailing segment if you need to refine the audience. You ca always matches the campaign database. If the selected activity segment differs from the campaign segı intersection between them: the message will be sent only to profiles that belong to both segments.

How intersection works

Partial segment overlap

The campaign is configured with the profile database `CRM_Base` and segment `Loyal customers`, which contai
A marketer wants to send an additional message only to those who recently placed an order. To do this, select a different segment — `Recent buyers`, which has 60 profiles. Of those, 40 overlap with users from th
In this case, the message will be received only by these 40 profiles — those who belong to both the `Loya` segments. Other campaign participants will not receive the message.

![img](/img/attachments/super-campaigns/eng/activities_scampaign.png)

This is useful when you need to configure a separate activity for a specific group of users without chang For example, you can select a narrower segment — active customers, promotion participants, or users fı them a message related to their interests or current marketing goals.

Segments do not overlap

The campaign uses the profile database `CRM_Base` and segment `Loyal customers` (100 profiles).
A broadcast is created with the segment `New registrations` (50 profiles), which includes users who only cr taken any actions.

Since there are no common users between the segments, no one will receive the message.

![img](/img/attachments/super-campaigns/eng/activities_scampaign_2.png)

Moving an activity

You can also add or remove an activity from the Messages menu. To do this, find the desired activity in t

![img](/img/attachments/super-campaigns/eng/activity_shiftng.png)

In the opened window, select where to move the activity — to the external list or to a specific campaign:

![img](/img/attachments/super-campaigns/eng/where-to-shift.png)

> INFO: Please note
>
> **Not all activities can be moved:**
>
> - **Activities with status “Active” or that have already been launched;**
> - **Scenarios with profiles;**
> - **An activity cannot be moved to a completed campaign.**

Configuring local control group (LCG)

What is a Local control group

Local control group (LCG) is a part of the campaign audience that does not receive activities from this c audience management and for analyzing the impact of campaigns on users. Currently, the LCG analytics platform.

> TIP: LCG and GCG
>
> **Local control group (LCG) works only within a single campaign and is configured in its settings.**
> **Global control group (GCG) applies to the entire profile database and excludes selected users from all a**
> **If a profile belongs to GCG, it cannot participate in a campaign, except when this campaign is added to GCG is configured in the profile database settings. More details in the separate article.**

Basic LCG setup

In the campaign settings, you can select what portion of the audience will fall into the local control grou

![img](/img/attachments/super-campaigns/eng/lcg_settings_12.png)

To configure LCG, specify the percentage of users to exclude from sends within the campaign. If you ena can set not a percentage but an exact number of profiles that will enter the LCG.

![img](/img/attachments/super-campaigns/eng/gg_settings.png)

TIP

If you plan to use a local control group, it is recommended to configure LCG before launching the camp

After changing the parameters, Audience building starts: during this time, all broadcasts and regular M process is complete.

To avoid delays and errors in campaign operation, configure the control group in advance, and then act

If the campaign is configured with an updatable segment, Audience building will be triggered with each

After campaign activation and completion of labeling, only new users will enter the local control group. the test group at the time of activation are not moved to LCG — the test audience composition is fixed

Stratification

Stratification is a method for more precise formation of the local control group.
With stratification, the audience is divided into subgroups by selected characteristics (for example, age, profiles is randomly selected from each subgroup for the LCG.

Stratification configuration is available in the LCG settings window via a dropdown menu. Select one or specify conditions for each group (for example, value ranges or lists of items):

![img](/img/attachments/super-campaigns/eng/strat-settings-eng.png)

Usage example:
A marketer needs to include men and women from two age groups in the control group:

- **from 20 to 25 years old (date of birth from 01.01.2000 to 01.01.2005);**
- **from 25 to 40 years old (date of birth from 01.01.1985 to 31.12.1999).**

To do this, they select the fields Gender and Date of birth, define two date-of-birth ranges, and add gend system will form four strata — men 20–25, women 20–25, men 25–40, and women 25–40 — and distribute control groups according to the settings.

CAUTION: Please note

It is not recommended to change stratification settings in an active campaign or for an already formed homogeneity of the control group may change.

More details on system behavior for each type of change — in the article Local control groups, section campaign.

Ignoring marketing activities

Ignoring marketing activities allows you to select Messages or scenarios within a campaign that will be participants. This is useful for messages that must reach the entire audience regardless of control grou need order notifications to reach the recipient even if they are in the control group.

To do this, in the LCG settings, select the desired activities via the dropdown menu and mark them as e sent to all users without restrictions:

![img](/img/attachments/super-campaigns/eng/ignoring-acrtivities.png)

More details on configuration in the article Local Control Groups.

Status management

Campaign activation

CAUTION

Activation is available only for campaigns that have at least one marketing activity. If no activities are be launched.

To activate the campaign, click the ▶ button on the top panel of the editor. After clicking, a window for c open.

![img](/img/attachments/super-campaigns/eng/activation-settings-eng.png)

Two activation options are available:

- **Activate now — the campaign launches immediately after saving. The process of labeling the audien begins, after which all activities within the campaign start working according to their settings.**
- **Activate on a specific date — the campaign is scheduled to launch at the specified time. The system campaign on the selected date and time.**

Be careful when selecting the date and time for launching the campaign. If you specify a past date, the c indefinitely and will not display a warning.

NOTE

Activating a campaign does not automatically launch activities inside it.
If Messages, scenarios, or other activities are in the “Inactive” status, they must be launched manually.

Completion settings

In the same window, you can set conditions for automatic deactivation or completion of the campaign:

- **Deactivation — temporary stop of the campaign on a specified date. All active Messages are paused membership.**
- **Completion — full end of the campaign on a specified date. All profiles are excluded from the audien**

> NOTE
>
> **A completed campaign cannot be activated again.**

More details on campaign statuses

Status	Description and impact on activities
Inactive	Assigned after campaign creation and remains until launch. Activities within the campaign do not run.
Audience building	Audience calculation and distribution of profiles between test and control groups is in progress. During calculation, broadcasts and regular Messa
Waiting	The campaign is activated with a delayed start. The campaign launch will begin automatically on the specified date.
Active	The campaign is launched, the audience is calculated. Activities run according to their settings.
Audience exclusion	The system excludes profiles from the campaign and detaches them from scenarios. New Messages do not launch.
Completed	The campaign is fully stopped. Activities stop working, re-launch is not possible.

Working with profiles

Forming test and control groups

After campaign activation, the Audience building process begins — the system distributes profiles betwe control group (LCG), if it is configured.

If LCG is not used, the entire campaign audience becomes the test group.

> NOTE
>
> **Audience building is triggered only after activation of the campaign.**

Audience expansion

Audience expansion is a mechanism that allows automatically adding new users to a campaign if they pa This mechanism helps increase campaign reach and keep the audience up to date without manual segm

> INFO: Please note:
>
> **When using an updatable segment as the main campaign segment, the audience expansion function is campaign audience only as a result of segment recalculation.**

When an activity runs with expansion enabled, the system checks whether recipients belong to the cam participating. New profiles are automatically distributed between the test group and the local control gr the test group will receive the activity, while LCG participants will not.

![img](/img/attachments/super-campaigns/eng/auditory.png)

Audience expansion examples

Scenario without audience expansion

The campaign uses the profile database `CRM_Base` and segment `Loyal customers` (100 profiles).
A broadcast is created with the segment `New registrations` (50 profiles). These users do not belong to the main segment.

When the broadcast is launched, the message will not be sent to anyone, because there is no intersectio profiles are not in the campaign.

![img](/img/attachments/super-campaigns/eng/activities_scampaign_2.png)

Scenario with audience expansion

If you add the broadcast to the Audience expansion list in the campaign settings, the system will include `registrations` segment in the campaign.

After that, profiles will be automatically distributed between the test group and the local control group.

Users from the test group will receive the message, while those who fall into LCG will not receive activit

Thus, an activity added to the audience expansion list not only performs the send but also replenishes th

More details on configuring and using this mechanism in the separate article on audience expansion.

Manual profile addition

INFO

Manual profile addition to the campaign audience is available only for static segments.

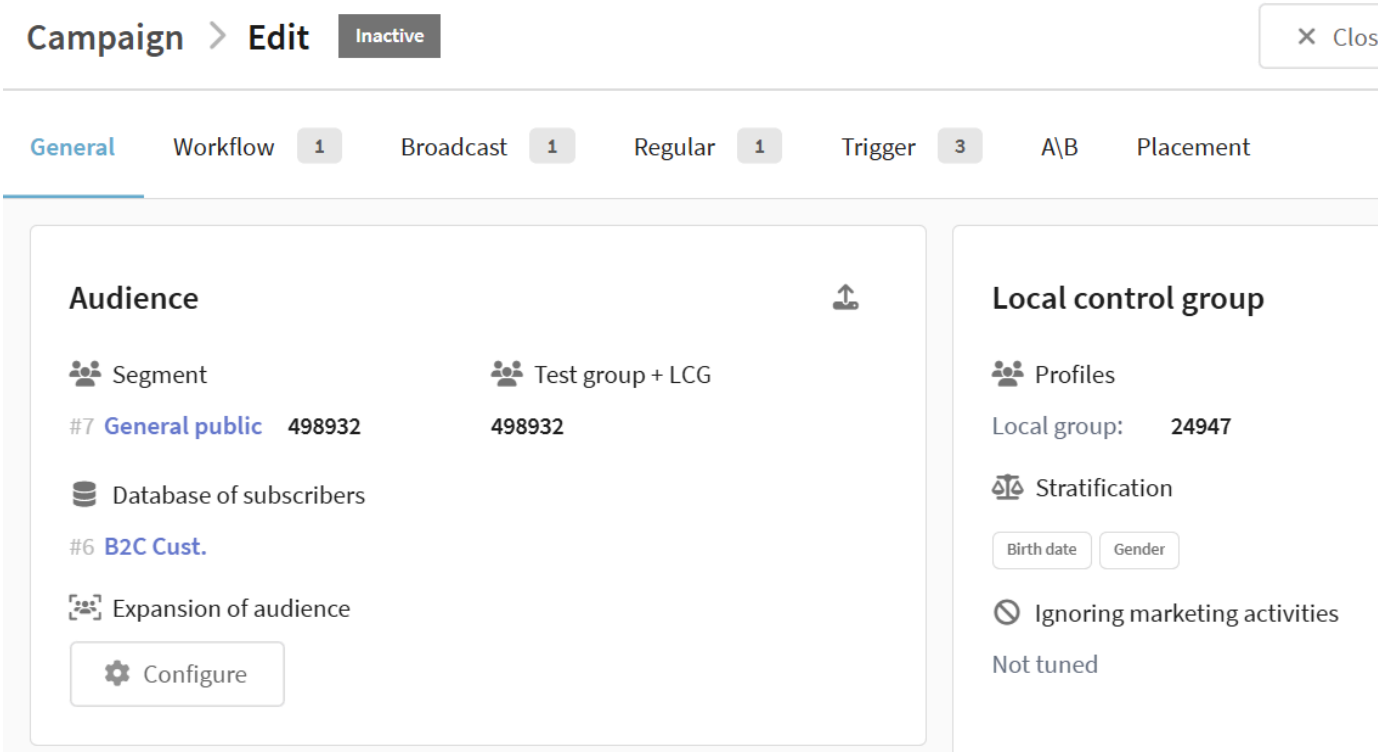
If needed, individual profiles can be manually included in the campaign, for example for testing or checki performed from the profile card: in the Interactions block, Campaigns section, select the desired campai

![img](/img/attachments/super-campaigns/eng/add-profile.png)

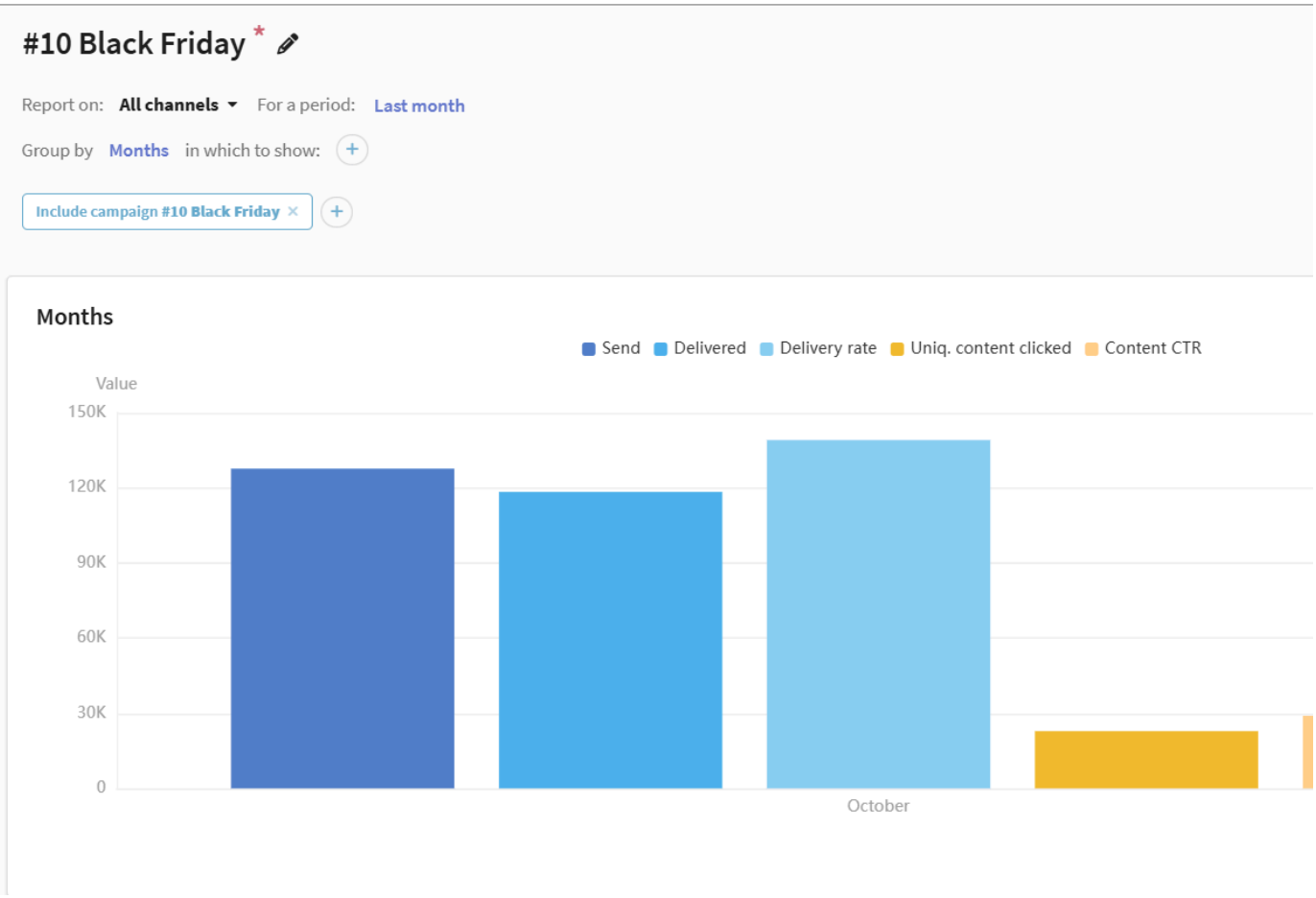
To make the profile participate in activities, check the box Send profile to test group even if it is in the c profile will be forcibly placed in the test group, even if it belonged to the local or global control groups.

Campaign analytics

To view the campaign report, click the Analytics button on the top panel of the editor.



After clicking, the system will automatically open the Analytics section with a filter already applied for t In this report, you can view metrics for each mailing and scenario included in the campaign:



Additional features

Profile export

Export of profiles from the campaign is available with various filtering parameters. The function is locat

![img](/img/attachments/super-campaigns/eng/export.png)

Available export parameters:

- **Filtering by subscription status**
- **Audience selection (Test group, Local control group, Global control group)**
- **Field selection for export (core, behavioral, system, custom)**
- **Profile quantity limit configuration**

> INFO
>
> **Profile export is available only after Audience building is complete.**

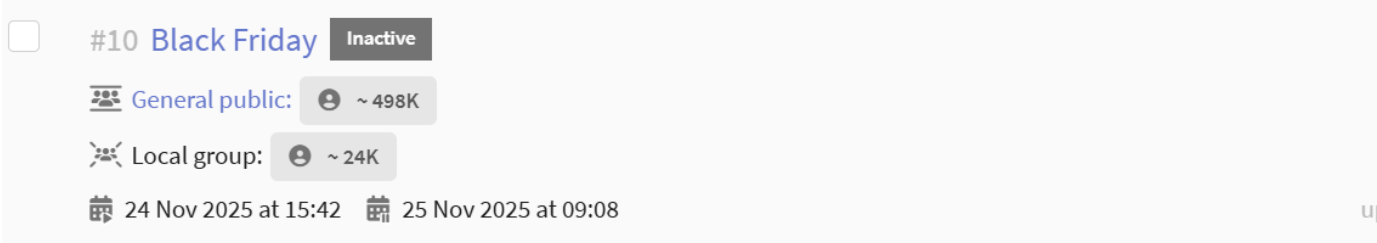
More details on working with data export in this article.

Campaign cloning

Cloning a campaign creates a full copy with all settings and marketing activities. This is useful when you a different audience or test changes on a copy of a working campaign.

To clone a campaign, in the campaign list click the icon

Clone in the context menu of the desired campaign.



After cloning, the editor of the new campaign will open, where you can change its parameters. All origina cloning, including the local control group, stratification, marketing activities, and audience expansion.

CAUTION: Please note

**The database cannot be changed during cloning — the new campaign will use the same database as the
the audience segment to configure the test group.**

**The new campaign is created in the “Inactive” status with inactive marketing activities. A separate activ
after configuring all parameters.**

Campaign deletion

**Deleting a campaign moves it, along with all related marketing activities, to the recycle bin. A campaign
“Inactive” status — provided there are no labeled profiles in it.**

To delete a campaign, open the campaign list and click

Delete campaign. Then confirm the action in the opened dialog window — the campaign will be moved to

CAUTION

**After deletion, the campaign and all activities included in it are moved to the recycle bin.
The campaign can be restored, but it will return in the Completed status and will not be able to be activ**

Local control groups (LCG)

Description

**Local control group (LCG) is a part of the campaign audience that does not participate in marketing acti
audience management and for analyzing the impact of campaigns on recipients. Currently, LCG reporting
Global control group (GCG) applies to the entire profile database and excludes a selected portion of prof
platform. If a profile belongs to GCG, it cannot participate in a campaign, except when this campaign is
configured in the profile database settings. More details in the separate article on GCG.**

Configuring local control group

**Configure the local control group via the Local control group widget on the main screen of the campaign
button:**
![img](/img/attachments/lcg/eng/lcg_settings_12.png)

Configuring group size

Specify the size of the control group in percent. Use fractional values for precise configuration:
![img](/img/attachments/lcg/eng/gg_settings.png)
**When the Audience limit option is enabled, you can set the maximum number of profiles for the control g
exceed this value, but may be smaller depending on the percentage and available audience.**

> INFO
>
> **Recommended LCG size — 5% of the campaign audience. Too small a group may not yield statistically s
> group reduces campaign reach.**

Stratification

Description

**Stratification is a method of random profile selection with preliminary division of the audience into subg
characteristics. The audience is divided into groups so that each reflects the composition of the entire p
From each group, a portion of profiles is randomly selected, proportional to the size of that group. This h
same distribution of characteristics as the test group, making campaign results more accurate. The max
the number of selected characteristics and the number of groups within them. These limits can be confi
Stratum — a group of conditions defined by one or more criteria. Each unique combination of criteria for
Field — a criterion by which strata will be defined.**

Stratification example

Suppose a marketer needs to divide the audience by three criteria:

- **City: Moscow, Saint Petersburg**
- **Gender: male, female**
- **Age: 18-30 years**

The system will create strata for all combinations of criteria:

- **Men aged 18-30 from Moscow**
- **Men aged 18-30 from Saint Petersburg**
- **Women aged 18-30 from Moscow**
- **Women aged 18-30 from Saint Petersburg**

Stratum share calculation: Suppose your target audience consists of:

- **50% men and 50% women**
- **40% aged 18-30 years**
- **60% from Moscow, 40% from Saint Petersburg**

**Then the share of the stratum “Men aged 18-30 from Moscow” of the entire audience will be: 0.5 (gender
Profiles for the control group are selected from each stratum proportionally to its calculated share.**

Stratification setup

**Stratification configuration is in the same window as the main LCG parameters and is available via a dro
![img](/img/attachments/lcg/eng/cg_stratification_settings.png)**
**Define strata by one or more fields, specifying conditions for each group: for example, date ranges, nume
list, etc. Stratification is configured through the user interface, where you can sequentially add strata an
![img](/img/attachments/lcg/eng/strat-settings-rus-copy.png)**

CAUTION: Please note

It is not recommended to change stratification settings in an active campaign or for an already formed l homogeneity of the control group may change.

More details on system behavior for each type of change — in the Stratification changes in an active ca

Stratification changes in an active campaign

Changing stratification settings in an active campaign has specific characteristics. The behavior of the s described below:

TIP

LCG is not replenished from the test group. The LCG is populated only with new profiles that enter the c — for example, when the segment is updated or the audience is expanded.

Removing a field value. Profiles that no longer match the remaining values are moved to the test group. T

Example

Stratification by field `region`: Moscow, Ryazan, Tula. The value “Tula” is removed.

Profiles from Tula are moved to the test group. Profiles from Moscow and Ryazan remain in the LCG.

Returning a removed value. If the value is added back, profiles from the test group do not return to the L added to the LCG only from new profiles that enter the campaign after the change.

Example

The value “Tula” was removed from stratification by `region`, and profiles from Tula were moved to the tes returned.

Profiles from Tula do not return to the LCG. Profiles from Tula will be added to the LCG only from new pr the value is returned.

Adding a field value. Current profiles in the LCG are not affected. Profiles from the test group that match LCG. Profiles with the new value will be added to the LCG only from new profiles that enter the campaig

Example

Stratification by field `region`: Moscow, Ryazan. The value “Kolomna” is added.

Profiles from Moscow and Ryazan remain in the LCG without changes. Profiles from Kolomna already in t LCG. Profiles from Kolomna will be added to the LCG only from new profiles that enter the campaign afte

Adding a field. The number of strata increases — calculated by multiplying the conditions of all fields. Pr the new conditions: those who have the new field filled and the value matches one of the conditions will the new field are moved to the test group.

Example

Stratification was by `region`: Moscow, Ryazan — 2 strata. The field `age` is added: 18–30, 31–50. There are Moscow 31–50, Ryazan 18–30, Ryazan 31–50.

Profiles in the LCG with `age` filled, falling into one of the ranges, remain. Profiles without an `age` value a

Removing a field. Profiles that match the remaining fields will stay in the LCG.

Example

Stratification was: `region` (Moscow, Ryazan) and `age` (18–30, 31–50) — 4 strata. The field `age` is removed,

All profiles in the LCG have `region` Moscow or Ryazan, and remain without changes.

Removing stratification. When the remove stratification button is clicked, all stratum settings are remove not moved — already distributed profiles retain their group.

Enabling stratification. If stratification was disabled and then enabled, the system checks all profiles in Matching profiles remain, non-matching profiles are moved to the test group.

Example

The campaign is activated without stratification, 10 profiles are in the LCG. Stratification is enabled by 30, 31–50).

Of the 10 profiles, those with both fields filled and values matching the conditions will remain. A profile “Tula” is moved to the test group.

Supported field types

For stratification, you can use core and additional fields of the following types:

- **enum — list of fixed values**

- **bool — boolean value**
- **int/float — numeric values**
- **string — string values**
- **date — date**

Conditions for field types

For each field type, different conditions are available:

- **Number — selection condition “in range” with “from” and “to” fields**
- **Float — “in range” condition with “inclusive/exclusive” selection**
- **String — selection condition “equals” with the ability to enter a list of values**
- **Boolean — conditions by values “true”, “false”, “missing”**
- **List — selection condition “equals” with multi-select capability**
- **Date — conditions “in relative range” (year/month/day) and “in range” (exact dates)**

Stratification limits

- **Minimum number of conditions — 2 in each field**
- **Maximum number of conditions — 64 for the entire stratification**
- **Maximum number of data fields — 32**

The limit is calculated by multiplying the number of conditions in the selected fields. For example: “City” with 2 conditions = 6 combinations.

TIP: Please note

Values within one field must not fully overlap:

- **Dates — ranges within one field must not overlap**
- **Numbers — ranges must not fully cover each other**
- **Strings/lists — full duplication of values is not allowed**

Recommendations on the number of strata

- **Segment up to 100k — 5-10 strata using key metrics (gender, age, region)**
- **Segment from 100k to 1M — 10-20 strata using key and behavioral metrics**
- **Segment from 1M — 20-30 strata with deep segmentation by demographics, behavior, and RFM**

Recommended number of conditions in one field — from 2 to 4. More than 5 conditions may lead to overl

Stratification error

The system may issue a warning if the LCG limit is too small for proportional filling of all strata. The war operation, but signals a possible violation of proportions. For example:

Parameter	Value
Segment	**10,000 profiles**
LCG	**10% (should be 1,000 profiles)**
Stratification	**Region: Moscow, Saint Petersburg**
LCG limit	**500 profiles**

Initial distribution

Stratum	In segment	Share	Exp
Moscow	**5,000**	**50%**	**250**
Saint Petersburg	**5,000**	**50%**	**250**

Proportions are maintained, no error. Then 5,000 new profiles from Moscow are added to the segment. N 10,000 from Moscow, 5,000 from Saint Petersburg.

Stratum	In segment	Share	Should be in LCG (without limit)	
Moscow	**10,000**	**66.7%**	**1,000**	**333**
Saint Petersburg	**5,000**	**33.3%**	**500**	**167**

With an LCG limit of 500, the required LCG size should be 1,500 (10% of 15,000 profiles in the segment). 33.3%, the correct distribution for 500 spots is 333 from Moscow and 167 from Saint Petersburg. The act does not match this. The LCG cannot be expanded due to the enabled limit, so the system issues a warn

How to fix: increase the limit to the actual LCG size (at least 1,500) or disable it entirely. In that case, th the required ratio during subsequent audience updates.

Ignoring marketing activities

Description

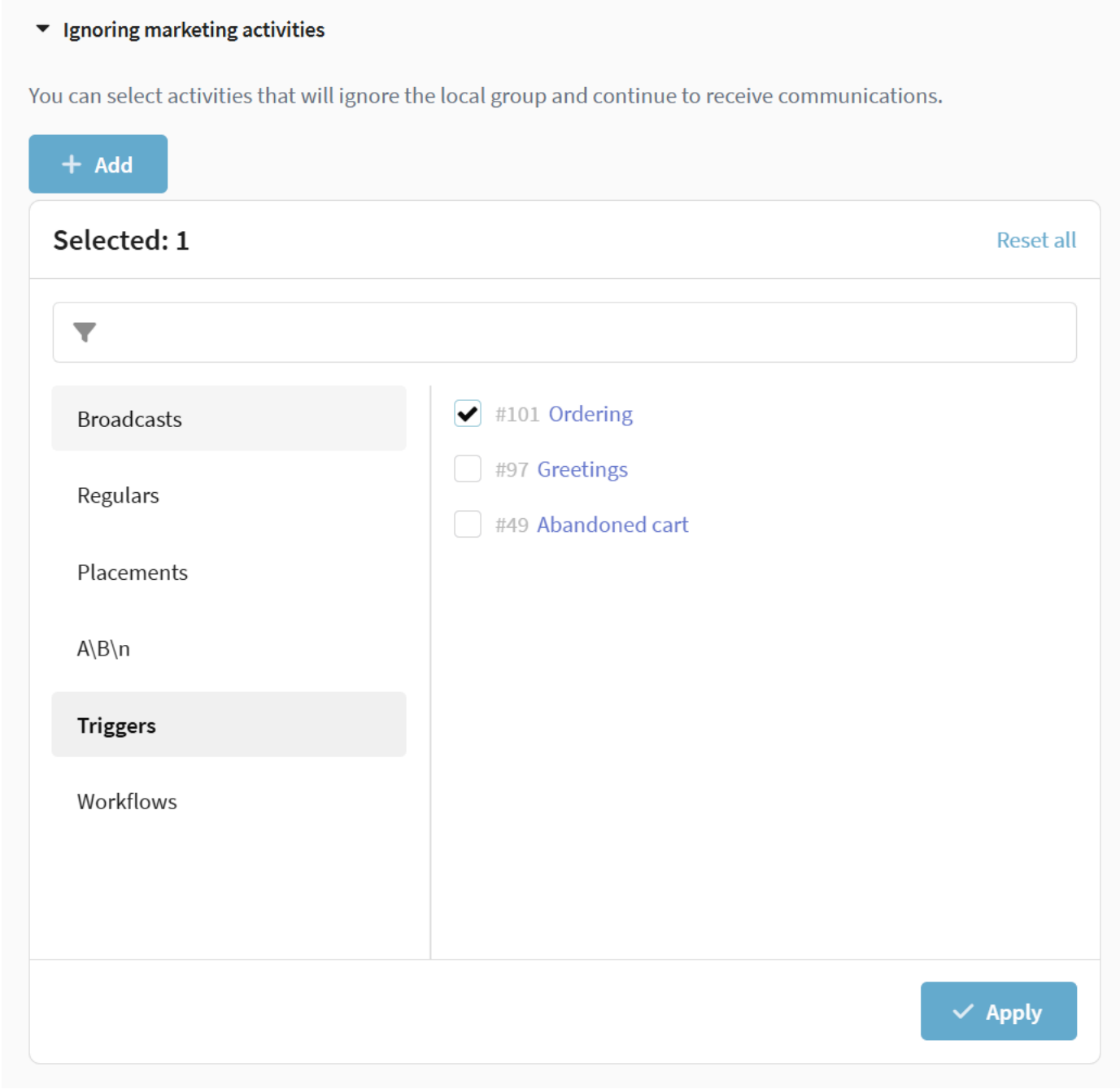
The Ignore marketing activities setting allows you to select specific activities within a campaign that wi including control group participants. This is useful for critical notifications that must reach the entire au participation.

TIP: Usage examples

- **Critical notifications — transactional emails, order status notifications**
- **Legal communications — notifications about changes to terms, privacy policy**
- **Technical notifications — messages about maintenance, system updates**
- **Emergency communications — important alerts concerning all users**

Ignore setup

The ignore activities setting is in the same LCG settings window and is available via a dropdown menu:



To edit the list of activities that ignore LCG, in the LCG settings click

Edit:
![img](/img/attachments/lcg/eng/ignoring-activities-edit.png)

Audience building process

After configuring LCG parameters, when the campaign is activated, the system starts the Audience build when LCG settings are changed in an already active campaign. During this time:

- **Profiles are distributed between the test and control groups**
- **Messages are paused until the recalculation is complete**
- **The campaign is assigned an additional status Audience building, while its main status is preserved**

Please note that when LCG is enabled in a campaign that was already active, the control group will be fo the campaign after its activation, for example as a result of audience expansion. Profiles that were place labeling remain in the test group.

> CAUTION: Attention
>
> **If broadcasts or regular Messages were active in the system at the time the audience recalculation was it is complete.**

LCG management

After configuring LCG, the following actions are available in the campaign interface:

- **Edit settings — changing LCG parameters, stratification, and exceptions;**
- **Delete LCG — complete removal of the local control group. All related settings will be deleted along Ignore marketing activities). As a result of this action, profiles from the control group will be moved**
- **Export profiles — ability to export LCG participants. After audience formation is complete, when con local control group participants for export in the filters. Export is available inside the Audience widg export can be found here.**

![img](/img/attachments/lcg/eng/lcg_delete.png)

LCG operation specifics with segment updates

Audience building is triggered automatically after each segment change. The system distributes only ne between the test and control groups. Profiles already in the campaign retain their group.

Stratification violation error when limit is reached

A stratification violation error occurs when, due to an imposed limit, the system cannot maintain proport local control group (LCG). The warning does not block campaign operation, but signals that the control g proportions.

The examples below use a database of 100 profiles. This small size is chosen intentionally — it makes it error. In real projects, databases can be much larger, but the principle of stratification remains the same

Database fields and their distribution:

Field	Values	
`int_field`	**20 / 80**	**20% / 80%**
`region`	**Moscow, SPb, Kazan**	**~34 / ~33 / ~33 profiles**
`age`	**18–54**	**uniform**

> DANGER: Important
>
> **The error notification appears only when the limit is reached during LCG selection. During LCG formatic system tries to correct using new profiles added to the campaign, but cannot do so because of the imp**

Example 1. Limit too small for the number of strata

Initial data

Parameter	Value
Database	**100 profiles**
Stratification	`int_field` **(20 / 80) and** `region` **(Moscow, SPb, Kazan)**
Number of strata	**6**
LCG size	**15% (should be 15 profiles)**
LCG limit	**12 profiles**

Stratum distribution in the database

Stratum	In database	Share	
int_field=20 × Moscow	**6**	**6%**	**0.72 → ~1**
int_field=20 × SPb	**7**	**7%**	**0.84 → ~1**
int_field=20 × Kazan	**7**	**7%**	**0.84 → ~1**
int_field=80 × Moscow	**28**	**28%**	**3.36 → ~3**
int_field=80 × SPb	**26**	**26%**	**3.12 → ~3**
int_field=80 × Kazan	**26**	**26%**	**3.12 → ~3**

Problem

Due to the limit of 12 profiles, the expected value for the stratum “20 × Moscow” is 0.72 — less than 1. T stratum.

How to fix

- **Increase the limit (for example, to 35 profiles) — in this case, the LCG size must also be increased pr**
- **Remove the limit entirely, leaving only the LCG percentage;**
- **Simplify stratification by removing fields with rare values (for example, leave only `int_field` without r**

Example 2. Critical heterogeneity in three-field stratification

Initial data

Parameter	Value
Database	**100 profiles**
Segment	**Profiles under 25 years old — 21 profiles**
Stratification	`int_field` **(20 / 80),** `region` **(Moscow, SPb, Kazan),** `age` **(18-21, 22-25)**
Number of strata	**2 × 3 × 2 = 12**
LCG size	**10%**
LCG limit	**8 profiles**

Stratum distribution in the segment and expected values in LCG

Combination	In segment	Share	
20 × Moscow × 18-21	**0**	**0%**	**0**
20 × Moscow × 22-25	**1**	**4.8%**	**0.3**
20 × SPb × 18-21	**1**	**4.8%**	**0.3**
20 × SPb × 22-25	**1**	**4.8%**	**0.3**
20 × Kazan × 18-21	**0**	**0%**	**0**
20 × Kazan × 22-25	**0**	**0%**	**0**
80 × Moscow × 18-21	**3**	**14.3%**	**1.1**
80 × Moscow × 22-25	**2**	**9.5%**	**0.7**
80 × SPb × 18-21	**2**	**9.5%**	**0.7**

Combination	In segment	Share	
80 × SPb × 22-25	5	23.8%	1.9
80 × Kazan × 18-21	2	9.5%	0.7
80 × Kazan × 22-25	4	19.0%	1.5

Problem

Due to the limit of 8 profiles, the expected value for strata with `int_field=20` is 0 or close to 0. Empty stra they may reflect the actual state of the database if such profiles truly do not exist. The error occurs bec in a stratum is too small and cannot be increased even when the campaign audience expands, as the lim stratification homogeneity is lost.

How to fix

Remove the limit — then the system will be able to add profiles from new arrivals. Or simplify stratificati ranges.

Example 3. Very small limit

Initial data

Parameter	Value
Database	100 profiles
Stratification	by `int_field` (20 / 80)
LCG limit	1 profile
LCG size	10% (10 profiles available, but limit restricts to 1)

Problem

With a limit of 1 profile, the system cannot maintain proportions of 20% and 80%. In a sample of one pro 0%/100% or 100%/0%. The deviation from the ideal proportion is 100%.

How to fix

Remove the limit or increase it to at least 10-20 profiles.

Audience expansion in campaigns

Description

Audience expansion is a mechanism that automatically adds new profiles to a campaign from the segme expansion mechanism helps increase campaign reach and keep the audience up to date without manual

Each campaign is created based on a static or updatable segment, which defines the target audience. H activity can be configured for any segment, even if it does not overlap with the main one.

For example, a campaign is targeted at the segment Loyal customers, but inside it a mailing is created a registrations. Without audience expansion, such users will not receive the message because they are no audience expansion is enabled for this mailing, the system will automatically add recipients from the seg campaign and distribute them between the test group and the local control group, if it is enabled.

INFO: Key limitations

- **Only activities of this campaign — you cannot select a mailing or scenario from other campaigns**
- **Only non-completed activities — you cannot select activities with status “Completed”**
- **Only new profiles — only profiles that are not yet in the campaign audience are added**
- **Only for static segments — audience expansion is not available for campaigns with an updatable se profiles enter the audience only as a result of segment recalculation**

How audience expansion works

The audience expansion mechanism allows dynamically replenishing the campaign with new profiles dir activities. Its operation can be described by the following key principles:

- **When an activity is launched, the system determines the audience of this activity (segment or entire the campaign audience profiles that are not yet part of it;**
- **New participants are immediately distributed between the test and control groups of the campaign;**
- **If a profile falls into the segments of several activities of audience expansion of one campaign, it wi Its group (test or control) will be fixed at the moment of first addition. This prevents repeated entry audience and prevents their accidental transition from the test group to the control group when subs**
- **Audience expansion works only for active campaigns.**

CAUTION: Please note

The activity segment may be built on the basis of another profile database. In this case, the intersectio be empty and the message will not be sent to anyone. Before adding an activity to a campaign, make su campaign segment are built on the basis of the same profile database.

Profile addition process

For each profile in the audience of the launched activity, the system checks:

- **Campaign participation: It is checked whether the profile belongs to the campaign audience at the ti**
  - **If yes — it will receive the message (if it is not in LCG).**
  - **If no — the profile will be distributed into the test or control group according to the local control the test group will receive the message, those that fall into the control group — will not.**

Usage example

Scenario without audience expansion

The campaign uses the profile database `CRM_Base` and segment `Loyal customers` (100 profiles).
A broadcast is created with the segment `New registrations` (50 profiles). These users do not belong to the main segment.

When the broadcast is launched, the message will not be sent to anyone, because there is no intersectio profiles are not in the campaign.

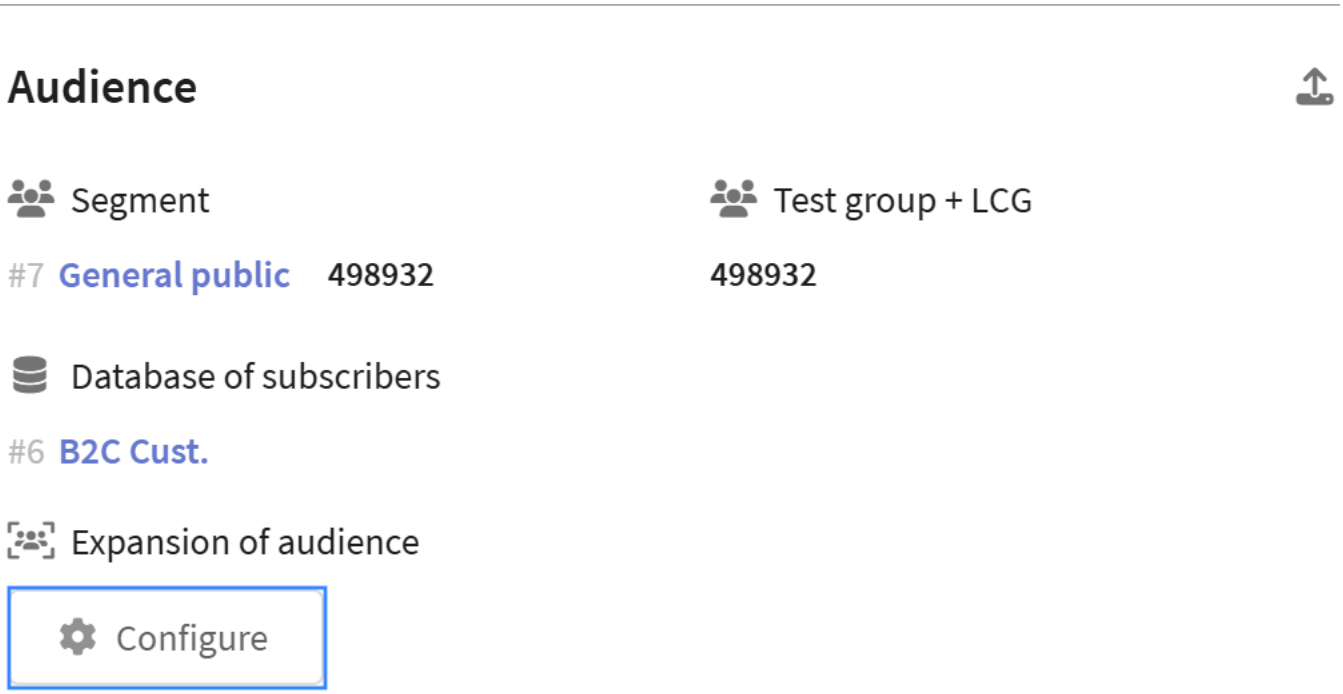
![img](/img/attachments/super-campaigns/eng/activities_scampaign_2.png)

Scenario with audience expansion

If you add the broadcast to the Audience expansion list in the campaign settings, the system will include `registrations` in the campaign.
After that, profiles will be automatically distributed between the test group and the local control group.

Audience expansion setup

Audience expansion configuration is located in the Audience section of the campaign editor in the Audie



Selecting activities

In the “Audience expansion” block, various types of activities are available for selection:

- **Broadcasts — one-time mass Messages**
- **Regular Messages — periodic communications on a schedule**
- **Triggered Messages — automated sends based on events**
- **A/B tests — comparison of variant effectiveness**
- **Scenarios — multi-step automations**
- **Placements — tool for content placement**

![img](/img/attachments/audience-expansion/eng/auditory_copy.png)

TIP

You can select multiple activities. Each of them will independently add new users to the campaign whe

Audience building

Process overview

Audience building is the process of distributing campaign profiles between the test group and the contro settings.

The system processes only profiles belonging to the profile database selected when the campaign was c

The process runs in the following cases:

- **When the campaign is activated — the initial distribution of profiles based on Local control group (LC**
- **When modifying an active campaign — editing LCG settings triggers recalculation**
- **When the segment is updated — newly added profiles go through incremental audience building. This**

Working with inactive campaigns

In an inactive campaign, profile-related actions behave differently. When a profile is removed from the se

- **The Left segment event is recorded in the profile card immediately**
- **The Left campaign event appears only after the campaign is activated**

If a profile is removed from the segment and then added back before activation, the system considers or segment. Campaign-level “enter/exit” events for these intermediate actions are not recorded, but the pro history.

LCG changes in active campaigns

Enabling the Local control group in an active campaign starts audience building. Scheduled mailings (bro paused during the process. Only new profiles entering the campaign audience (for example, through audi LCG.

Removing LCG in an active campaign redistributes the audience — all profiles from LCG are moved to the

Campaign completion

When a campaign is completed, the system automatically records the Left campaign event for all partici

Market

Market allows you to transfer product and order data to the platform and use it in customer communicati automatically via API or a product feed, or manually by uploading a file.

The imported data can be used for the following purposes:

- **inserting order details into message templates;**
- **triggering events based on order status changes;**
- **segmenting customers based on purchase data.**

Before you get started, make sure you have the required roles enabled for the user and token:

To work with the Market module, we recommend using API requests. If you are using the platform interfa below:

Market settings

![img] (/img/attachments/56251102/56256318.png)	This instruction contains information on such market entities as sales channels, custom fields, order item statuses, regions, ca section, and then "Settings".

Sales channels

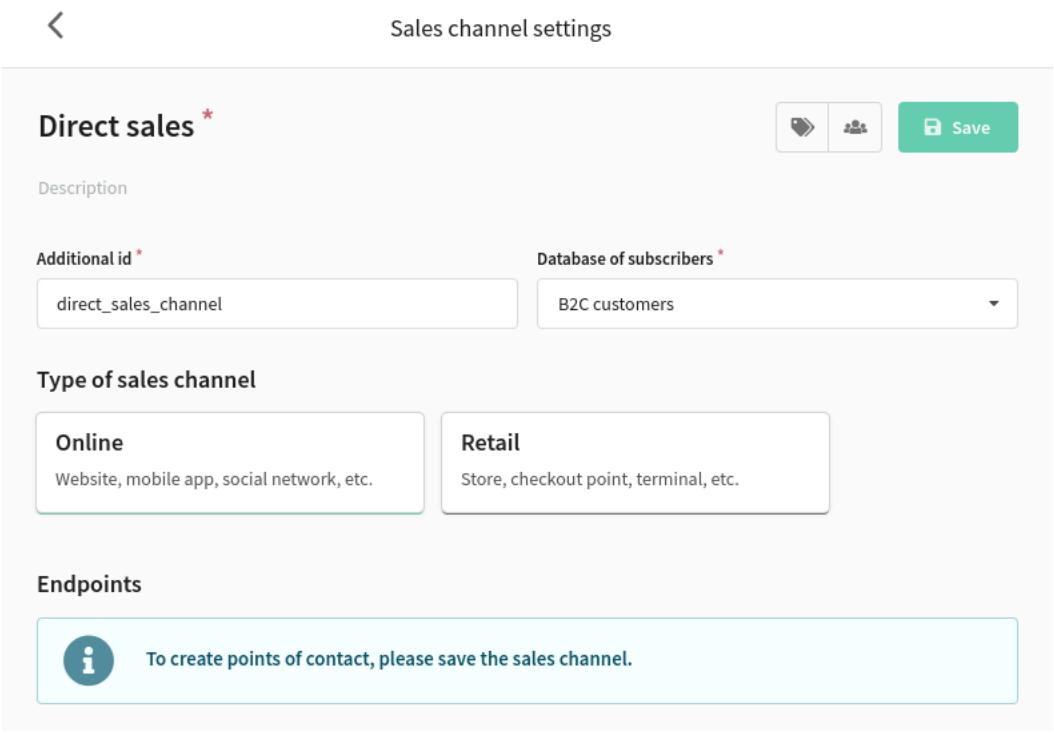
A sales channel is the way customers get to a company to order products they are interested in. It cover (stores, websites, terminals or cash registers where the order is placed). The key task of the channel is touchpoints.

Click Create to create a new sales channel. You will see a window, configure the channel in it:

- **type a name and description for the channel**
- **additional ID**
- **add tags**

  and access groups

- **specify the profile database (channel purchasers)**
- **select the type of channel (online or retail)**



CAUTION

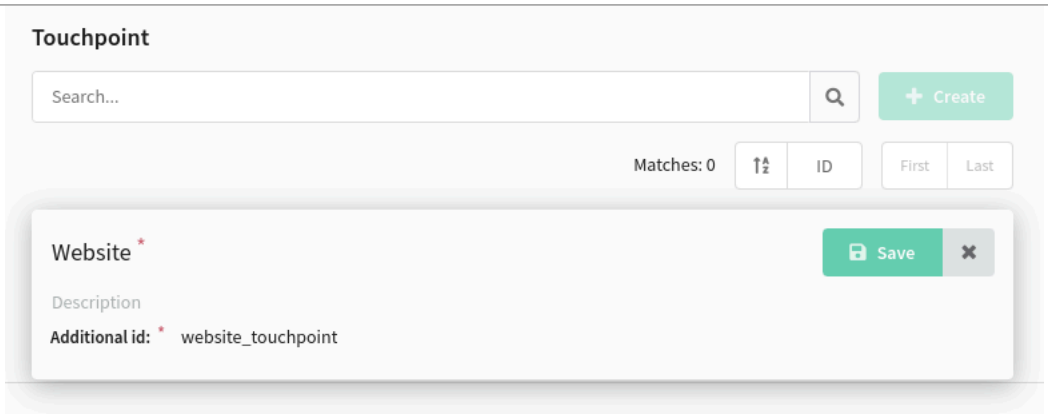
Once the channel has been created, the specified profile database cannot be changed.

Save the channel and then select it from the general list to create a touchpoint.

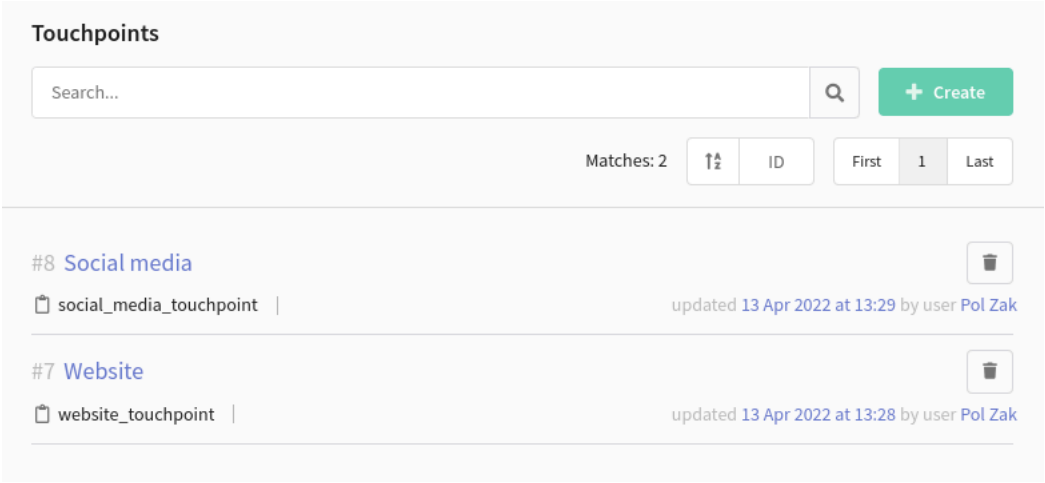
TIP

A touchpoint is a specific platform or tool by means of which you sell a product to your customer. Touch and the order. Some examples of touchpoints: a website, a store, a parcel terminal, a cash register whe network, a call centre, etc.

Type a name, description and additional ID for the touchpoint. Then click "Save:



Each touchpoint can be linked to only one channel. At the same time, the channel may have several touc

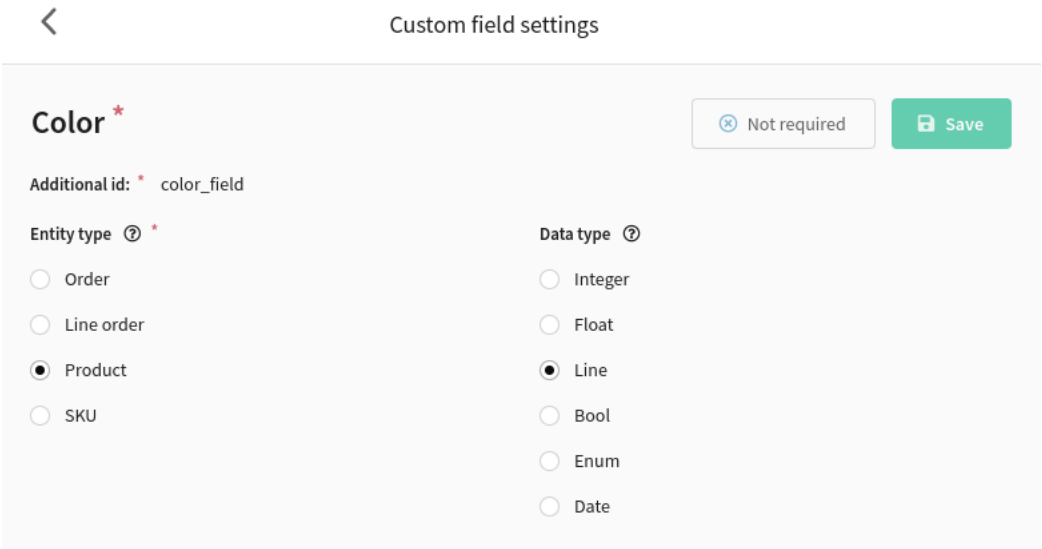


Custom fields

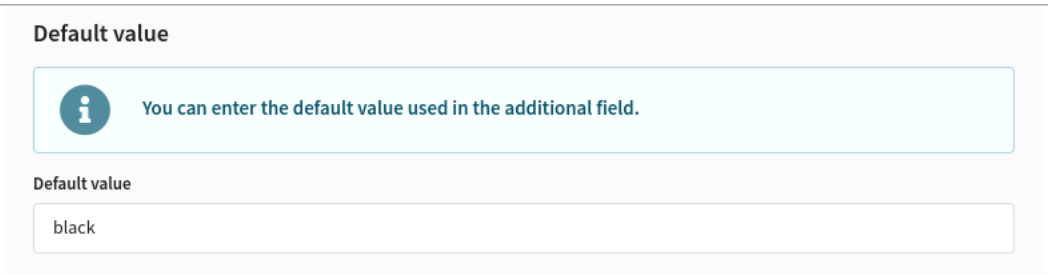
Custom fields are set for entities such as order, order item, product and SKU. There you can store inform features — colour, size, volume, material, model, weight, etc.

Select the type of entity and specify the type of data that will be stored in the new field:

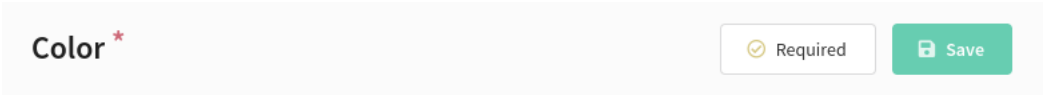
- **whole number**
- **floating-point number**
- **line**
- **boolean**
- **list**
- **date**



It is possible to set a default data value. It means that the value will be automatically substituted in an c



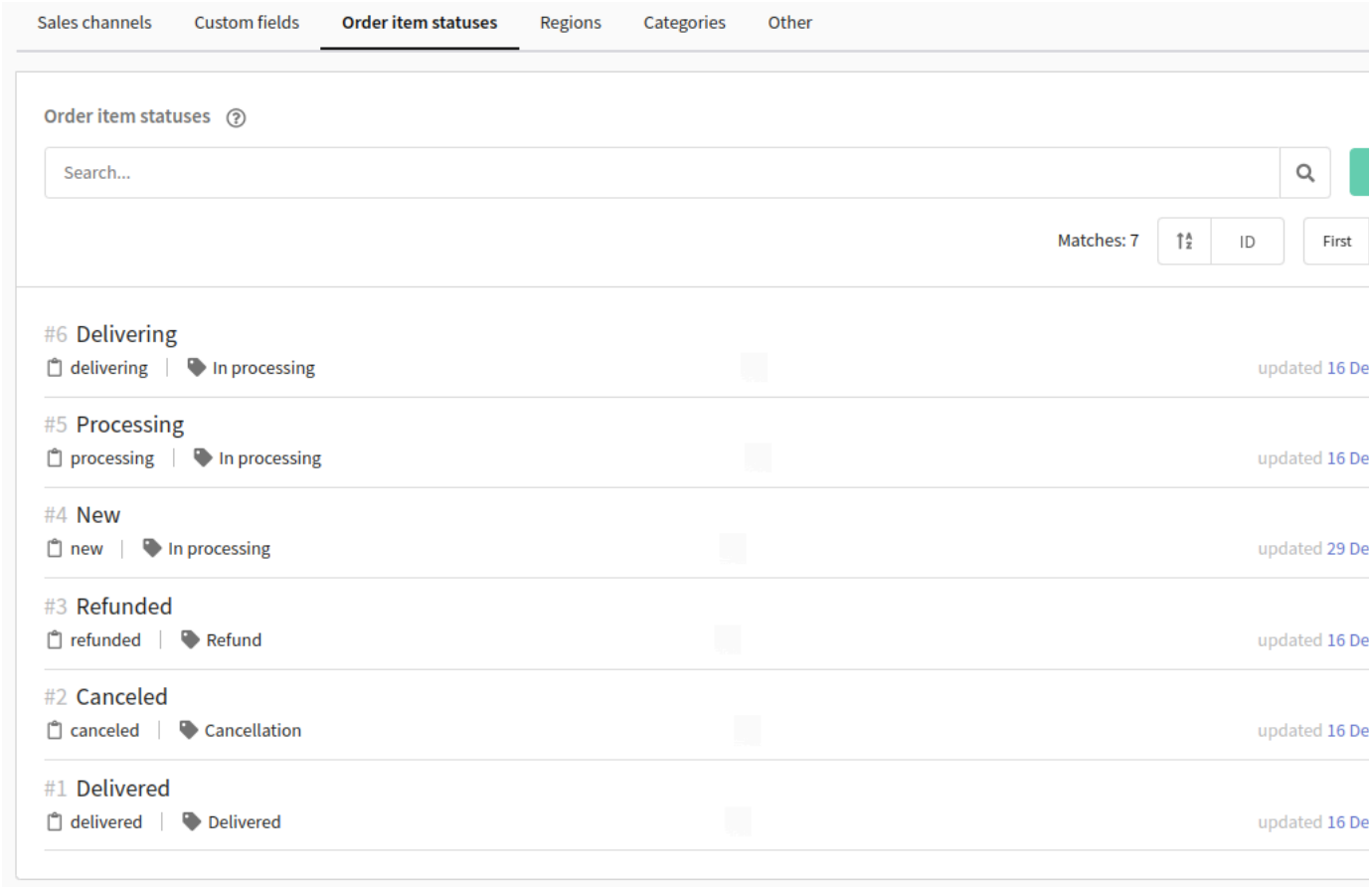
If you need to make the field required, change Optional to Required":



Order item statuses

Statuses are available by default:

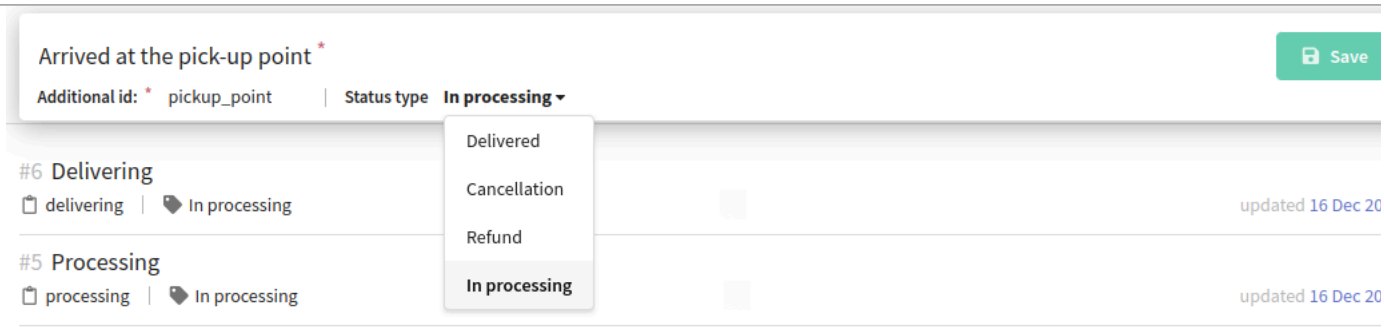
- **delivering**
- **processing**
- **new**
- **refunded**
- **canceled**
- **delivered**



You can also create your own statuses. For example, if you need to segment the delivery process into se*service", "arrived at the pick-up point", "arrived at the local courier facility".*

Use the "Create button to add a new status. Enter a name, ID and type for the status. Four types of statu

- **delivered**
- **cancellation**
- **refund**
- **in processing**

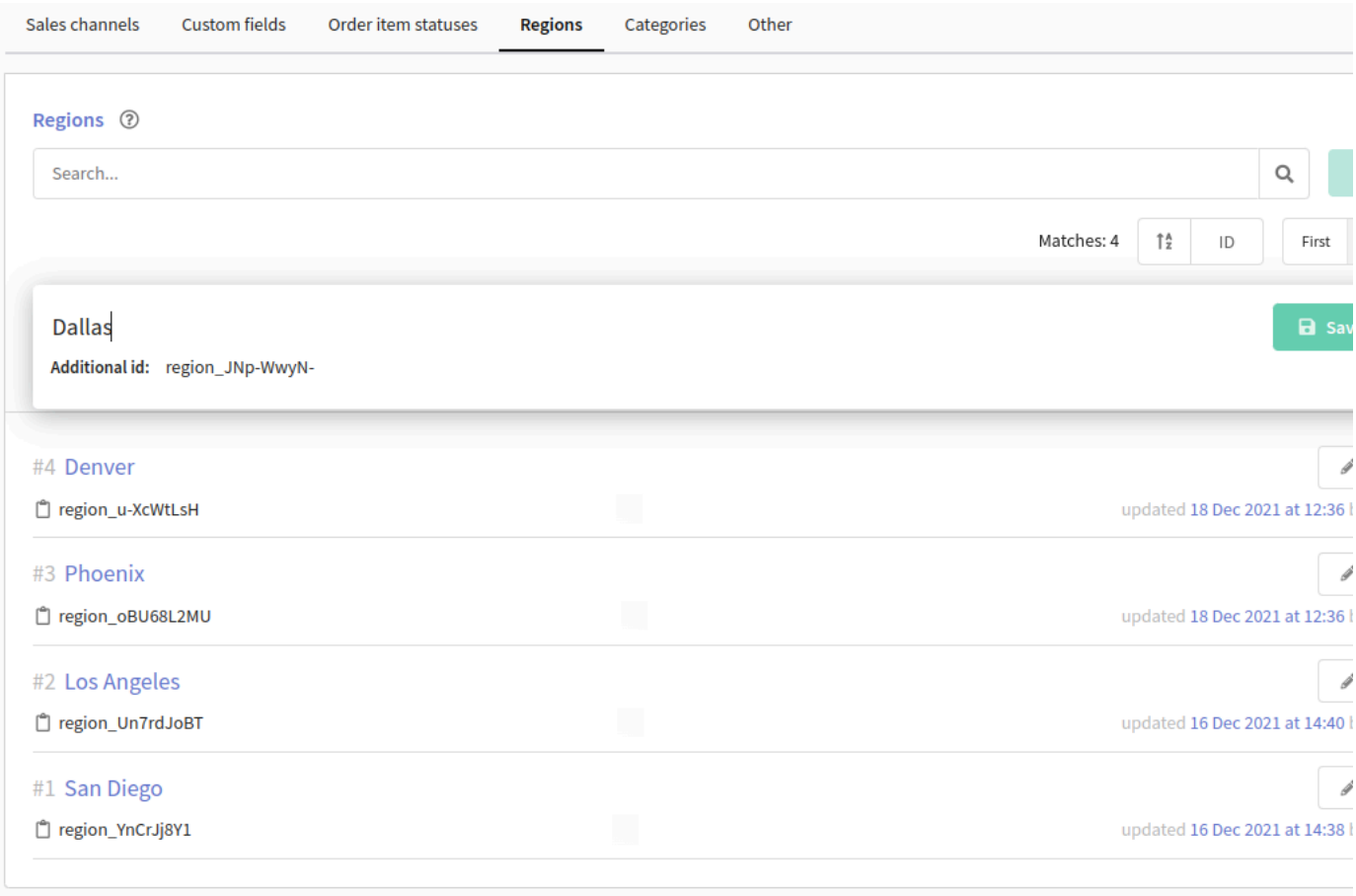


After filling in all the fields, click Save.

Regions

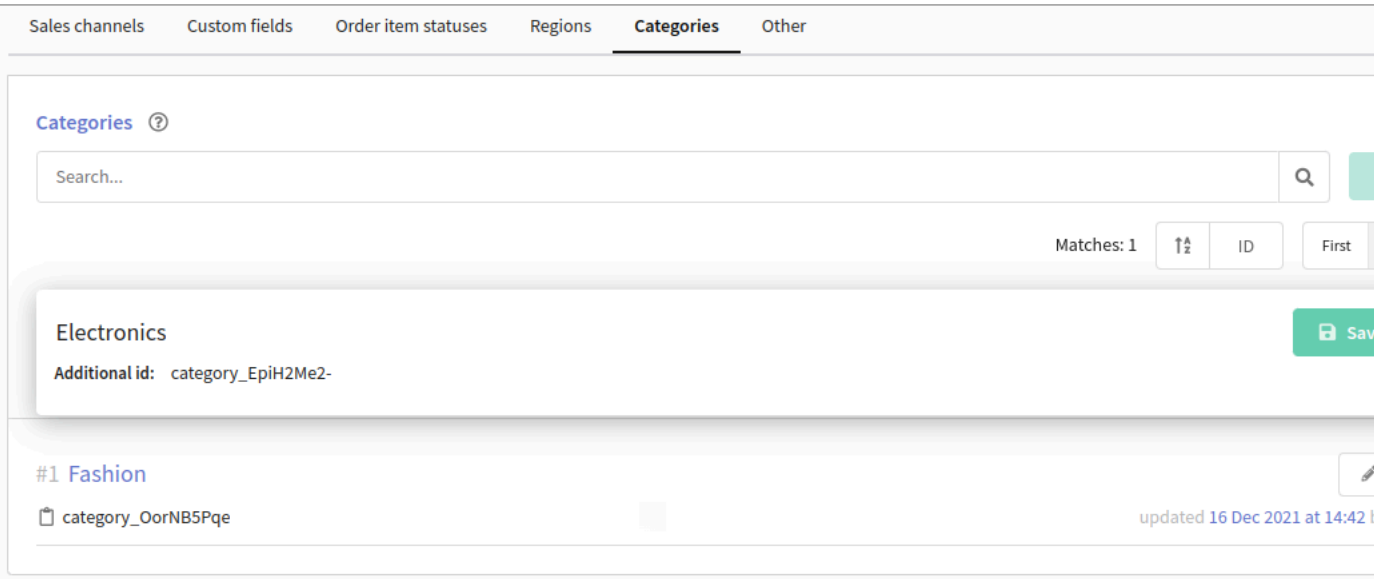
Regions are specified when setting up the order. You can set regional data in products and SKUs. For ex details, etc.

When creating a region, enter its name and edit the ID if necessary. Then click Save:

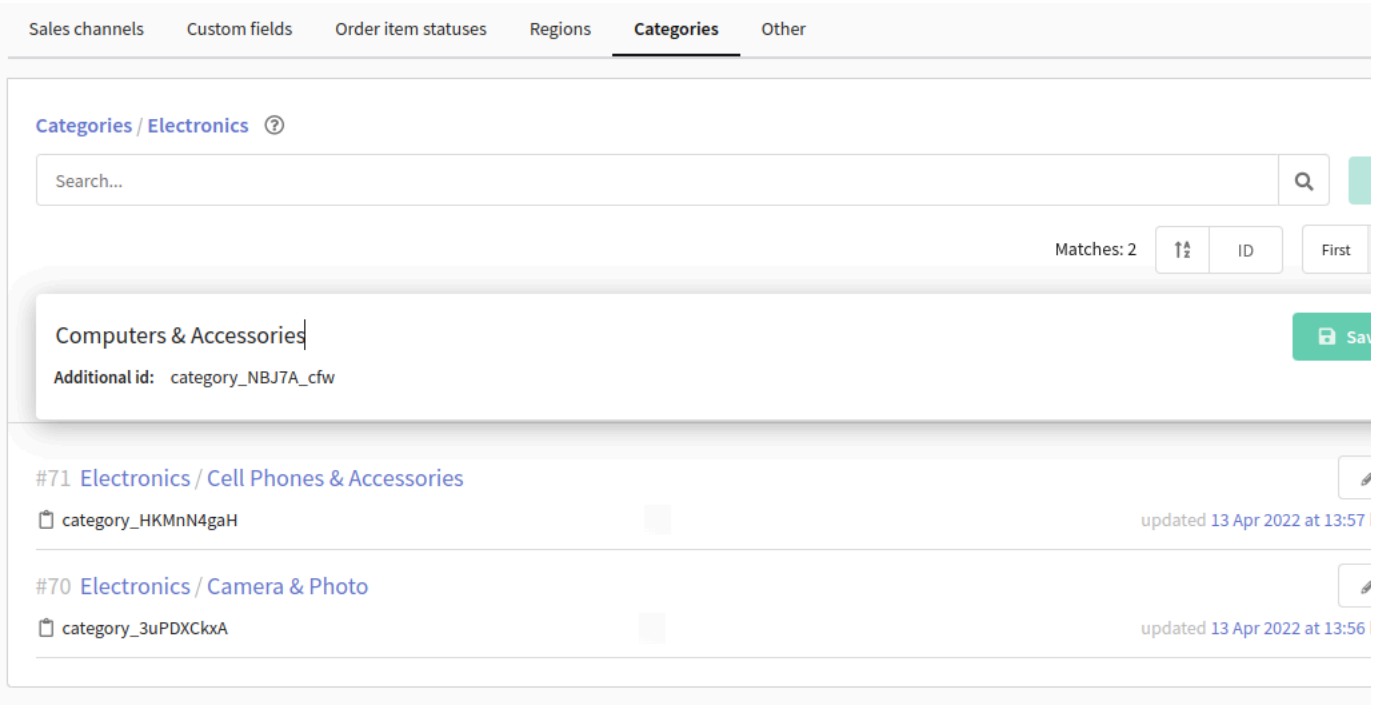


Categories

A category is a group of products that seem similar to the customer. You can combine products and SKU
Click Create" to create a new category. Type a name for the category and change its ID if necessary. Th



Each category may be divided into several subcategories. Click on the name of the category if you want save the object:

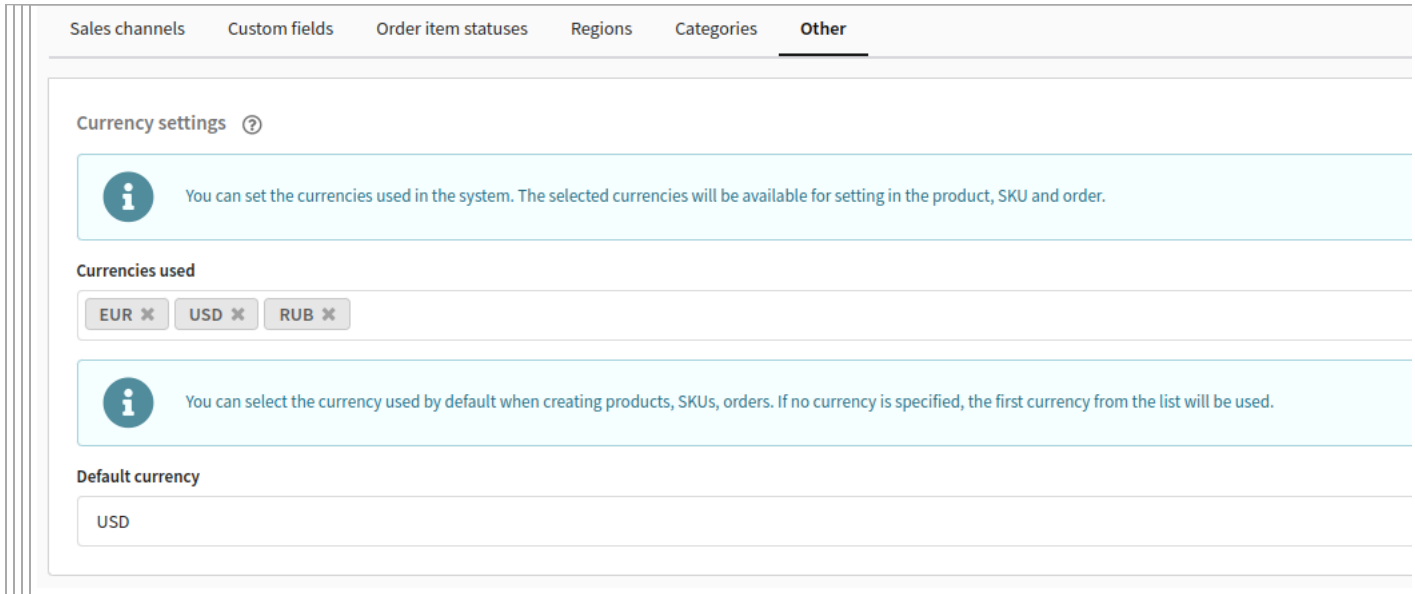


Other

Here you can set currencies used in the market. Select necessary currencies from the drop-down list.

CAUTION

Product or SKU will not be imported through API if its currency is not stated in the Currencies field.



You can select a default currency. It will be automatically set when creating products, SKUs and orders.

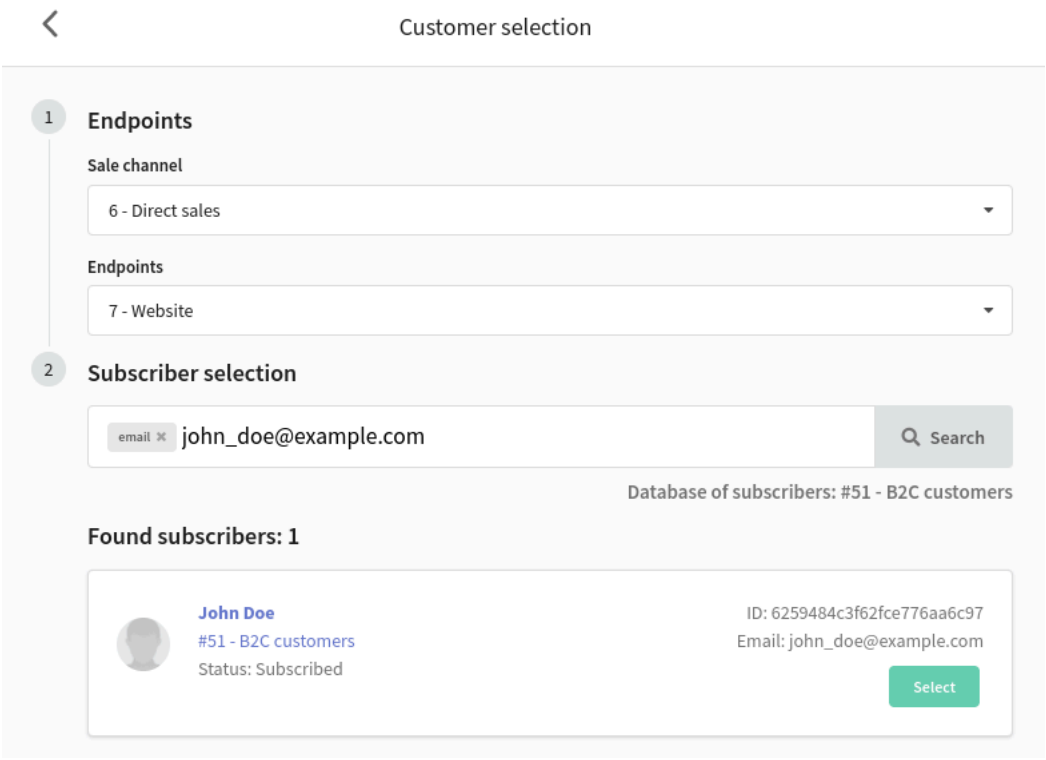
Orders

![img] (/img/attachments/49710303/56256594.png)	**You can create and process orders coming from various sales points in the "Orders" section. It stores information on orders tha To create a new order, select a customer and add products or SKUs to the order.** You can [import orders using API](/en/developer-guide/market/orders/56249746/) or create it manually in the platform interface. To get started with orders, go to <strong

Selecting a customer

To create an order manually, enter its number. Then specify the buyer by clicking " Select a customer ":

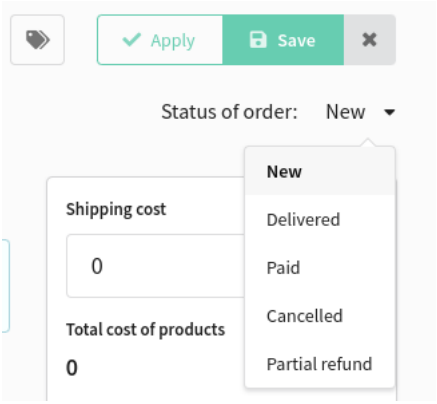
A new window will open. Select a sales channel and a touchpoint in it. Each channel has a profile databa customer" button to find the desired profile in the database by its email, phone number, custom field or a window will automatically close when you click "Select":



Current order status

Available order statuses:

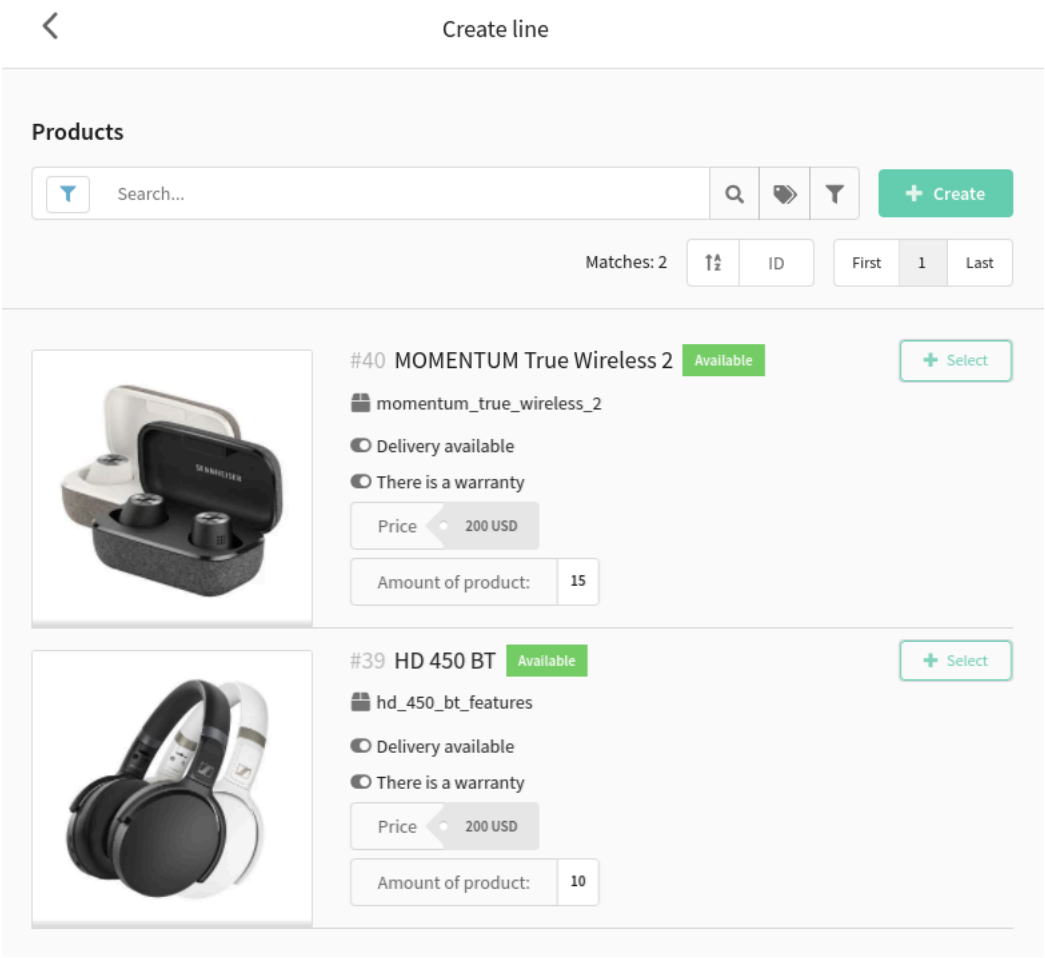
- **new**
- **delivered**
- **paid**
- **cancelled**
- **partial refund**



Adding an item to an order

An order comprises items. These can be products or SKUs. Several items of the same product in an orde
To add new items to an order, click “Add”:

In the new window, select the required items:

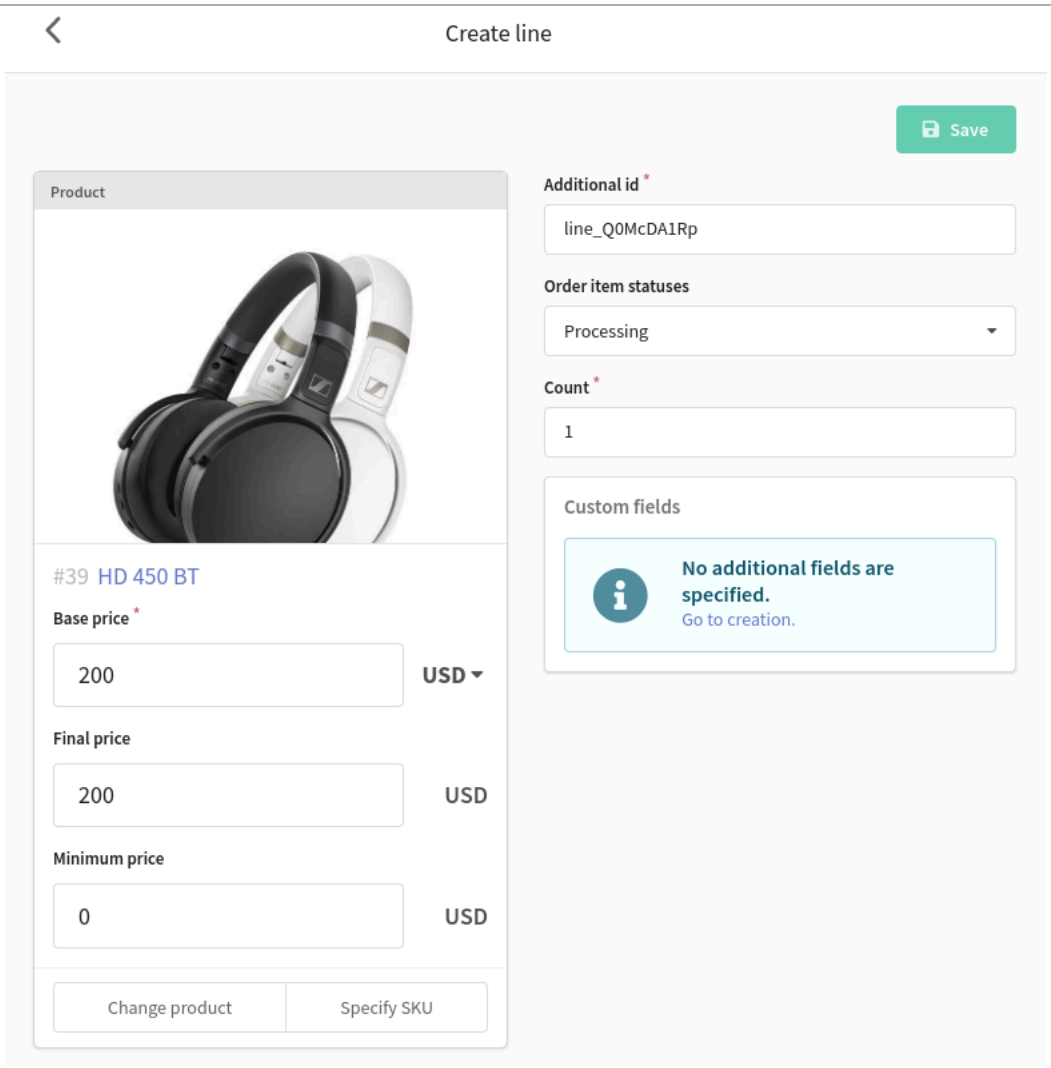


Configuring an item

You can edit each item if necessary. To do this, click on an item in the general list — a window for editin the right:

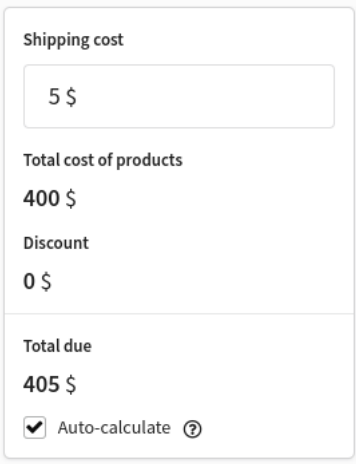
- **product price**
- **additional identifier (used in API requests);**
- **current status of the order item;**
- **number of products in the item;**
- **custom fields.**

In this window, you can also change the product or specify the SKU.



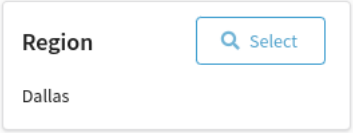
Order price

The total price of the order is shown in the right window. It comprises delivery price and the price of eac
If you need to recalculate the price manually, activate the “Auto Calculate” option.



Region

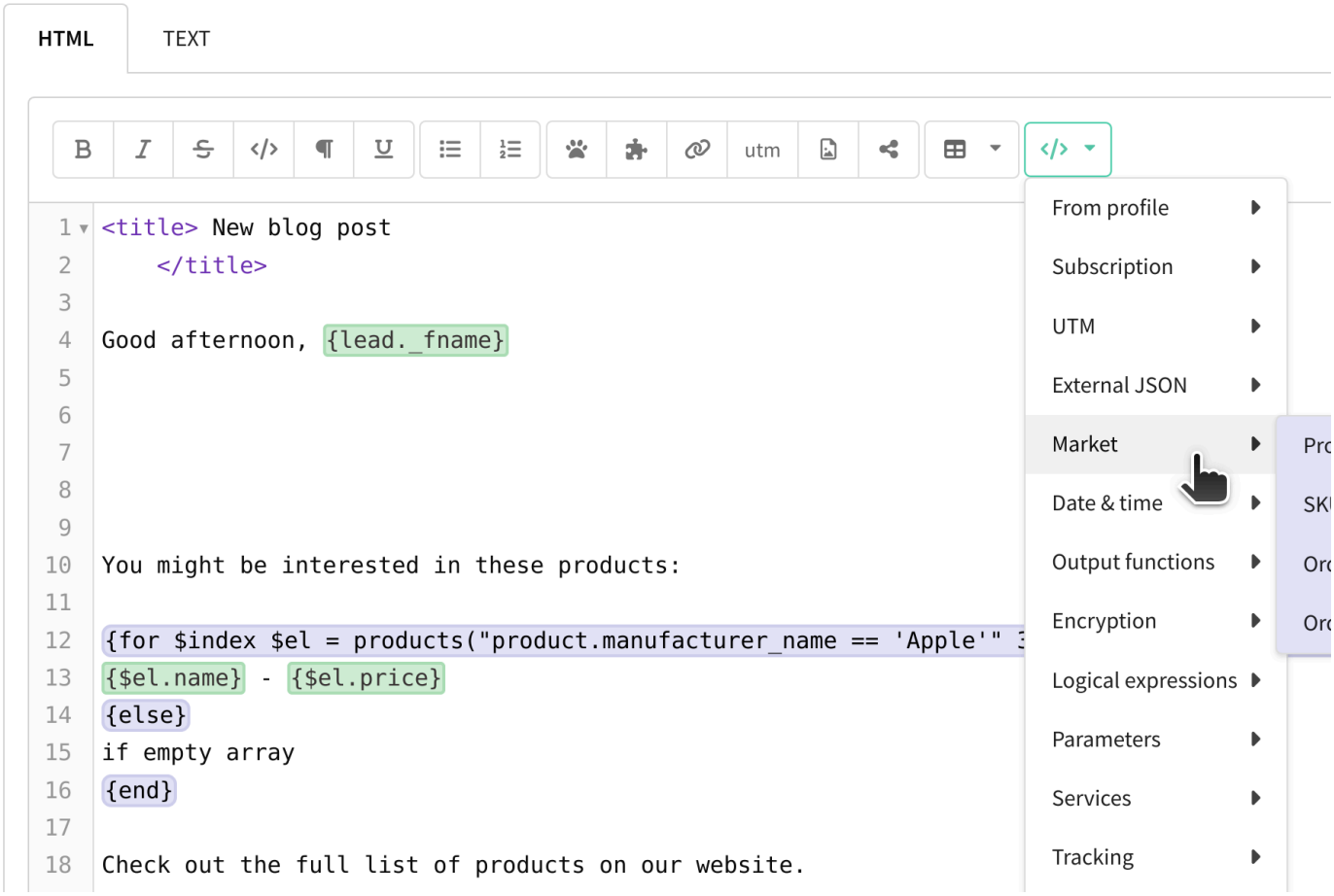
Select the region where the order was placed. Some characteristics of the order (price, values of custor change depending on the region.



The order configuration is complete. Save it using the top bar buttons.

Market variables in message template

To use information about orders and products in your mailings, you need to add Market variables to your
Open the [message template] and click the `</>` button in the editor. In the list, select Market and insert o



TIP

For more information on using marketplace variables in messages, please refer to [this article].

Guide: how to send an order confirmation email

Description

This guide will show you the algorithm for working in the Altcraft Platform market.
Let's imagine that we have an online store for smartphones. A buyer visits a website, adds the product t
task is to send them an order confirmation email with the order number and price.
The guide comprises the following steps:

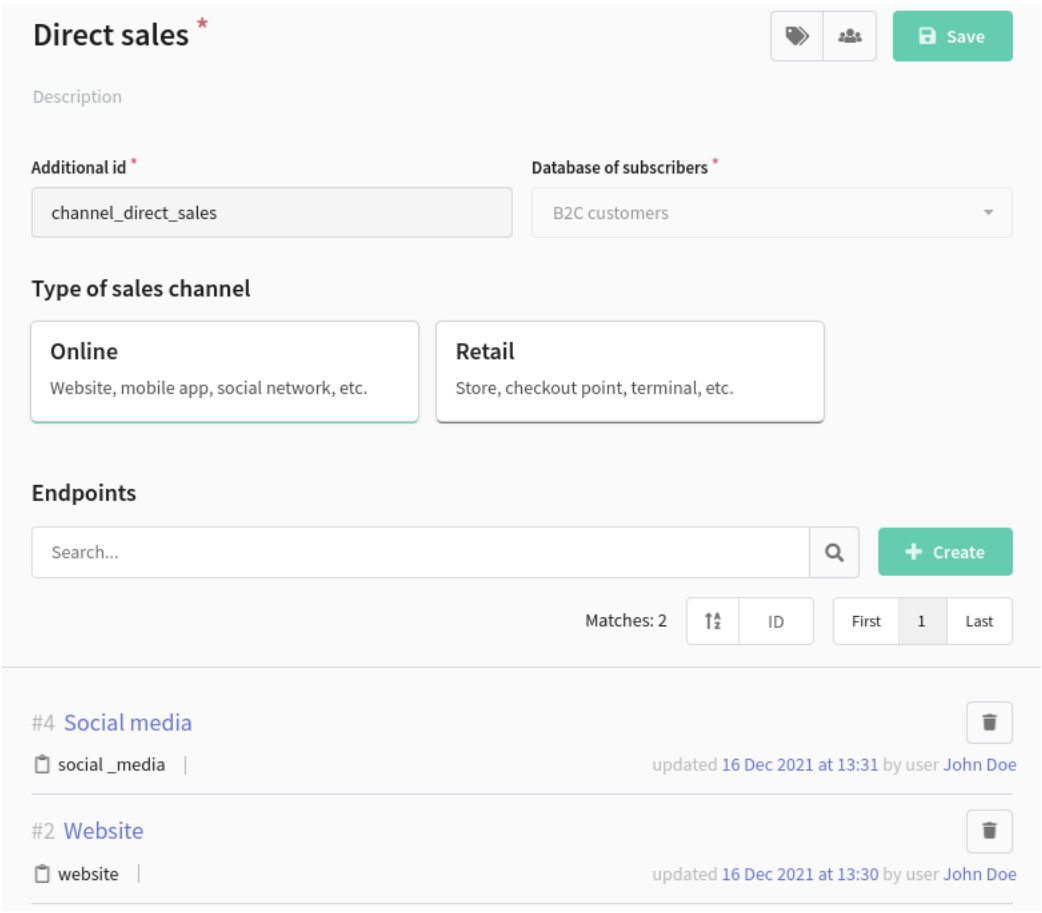
1. **Creating a sales channel and touchpoins**
2. **Importing product and SKU data using a yml file**
3. **Setting up an API request to add order data to the market**
4. **Creating a template and substituting order data using variables**
5. **Setting up a trigger mailing**

Step 1. Set up a sales channel

You need a sales channel to start working in the market. This channel defines a profile database and tou
A profile database contains the contact details of customers who place orders in a certain channel. You
or existing database. To do this, use import from a file or via API.
A touchpoint is a platform or a tool using which you sell products and services to your customers (a web
social network, a call center and so on). A touchpoint connects a sales channel a an order.
To create a channel, go to Market —> Settings in the main menu. Create a new channel on the Sales cha

- **a channel name**
- **an additional identifier**
- **a profile database**
- **a channel type: online or retail**

Next, save the channel and go back to its settings. Create touchpoints.
Please note that each touchpoint can only be linked to one channel. The touchpoint ID will be used in the



Step 2. Upload product data to the platform

Products in the market are represented by two entities: products and SKU.

INFO

A product is goods or services for sale. SKU is a product modifier, it is used for variations of one produc size, weight, model, configuration and so on can vary).

SKU is connected to a certain product and cannot exist without it. For example, `Apple iPhone 12` is a pro `Apple iPhone 12 256GB` are its SKUs.

There are three ways to add product data to the platform:

- **manually**
- **import it from a file**
- **import it via API**

Let's import product data from a file. To do this, you need a yml file.

TIP

If you have any questions about import, read this detailed guide.

Below you can see examples of yml files:

Product

A file with such content will add two products to the market: *Apple iPhone 12* and *Xiaomi 12X.*

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
    <shop>
    <name>BestSeller</name>
    <company>Tne Best inc.</company>
    <url>http://best.seller.ru</url>
    <categories>
      <category id="10">Смартфоны</category>
      <category id="101" parentId="10">Apple</category>
      <category id="102" parentId="10">Xiaomi</category>
    </categories>
    <offers>
      <offer id="iphone12" available="true">
        <name>Apple iPhone 12</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12348</url>
        <price>83990</price>
        <oldprice>89990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>101</categoryId>
        <picture>https://example.com/img/apple_phone.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
           </offer>
      <offer id="xiaomi12" available="true">
                 <name>Xiaomi 12X</name>
        <vendor>Xiaomi</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12345</url>
        <price>62990</price>
        <oldprice>69990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>102</categoryId>
        <picture>https://example.com/img/xiaomi_phone.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0123456789379</barcode>
           </offer>
    </offers>
  </shop>
</yml_catalog>
```

SKU

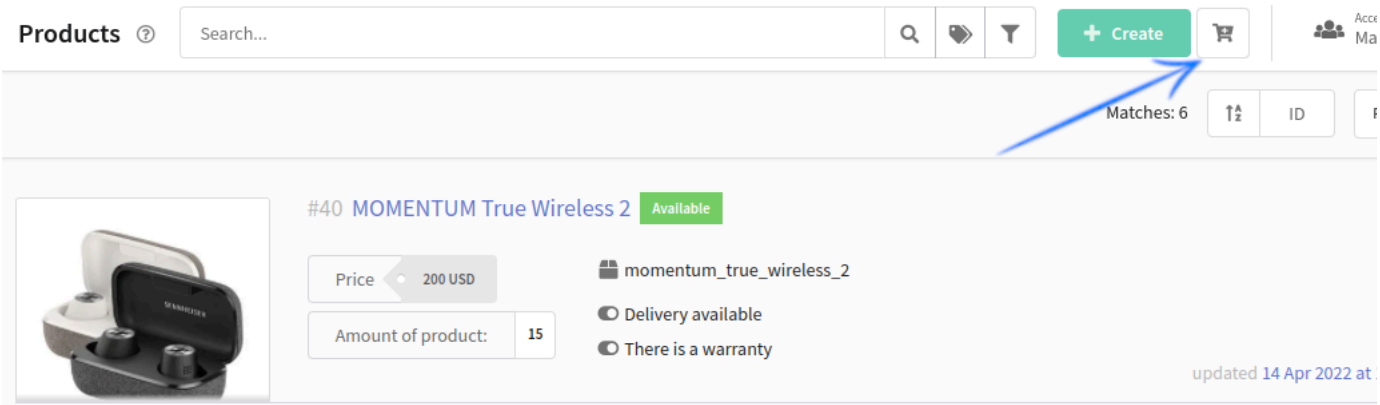
A file will add two SKUsfor *Apple iPhone 12*: *Apple iPhone 12 128GB* and* Apple iPhone 12 256GB*.

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
    <shop>
    <name>BestSeller</name>
    <company>Tne Best inc.</company>
    <url>http://best.seller.ru</url>
    <offers>
      <offer id="apple12_128gb" group_id="iphone12" available="true">
        <name>Apple iPhone 12 128GB</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12358</url>
        <price>71990</price>
        <oldprice>72990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>101</categoryId>
        <picture>https://example.com/img/apple_phone_128.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
           </offer>
      <offer id="apple12_256gb" group_id="iphone12" available="true">
                 <name>Apple iPhone 12 256GB</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12359</url>
        <price>83990</price>
        <oldprice>89990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>102</categoryId>
        <picture>https://example.com/img/apple_phone_256.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
           </offer>
    </offers>
  </shop>
</yml_catalog>
```

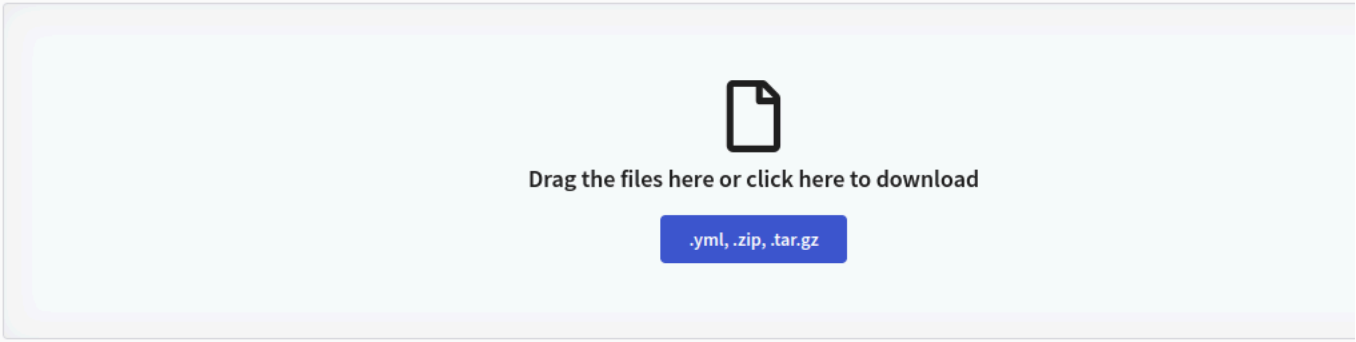
TIP

A list of all products and SKUs is stored in the `offers` element. One product or SKU is stored in a separ
The `id` attribute, which is unique for each product, is specified for the `offer` element.
The structure of the yml file is almost the same for products and SKUs. The only difference is the follov
`offer` element you need to specify not only `id`, but also `group_id` — a product identifier.
Please note that SKUs are different variations of one product, so the value of the `group_id` attribute mus
product.

To import a file with a list of products, go to Market —> Products. Click on the icon on the top toolbar, an
to import — a product.



Click on the download area and select a file on your device:



Start the import process by clicking "Import" on the top toolbar.
Then import SKUs. To do this, follow all the steps described in this paragraph.

Step 3. Set up sending an API request when adding an order

When a visitor of an online store places an order, an API request with the order data must be sent to the

URL

Method: POST

`https://example.com/api/v1.1/market/orders/import`

Request parameters

Parameter	Type	Example	Required	
token	string	“abcdefghijklmnqrstuvwxyz”	Да	API token. Make sure that roles that allow you to
data	array of objects	[{ … }]	Да	Order data. Learn more.
Data object				
external_id	string	“1234001”	Yes	Order ID
matching	string	“email”		Subscriber search mode. By default — email. For each type of matching, certain fields must be fieldshere.
endpoint	string	“website”	Yes	Touchpoint ID
status	string	“new” “delivered” “paid” “canceled” “part_refunded”	Yes	Order status ID
total_price	string	“71990”	Yes	Total cost of the order. It consists of the price of
currency	string	“RUB”	Yes	Currency of the order price.
create_time	string	“2022-04-29T12:48:23Z”	No	Order creation time
update_time	string	“2022-04-29T12:48:23Z”	No	Order update time
lines	array	[{ … }]	Yes	Order lines data. At least one item.
Lines object				
external_id	string	“line_01”	Yes	Line ID
product	string	“iphone12”	Must not be present in the request if `sku` is specified	Product ID
sku	string	“apple12_128gb”	Must not be present in the request if `product` is specified	SKU ID
base_price_per_item	string	“71990”	Yes	Base price
min_price_per_item	string	“71990”	Yes	Minimum price
final_price_per_item	string	“71990”	Yes	Final price
count	int	1	Yes	Quantity of goods per item

Request example

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "data": [
    {
      "external_id": "1234001",
      "matching": "email",
      "email": "johndoe@example.com",
      "endpoint": "website",
      "status": "new",
      "total_price": "71990",
      "create_time": "2022-04-29T12:48:23Z",
      "update_time": "2022-04-29T12:48:23Z",
      "currency": "RUB",
      "lines": [
        {
          "external_id": "line_01",
          "product": "iphone12",
          "base_price_per_item": "71990",
          "min_price_per_item": "71990",
          "final_price_per_item": "71990",
          "count": 1
        }
      ]
    }
  ]
}
```

Response example

```
{
  "error": 0,
  "error_text": "",
  "result": [
    {
      "error": 0,
      "external_id": "1234000"
    }
  ]
}
```

Response parameters

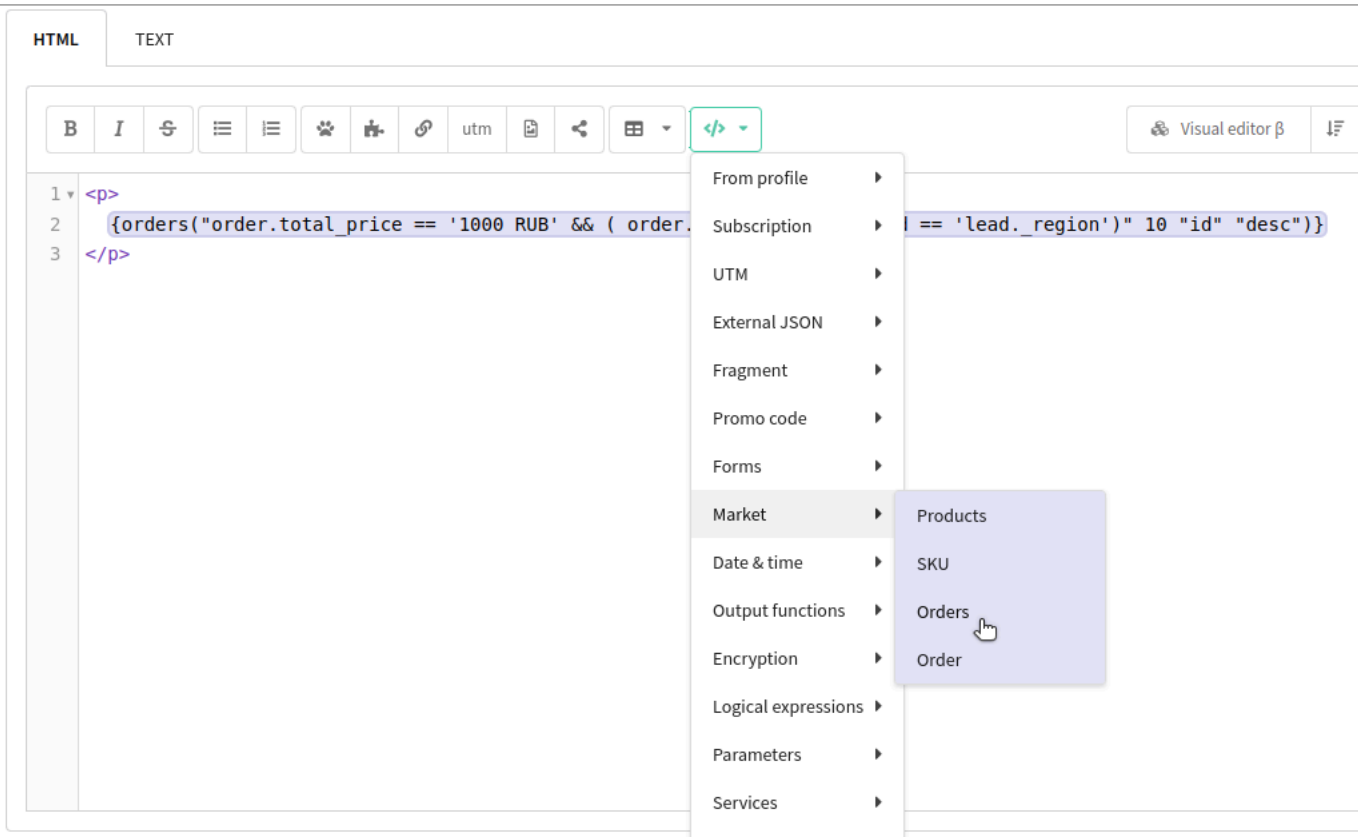
Parameter	Type	Description
error	int	Error code
error_text	string	Error text
result	array	IDs of orders that were successfully imported

Step 4. Create an email template

Go to the Templates section. Add a new template for the email channel.

Add the market variables to the email template to use the order information in your mailing. To do this, c select “Market and insert the Orders” variable.

Theordersfunction pattern will be added to the editor. This function selects orders from the list of profile



We only need to add information about the last order with the status “New” to our email. To do this, we v a function:

```
{orders("order.status == '1'" 1 "create_time" "desc")}
```

The function consists of four arguments:

Argument	Example	Description
Filter	**“order.status == ‘1’”**	**The function will select only those orders that match the condition “order status is new”.**
Limit	**1**	**The function will select only the first element in the list of orders.**
Sorting field	**“create_time”**	**Selected elements will be sorted by the “create_time” field (order creation time).**
Sorting direction	**“desc”**	**The elements will be sorted in descending order of the value in the “create_time” field.**

An array of orders is the result of the orders function.In this case, it will be an array with one item. Use I

In the email template, substitute two variables:

- `{$el.external_id}` **— additional order ID.**
- `{$el.total_price}` **— total order price.**

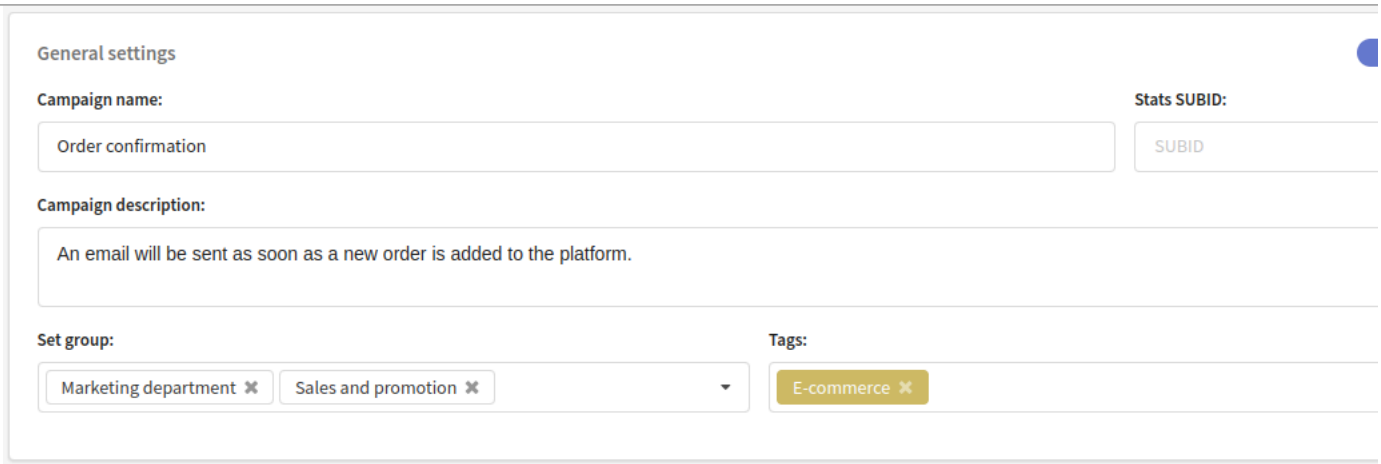
```
{for $index $el = orders("order.status == '1'" 1 "create_time" "desc")}
Thanks! Your order no.{$el.external_id} is confirmed.
Order total: {$el.total_price}
{else}
if empty array
{end}
```

TIP

Read more about market variables in this article. There you will find information about fields that can b the template.

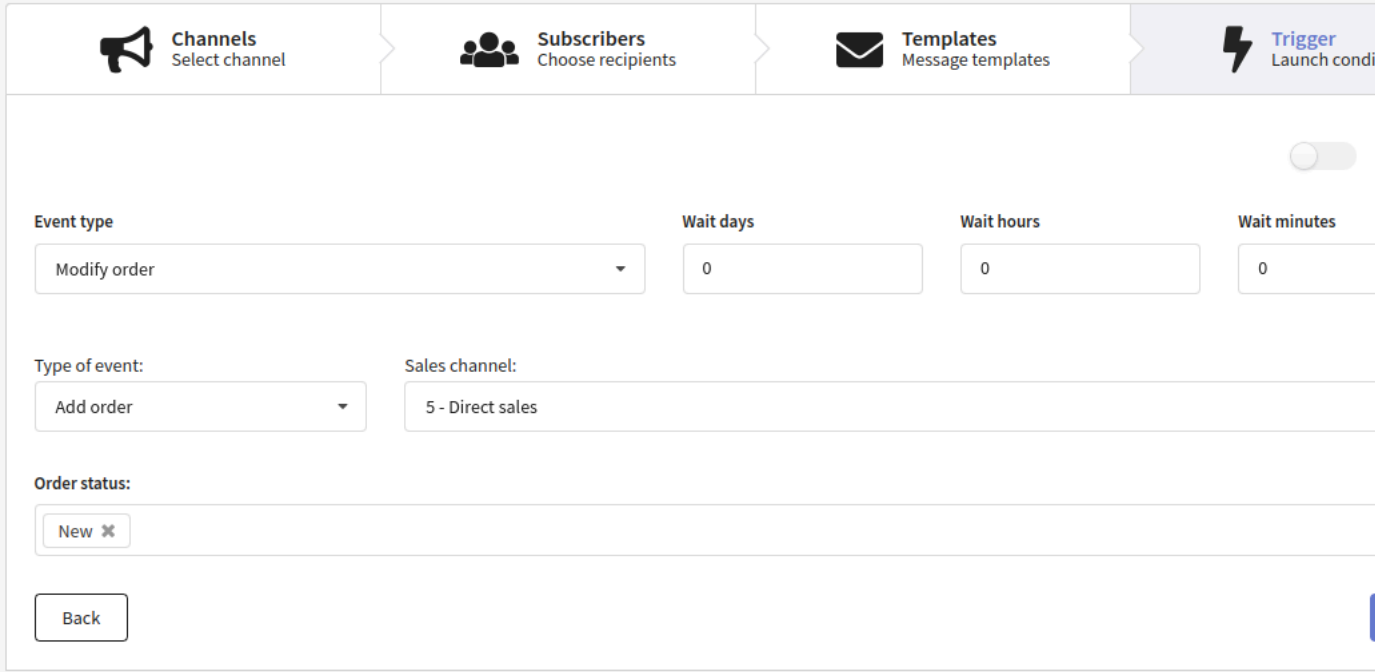
Step 5. Set up a trigger mailing

Open the Mailings section → Triggers. Create a new mailing. Don’t forget to activate it.



Select the email channel and add the desired email template. On the Trigger tab specify:

- **Event type: when changing an order —> adding an order. The trigger will be activated as soon as a n**
- **Sales channel: here specify the profile database with customer data of your online store.**
- **Order status: new. The trigger will be activated if an order with the status "new" will be added to the**



Click Finish and save the mailing.

TIP

To test how the trigger works, you can [add an order manually.]

Products

![img] (/img/attachments/49710301/56256323.png)	**Products are goods or services for sale. Each product has its own card with a name, description, image, and other information. ` product.** To start working with products, open the " Market" section → "Products". </td> </tr>

There are several ways to create a product or SKU:

1. **Manually**
2. **Import a product from a file**
3. **Import via feed on schedule**
4. **Import via API**

How to create a product manually

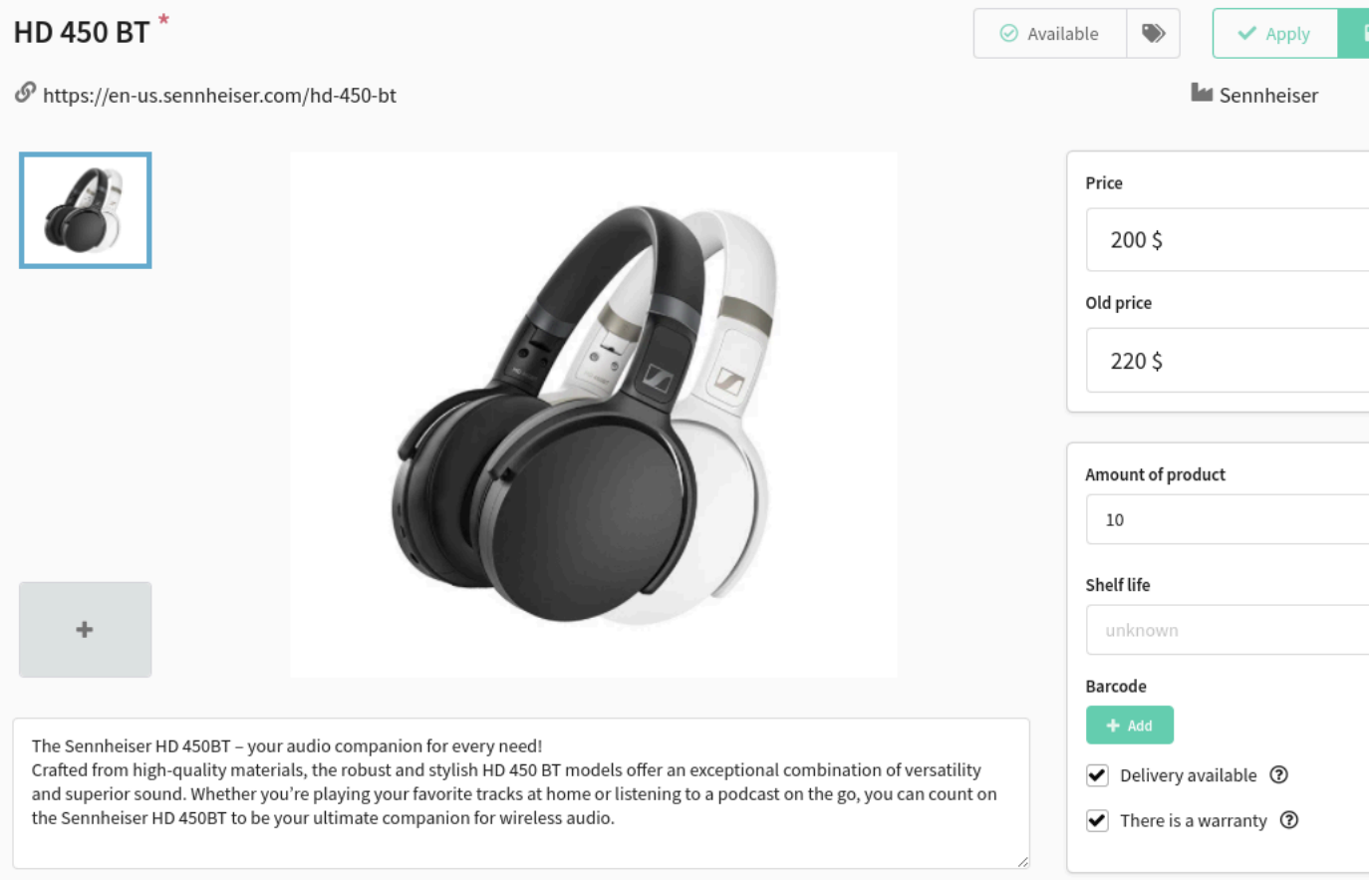
![img] (/img/attachments/56256325/56256327.png)	**In this guide, you will learn how to create a product or SKU manually in the Altcraft market.** To get started, go to the "Products" section and click "Create" in the upper panel. </td> </tr>

Basic product information

When creating a new product you must specify its name and additional id. The latter can consist only of contain up to 20 characters.

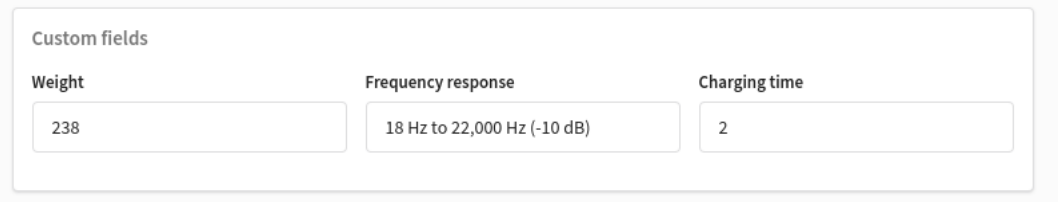
Other fields are optional:

- **product link — the URL of a product page on the store's website**
- **product images**
- **description**
- **available/unavailable — in stock or out of stock**
- **tags for quick search**
- **manufacture 's name**
- **price (select currency) — the current price of a product**
- **old price**
- **amount of product available for sale**
- **shelf life or service life (in days)**
- **barcode from the manufacturer in one of the formats (EAN-13, EAN-8, UPC-A, UPC-E), it is possible to**
- **delivery — the possibility of courier delivery (for all regions to which the store delivers).**
- **warranty — whether or not the product has an official manufacturer's warranty**



Custom fields

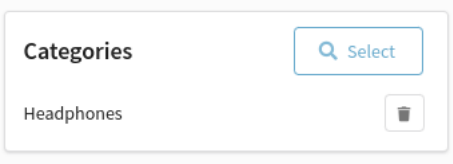
Fill in the custom product fields. For example, you can specify important characteristics of a product suc weight, model, etc.



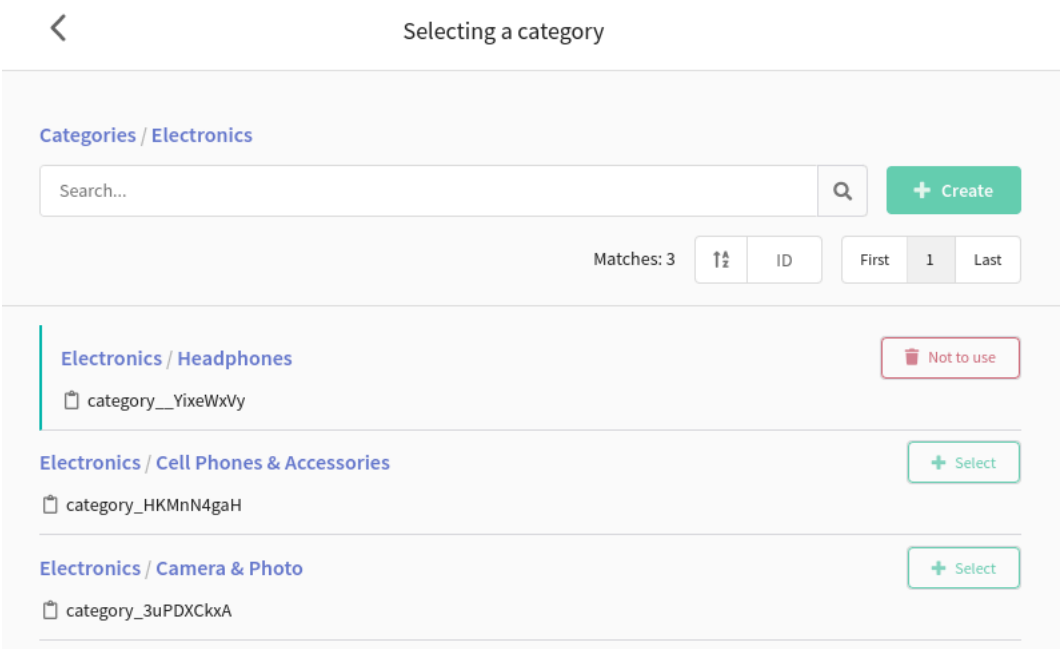
You can create custom fields in “Settings”.

Categories

Specify categories for a product by clicking “ Select”:

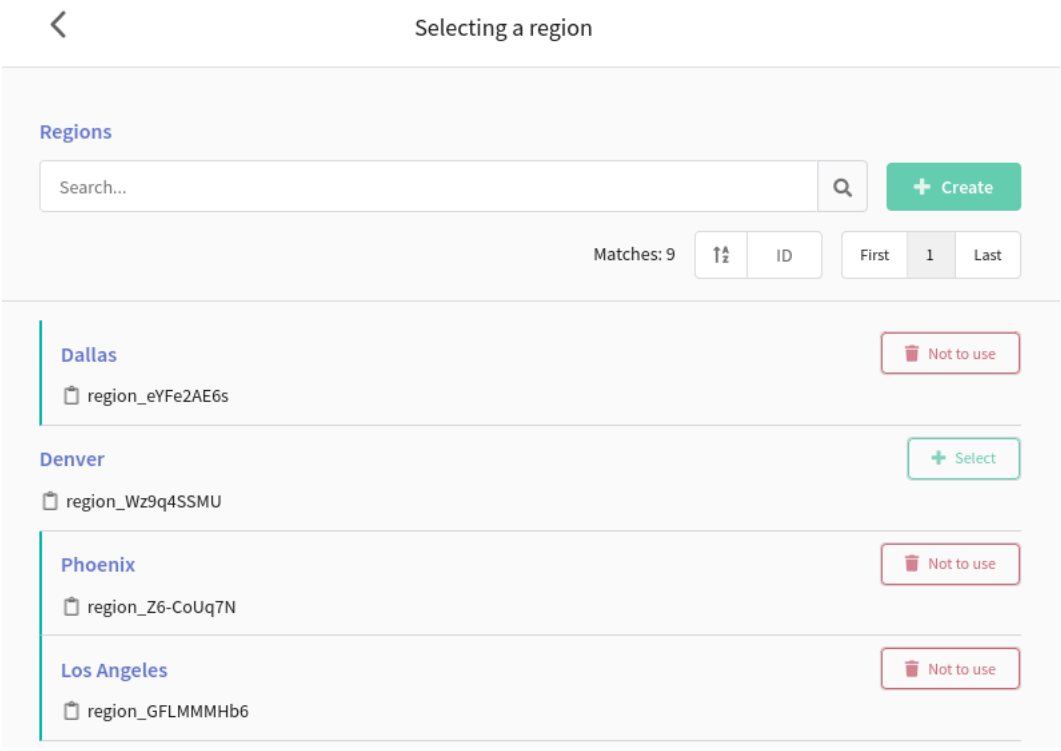


You can create categories in the market settings. Click “Create” to quickly jump to this section.
If you have already created the necessary categories, click “Select” to add a product there:

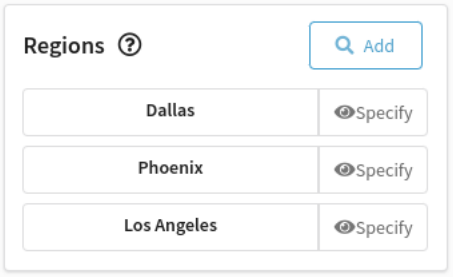


Regions

Add regions where your product is available:

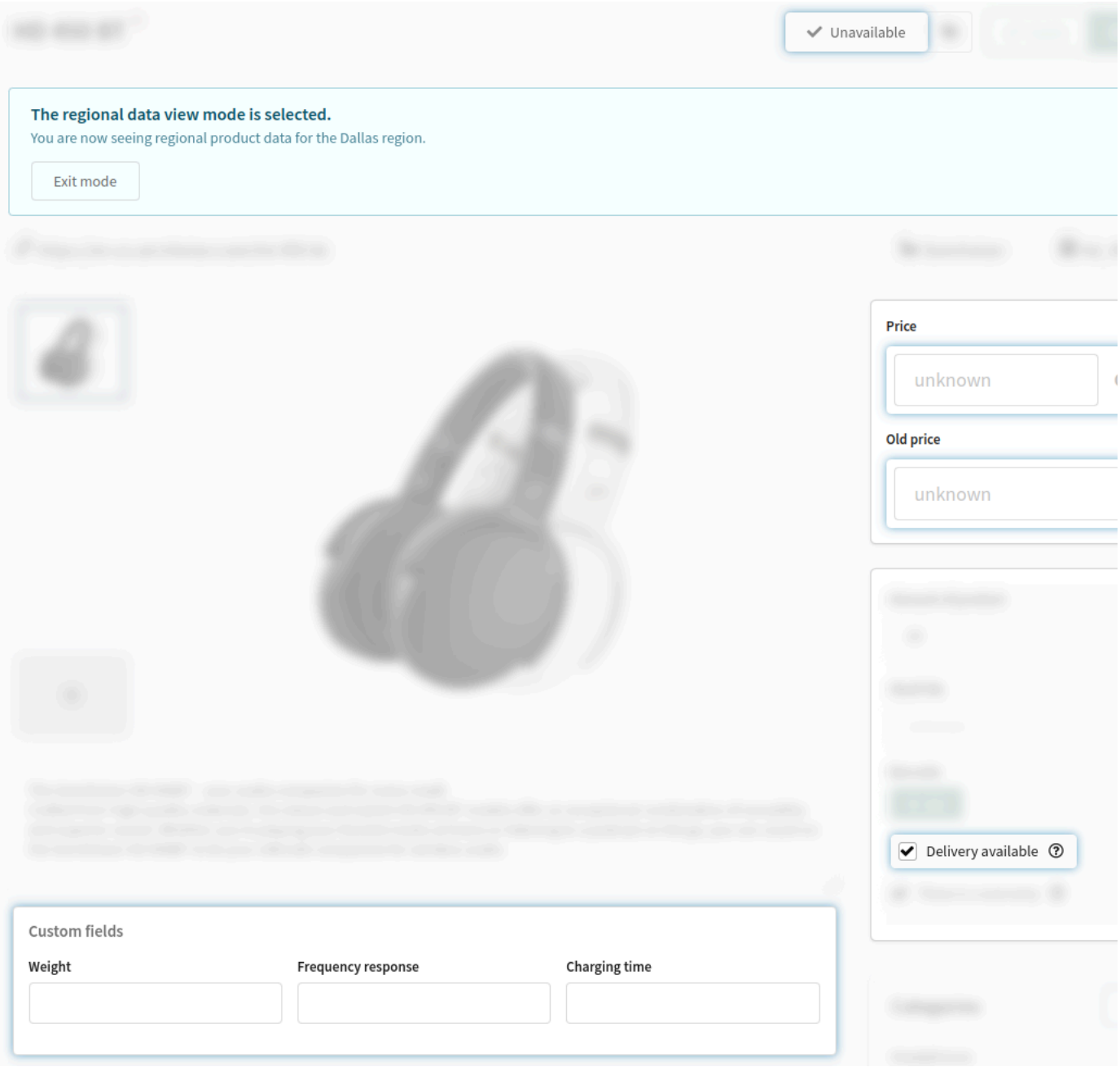


You can change product parameters depending on the region. If you do not deliver your product to some available” parameter for it.
Click “Specify” to edit product information for a certain region — you will switch to the regional data vie

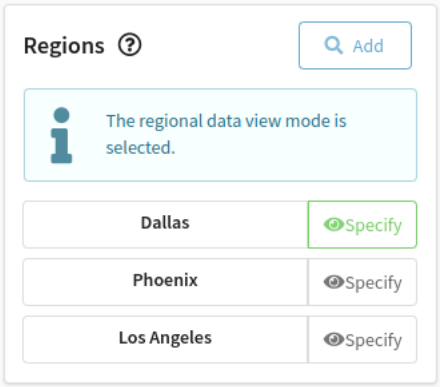


In this mode, you can change the following product information:

- **availability in the selected region;**
- **price;**
- **old price;**
- **delivery to the selected region;**
- **additional fields.**



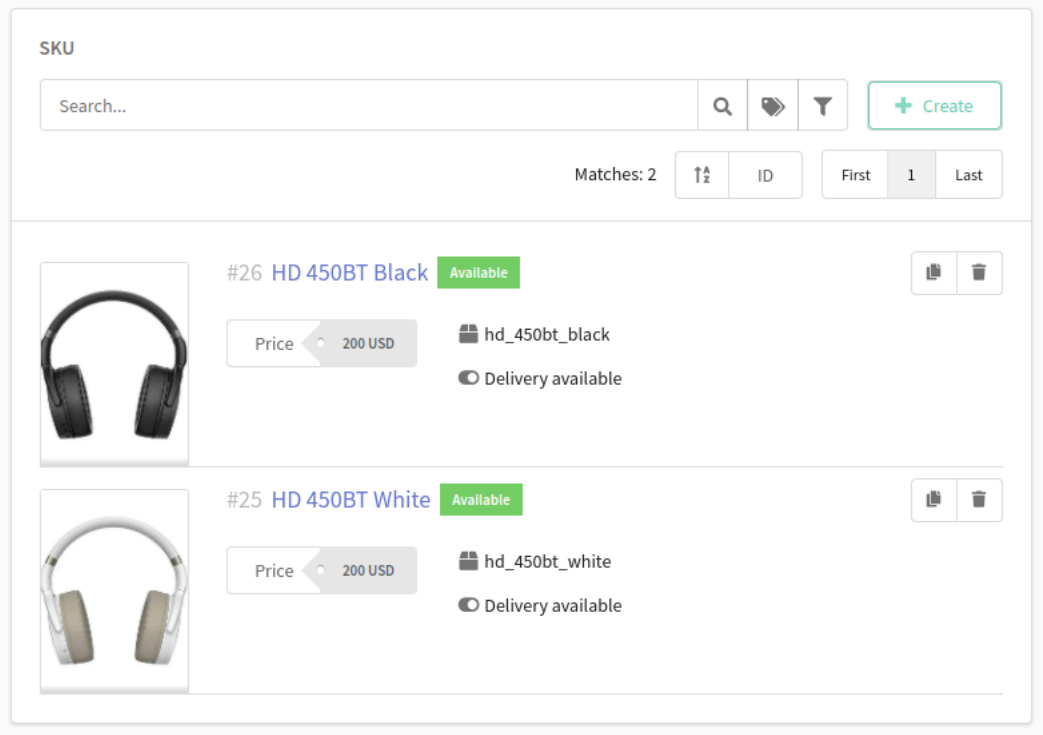
After filling in the regional data, click “Specify” next to a region to save changes:



SKU

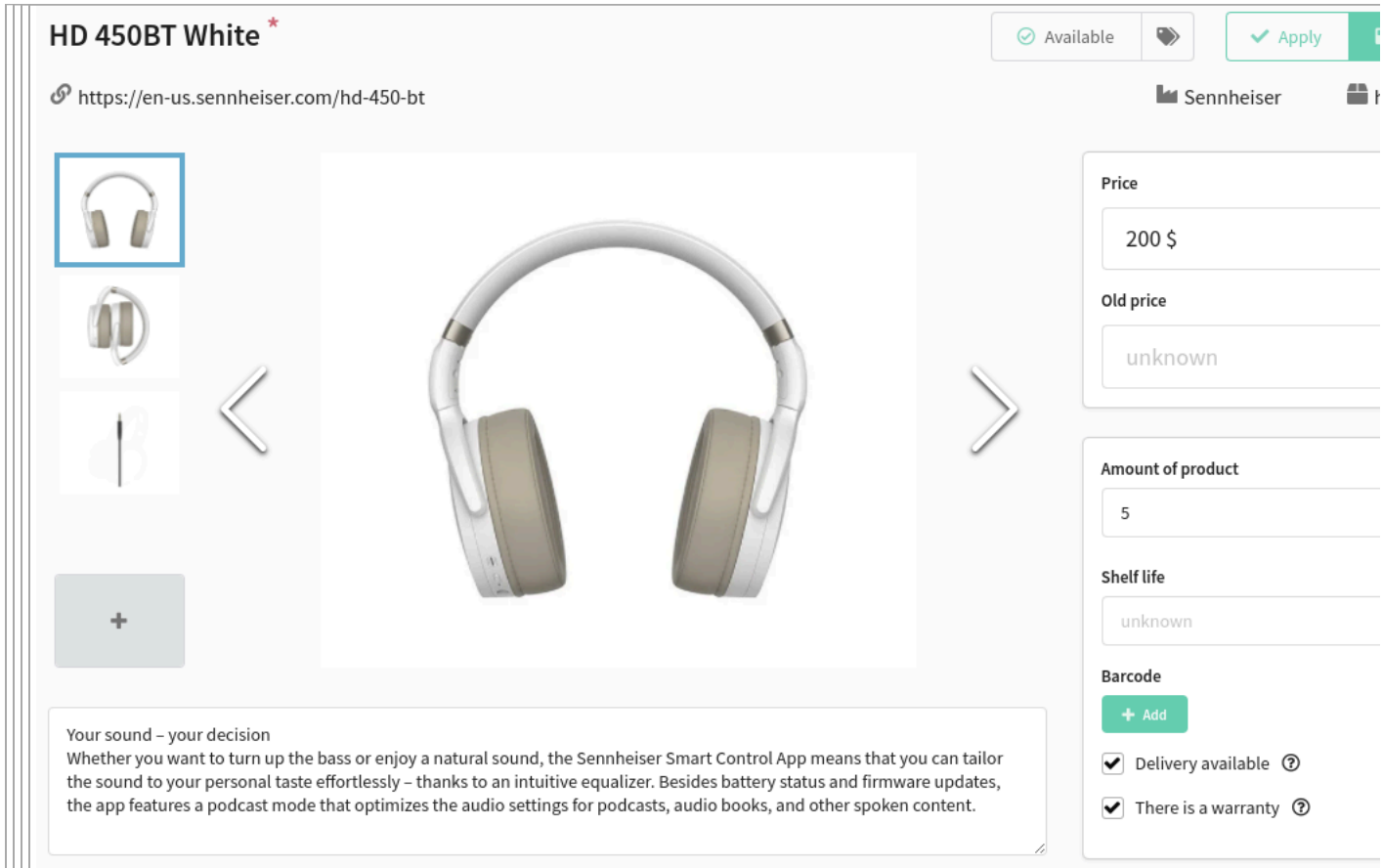
TIP

A stock keeping unit (SKU) is a product modifier that is attached to a group of products that differs fron (colour, size, weight, model, configuration, etc.). Each product has a unique SKU.



The process of creating a SKU is similar to creating a product:

- **enter a name and an additional identifier;**
- **fill in a product card (link, image, manufacturer, price, delivery, warranty, and so on);**
- **add a SKU to a specific product category;**
- **add regions and specify regional data for each of them;**
- **fill in the custom fields.**



Then save a product. Now you can use this product or its SKU when creating an order.

How to import a product from a file

This article describes how to import product data from a file into the Altcraft platform market. Products entities: products and SKU.

Product is a good or service for sale. SKU is a product modifier, i.e. a group of goods that stands out in o parameters (color, size, weight, model, equipment, etc.).

A SKU is tied to a specific product and cannot exist without it. For example, Apple iPhone 12 is a produc Apple iPhone 12 128GB are its SKU.

To upload product data to the platform, use a file in yml format or a .zip/.tar.gz archive containing yml file

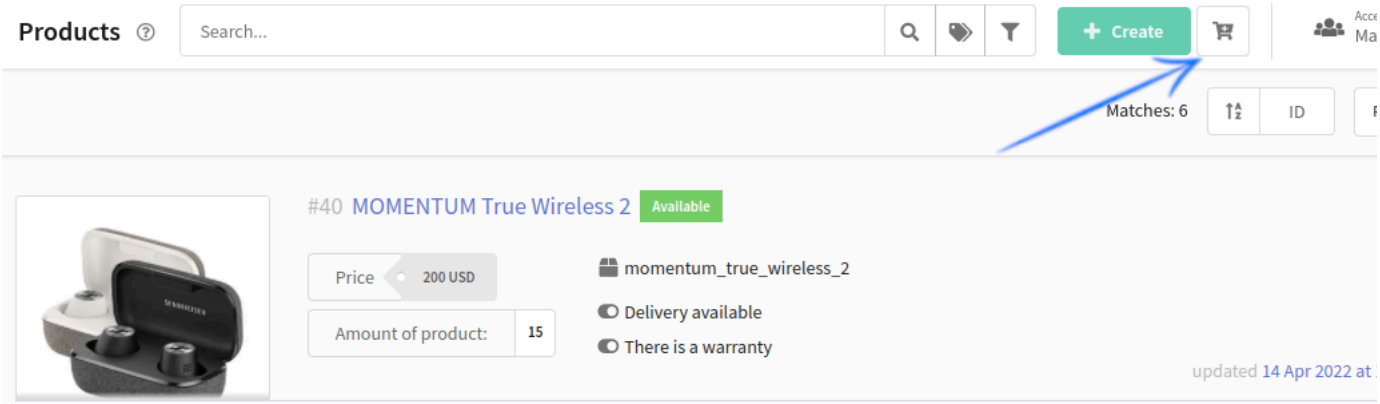
INFO

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. structure of the yml file, its parameters and properties, as well as examples of ready-made yml feeds w

File uploading

Click

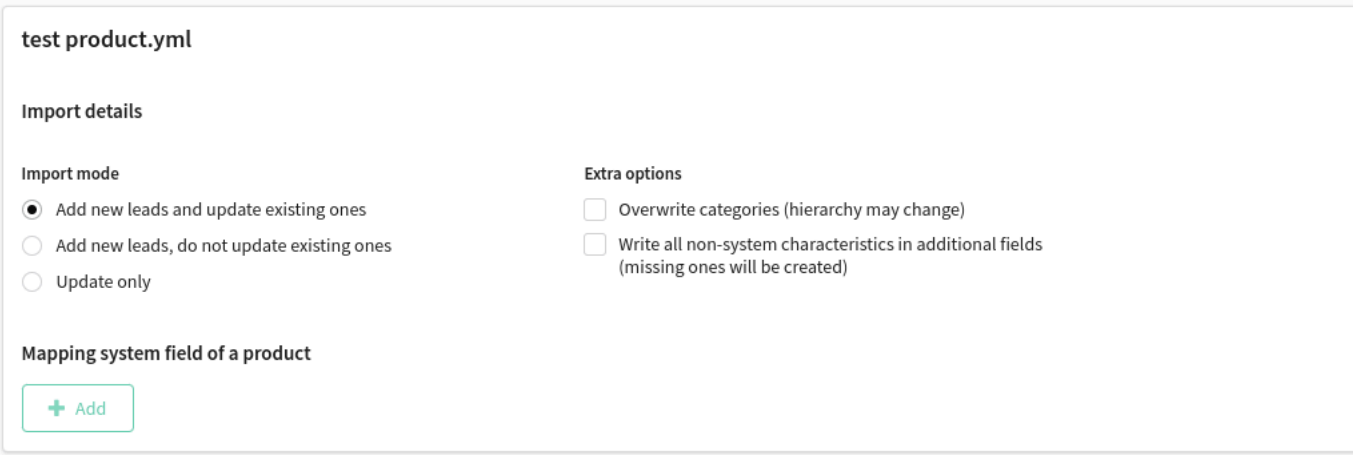
on the upper panel to import a product or SKU from a file. Select what you want to import: product or S



Import details

Specify the import mode. It depends on whether you are adding data about new products or updating inf already in the market:

- **Add new products and update existing ones**
- **Add only new products and do not update existing ones**
- **Update only existing ones, do not add new ones**

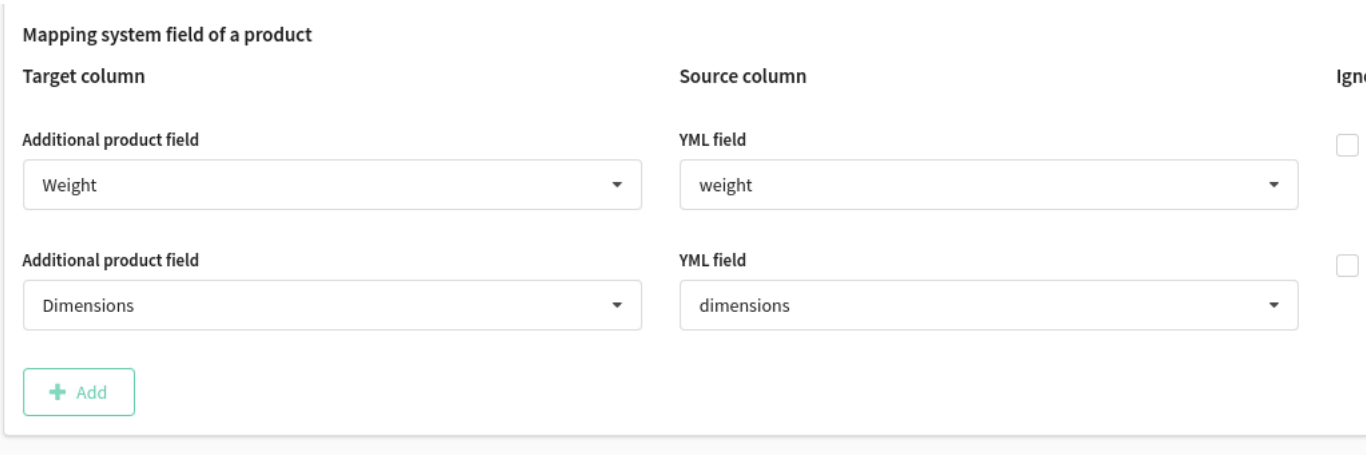


Extra options are activated in the “Import details” section:

- **Overwrite categories (hierarchy may change)**
- **Write all non-system characteristics in additional fields (missing ones will be created)— the platform param element in a yml file.**

Mapping system fields

In this block, you can configure the import of additional product information into the platform.
The target column is a custom field in the market, which is created in the "Settings" section.
The source column is a standard field in the yml file, the data from which you want to write to a custom



Now the import setup is complete. Start the process by clicking "Import" in the upper panel.

Scheduled product import

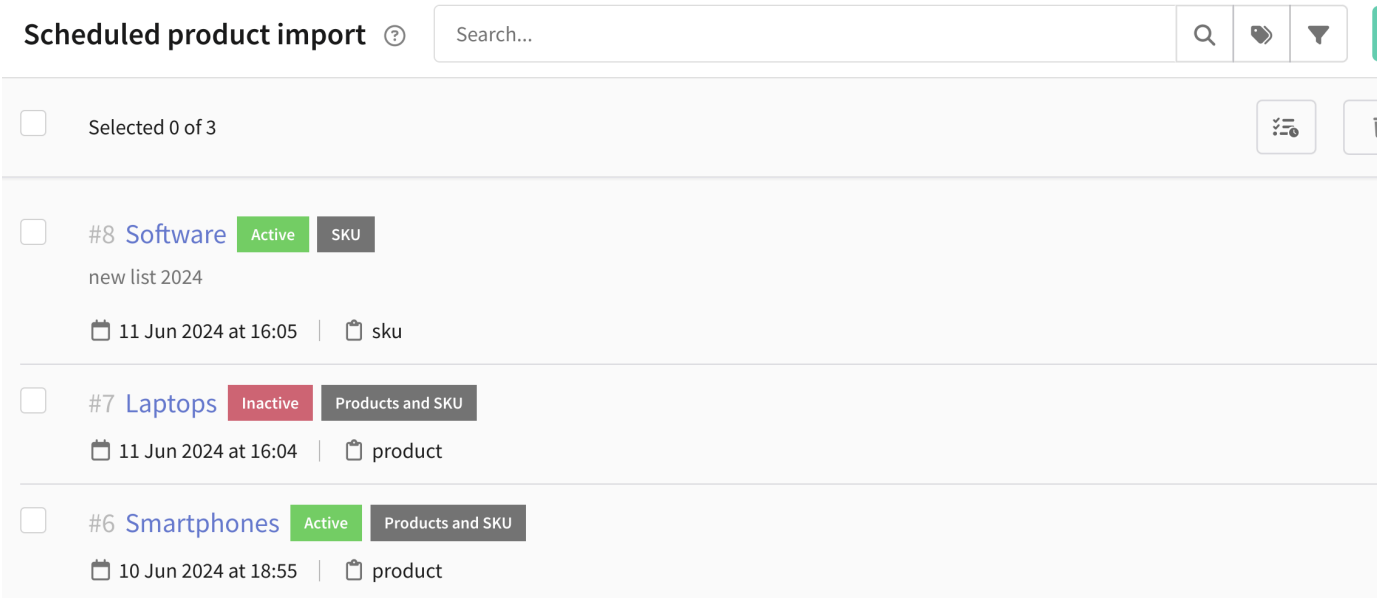
This article describes how to synchronize products and SKUs in Market with your product feed data. This to download a new yml file every time your feed is updated.
After setting up the import, new information about products will be automatically uploaded to the Market Information will be loaded from the yml feed with products, to which you will need to provide a link.

INFO

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. structure of the yml file, its parameters and properties, as well as examples of ready-made yml feeds w

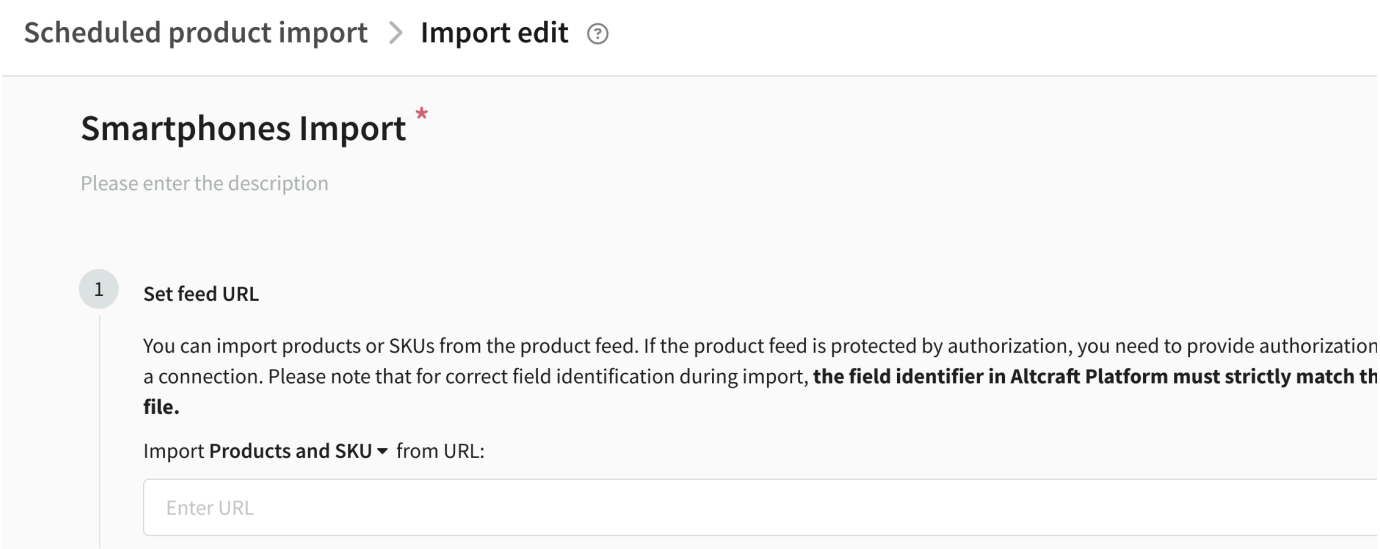
Step 1. Create a new import

Go to the Market section, select Scheduled product import. A list of existing imports will appear in front them. Here you can also start, pause, clone or delete an import.
Click the Create button in the upper right corner:



Step 2. Specify the feed URL

Insert the URL of your feed into the corresponding field. In the item Import products and SKU from URL will be loaded from the feed - products and their SKUs or only SKUs. In the first case the platform will fir all relevant SKUs, in the second case - new products will be completely ignored and SKUs will be loaded

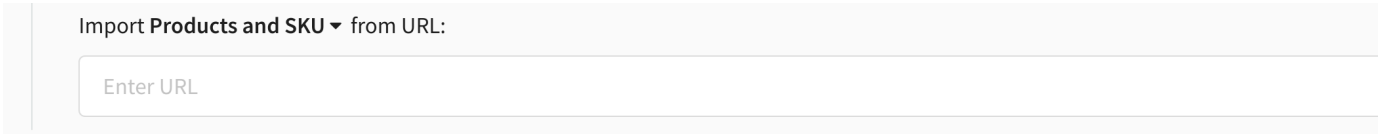


CAUTION

If your product feed has fields that are not provided by default in the Market platform, make sure you ha additional fields in Market Settings, otherwise the import will fail.

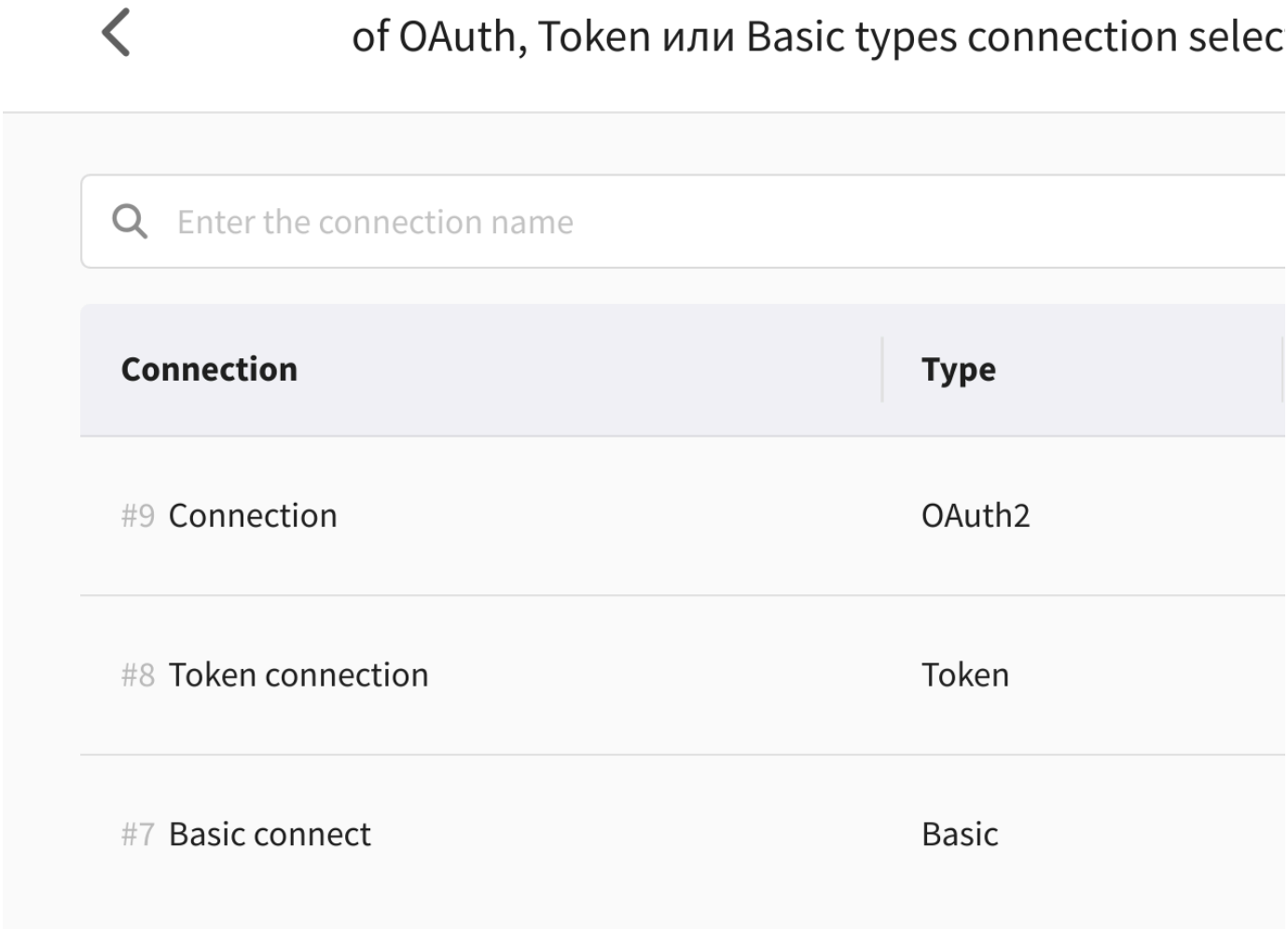
Connection selection

If your product feed is protected by authorization, you need to add connection to the import. You can do the key icon:



A connection selection window will appear in front of you. There are three types of authorization connec

- **OAuth2 is an authorization protocol that allows you to grant one service (application) the rights to ac service. The user does not need to trust the application with a login and password.**
- **Basic Authentication - HTTP authentication protocol that uses login and password to gain access to**
- **Token Authentication - a protocol that uses a special token that is provided to the user after success protected resources.**



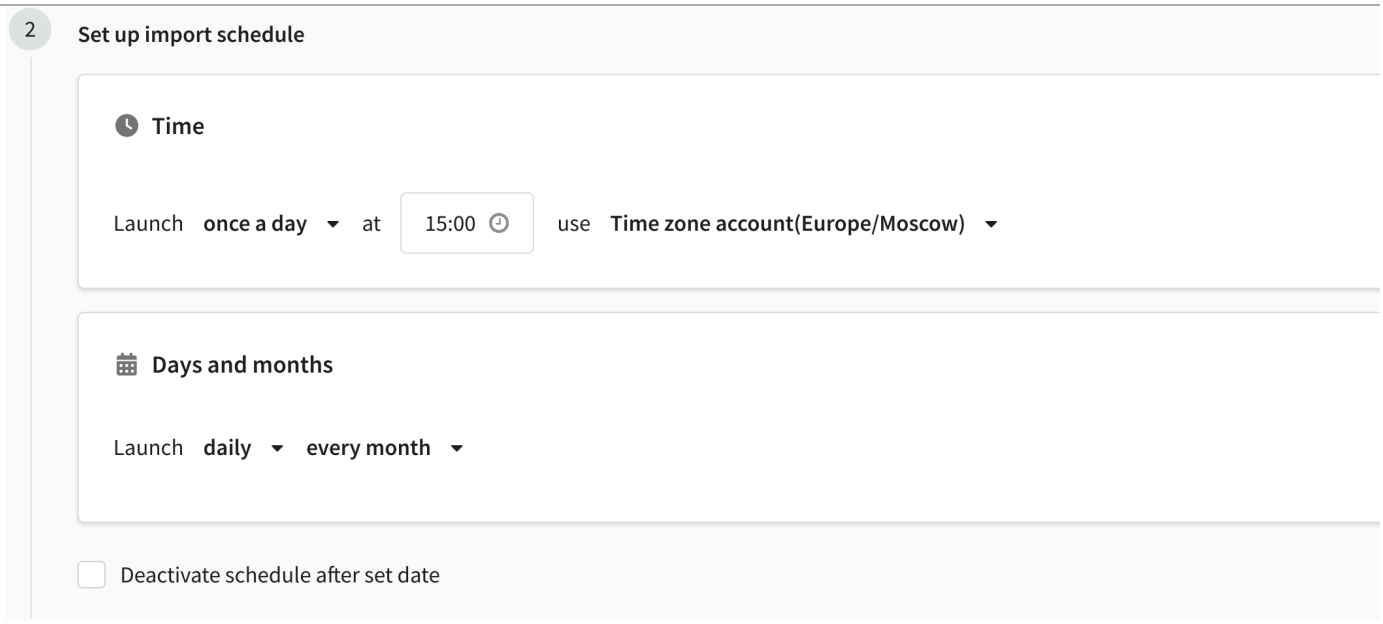
If there are no connections, you can create a new one directly in this window. To do this, click + Create a OAuth2, Basic Authentication or Token Authentication. Fill in the required fields and click **Save **.

TIP

More detailed information about creating and configuring a connection is available in the "Connections'

Step 3. Set up the import schedule

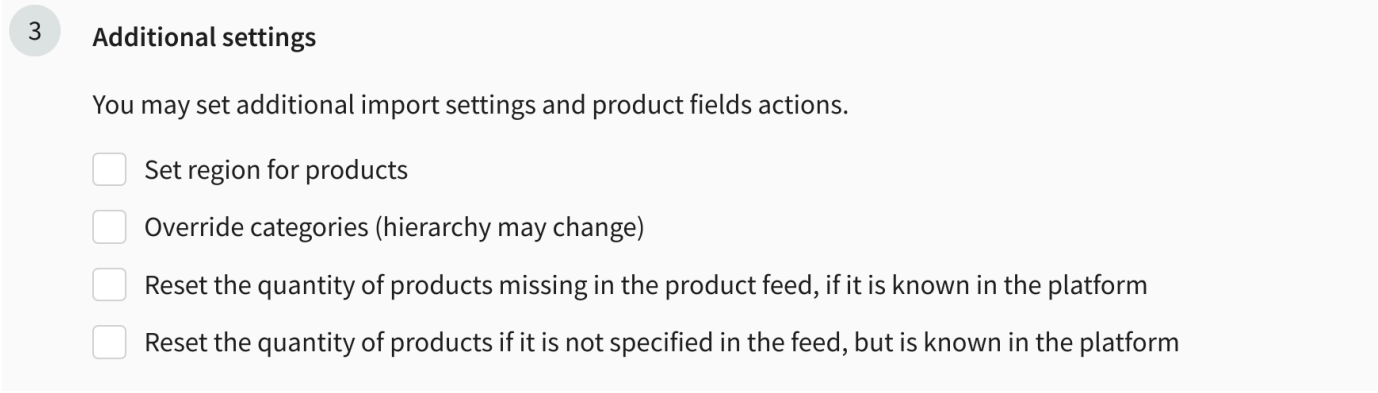
From the drop-down menu, select a schedule plan and customize depending on how often you want the p schedule plan, you can set specific time zone. You can also specify the import deactivation date:



Step 4. Specify additional settings

After setting the basic parameters, you can specify the necessary additional options:

- **Specify region for products - sets the region for imported products. The region must be created on th**
- **Override categories (hierarchy may change) - product categories will be overwritten in accordance v**
- **Reset the quantity of products missing in the product feed, if it is known in the platform - if any prod its quantity on the platform will become 0.**
- **Reset the quantity of products if it is not specified in the feed, but is known in the platform - if the qu the feed, then its quantity on the platform will become 0.**



Step. 5. Saving and starting import

To save the created import, click the Save button. You will see an overview of your import.

The following functions are available in the context menu:

— edit import

— launch and activate import right now

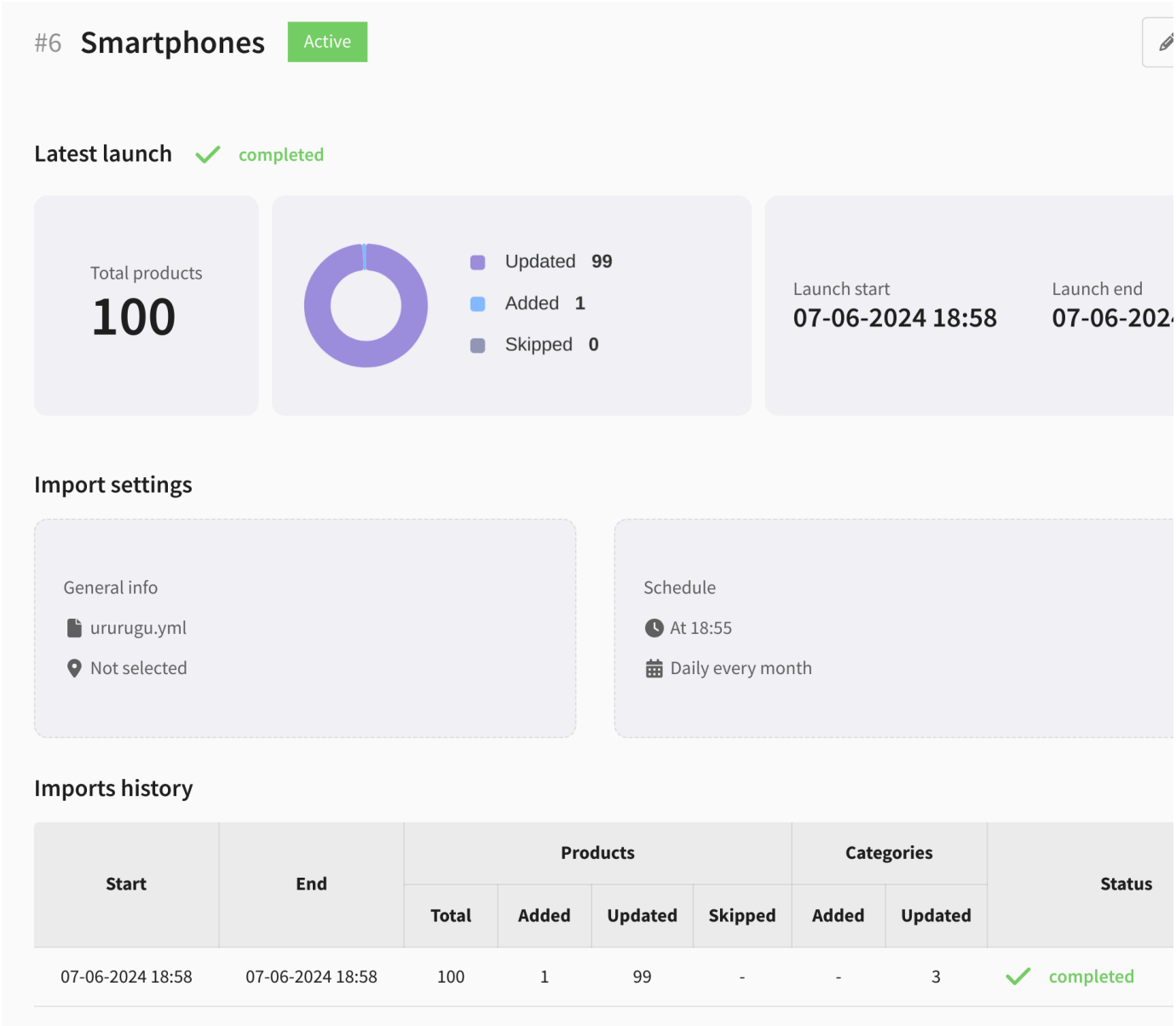
— activate/deactivate import

— assign tag

— assign access groups fot his import

— delete import

After the first launch, the imports history and information about added products will appear in the table l status and number of errors that occurred during the import. To switch the type of data displayed (produ Products/SKU switch:



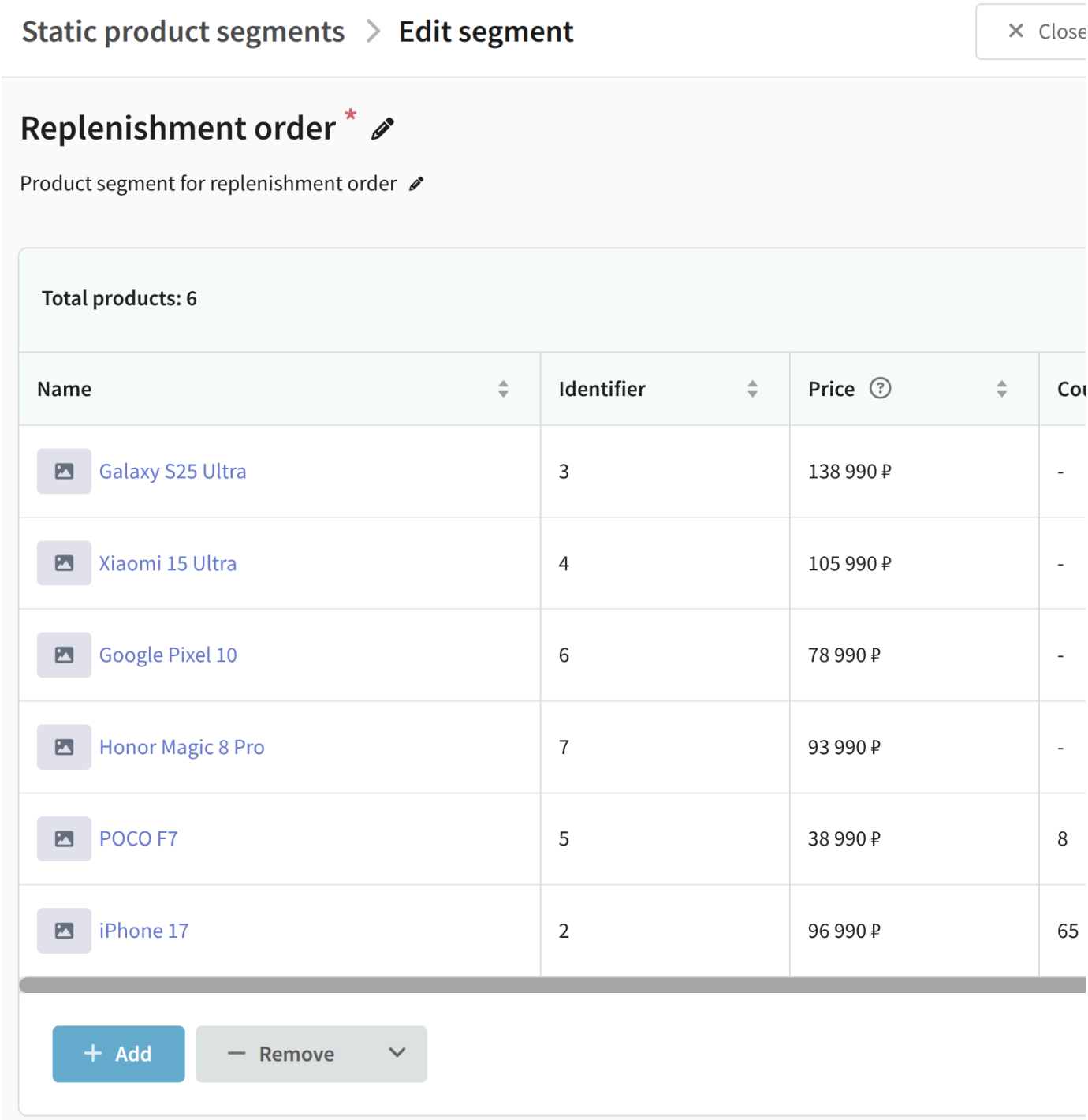
Product and SKU Segments

TIP: Segmentation in Altcraft

Product and SKU segmentation works the same way as profile segmentation in the platform. More deta segmentation logic are available in [this section].

Description

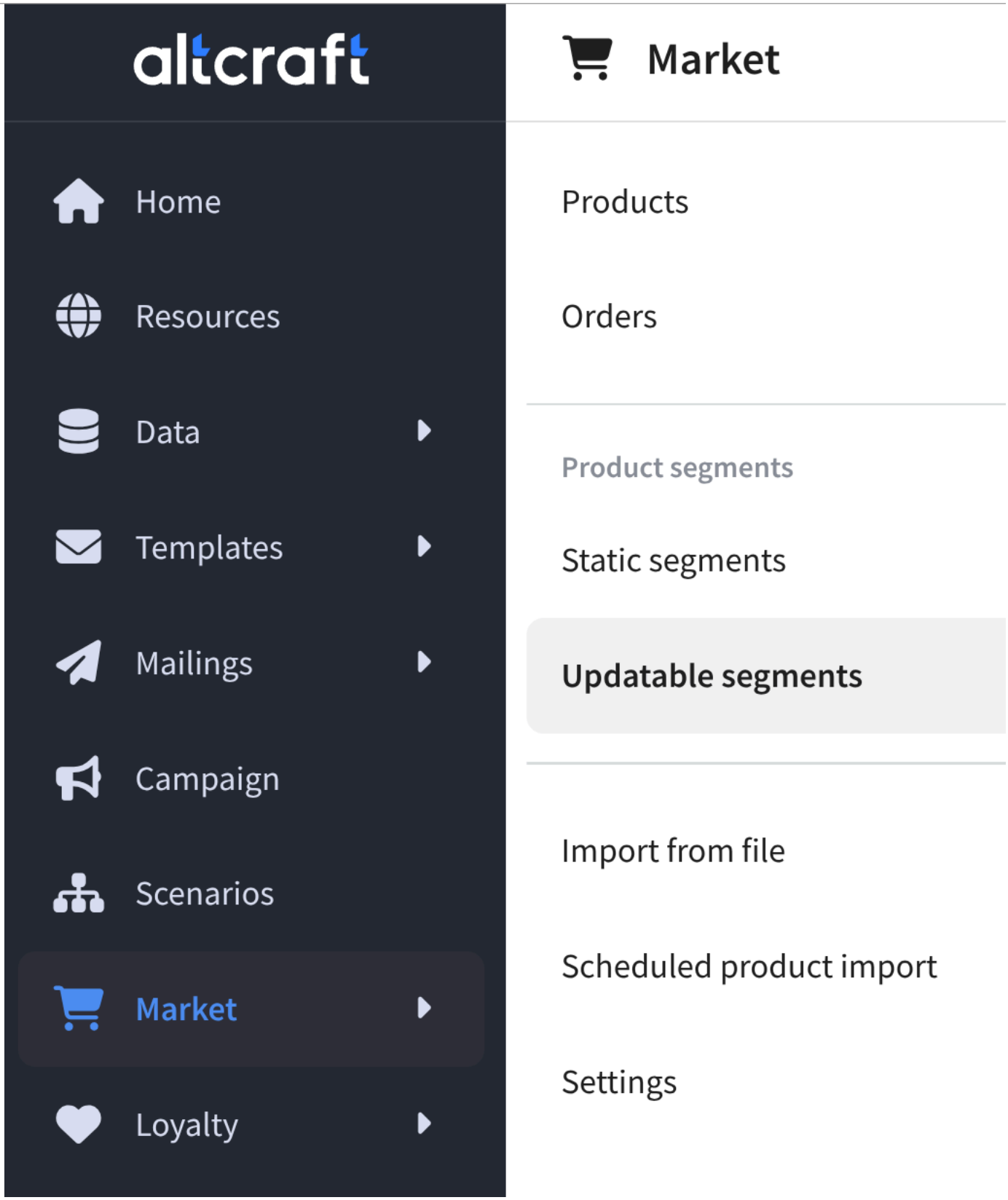
Product and SKU segments are sets of products and their modifiers (SKUs) grouped by a specific criteri exported and used for sales data analysis. There are two types of segments: static and updatable.



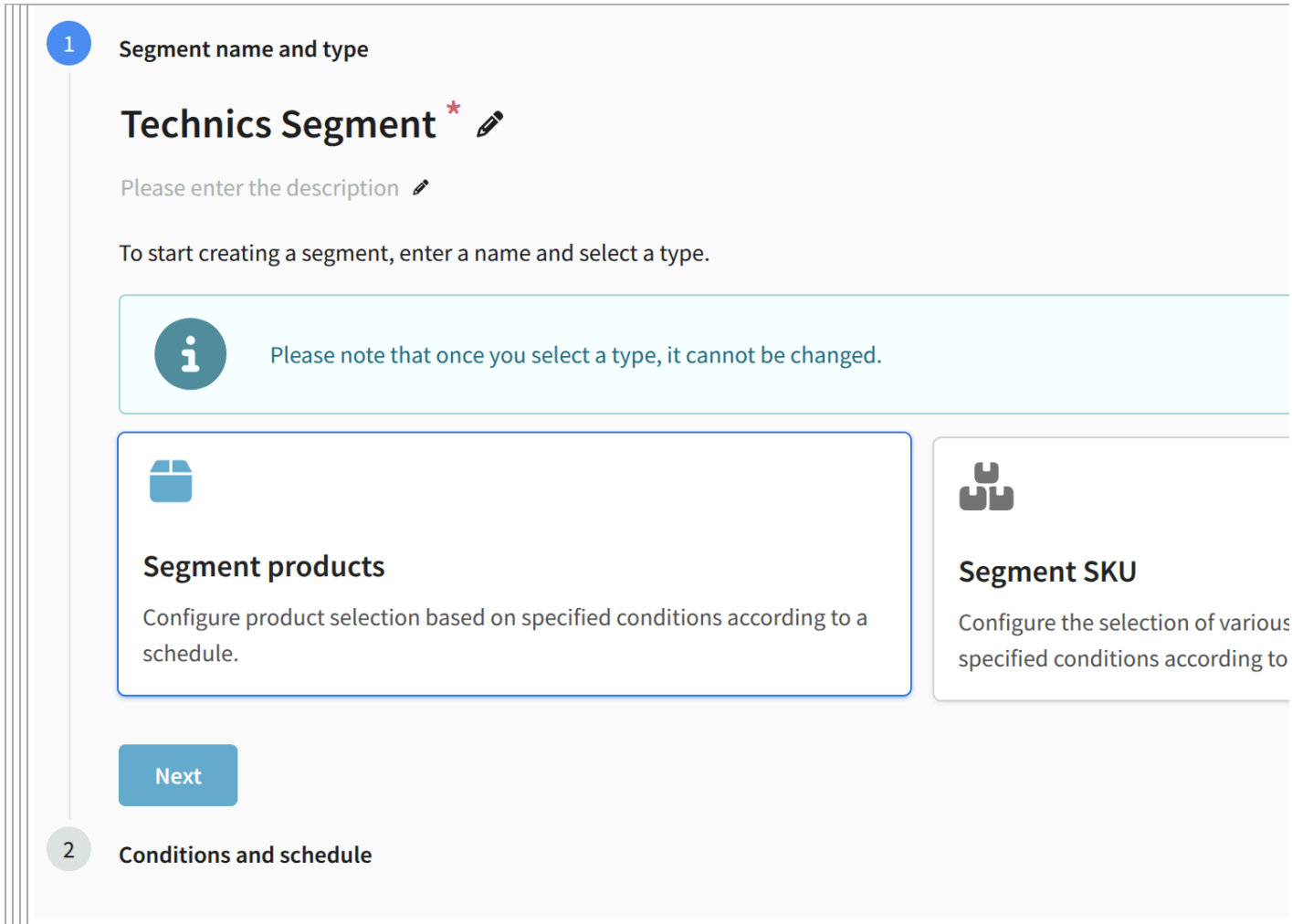
Updatable Segments

Updatable segments are product and SKU sets that are recalculated on a schedule based on defined cor recalculated manually outside the schedule.

To create an updatable segment, go to Market — Updatable segments:

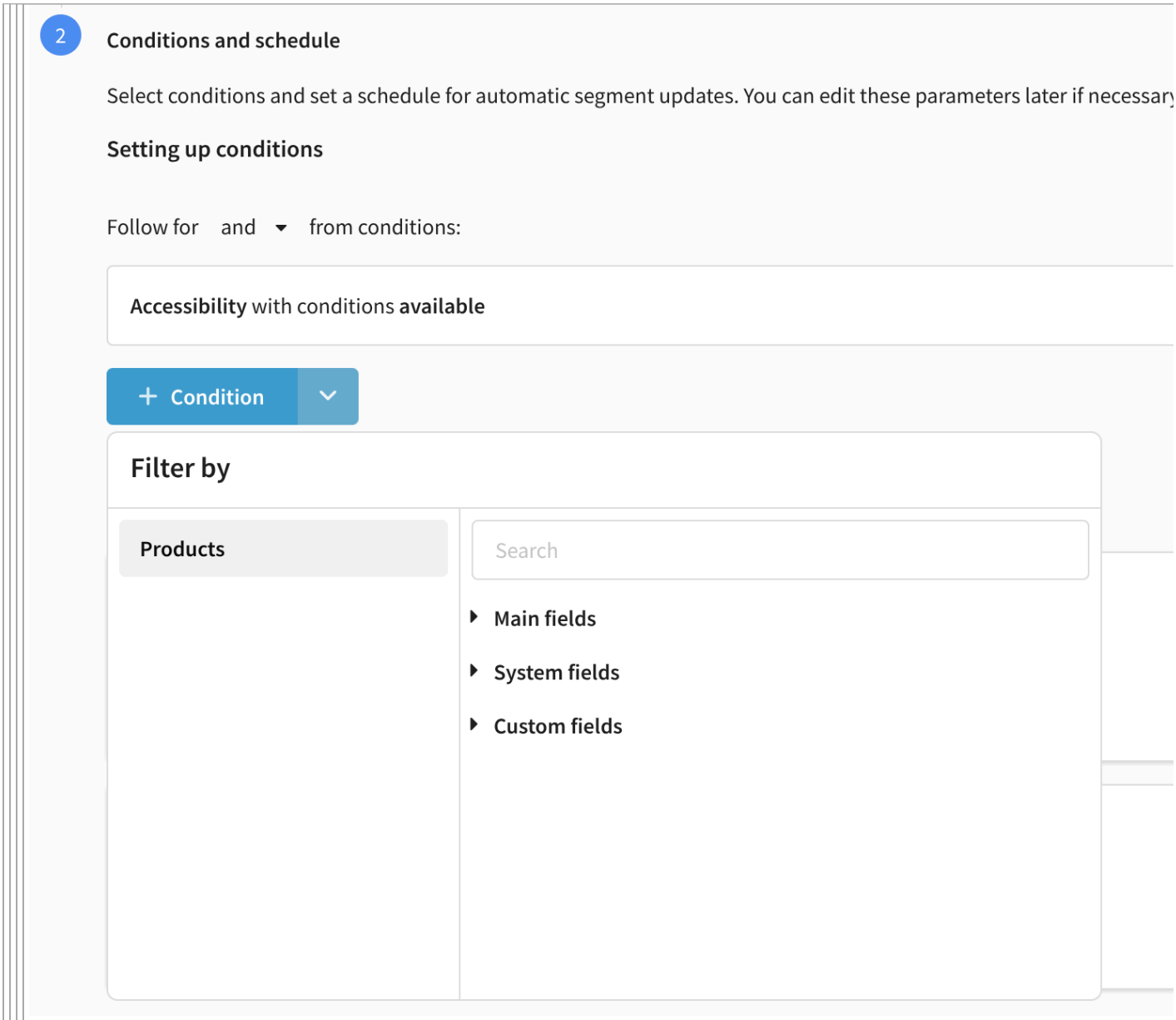


Click “Create”. Specify a name, description, and select entities for segmentation — products or SKU. Aft be changed:



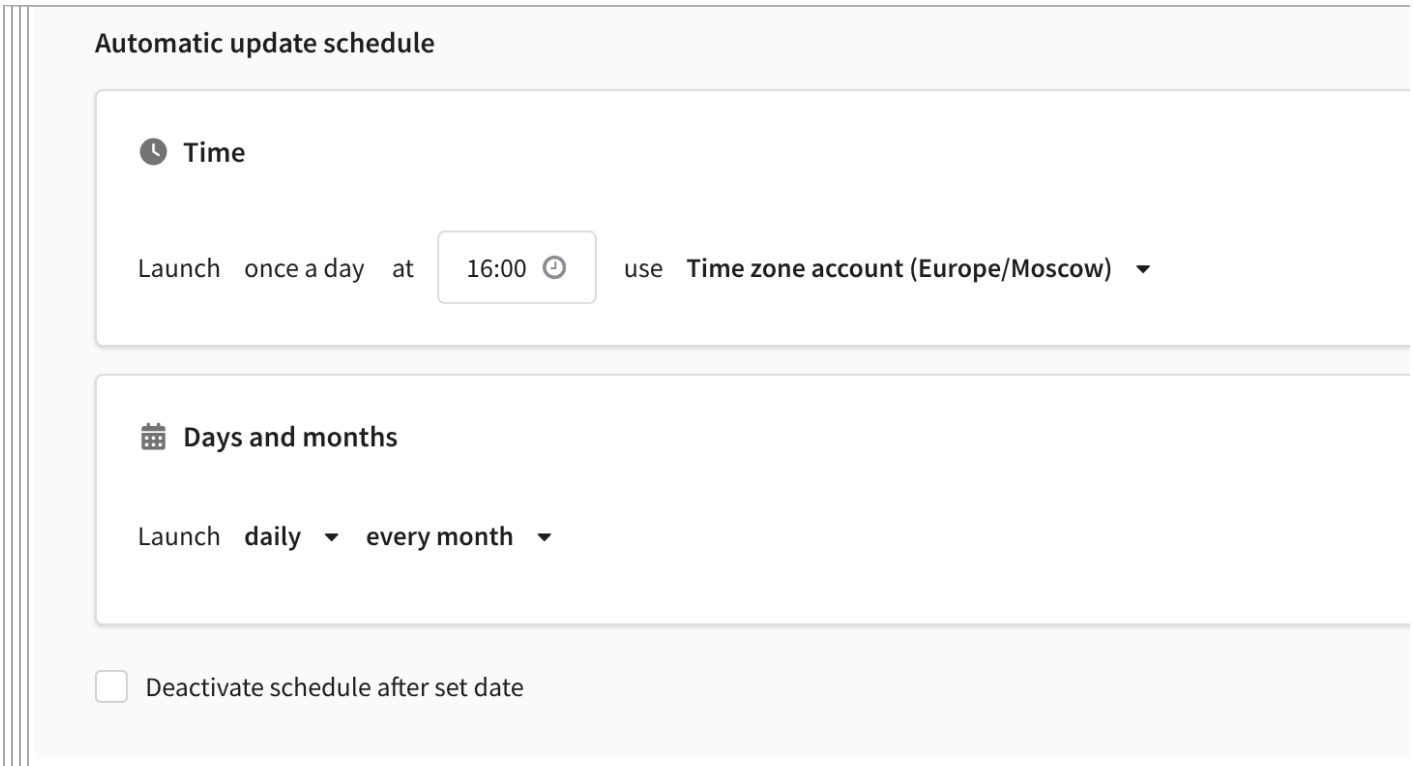
Then the Conditions and schedule block opens. Select conditions or condition groups and set a schedule inside a group can be combined using match all conditions (AND) or match any condition (OR). These set Conditions are divided into three groups:

- **Core fields — filtering by name, region, category, product quantity, expiration date, price, delivery, w and tags.**
- **System fields — filtering by identifier, additional (string) identifier, creation date, and update date.**
- **Custom fields — filtering by user-defined fields.**

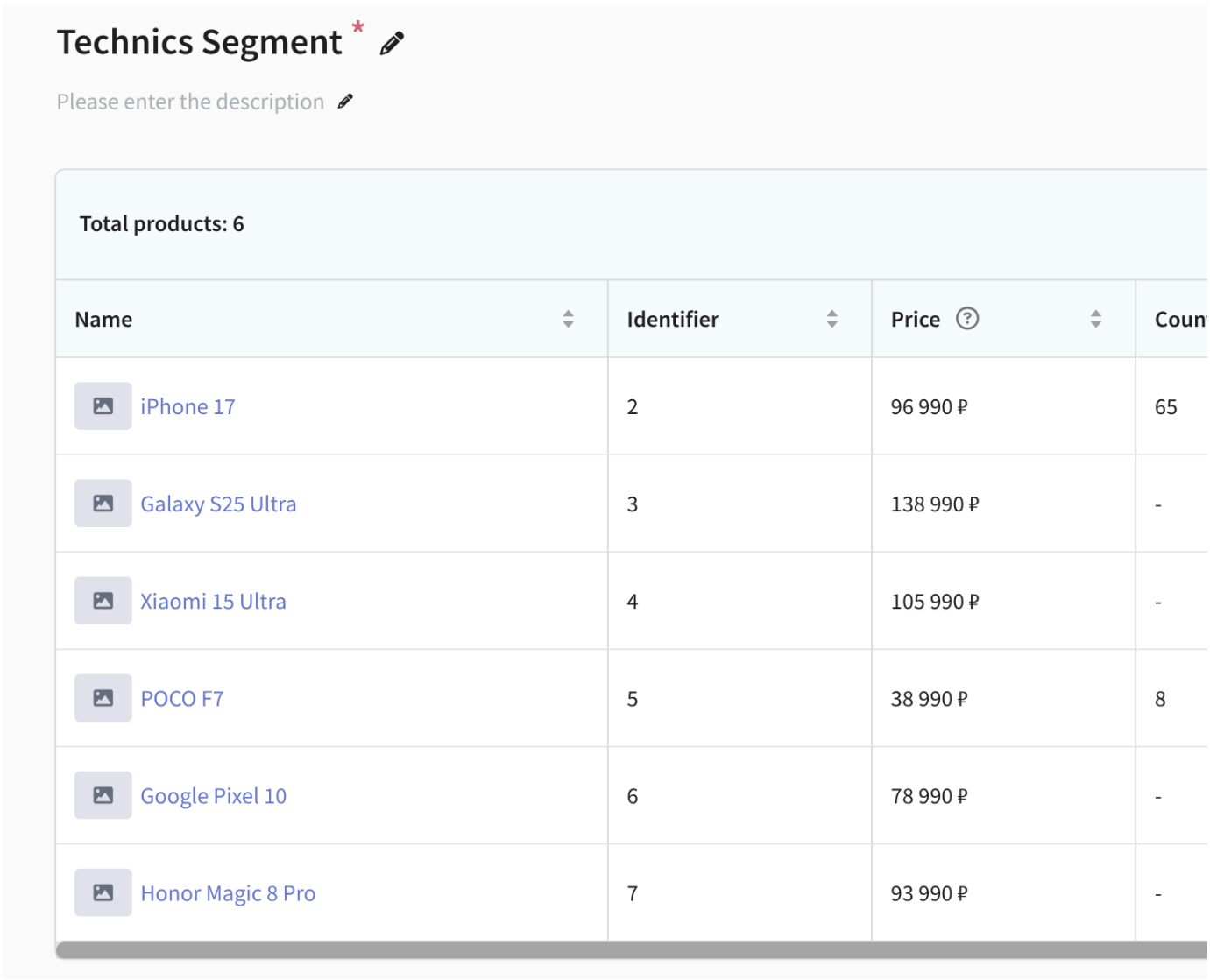


The same block contains the Script button, which opens an editor for manual condition writing. After set product count limit. This limit applies to the selection result after condition filtering.

A segment can be updated daily or on specific days. Update time and time zone are set by the user:



You can also configure automatic segment deactivation after a specified date.
After calculation, the segment view displays a table with all products included in the segment. It shows "Price", "Quantity", and a button to open product editing:



Name	Identifier	Price	Coun
iPhone 17	2	96 990 ₽	65
Galaxy S25 Ultra	3	138 990 ₽	-
Xiaomi 15 Ultra	4	105 990 ₽	-
POCO F7	5	38 990 ₽	8
Google Pixel 10	6	78 990 ₽	-
Honor Magic 8 Pro	7	93 990 ₽	-

Available actions for this table:

- **Search**

  img

  : find a specific product.
- **Export products**

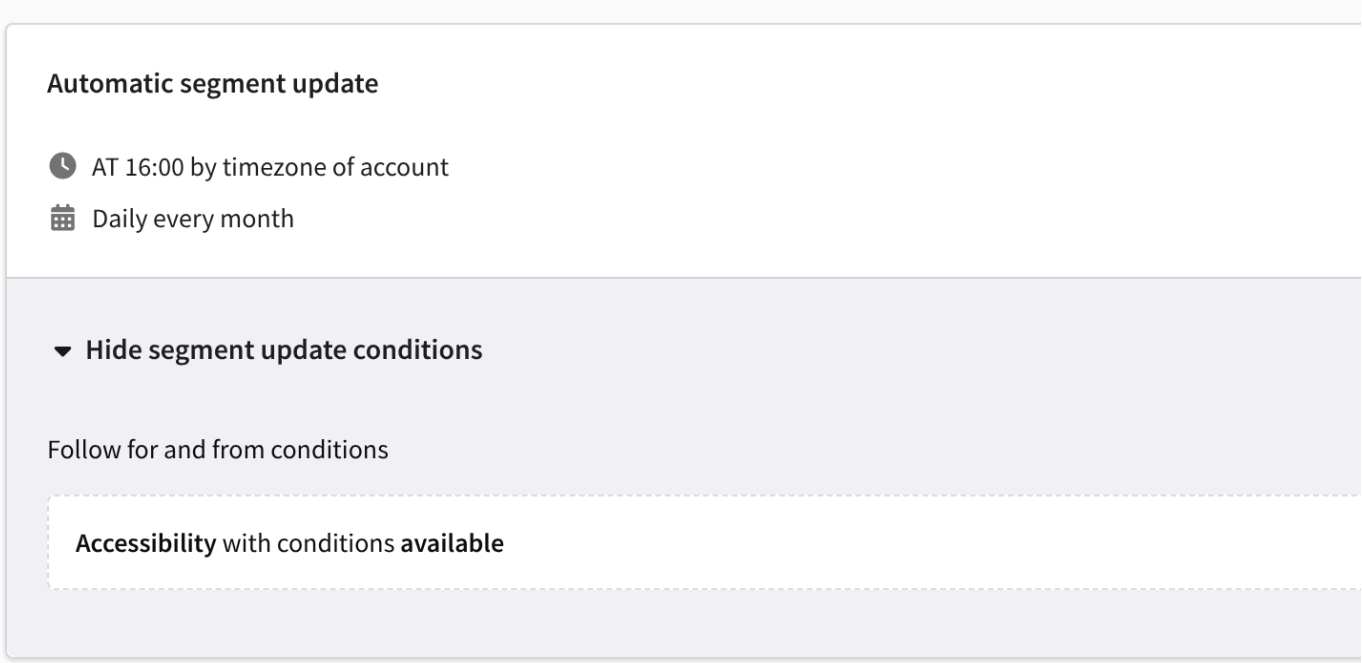
  img

  : export the displayed list to an external file.
- **Configure displayed fields**

  img

  : add or remove table columns by dragging a Field from the Add field sidebar to Displayed fields or ba

Below the product table are the segment update schedule and conditions. Here you can edit them or rec schedule:



Below is the Update history block. It visually shows changes in the total number of products in the segm specify a date range and group data by days or weeks.

Below the update history is the Log. It contains a detailed record of all operations affecting the segment following is shown:

- **Start date: time when the segmentation process started (adding/removing by conditions).**
- **End date: time when the process finished.**
- **Initiator: name of the user who initiated the operation.**
- **Added: number of products added as a result of the operation.**
- **Removed: number of products removed as a result of the operation.**
- **Status: current operation state (for example, “Completed”, “In progress”, “Error”).**

Log

Start date	Finish date	Type	Initiator	Added	
10-01-2026 15:36	10-01-2026 15:36	Manual	Marketing User	65	(
13-01-2026 15:57	13-01-2026 15:57	Manual	Marketing User	13	(
15-01-2026 15:58	15-01-2026 15:58	Manual	Marketing User	19	8
17-01-2026 16:00	17-01-2026 16:00	Manual	Marketing User	29	(

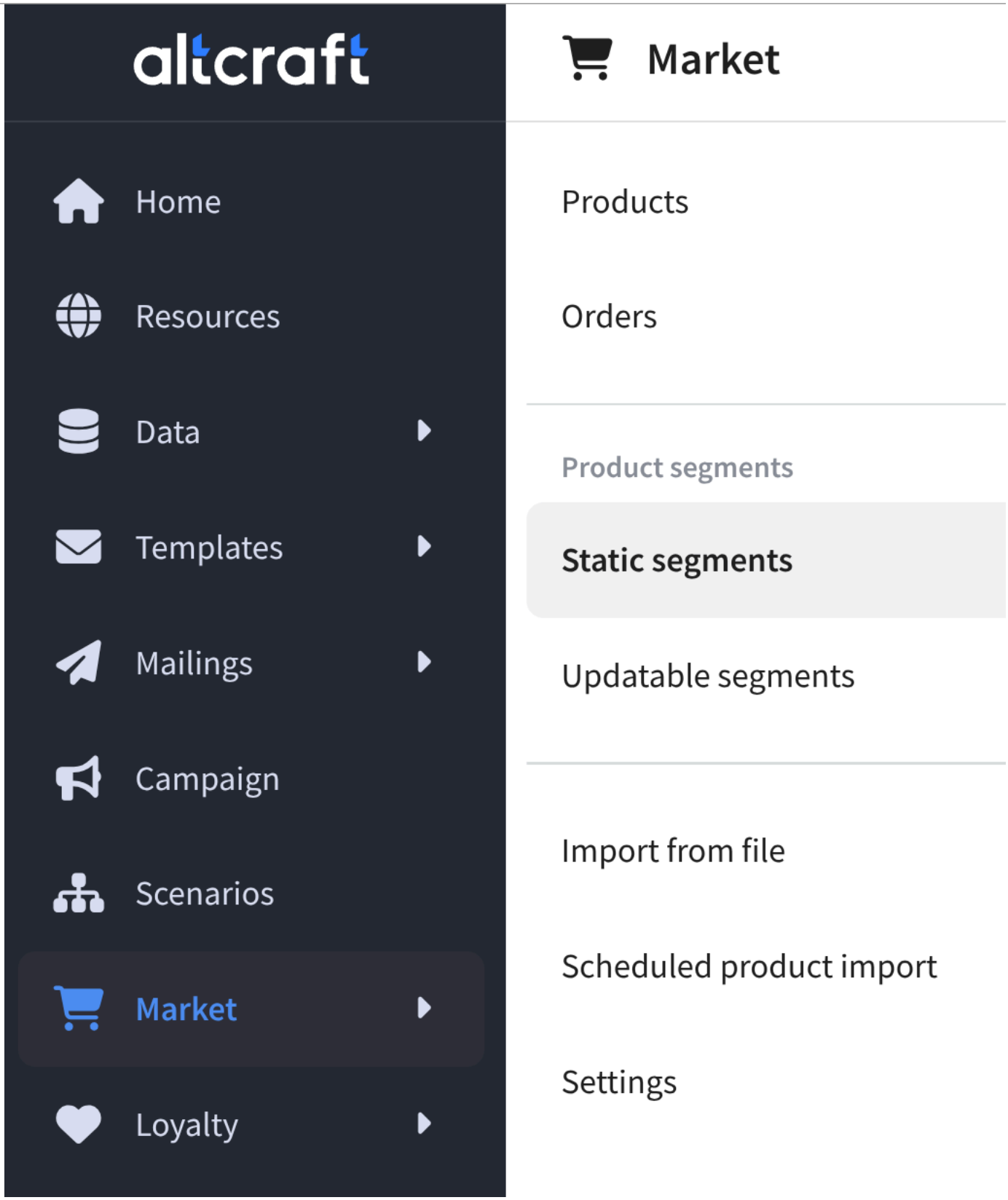
TIP: Segment actions

At the top of the screen, a quick actions menu is available, where you can assign tags, configure acces another segment.

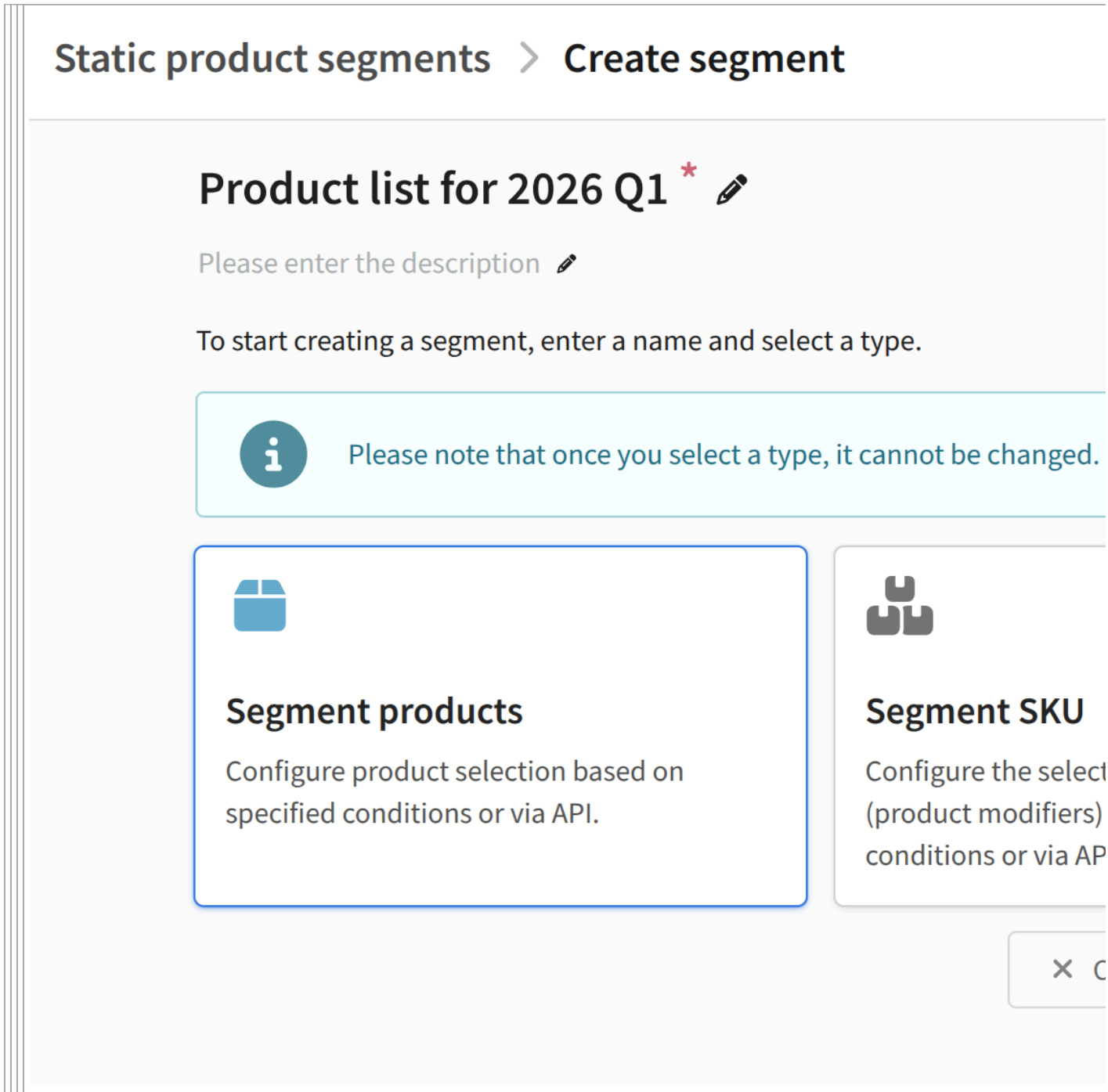
Static Segments

Static product segments are sets of products and their modifiers (SKU) grouped via API or manually usin

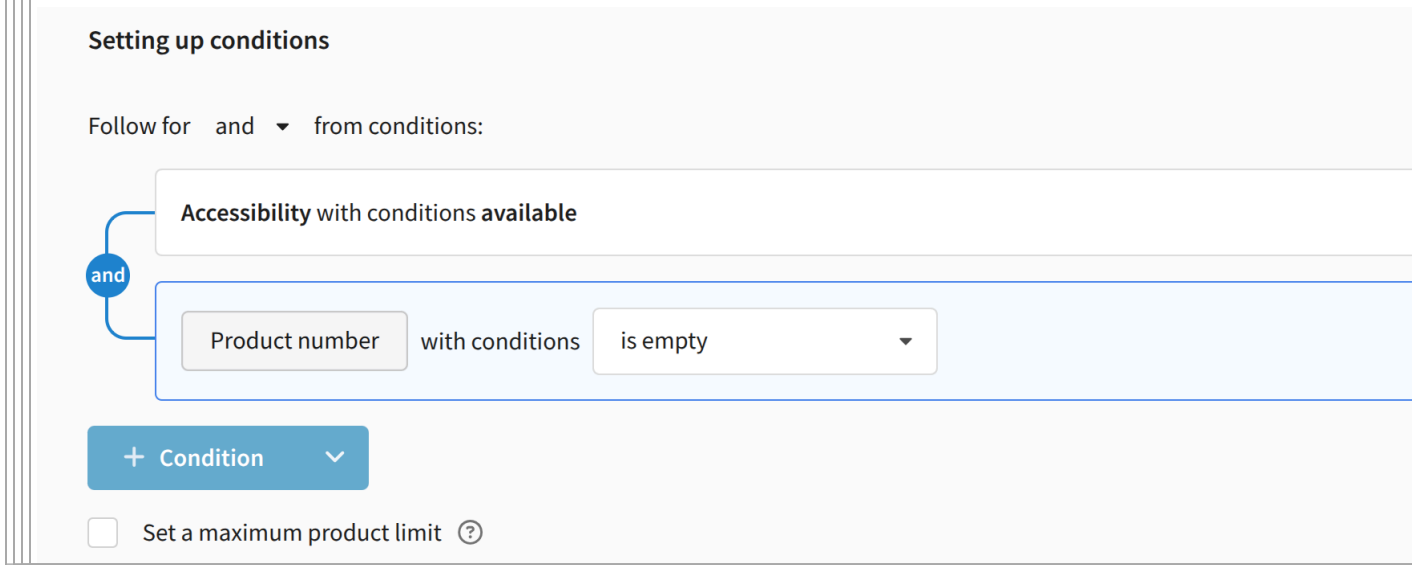
To create a static segment, go to Market — Static segments:



Click “Create”. Specify a name, description, and select entities for segmentation — products or SKU. Aft be changed:



The segment is created. To add products, click “Add” — the Conditions block opens.
To add a rule, click + Condition. The dropdown menu contains the Add condition group option. Conditions using match all conditions (AND) or match any condition (OR):



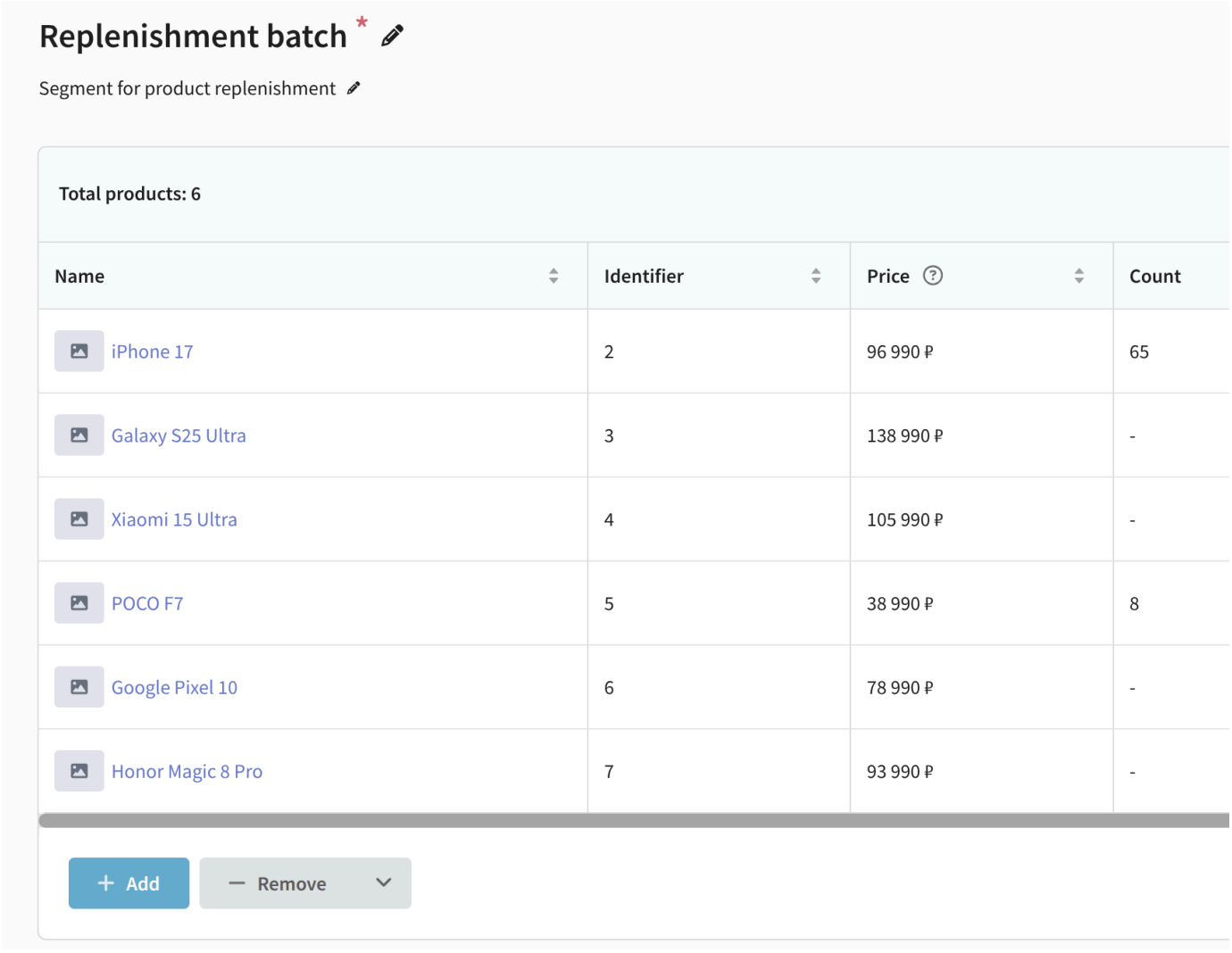
Conditions are divided into three groups:

- **Core fields — filtering by name, region, category, product quantity, expiration date, price, delivery, w and tags;**
- **System fields — filtering by identifier, additional (string) identifier, creation date, and update date;**
- **Custom fields — filtering by user-defined fields.**

The same block contains the Clear button, which fully removes all defined rules, and the Script button, w condition writing. After setting conditions, you can define a product count limit. This limit applies to the filtering.

After clicking “Add”, products matching the selection rules appear in the segment. To remove products f removal function works the same way as adding products.

After calculation, the static segment view displays a table with all products included in the segment. It s “Price”, “Quantity”, as well as a quick actions menu for removal or opening product editing:



Available actions for this table:

- **Search**

  img

  : find a specific product.
- **Export products**

  img

  : export the displayed list to an external file.
- **Configure displayed fields**

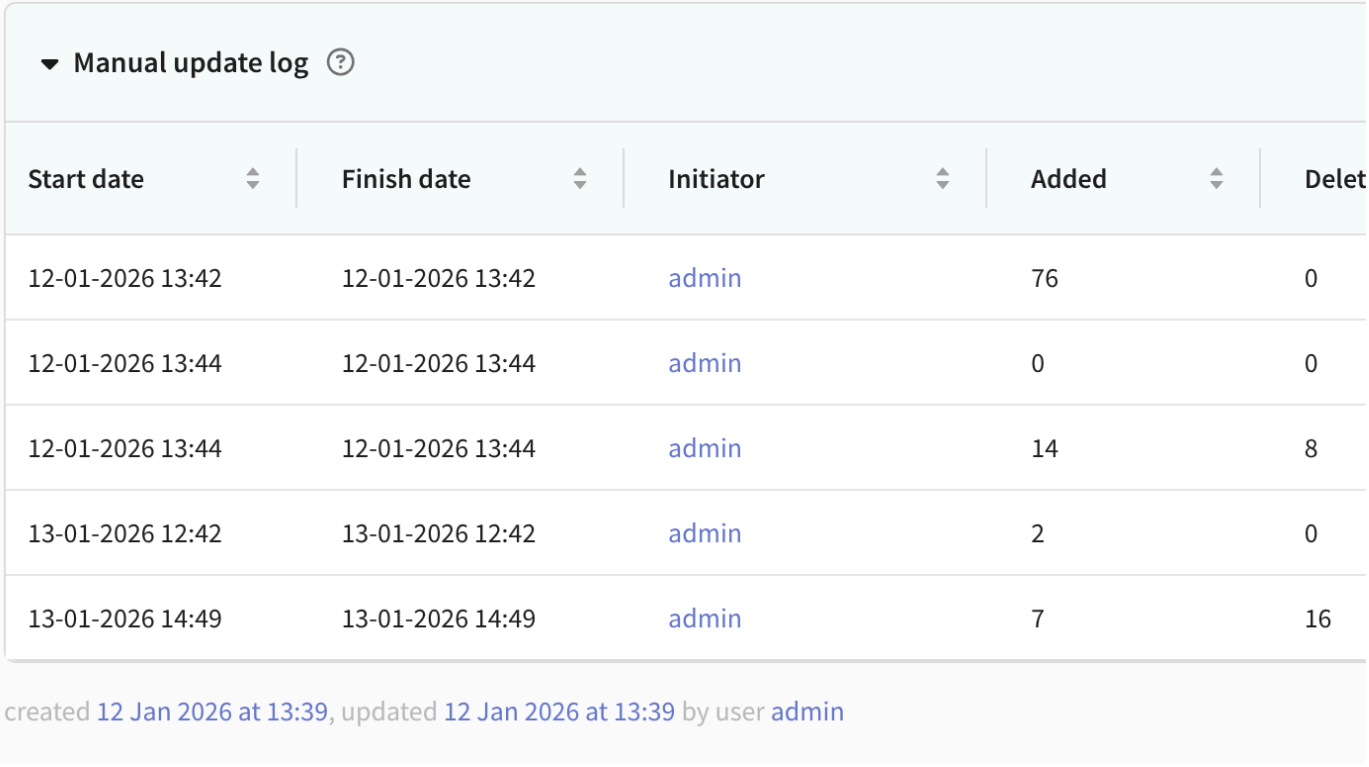
  img

  : add or remove table columns by dragging a Field from the Add field sidebar to Displayed fields or ba

Below is the Update history block. It shows changes in the total number of products in the segment for t date range and group data by days or weeks.

Below the update history is the Manual changes log. It contains a detailed record of all operations that a each operation, the following is shown:

- **Start date: time when the segmentation process started (adding/removing by conditions).**
- **End date: time when the process finished.**
- **Initiator: name of the user who initiated the operation.**
- **Added: number of products added as a result of the operation.**
- **Removed: number of products removed as a result of the operation.**
- **Status: current operation state (for example, “Completed”, “In progress”, “Error”).**



TIP: Segment actions

At the top of the screen, a quick actions menu is available, where you can assign tags, configure acces another segment.

Conditions

Conditions for selecting products and SKUs are divided into three groups: Main Fields, System Fields, an of operators depends on the type of selected field.

Main Fields

Main fields include standard product or SKU fields: name, category, price, quantity, and others.

Product Name

A string field containing the product name.

- **is equal to / is not equal to**
- **contains / does not contain**
- **begins with**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Region

A field for specifying the sales region or product availability.

- **exists / does not exist**
- **is equal to / is not equal to**
- **is in list / is not in list**

Category

A field containing one or more categories to which the product belongs.

- **exists / does not exist**
- **contains at least one value from the list**
- **does not contain any values from the list**

- **contains all values from the list**
- **does not contain all values from the list at the same time**

Product Number

An integer reflecting the remaining stock or available quantity.

- **is empty / is not empty**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **in the range**

Expiration Date

A *date* type field indicating the product's expiration date.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **exists / does not exist**
- **in the range**

Price

A numeric field with the current product price.

- **exists / does not exist**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **is in list / is not in list**
- **in the range**

Old Price

A numeric field with the price before discount (if applicable).

- **exists / does not exist**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **is in list / is not in list**
- **in the range**

Delivery

A boolean field (true/false) indicating whether delivery is required for the product or available.

- **true**
- **false**

Guarantee

A boolean field (true/false) indicating whether the product has a warranty.

- **true**
- **false**

Accessibility

A switch field reflecting the product's availability status.

- **available**
- **not available**

Manufacturer

A string field with the manufacturer's company name.

- **exists / does not exist**
- **is equal to / is not equal to**
- **contains / does not contain**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Tags

A list field used for flexible product tagging.

- **exists / does not exist**
- **contains at least one value from the list**
- **does not contain any values from the list**

- **contains all values from the list**
- **does not contain all values from the list at the same time**

System Fields

Fields automatically filled by the platform or used for internal identification.

Identifier

A unique numeric product ID in the system.

- **is equal to / is not equal to**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Additional Identifier

A string field for storing an external ID (e.g., SKU from 1C or product code).

- **is equal to / is not equal to**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Product/SKU Creation Date

A *date* type field recording the moment the record was created in the platform.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **month is / month is not**
- **month is the same as today**
- **month day is / is not / is greater than / is less than / is the same as current**
- **day and month are / are not / are greater than / are less than / are the same as current**
- **is in list / is not in list**
- **is in a date range**
- **date relative to current date — selection based on a moment in the past or future (e.g., “created in th**

Product/SKU Update Date

A *date* type field recording the moment the record was last modified.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **month is / month is not**
- **month is the same as today**
- **month day is / is not / is greater than / is less than / is the same as current**
- **day and month are / are not / are greater than / are less than / are the same as current**
- **is in list / is not in list**
- **is in a date range**
- **date relative to current date**

Custom Fields

You can create your own fields in the catalog structure to store specific product information. For segmen the created field type (string, numeric, date, boolean, etc.) are available.

Preparing the YML file

Products in the Market are represented by two entities: products and SKU.

A product is a good or service for sale. SKU is a product modifier, i.e. a group of goods that stands out in parameters (color, size, weight, model, equipment, etc.).

To upload product data to the platform, a file in yml format or a .zip/.tar.gz archive containing yml files is a schedule, you must provide a link to the yml file.

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. link.

The structure of such files looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
  <shop>
    ...
    <offers>
      ...
    </offers>
    <gifts>
      ...
    </gifts>
    <promos>
      ...
    </promos>
  </shop>
</yml_catalog>
```

If you want to upload product data to the platform, use yml files or .zip/.tar.gz archives containing yml fil

A list of all products and SKUs to import is passed in the `offers` element. More information you will find I

Each product and SKU is described in a separate `offer` element. In this element you need to specify an item:

```
<offer id="9012">
```

Importing a SKU is similar to importing a product. The only difference is the structure of a yml file.

When importing a SKU, you'll have to specify not only `id` , but also `group_id` (the product identifier) for th SKU is always tied to a specific product.

```
<offer id="1234" group_id="9012">
```

> **TIP**
>
> **Please note that SKUs are different variations of the same product, so the value of the `group_id` attribut the same product.**

Examples

Product

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2020-11-22T14:37:38+03:00">
    <shop>
        <name>BestSeller</name>
        <company>Tne Best inc.</company>
        <url>http://best.seller.ru</url>
        <currencies>
        </currencies>
        <categories>
            <category id="1">Appliances</category>
            <category id="10" parentId="1">Small kitchen appliances</category>
        </categories>
        <delivery-options>
        </delivery-options>
        <offers>
            <offer id="9012">
                <name>Cream freezer Brand 3811</name>
                <url>http://best.seller.ru/product_page.asp?pid=12345</url>
                <price>200</price>
                <currencyId>USD</currencyId>
                <categoryId>10</categoryId>
                <delivery>true</delivery>
                <delivery-options>
                </delivery-options>
                <param name="Color">white</param>
                <weight>3.6</weight>
                <dimensions>20.1/20.551/22.5</dimensions>
            </offer>
        </offers>
        <gifts>
            <!-- gifts not from the price list -->
        </gifts>
        <promos>
            <!-- promotions -->
        </promos>
    </shop>
</yml_catalog>
```

SKU

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2020-11-22T14:37:38+03:00">
    <shop>
        <name>BestSeller</name>
        <company>Tne Best inc.</company>
        <url>http://best.seller.ru</url>
        <currencies>
        </currencies>
        <categories>
            <category id="1">Appliances</category>
            <category id="10" parentId="1">Small kitchen appliances</category>
        </categories>
        <delivery-options>
        </delivery-options>
        <offers>
            <offer id="1234" group_id="9012">
                <name>Cream freezer Brand 3811 v.2</name>
                <url>http://best.seller.ru/product_page.asp?pid=12345</url>
                <price>200</price>
                <currencyId>USD</currencyId>
                <categoryId>10</categoryId>
                <delivery>true</delivery>
                <delivery-options>
                </delivery-options>
                <param name="Color">white</param>
                <weight>3.6</weight>
                <dimensions>20.1/20.551/22.5</dimensions>
            </offer>
        </offers>
        <gifts>
            <!-- gifts not from the price list -->
        </gifts>
        <promos>
            <!-- promotions -->
        </promos>
    </shop>
</yml_catalog>
```

Loyalty programs

Description

Loyalty programs can be used in marketing to increase customer involvement in buying process and to k purchase.

In this section

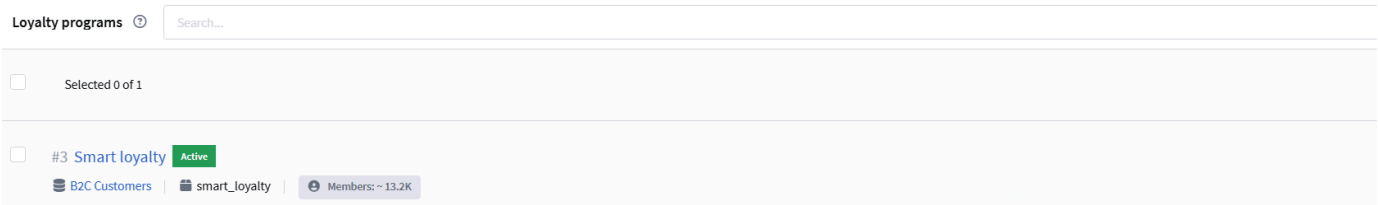
Loyalty programs

A loyalty program is a customer reward system for target actions such as purchases, registration, conte earn points that can be spent on discounts, movies, plans, and services. The program includes tiers (bro points accrual and expiration, as well as integrations with other services and communication channels.

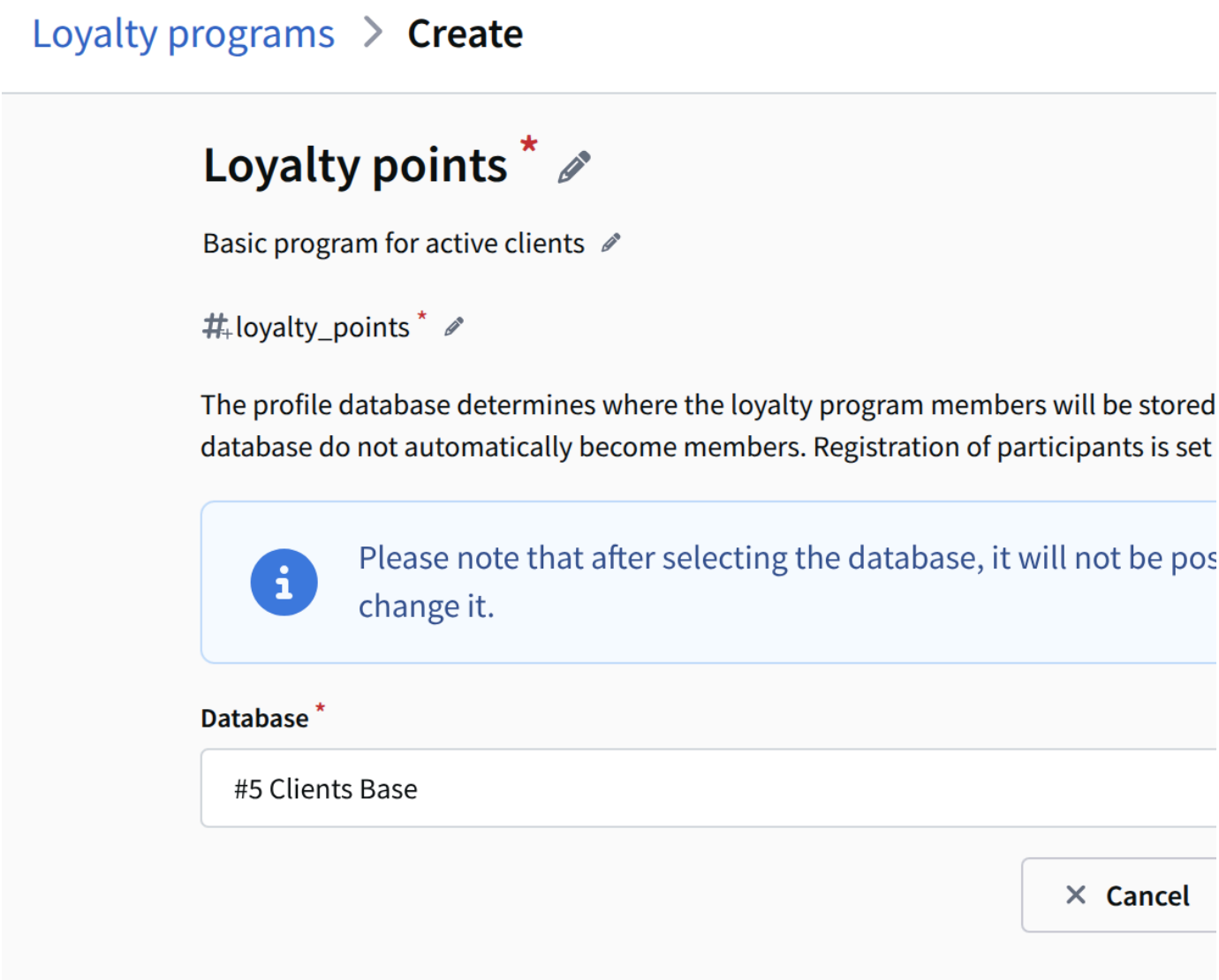
Creating a loyalty program

![img](/img/attachments/loyalty-programs/en/choose_loyalty.png)	To manage loyalty programs, go to **Loyalty — Loyalty programs** in the main program, click its title.

To start a new loyalty program, click + Create at the top of the page:



The loyalty program settings window will open. Specify the name of your program and its external identi

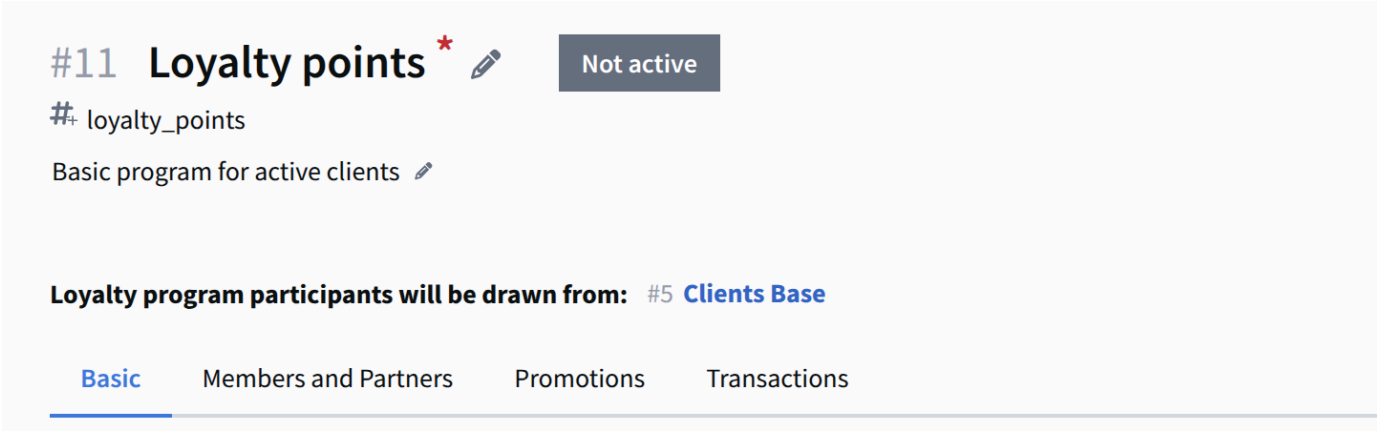


Select the database from which participants will be taken.

> **CAUTION**
>
> **Please note: after selection, the database cannot be changed; only one loyalty program can be active ir**

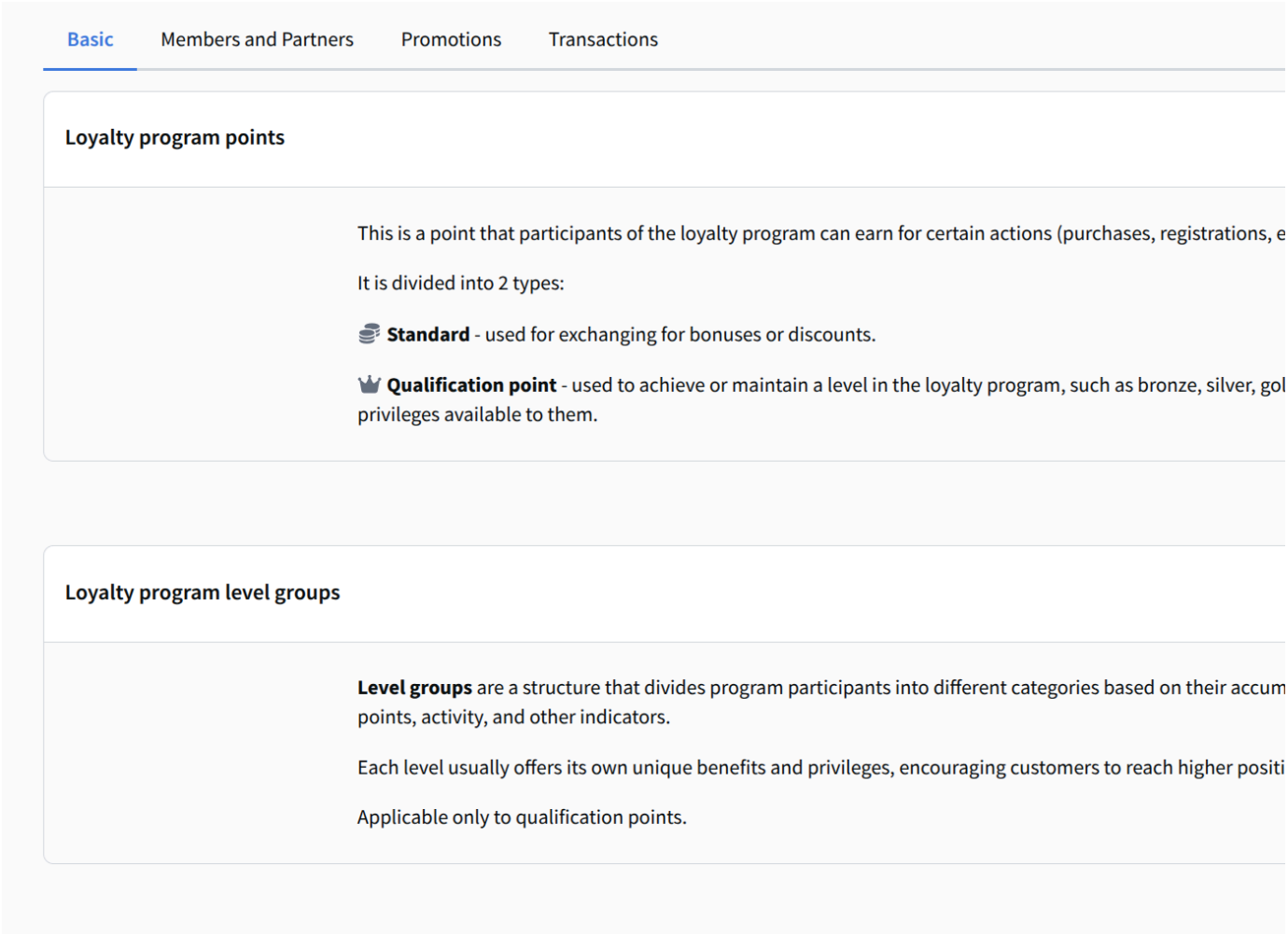
Editing a loyalty program

After creation, you will be taken to the loyalty program editor, which consists of several tabs: General, M and Transactions:



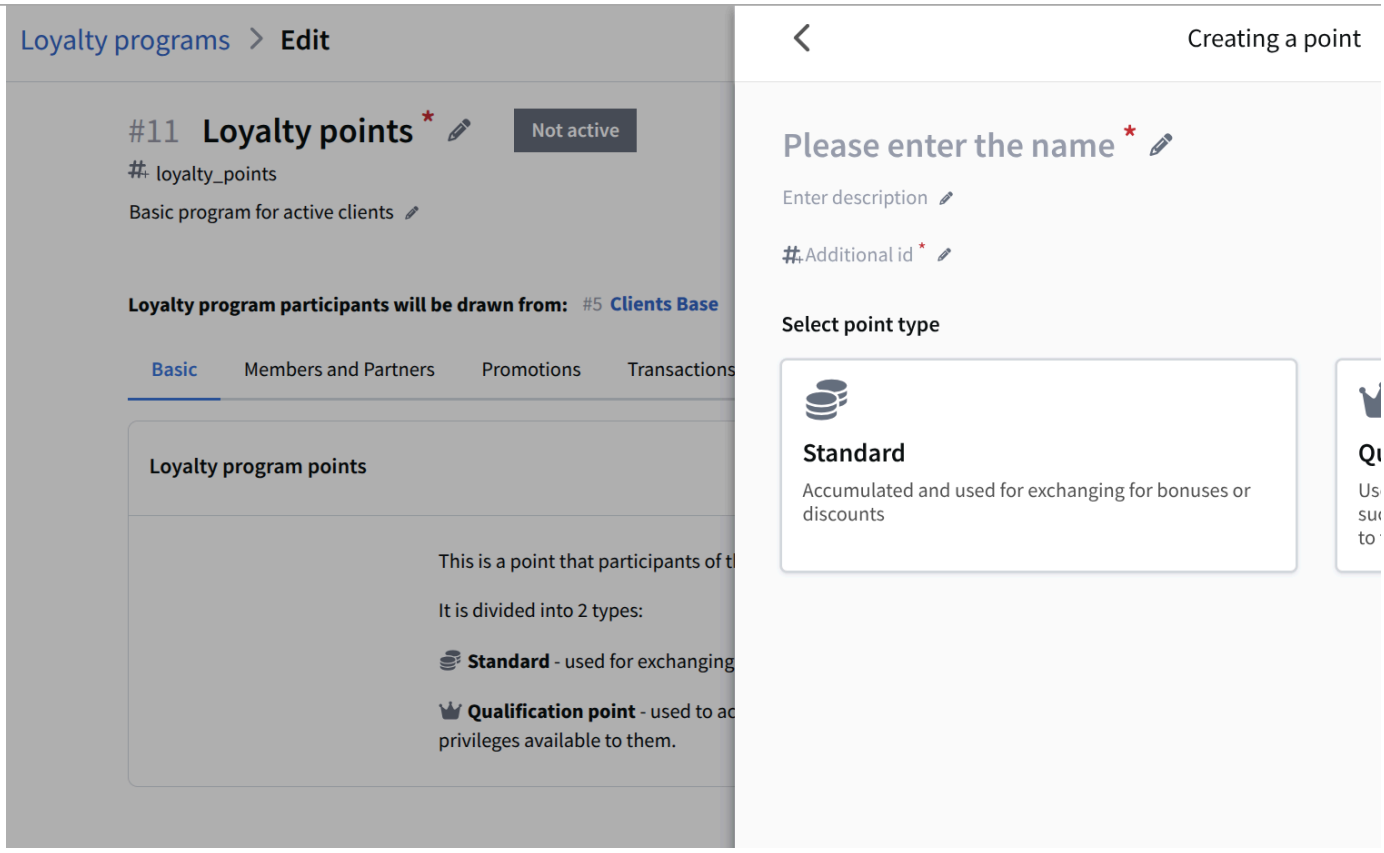
General

The General section is used to configure the basic structure of the loyalty program. It defines the key pa are accrued, used, and tracked, as well as how customers progress through loyalty tiers.



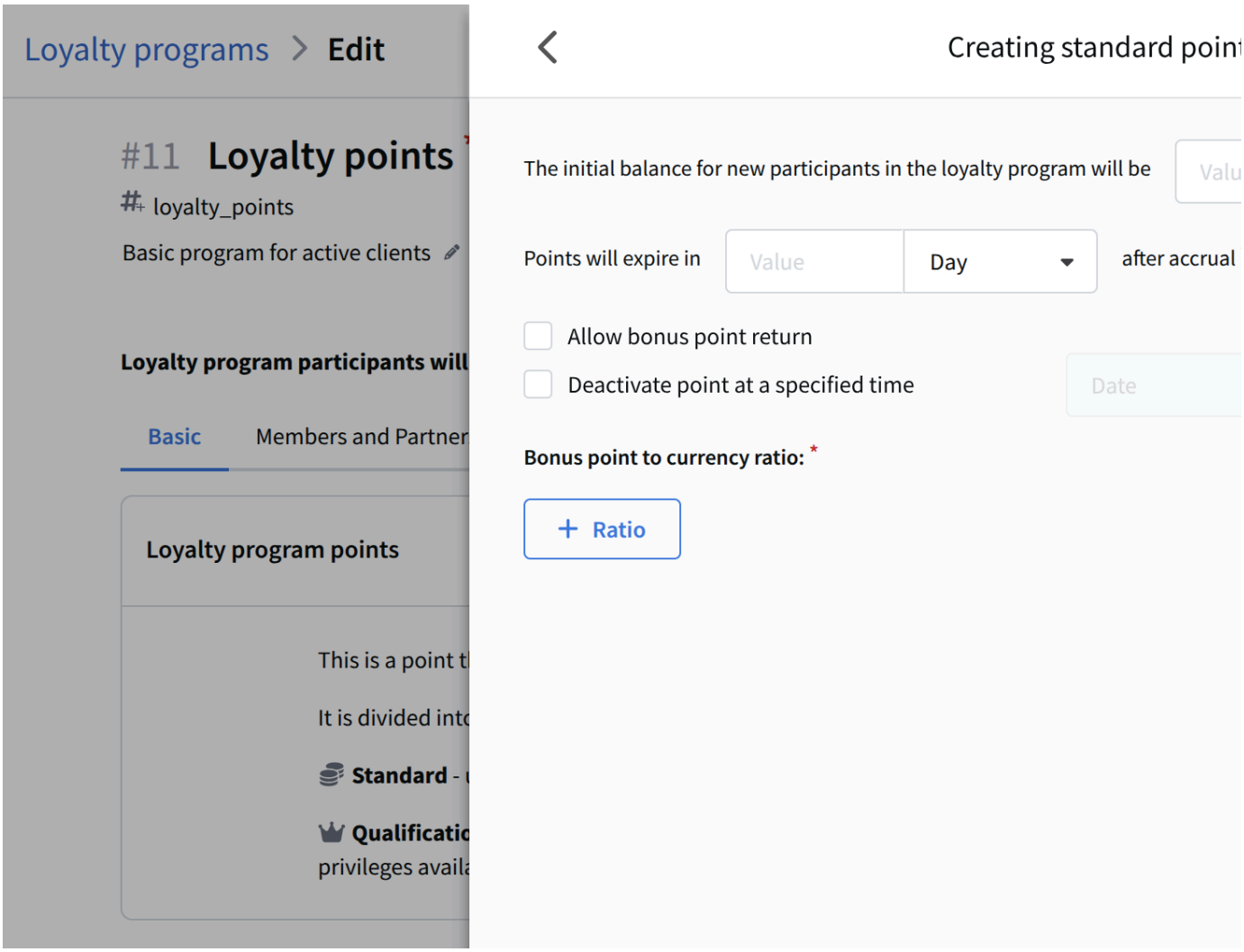
Loyalty program points

Loyalty program points — this block is used to create and manage points types within the loyalty progra external identifier, and optionally add a description.



There are two types of points:

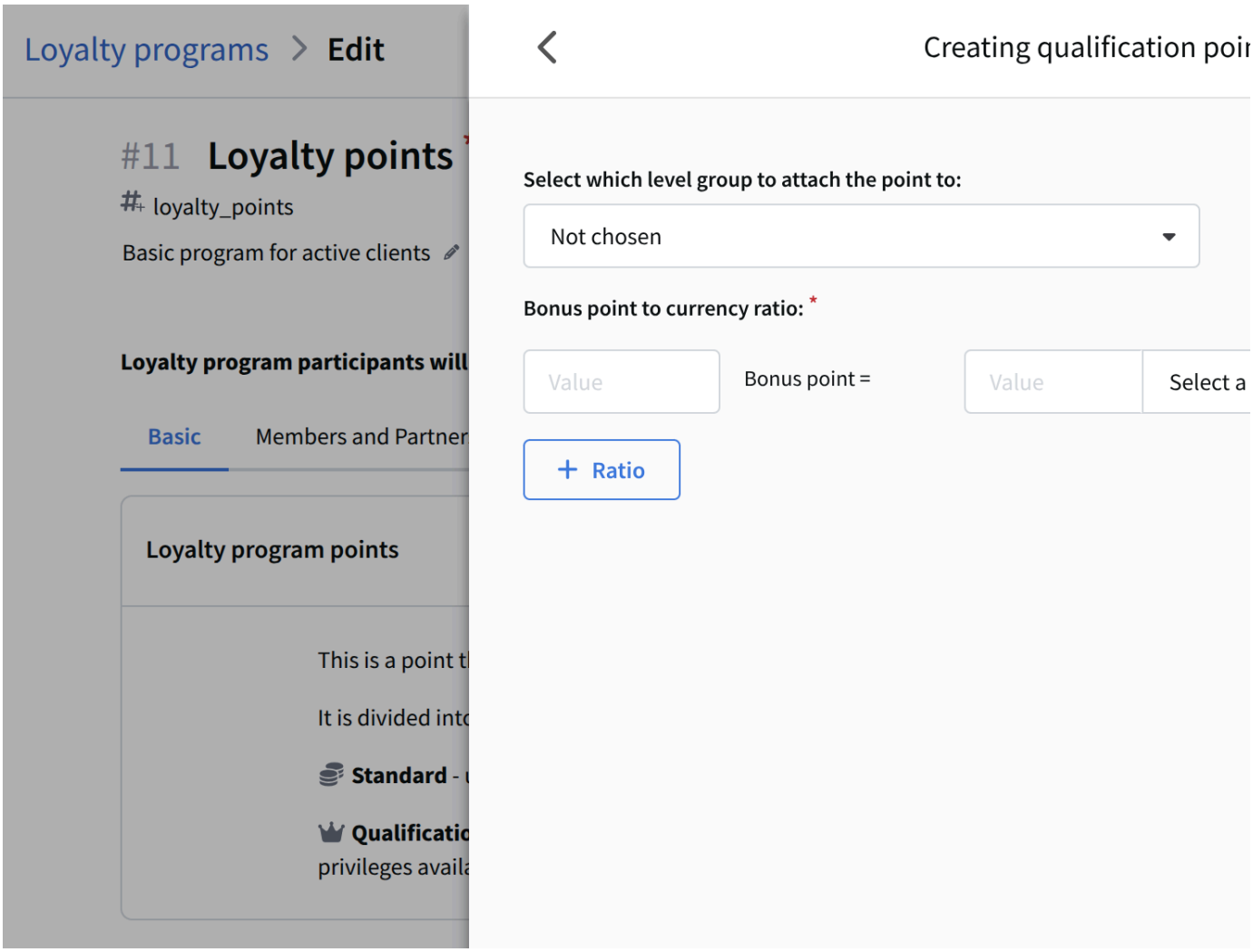
Standard points — points credited to the bonus account that the customer can later use to fully or partia



When creating standard points, you can configure the following parameters:

- **Initial balance for a new loyalty program member — the number of bonus points automatically grante program. The value is specified in points.**
- **Points validity period — defines how long accrued points remain valid. The period can be set in days, expiration, points are canceled.**
- **Allow bonus points refund — enables returning bonus points to a member, for example, in case of a p cancellation.**
- **Deactivate points at a specific time — sets a date after which the points become inactive (not accru seasonal programs.**
- **Bonus-to-currency ratio — a required parameter defining how many bonus points correspond to a mo 1 USD or 10 points = 1 USD. Used for discount and redemption calculations.**

Qualification points — used to evaluate customer activity and determine their tier in the loyalty program



When creating qualification points, you can configure:

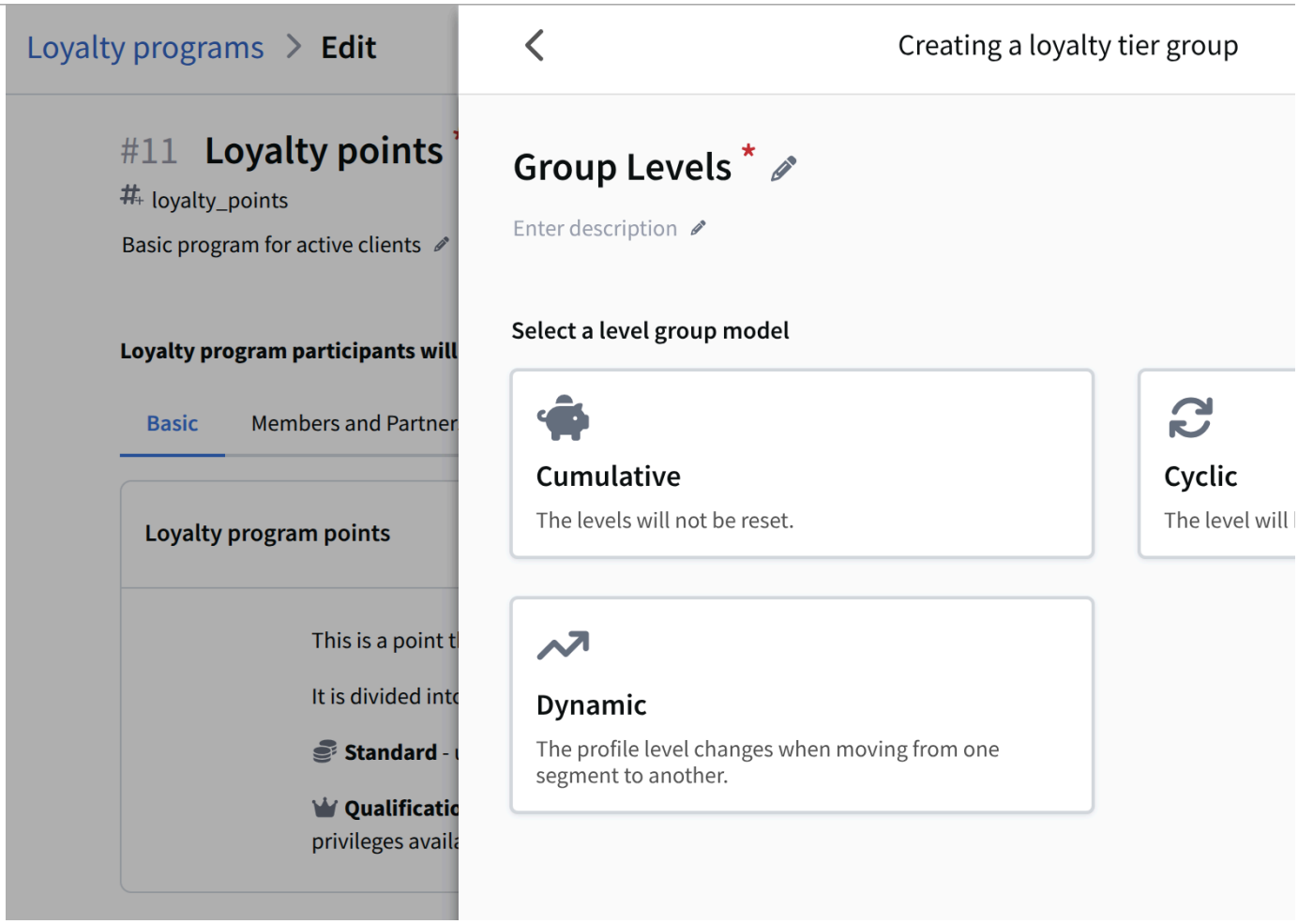
- **Allow bonus points refund — enables returning bonus points to a member.**
- **Tier group binding — links the points to a specific loyalty tier group (for example, Bronze, Silver, Gold different accrual rules or rates depending on the customer's tier.**
- **Bonus-to-currency ratio — defines the conversion rate between points and monetary value.**

Do not forget to activate the created points. If needed, points can be cloned to create another type base

Loyalty tier groups

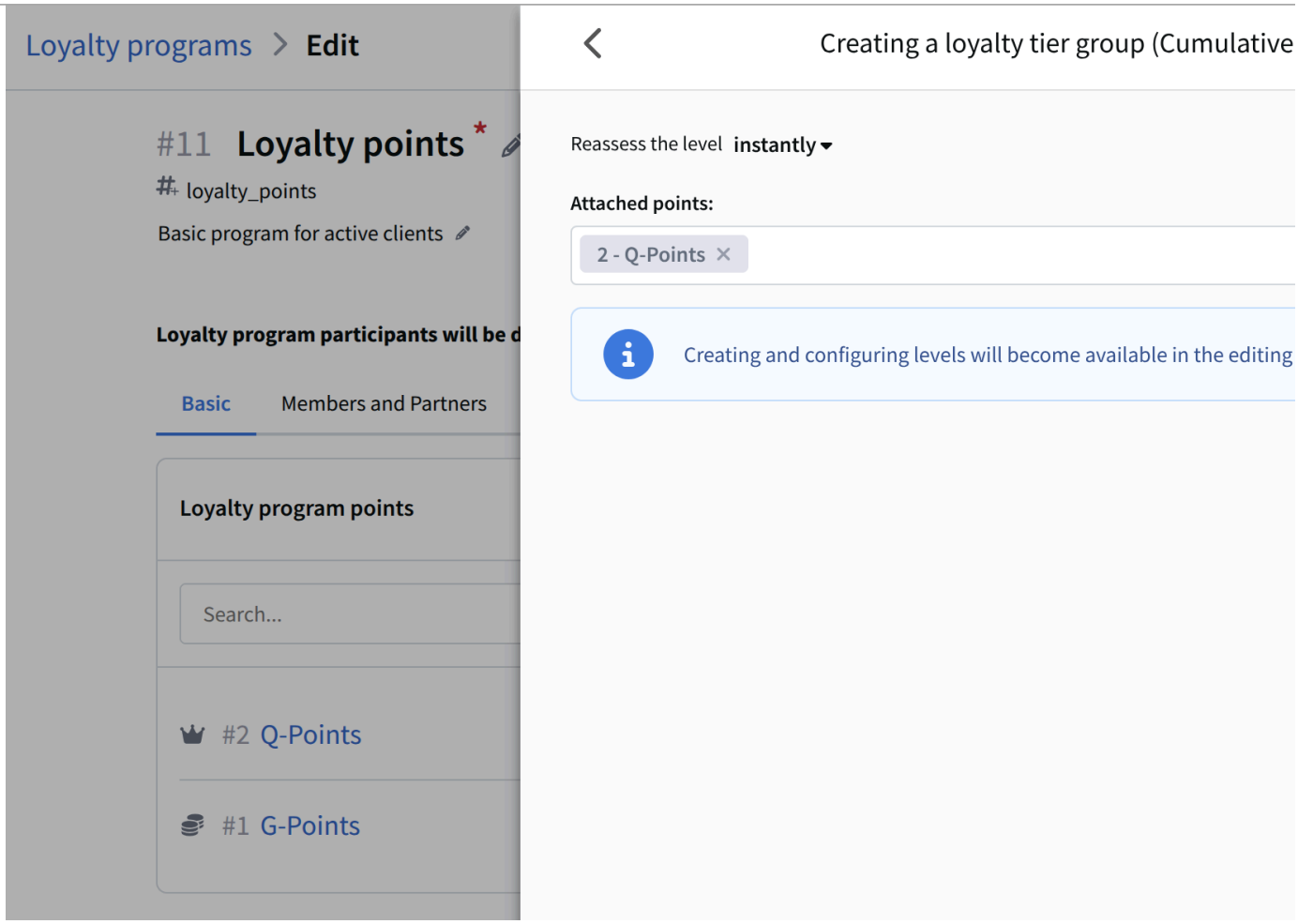
Loyalty tier groups allow you to segment loyalty program members into different levels or statuses based other criteria.

Specify the name and external identifier of the tier group, and optionally add a description:



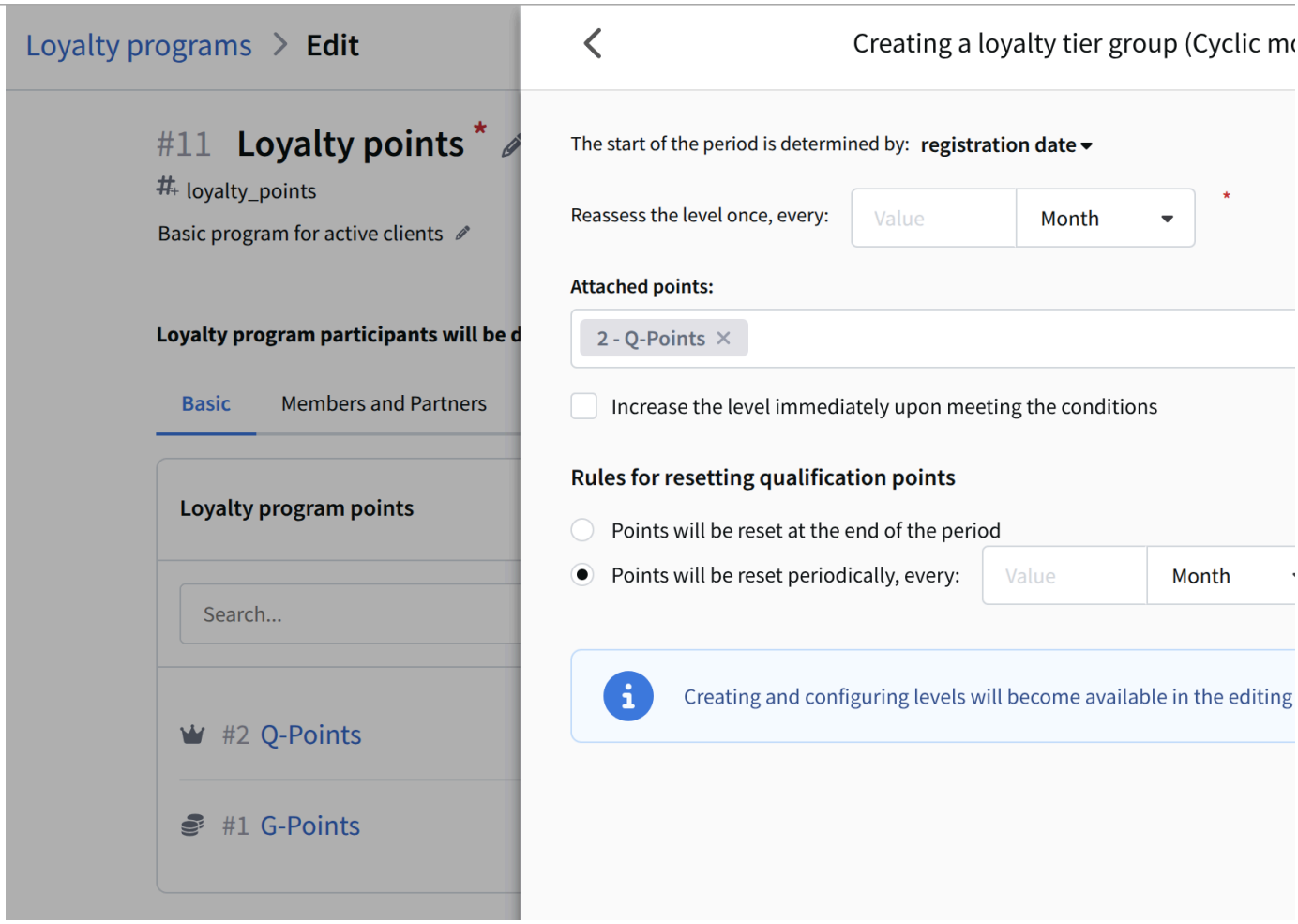
Several tier group models are available:

- **Cumulative model — once a member reaches a tier, it is retained permanently. After a user accumula necessary conditions, they receive a status that stays with them and does not reset over time. This i long-term motivation and rewarding loyalty.**



Specify the points type bound to this model.

- **Cyclic model — the achieved tier is valid only for a defined period. After the period ends, the system they have not confirmed their activity or met the required conditions, the tier may be lowered.**



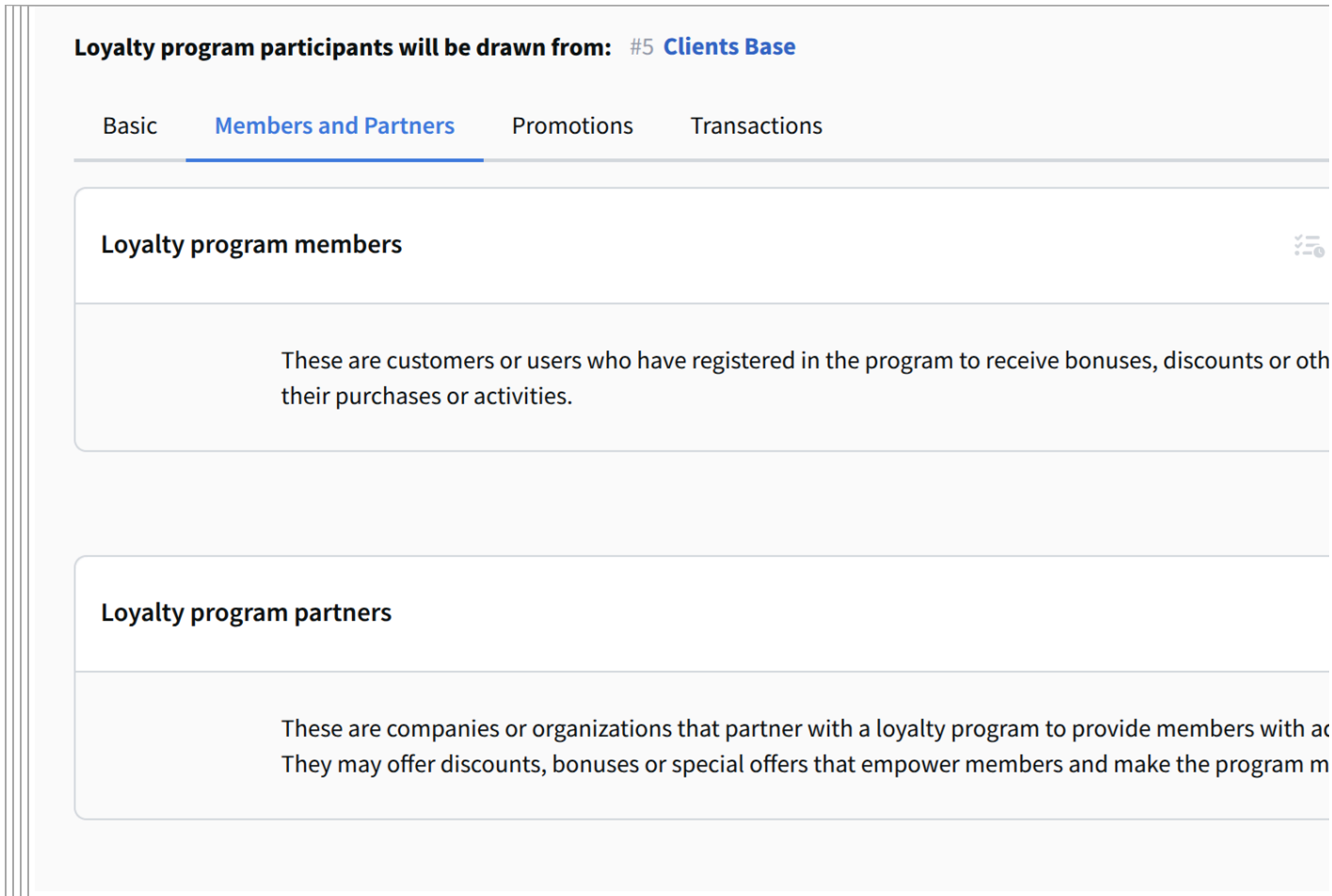
Configure the model:

- **Set the interval for tier reevaluation;**
- **Specify the points type bound to the model;**
- **Enable Upgrade tier immediately when conditions are met to apply tier changes instantly;**
- **Configure qualification points reset rules, you can choose reset at the end of each period or at fixed months);**
- **Optionally enable Extend tier until the end of the month — the tier is retained until the end of the cu member did not meet the conditions to maintain it.**
- **Dynamic model — the member's tier directly depends on their current characteristics or activity and conditions change.**

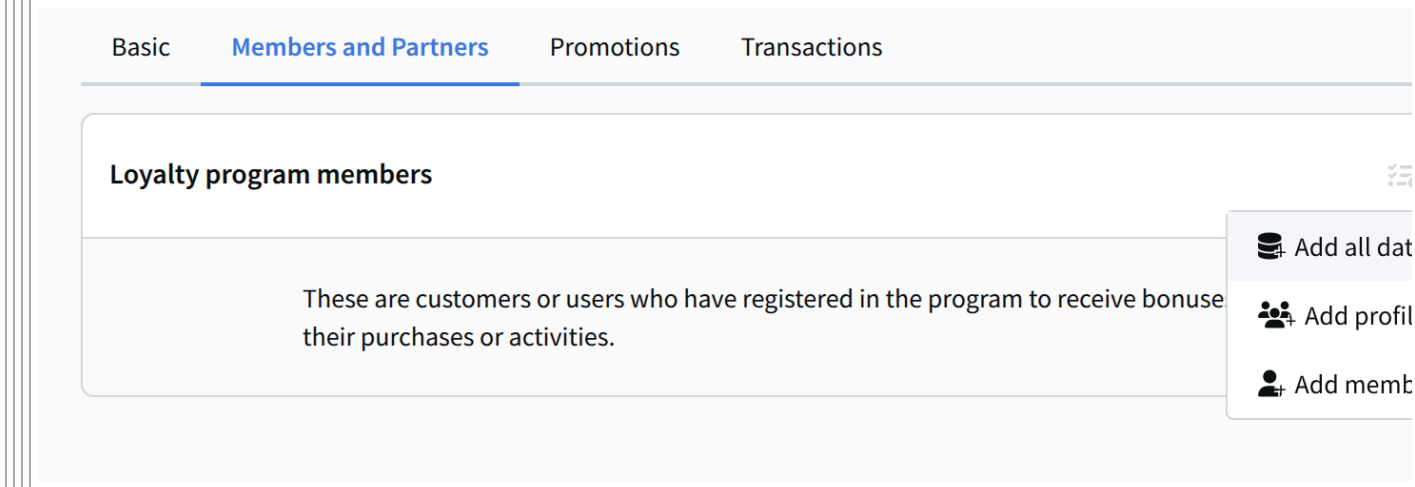
After completing the basic settings, go to Members and partners.

Members and partners

The Members and partners section is used to manage loyalty program members (end users) and partner benefits).



This section includes the following elements:
Loyalty program members — customers registered in the loyalty system. They earn points for specific ac
content viewing, data confirmation) and can redeem points for discounts, content, or additional options.
To add members, click + Add. Three options are available:



1. Add all database profiles — mass registration of all customers in the database. 2. Add profiles from a
group of users, for example, active customers, those who recently paid for a service, etc. 3. Add a memb
customer.

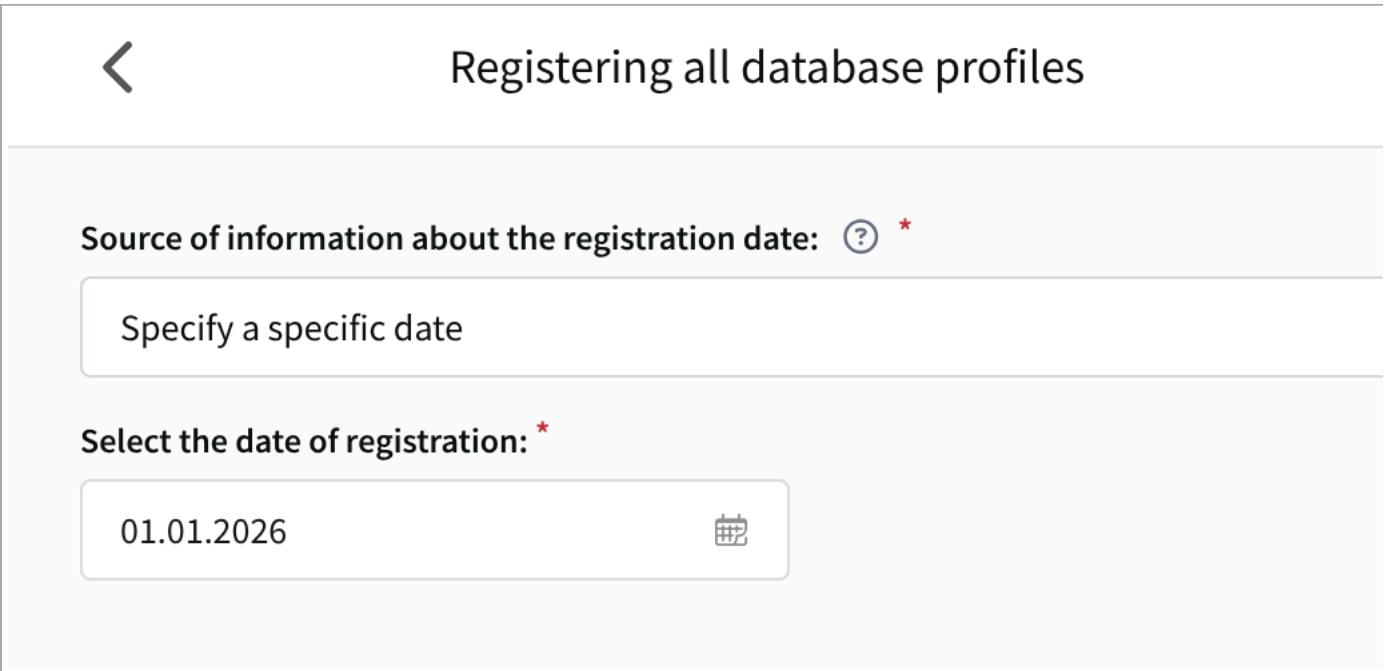
INFO

Additionally, the following methods for adding members are available:

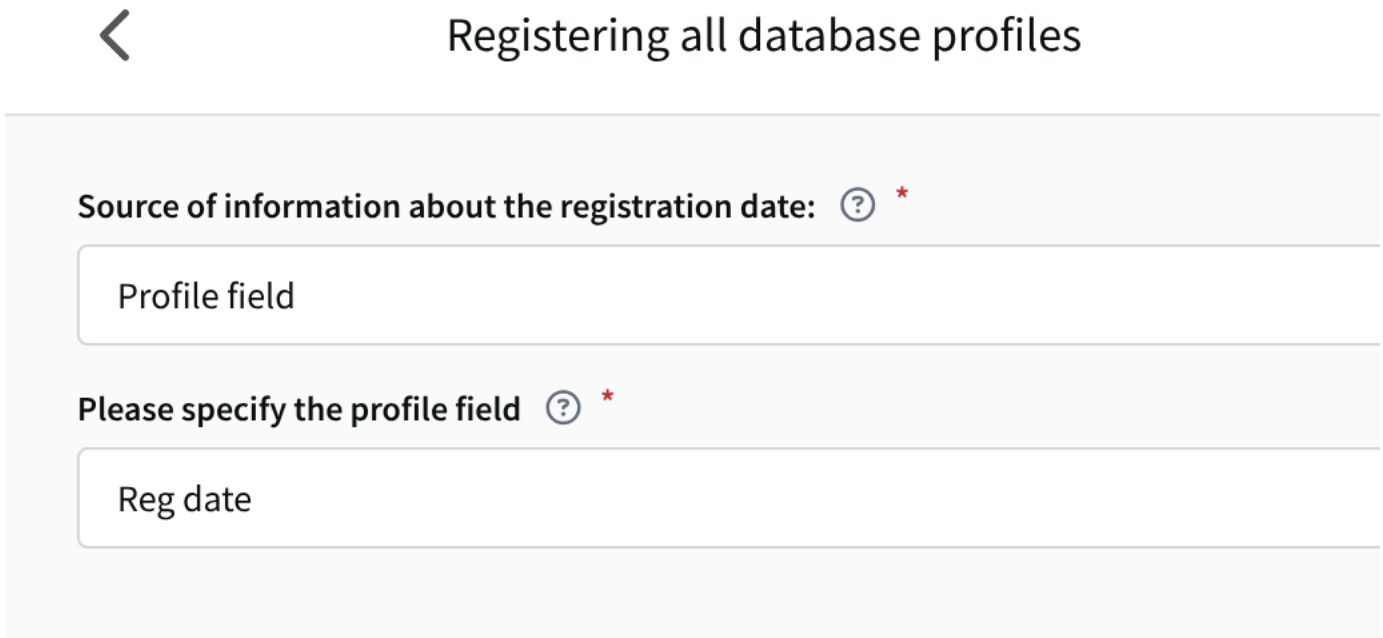
- Automation via the "Loyalty Participation" node — automatically adding customers within scenarios w
for example, after submitting a form.

- Integration via API — adding members programmatically using an API call. Suitable for connecting ext

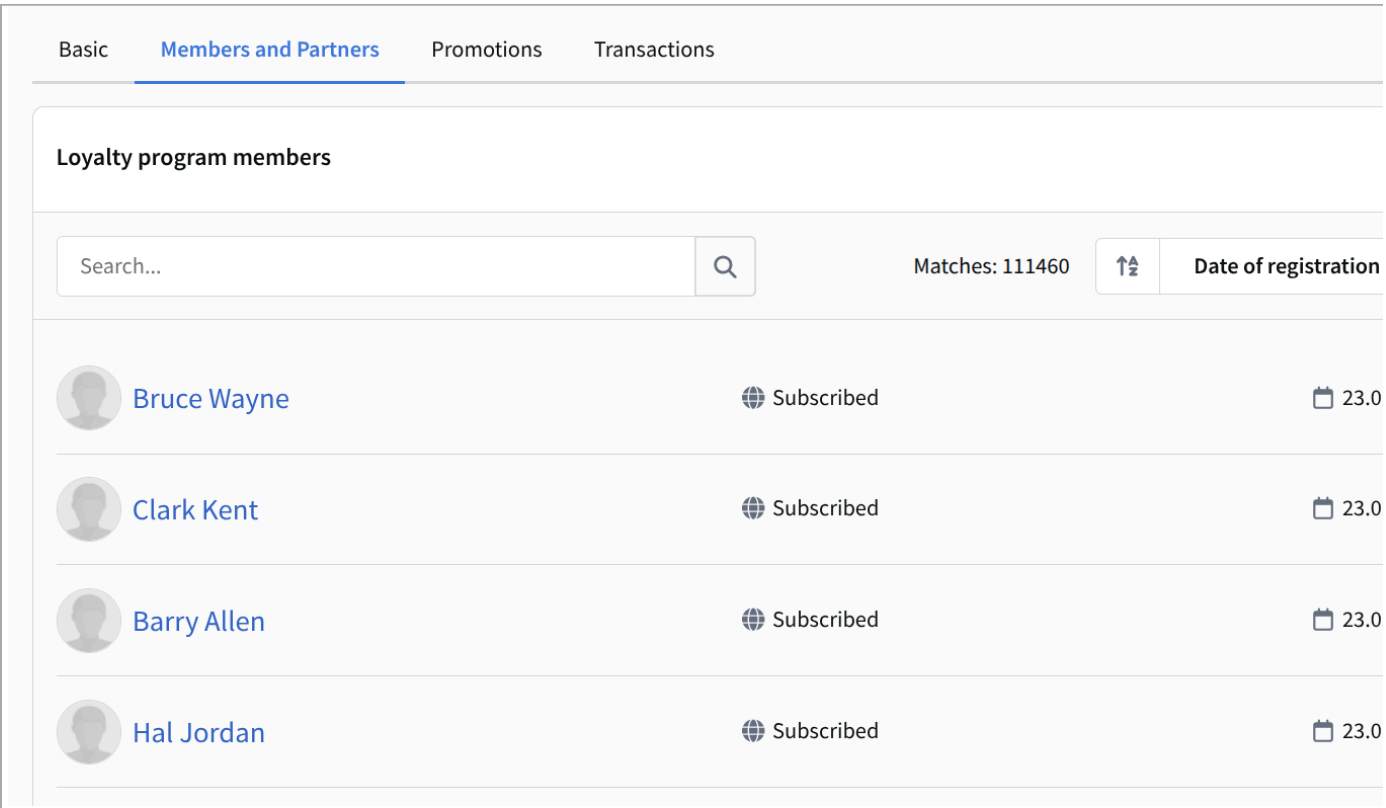
Specify the registration date source.
It can be a specific date:



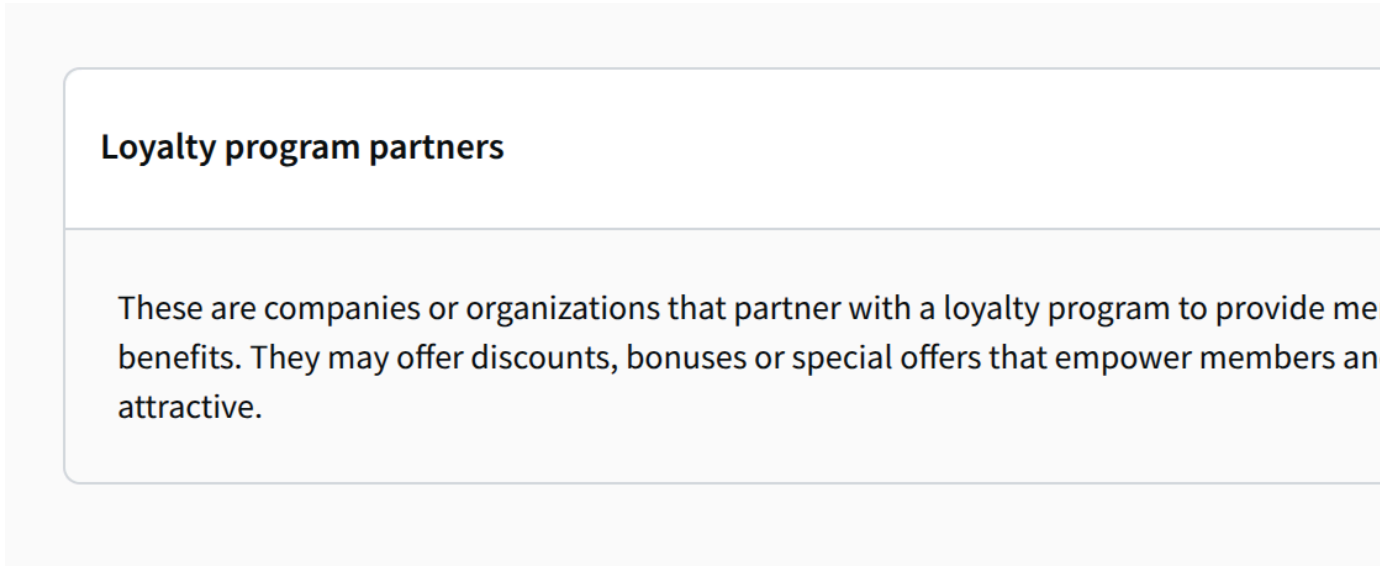
Or a profile field:



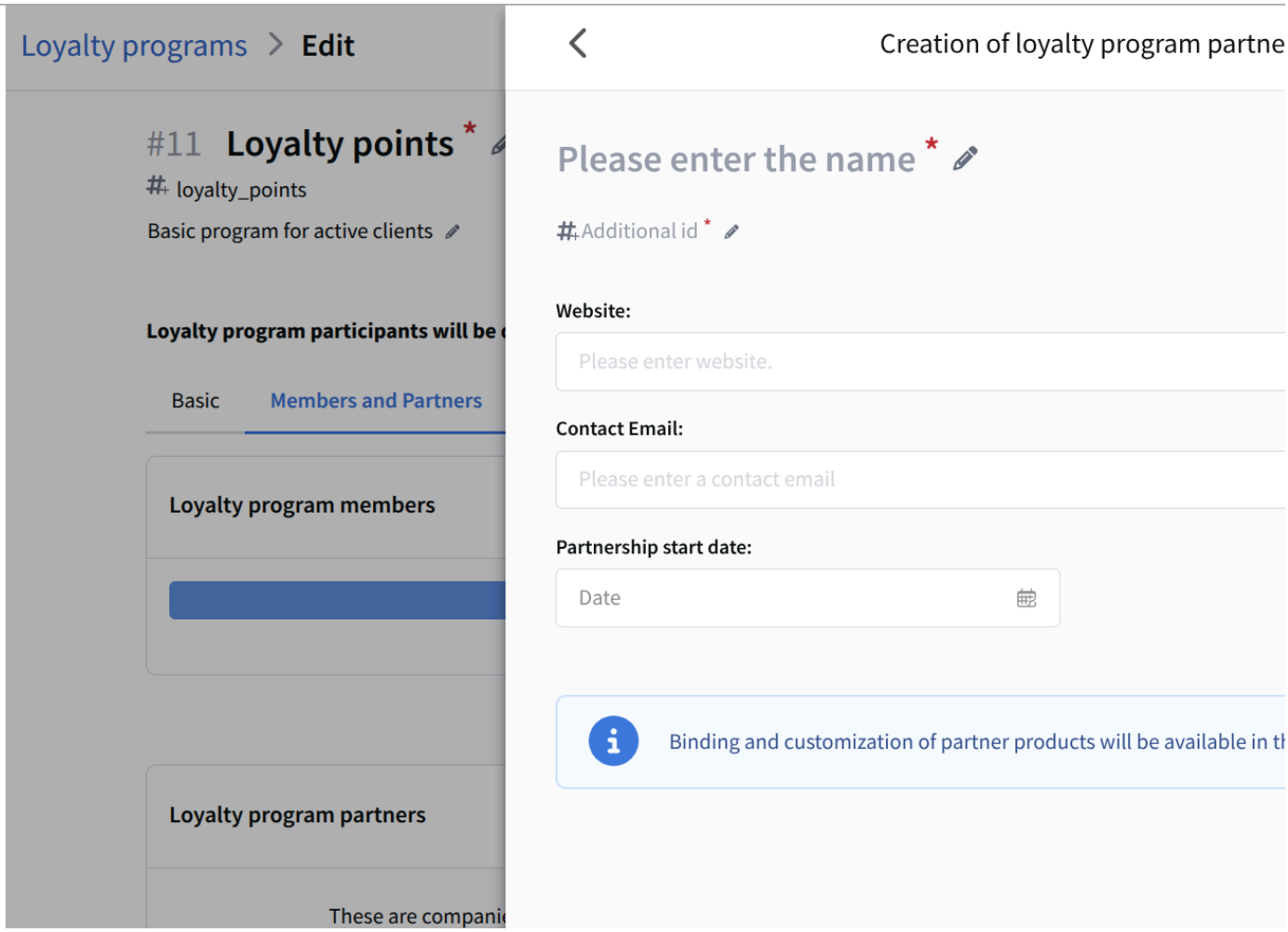
After members are added, you can remove individual members or all members using Clear members list, corresponding button.



Loyalty program partners — this section manages organizations cooperating with the loyalty program to provide participants with additional privileges — from discounts to exclusive offers.



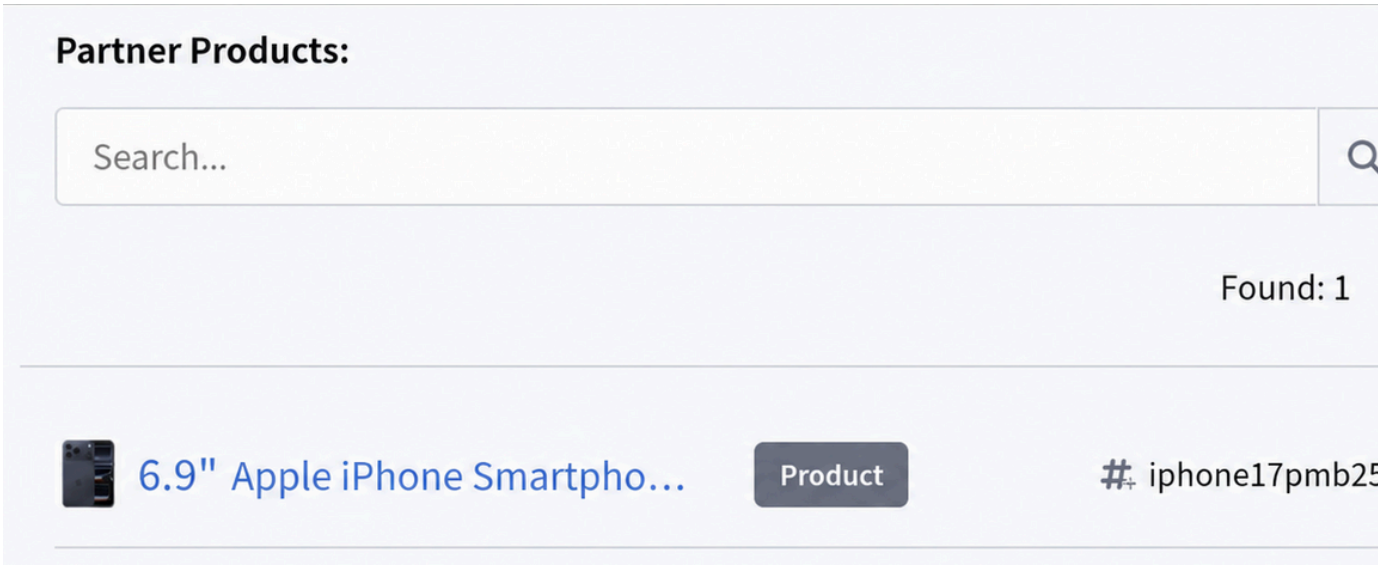
Click + Create to add a new partner. The partner settings window will open:



Specify the partner name and external identifier. Optionally, add a link to the partner's official website fo and set the partnership start date.

After creating and saving the partner, you can link Partner products from the Market to which partner te

For each product, a discount percentage is configured, which will be provided to loyalty program particip products. These settings are used in promotions with the "Discount from a partner" benefit and are take order.



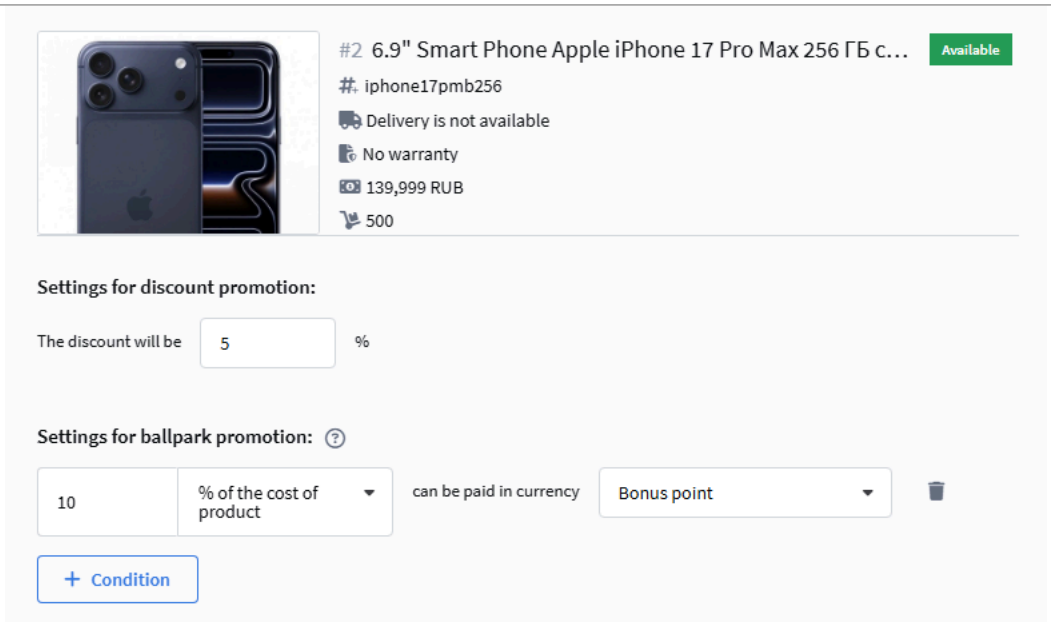
When adding partner products, you can also set application conditions depending on the type of promoti

Settings for discount promotions:

- **Discount will be % — discount type;**

Settings for points promotions:

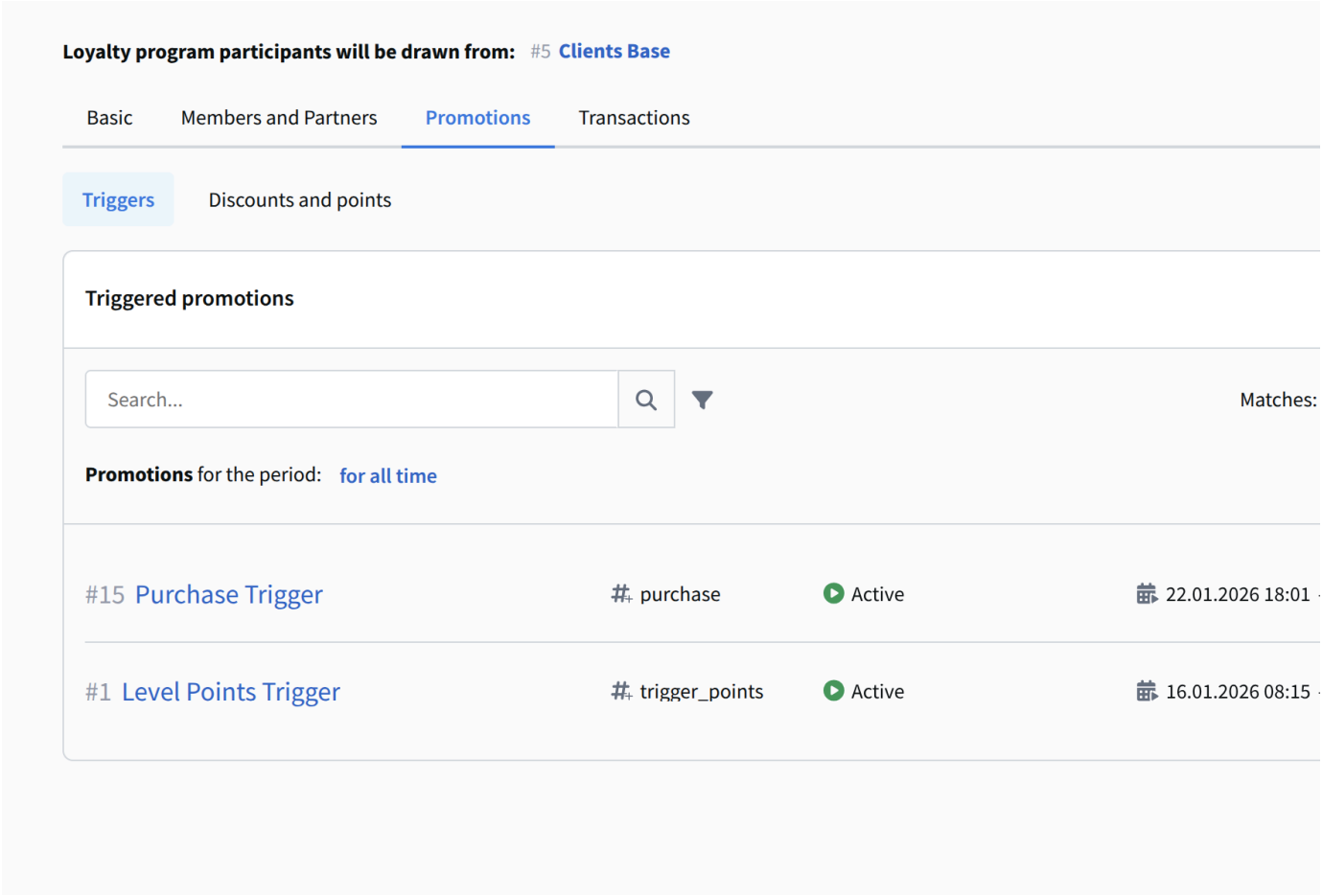
- **Value — the number of points or the percentage of the product price to be used for calculation;**
- **Points / % of product price — choice of calculation mechanics: points accrual/deduction or applying**
- **Can be paid with currency — determines how many points can be used as a monetary equivalent:**



Partner products are used to form special offers within the loyalty program. Such offers are available onl partnership terms apply and are configured through corresponding promotions with the “Discount from a

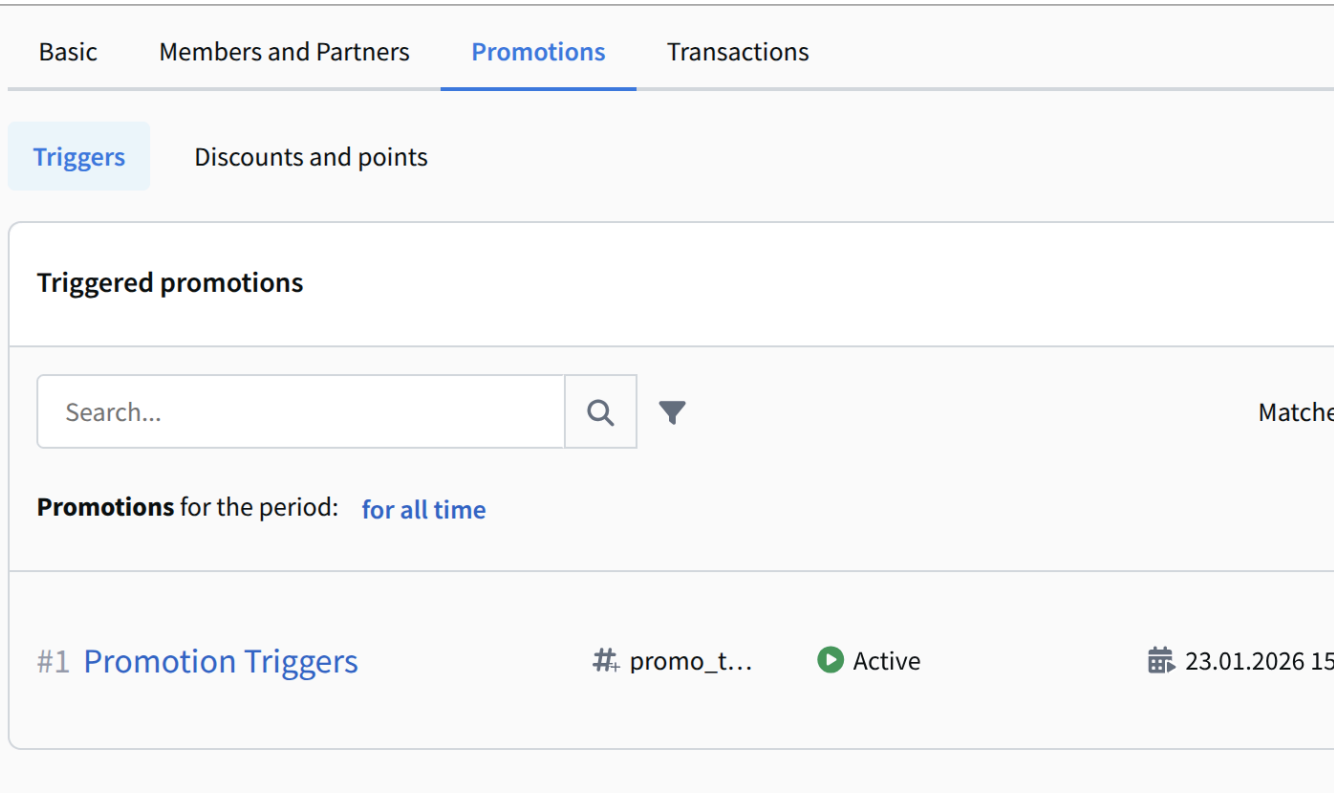
Promotions

The Promotions section is used to create and manage loyalty program promotions that automatically acc discounts, and grant other benefits to orders and products in the Market.



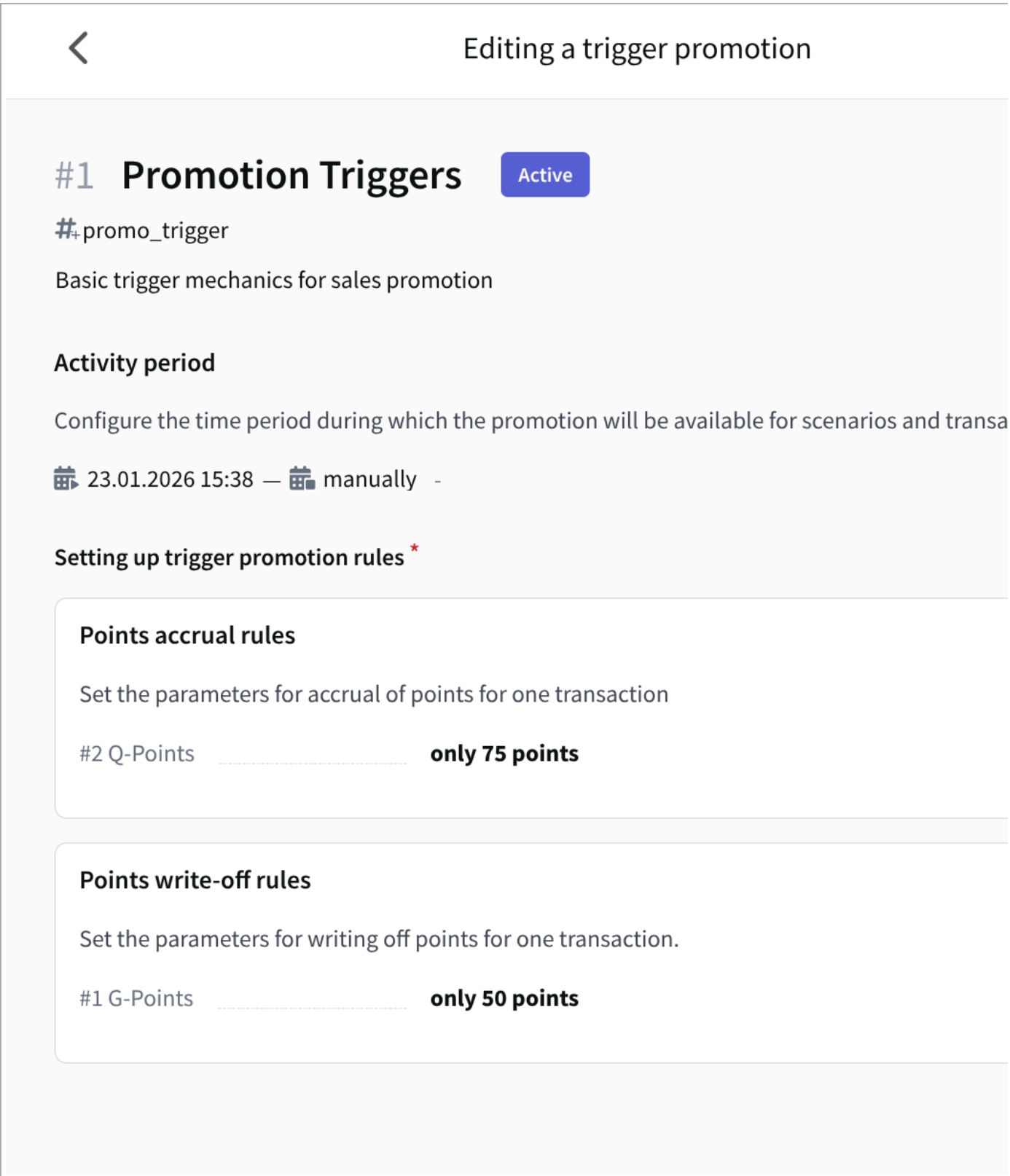
Promotions are divided into two groups: trigger and discount and points promotions.

Trigger promotions



Trigger promotions are marketing activities in which, when rules are met (triggers are activated), a parti bonus points. In other words, this is a set of rules according to which a customer can earn (and later los are linked to triggers when creating campaigns and scenarios.

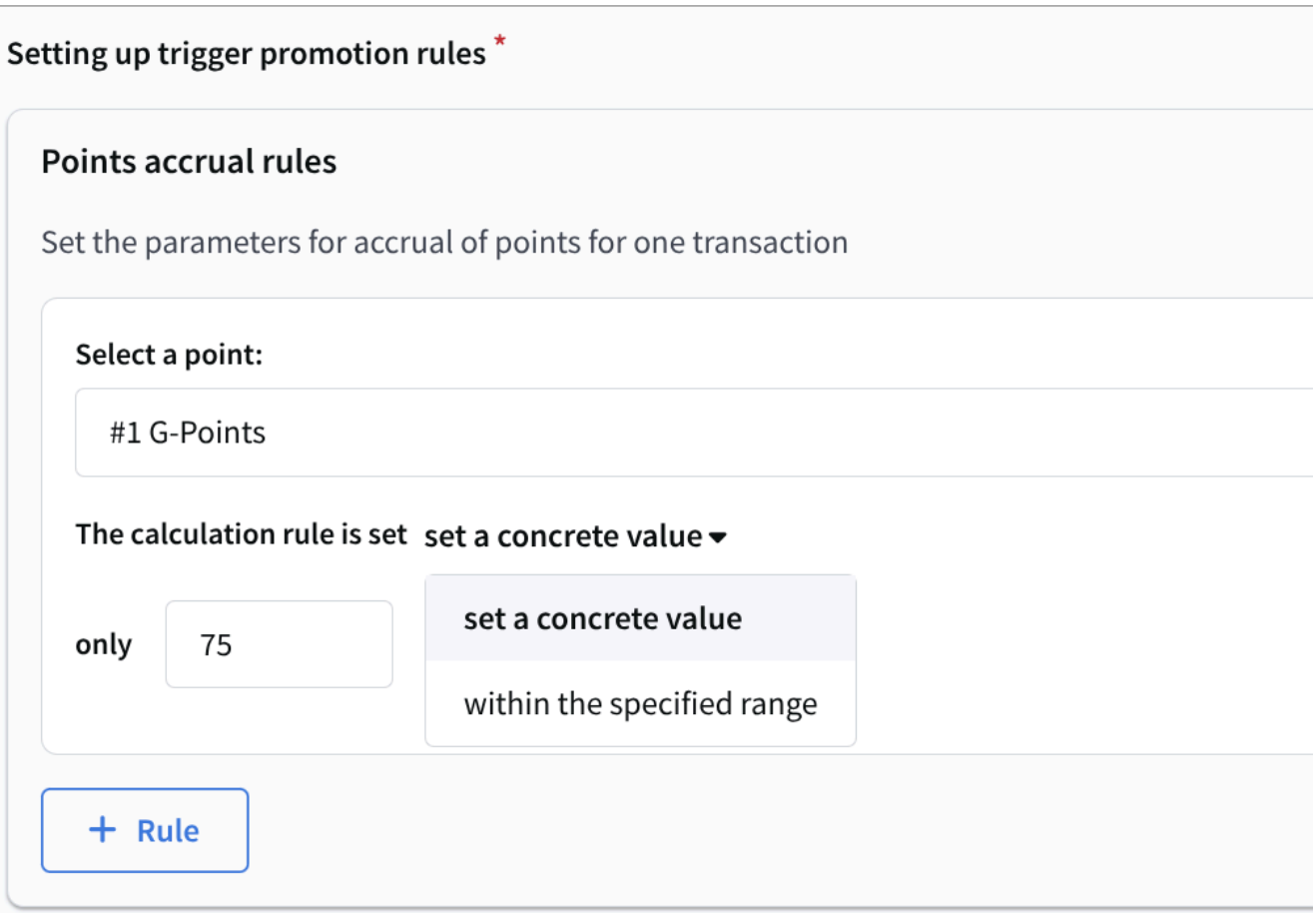
To create a new trigger promotion, click + Create. A promotion setup window opens:



Specify the name of the promotion and its additional identifier. Add a description if needed.
Then set the start and end dates of the promotion:

- **Promotion start — starts immediately after activation or at a specific date and time;**
- **Promotion end — ends manually, at a specific date and time, or after a specified period.**

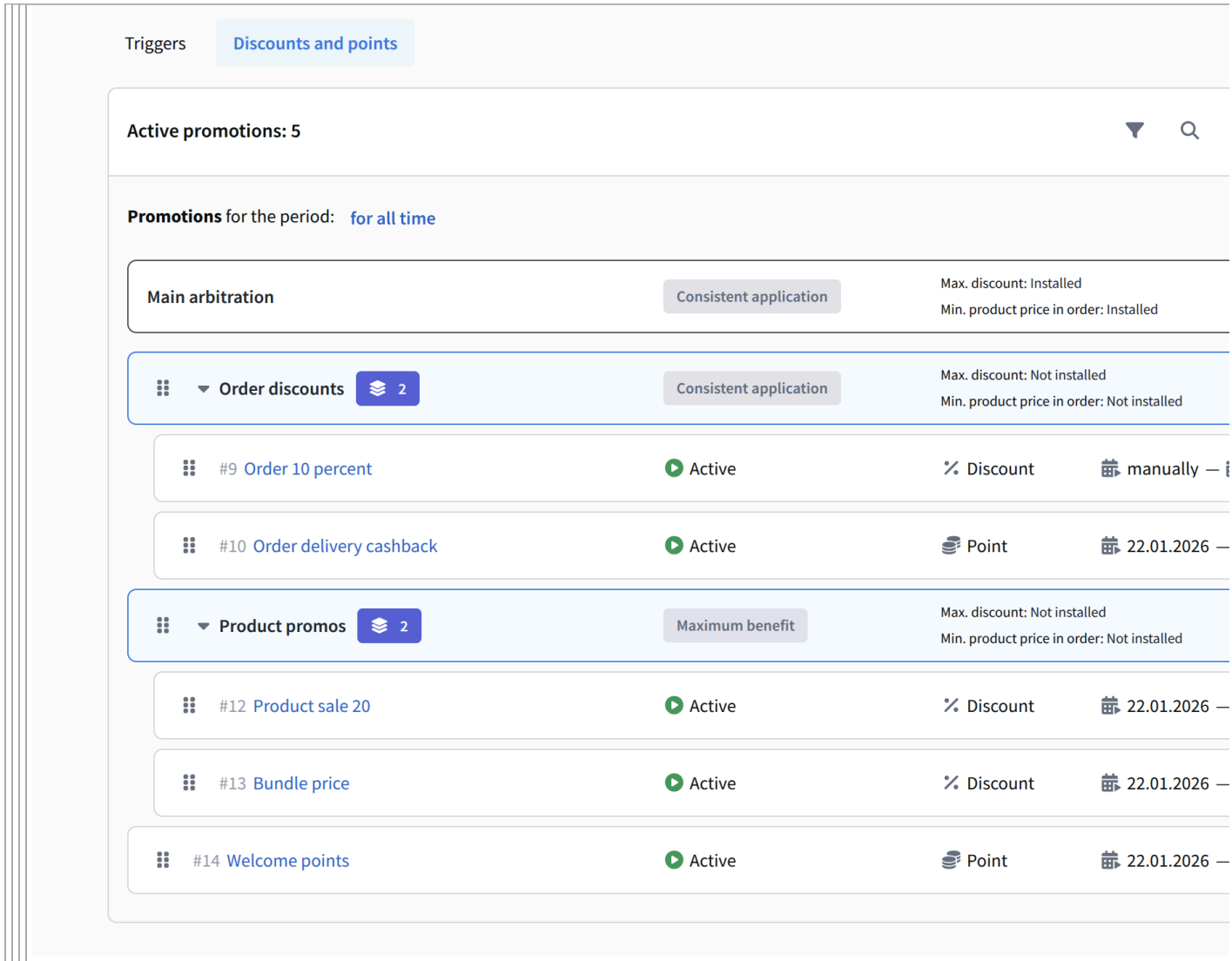
Next, define the promotion rules. They control how points can be accrued and deducted for participants.
configured as separate conditions. Specify the point type to which the rules apply and the exact value o



After saving and activating the promotion, it can be used in trigger-based campaigns and scenarios to au you can set up a promotion that awards bonus points for placing a first order, purchasing products from a specific order amount.

Discount and points promotions

Points and discount promotions are loyalty program promotions that are automatically applied at checko Market. They manage point accrual and deduction, discount application, and other benefits based on cor application rules.



Points promotions manage a participant's loyalty balance. They accrue or deduct points, both standard a conditions are met, such as purchasing products, reaching an order total, participating in a promotion, o promotions do not change product prices directly; they affect the participant's future benefits and loyalt can accrue both standard and qualification points, depending on the selected point type and configured simultaneously manage the participant's bonus benefit and their loyalty program tier.

Discount promotions are applied to orders or individual products and change their price. They are used t discounts, special prices, and other incentives.

Arbitration

Promotions can run simultaneously and apply to the same order or product, so the system requires a me together. For this purpose, the loyalty program includes arbitration.

Arbitration is a promotion application rule that defines how the system processes multiple eligible prom determines:

- **which promotions are applied;**
- **the order in which they are applied;**
- **whether promotions can be combined;**
- **which promotions apply together and which do not.**

Arbitration is used whenever a single order or product meets the conditions of multiple promotions. It is works hierarchically.

Main arbitration is the core arbitration applied to all points and discount promotions within the loyalty pr calculation logic and always affects the final result, regardless of promotion structure.

Even if promotions are grouped or have their own rules, the final calculation always goes through the ma

Group arbitration allows promotions to be combined into groups, each with its own arbitration settings. I group interact. Groups can also contain subgroups.

Group arbitration works locally, only within its group. Its result is then passed upward and processed by arbitration.

In-promotion arbitration applies when a single promotion contains multiple benefits. It defines whether b sequentially.

There are several types of arbitration:

- **Consistent application — promotions are applied one after another in the order defined within the gr calculated based on the result of the previous one. For example, if one discount is applied first and a is calculated from the reduced price. Suitable when order and step-by-step price or balance changes**
- **Incompatibility — only one of the eligible promotions is applied. At the product or order level, the firs is applied, and the rest are ignored. Used when promotions logically exclude each other, for example**
- **Summation — all eligible promotions are applied simultaneously. Discounts are added together, and a result is calculated as the combined effect of all promotions in the group. Suitable when multiple pr**
- **Maximum benefit — the system evaluates possible promotion combinations and selects the one that participant. Used in complex promotional mechanics where the best option should be selected auto setup.**

CAUTION: Arbitration type limitations

Not all benefits are supported by some arbitration types:

For “Maximum benefit” arbitration, the following are unavailable:

- **Display templated text**
- **Issue a promo code**

For “Summation” arbitration, the following are unavailable:

- **Round price / Round order price**
- **Use price from field**

- **Award points from a custom field**

Additional limits can be configured for arbitration to control the final result and prevent excessive discou

- **Maximum discount — limits the maximum discount that can be applied regardless of the number or c If the total discount exceeds the limit, the system caps it automatically.**
- **Minimum product price — sets the lowest allowed product price after all promotions are applied. If p price below this minimum, the system adjusts the result and stops further reduction.**

Limits configured in the main arbitration apply to all promotions in the loyalty program.

Arbitration keeps promotion outcomes predictable and controlled, even with complex discount and bonu

TIP: Practical example

Assume a customer wants to buy an all-in-one PC for ₽100,000. They have access to a 20% birthday dis price with bonus points. Here is how it works with different arbitration types:

Consistent application 100,000 → −20% = 80,000 → −15% = ₽68,000 Promotions are applied one after ar previous one.

Incompatibility 100,000 → either −20% = ₽80,000, or −15% = ₽85,000 Only one promotion is applied.

Summation 100,000 → −(20% + 15%) = ₽65,000 All promotions are added together.

Maximum benefit Comparison: 80,000 / 85,000 / 68,000 / ₽65,000 The lowest price is selected.

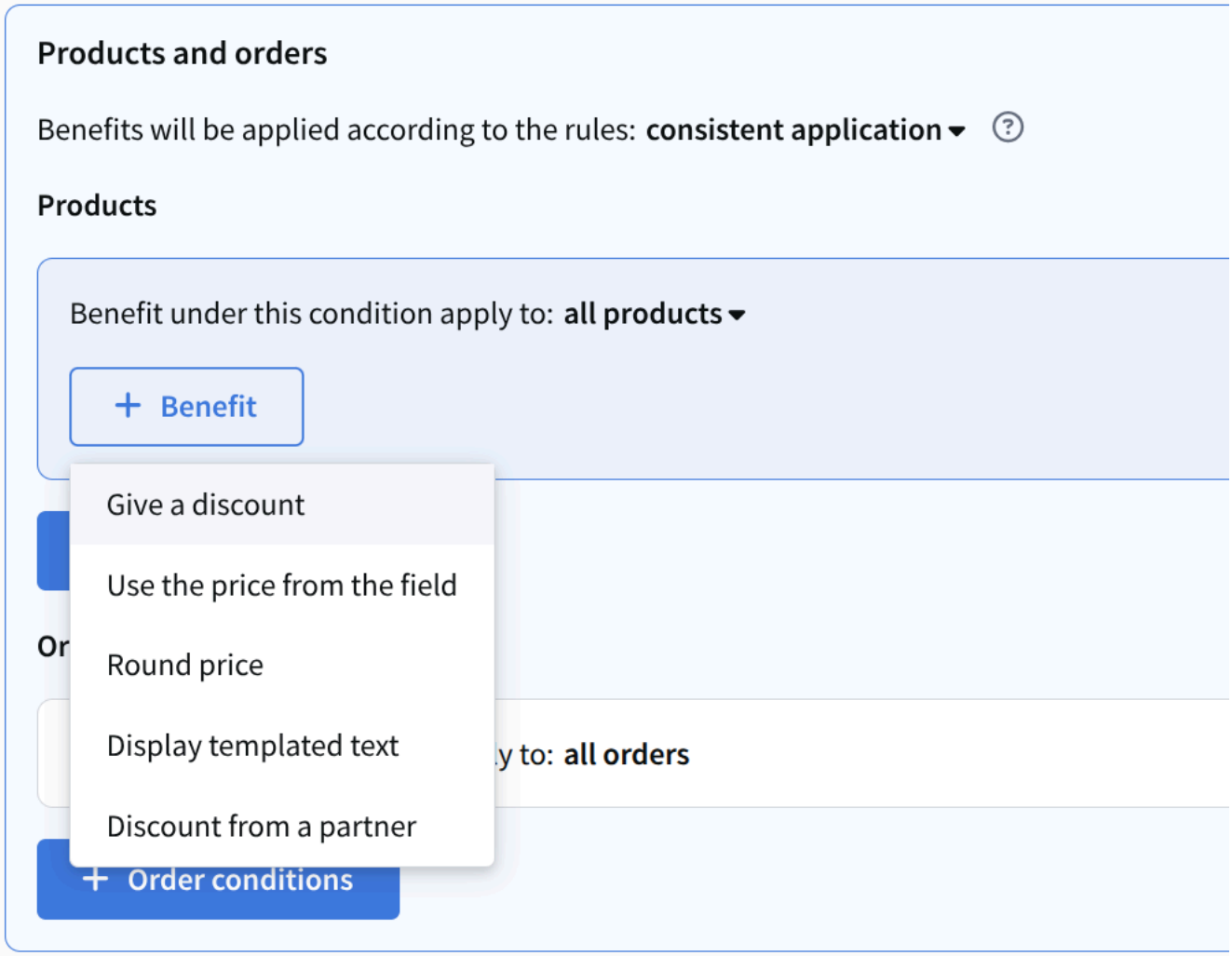
Benefits for discount promotions

For products:

- **Give a discount — a discount in a specified amount or percentage of the product price;**
- **Use the price from the field — uses a price from a custom product field. This price can be multiplied 0.75 for a 25% discount;**
- **Round price — rounds the product price to an integer, half, nearest integer down, tens, or nearest ter**
- **Display templated text — passes text with the order, supporting variables and emoji. The text is sent via API, for example to an email or POS terminal;**
- **Discount from a partner — grants a discount on products from selected partners according to partne**

For orders:

- **Give a discount — a discount in a specified amount or percentage of the order value. You can limit th order;**
- **Round order price — rounds the final order total;**
- **Display templated text — passes text with the order;**
- **Issue a promo code — selects a promo code from a specified pool and links it to the participant. You template that the participant receives with the promo code.**



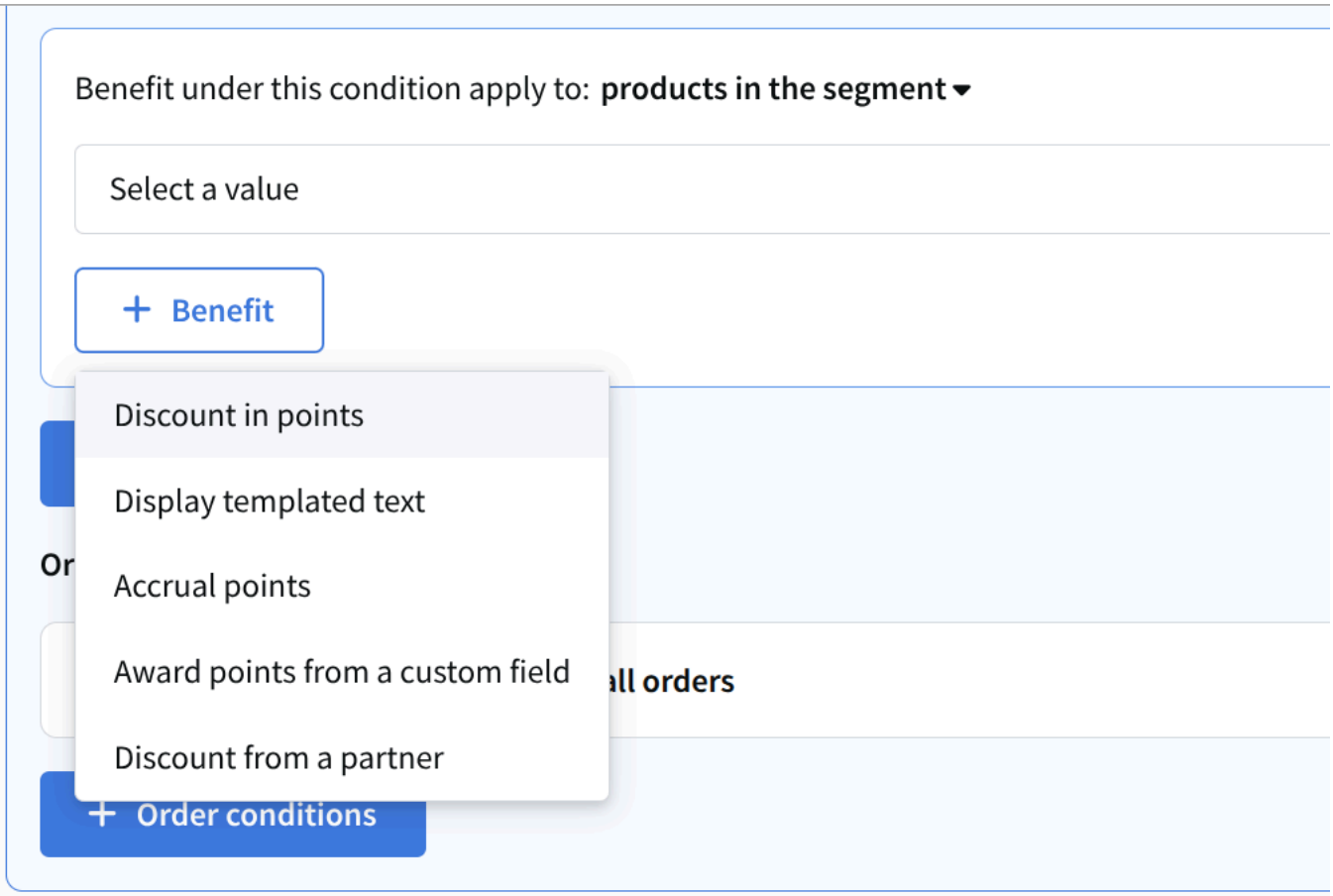
Benefits for points promotions

For products:

- **Discount in points — a discount issued in points in a specified amount or percentage of the product p can limit the number of discounted items per order;**
- **Display templated text — passes text with the order;**
- **Accrue points — credits specified loyalty points to the profile. You can configure when points are cre or after delivery;**
- **Accrue points from a custom field — credits the number of points taken from a custom product field. number of points for each currency unit spent, for example, 200 points for every ₽1,500 spent;**
- **Discount from a partner — grants a points-based discount on products from selected partners accor**

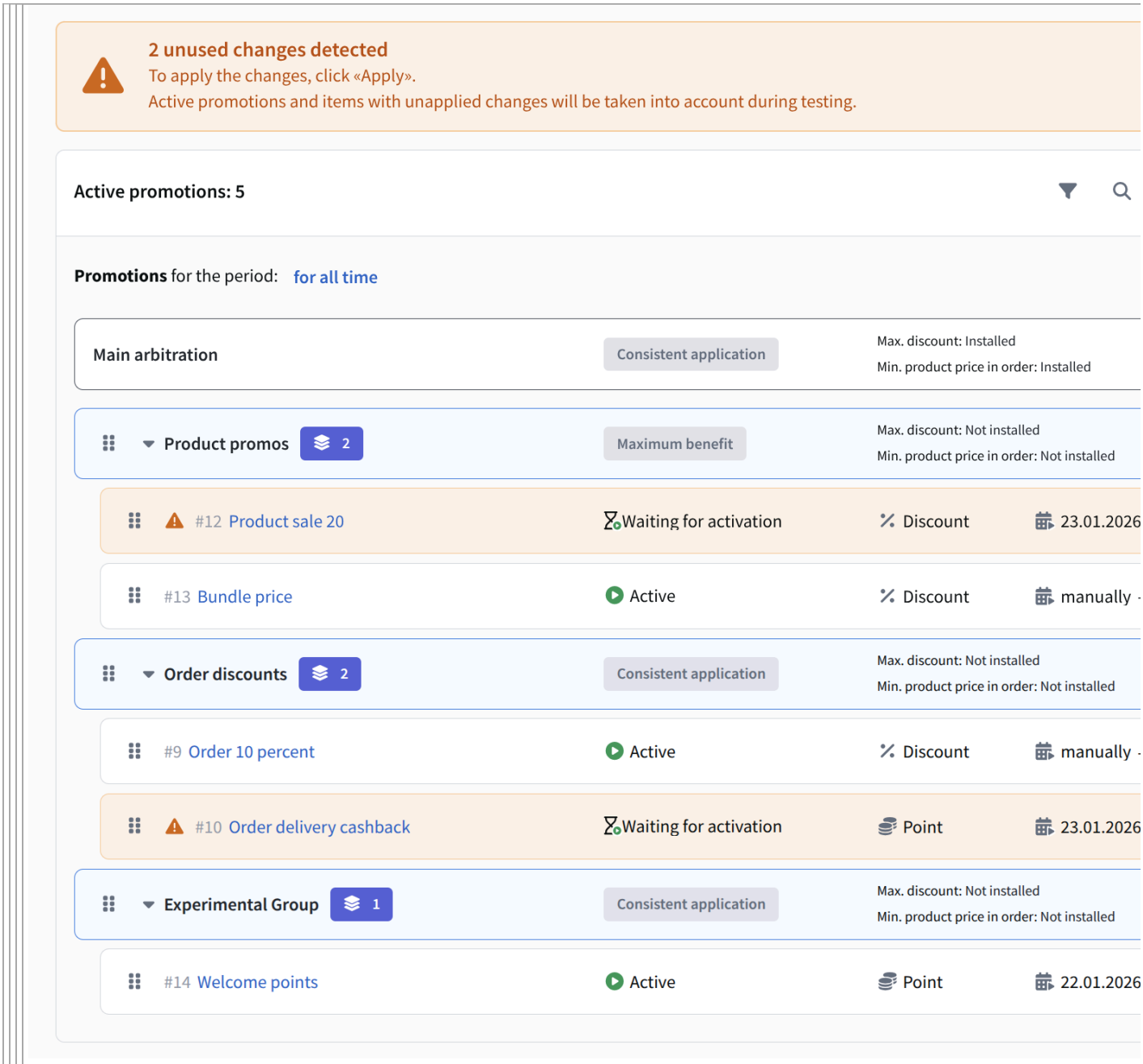
For orders:

- **Discount in points — a discount issued in points as a percentage of the order value or a fixed numbe**
- **Display templated text — passes text with the order;**
- **Accrue points — credits points for the entire order.**

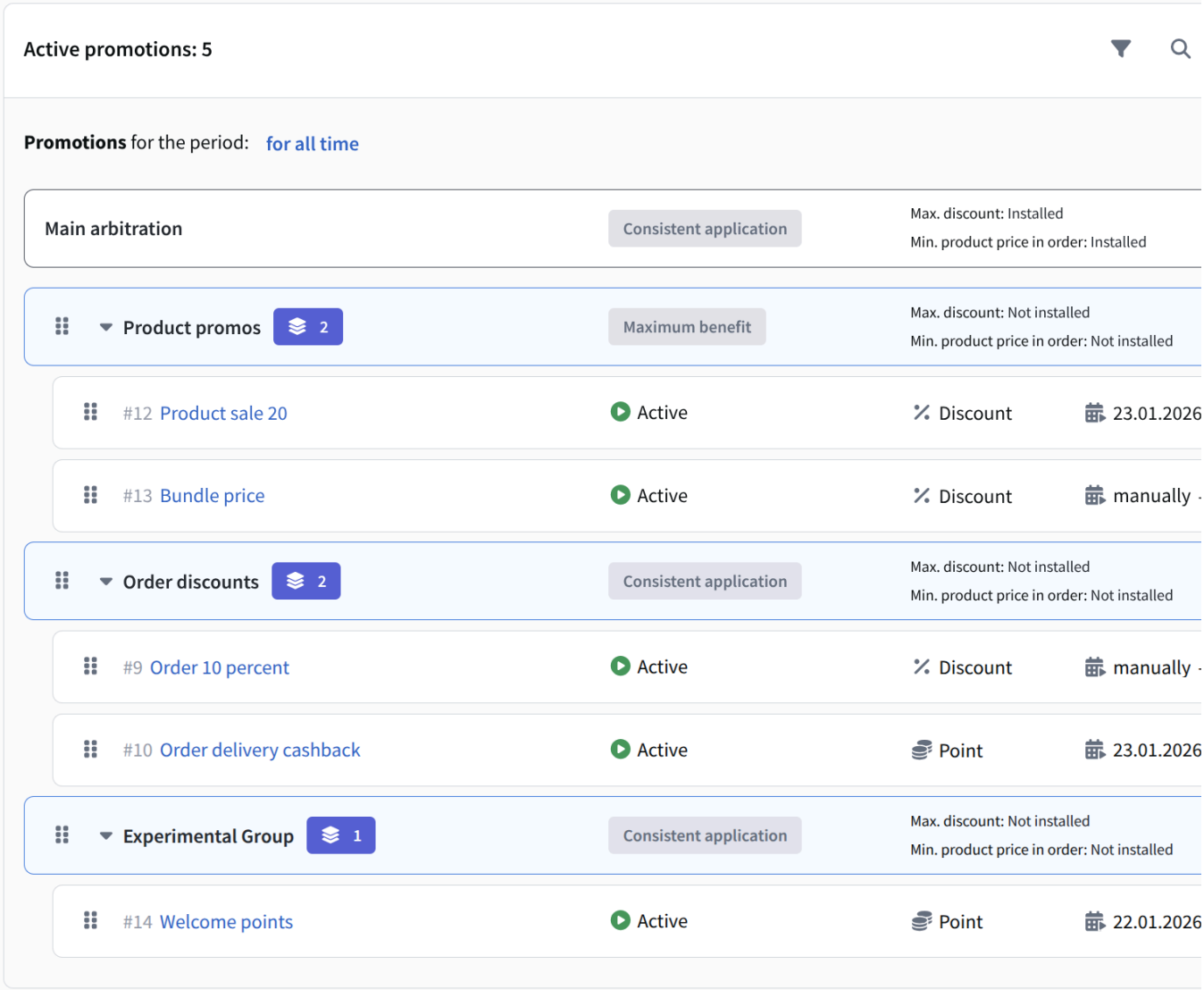


After completing all required sections, the promotion can be saved or activated. An activated discount p calculations according to configured arbitration and conditions. If saved, the promotion appears as a dra
When working with promotions, the top of the page shows the change status and quick actions.
If there are unapplied changes in the tree, the system displays a notification: “Unapplied changes detect

- **Apply — makes changes effective for real calculations.**
- **Revert — discards unapplied changes and restores the last applied version.**



Unapplied changes are considered in test mode only. Real calculations use changes only after clicking A
Below is the list of promotions and groups:

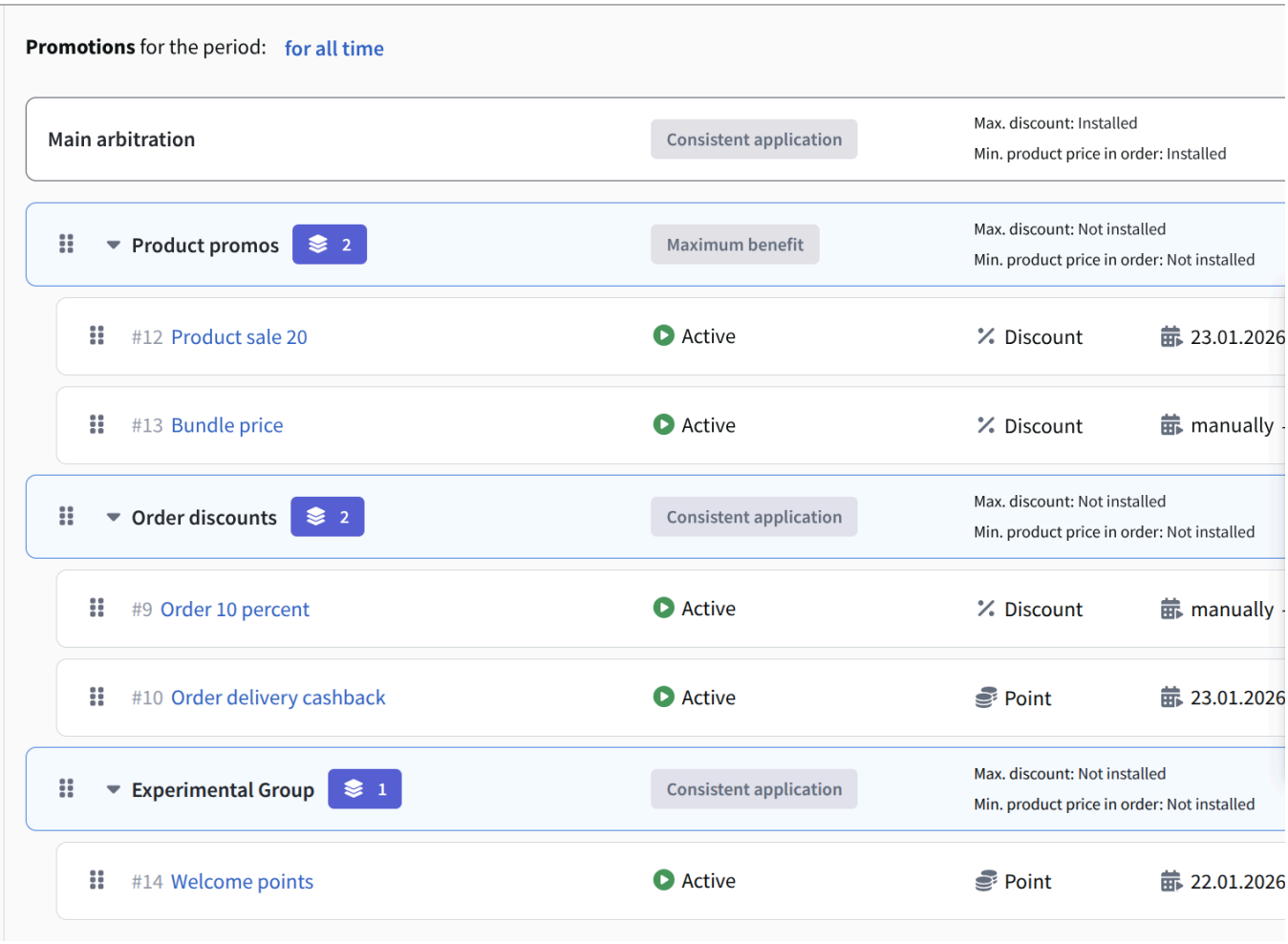


- **the header shows the number of active promotions;**
- **quick actions on the right: filters/search, Test, Create;**
- **the tree shows the Main arbitration and its limits, such as maximum discount and minimum product**
- **groups and promotions have action menus, and items can be dragged within the tree and moved bet**

For created promotions, a quick actions menu is available, which can be used to:

- **edit the promotion;**
- **deactivate it (if active);**
- **move it to another group;**
- **clone it;**
- **end the promotion (all related processes stop and it cannot be reactivated).**

Group management



Promotion groups are used to logically organize promotions and manage their interaction through separa
A group is a node in the promotion tree where its own arbitration type is defined, limits for maximum dis configured, and the application order of included promotions is set.
Groups can be nested (no more than 9 levels) and placed inside other groups or within the loyalty progra
When creating a group, specify:

- **group name;**
- **parent group or main arbitration;**
- **arbitration type and limits.**

After creating a group, promotions can be:

- **moved into it during creation or editing;**
- **dragged between groups directly in the list;**
- **reordered within the group to affect the result (except for Summation);**
- **deactivated in bulk via the quick actions menu, and the group can also be archived.**

As with promotions, changes to group structure and promotion order must be applied. Until applied, cha mode. The final result of applying promotions is always additionally processed by the loyalty program's r to a group (activation, deactivation, archiving) cascade to all nested promotions and subgroups.

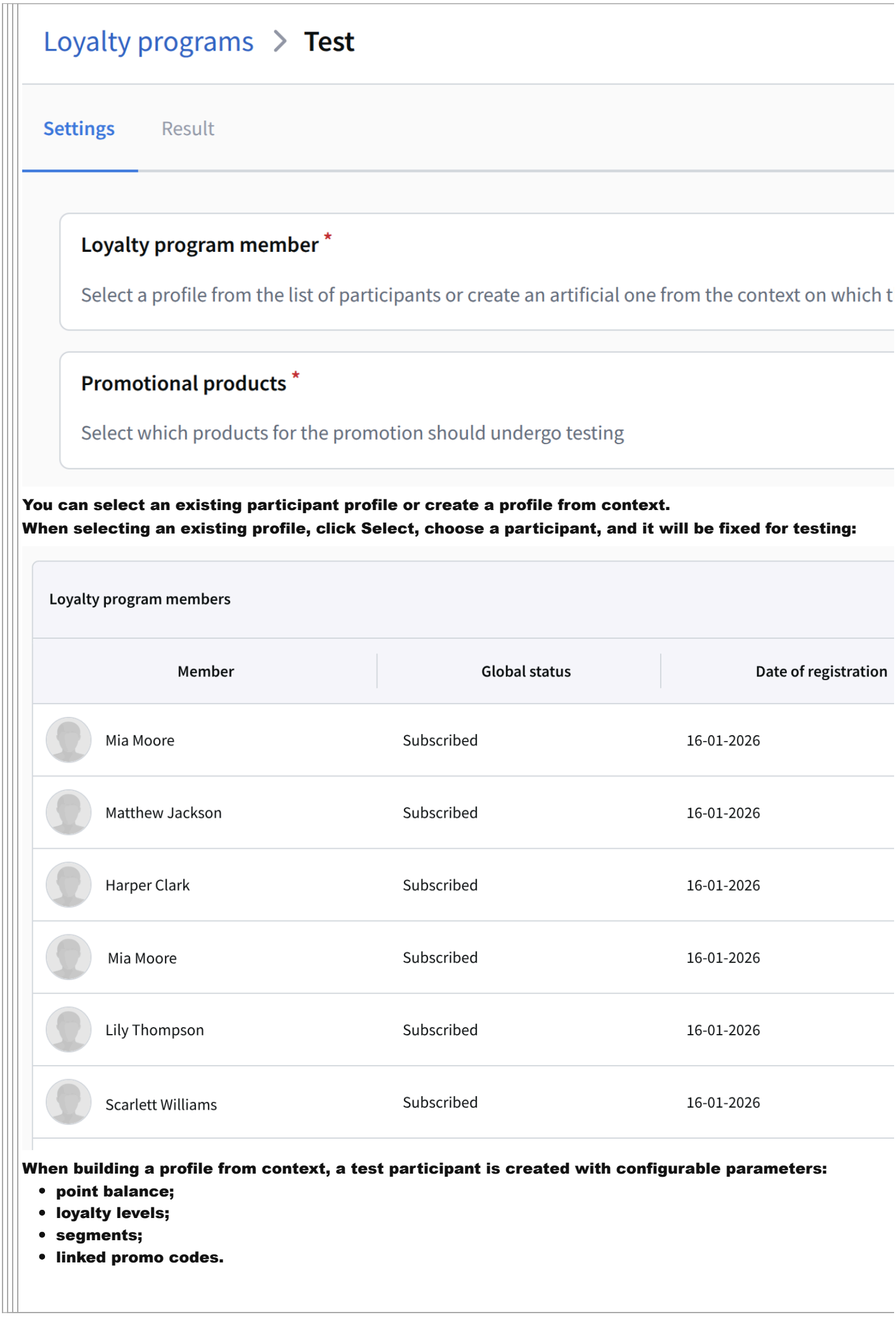
INFO

A group is deleted only if one of the following conditions is met:

- **there are no promotions inside the group;**
- **all promotions inside the group have the "Draft" status.**

Promotion testing

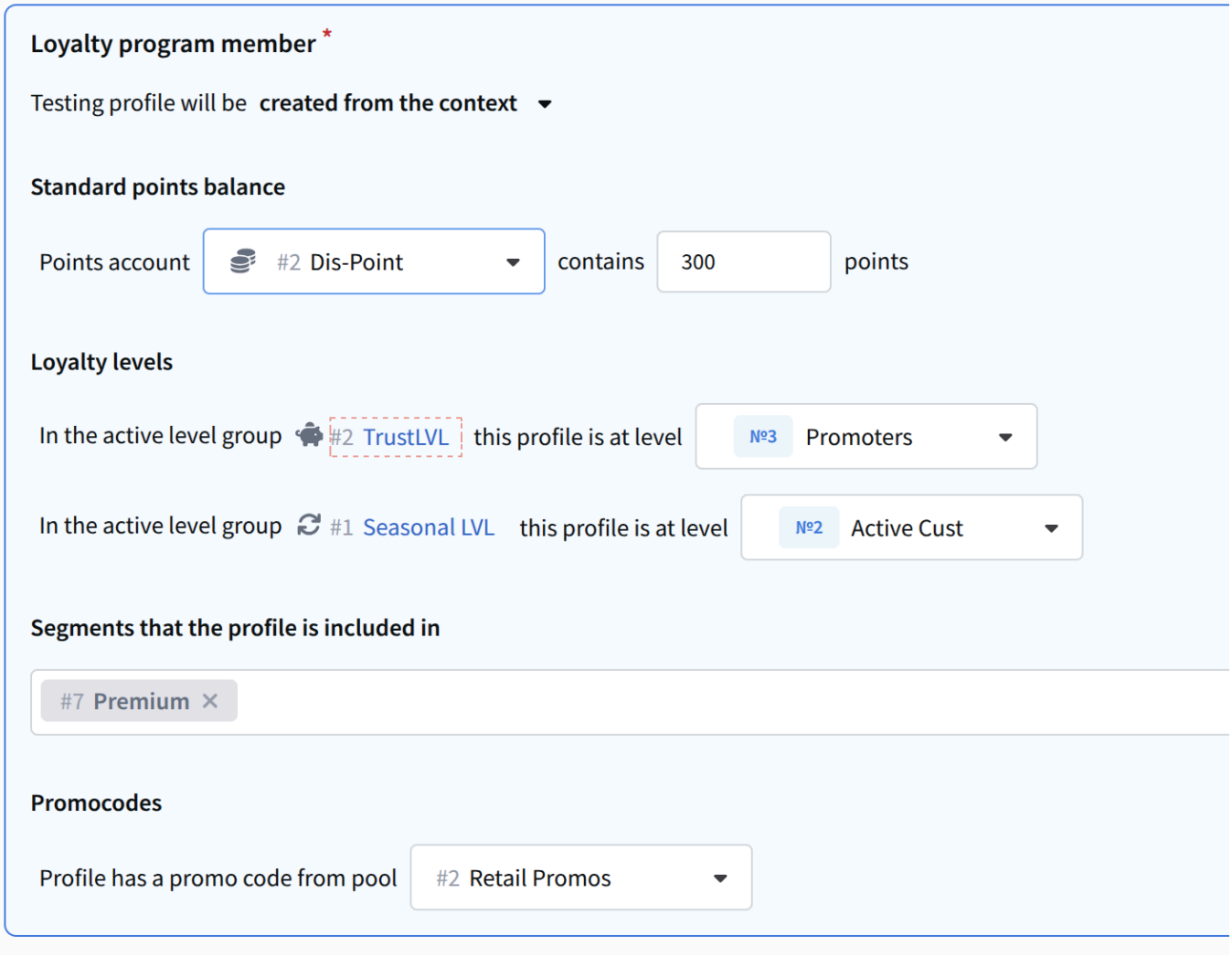
To check how promotions apply to an order, click Test to open test mode, where you can set order and p result, including unapplied changes.
Select a member and products for testing:



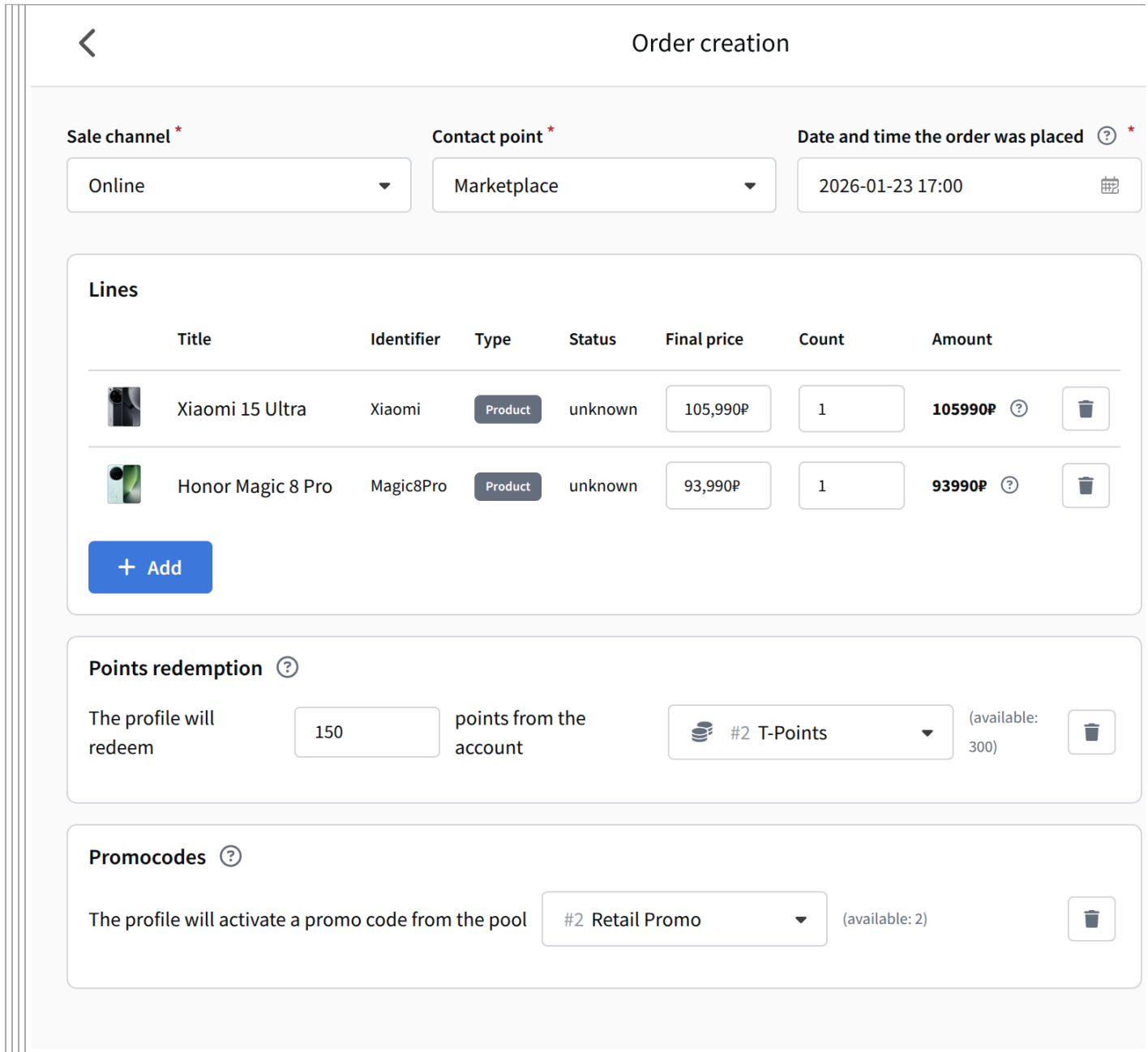
You can select an existing participant profile or create a profile from context.
When selecting an existing profile, click Select, choose a participant, and it will be fixed for testing:

When building a profile from context, a test participant is created with configurable parameters:

- **point balance;**
- **loyalty levels;**
- **segments;**
- **linked promo codes.**



Next, define the products for the promotion within an order:

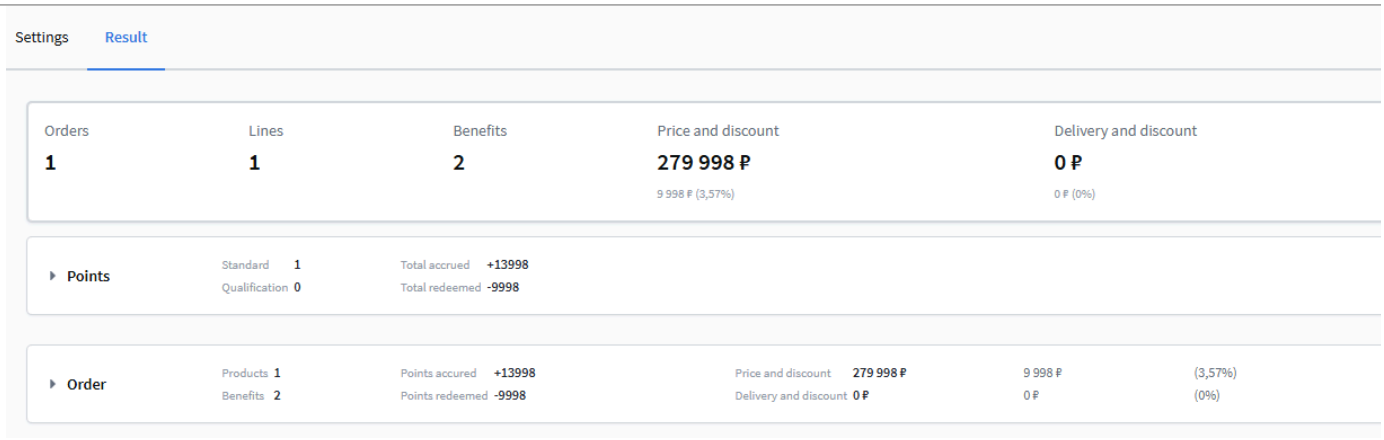


For a test order, specify:

- **Sales channel and contact point — the channel through which the customer placed the order;**
- **Order date and time, including time zone;**
- **Lines — products, quantity, and final price. After selecting a product, you can edit it. Changes apply Market data. This is useful for testing promotions on new prices;**
- **Point redemption — amount and type of points to deduct within the order;**
- **Promocodes — promo codes activated by the profile;**
- **Shipping cost;**
- **Region.**

If Auto-calculate is enabled, the final price is calculated as the sum of products and delivery.

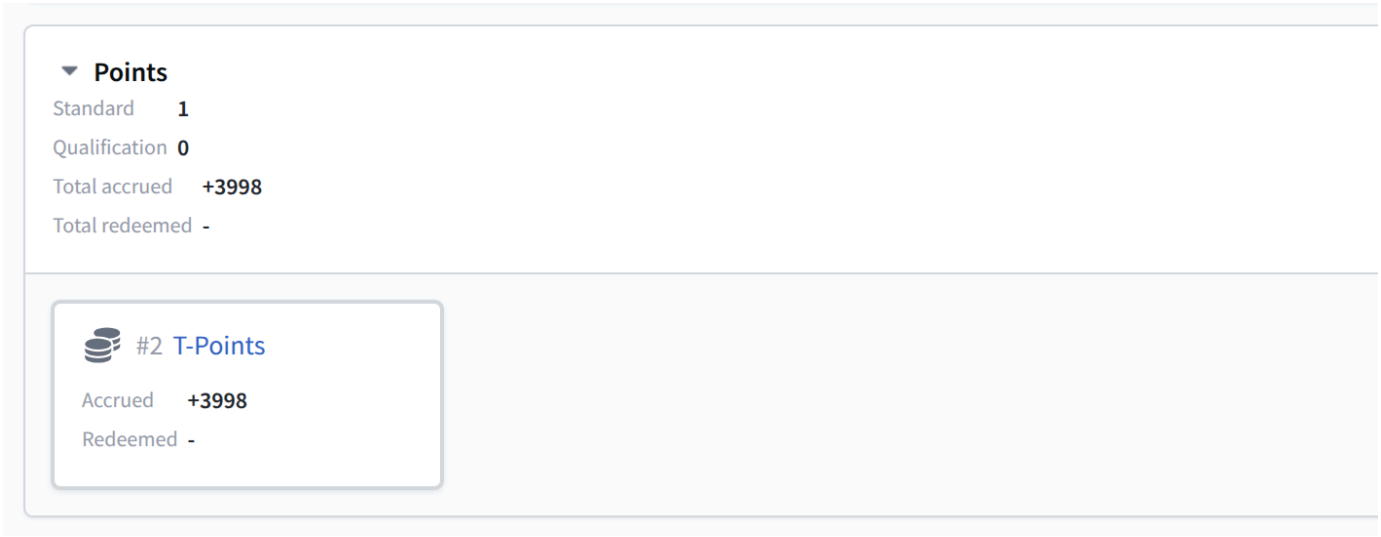
After configuring the participant and products, click Test. The Result tab shows the final benefit applicat



The panel shows:

- **number of orders;**
- **number of items;**
- **number of applied benefits;**
- **total order value (before discount) and discount amount (under the value);**
- **delivery cost and discount;**
- **final order total with applied discount (discount amount shown under the total).**

The Points tab shows loyalty balance changes after the order:



The Order tab contains a table of benefit-related operations. Each row includes:

- **order item number;**
- **line (product);**
- **benefit;**
- **benefit result;**
- **applied promotion;**
- **promotion type.**

Positions details

Position number	Position	Benefit	Result	Pro
1	Xiaomi 15 Ultra	Issue a discount	-15 898,5 ₽ (-15%)	#9 Order
1	Xiaomi 15 Ultra	Accrue points	+1999 to #2 T-Points	#14 Welc
2	Honor Magic 8 Pro	Issue a discount	-14 098,5 ₽ (-15%)	#9 Order
2	Honor Magic 8 Pro	Accrue points	+1999 to #2 T-Points	#14 Welc

The table supports search and configurable visible columns via the actions menu.

Transactions

The Transactions section contains a list of all transactions within the loyalty program. Here you can also or redeem points.

Basic | Members and Partners | Promotions | Transactions

Transaction list

ID	Addtl.ID	Created	Transaction type
aaa5ca4c-cc40-4422-bde5-a854dfb81003	transaction_welcome	23-01-2026 17:34	Accrual

The transactions table provides detailed information about each balance change. It includes the followin

- **ID — transaction ID**
- **External ID — external transaction identifier**
- **Created at — transaction creation date**
- **Transaction type — transaction type**
- **Member — loyalty program member profile**
- **Points — points type used**
- **Balance before — balance before the transaction**
- **Value — number of accrued or redeemed points**
- **Balance after — balance after the transaction**
- **Source type — entity type that initiated the transaction**
- **Source — link to the source entity**

To create a new transaction, click + Create. The transaction setup window will open:

Create a points transaction

#welcoming

Promotion member *

The member whose account needs to be manipulated

John Wayne

Trigger promotion *

Select the TPA within which to conduct the transaction

1 - Promotion Triggers

Points *

Select the points to whose account the transaction will be made

2 - Q-Points

Operation value *

Configure the type and value of manual transaction for selected member

- **Accrue points to account**

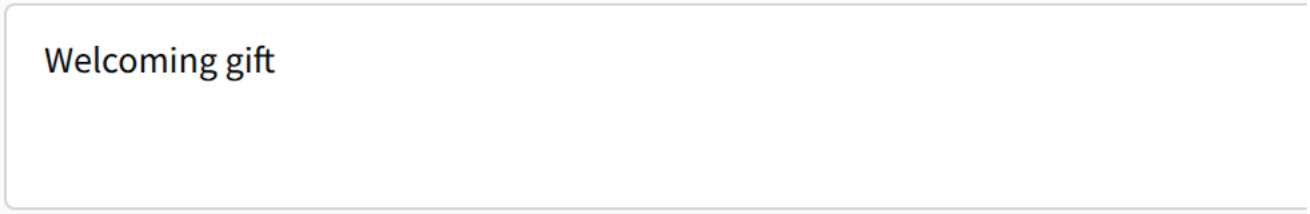
  Specify the number of points to be credited to the account, and configure activa
  and expiration of credited points

  Accrue **only 75 points** according to the rule accrual

- **Redeem points from account**

  Specify the number of points to be debited from the account

Comment *



- **Optionally specify an external identifier;**
- **Select the member — any profile registered in this loyalty program;**
- **Select the trigger promotion under which the transaction will be conducted;**
- **Select the points type;**
- **Specify the operation type — accrual or redemption;**
- **Add a transaction comment;**
- **Optionally specify additional JSON fields.**

Integrating loyalty programs with an external order system

A loyalty program in Altcraft Platform can be used in two ways, depending on whether product and order
via the Market module.

Market allows you to transfer product and order data to the platform and use it in customer communicati
automatically — via API or product feed, or manually — by uploading a file.

The availability of data in Market determines which loyalty mechanisms are available to you:

- **Without Market, you cannot automatically calculate discounts, accrue or redeem points during checl**
  operations are only available through trigger promotions and the Loyalty API.
- **With Market, in addition to basic operations via trigger promotions, discount and points promotions l**
  automatically accrue and redeem points during checkout, apply discounts, and calculate all applied
  arbitration. The partner products functionality also becomes available — for individual products, you
  increased cashback or an exclusive discount for program participants. Partner products are taken in
  via `preorder` / `preorder_commit` just like regular promotions.

Loyalty with Market integration

This option is suitable for automatically accruing and redeeming points during checkout, as well as appl
discount promotions. To make this work, you need to:

- **Upload product data to Market. Data can be uploaded via API, product feed, or from a file.**
- **Create discount and points promotions — these will be automatically applied when calculating an or**

Order processing with calculation of applied benefits occurs through a two-stage API:

- **At the first stage, the `preorder` method is called — it performs a preliminary calculation when the cus**
  includes the order composition and participant data (email, phone, profile_id). The platform returns t
  amount of points accrued and redeemed, and a list of triggered promotions. The order is not created
  calculation.
- **At the second stage, after the order is placed or confirmed, the `preorder_commit` method is called. The**
  points to the participant's account, deducts used points, and records all applied promotions.

> INFO: Please note
>
> **Automatic point accrual and redemption during checkout works only when using the two-stage API (`pre`**
> **If an order is created in any other way (for example, via import, direct addition in the interface, or custo**
> **not trigger. In such cases, point management must be implemented separately — through scenarios wit**
> **to the accrual and redemption API.**

Loyalty without Market integration

In this scenario, points are accrued and redeemed for user actions: filling out forms, installing apps, achi
promotions and related functions (testing, benefit application calculation within arbitration) are unavaila

Point management is based on using trigger promotions — they set the rules for accrual and redemption
moment a promotion is executed is determined either in automation scenarios (via the "Loyalty points m
calls to the Loyalty API (accrual, redemption, member registration). The `preorder` and `preorder_commit` met
they are only available when working with Market.

Examples of implementing this approach — accruing points for app installation, form submission, and no
be found in the article Basic loyalty program use cases.

Creating a loyalty program from scratch

This article outlines the basic workflow for creating Loyalty Program in Altcraft Platform.
Goal — create your first loyalty program, accrue points to a member, and understand the logic of the mo
Step-by-step guide: Loyalty program — Points — Members — Promotion — Testing

Step 1. Create a loyalty program

A loyalty program is a customer reward system for target actions. It stores points, configures tiers, and redeeming them.

1. Go to **Loyalty — Loyalty programs** and click: ![img](/img/attachments/loyalty-cases/en/creation-button.png)	![img](/img/attachments/loyalty-programs/en/choose_loy

2. Enter: - **Name** - **External ID*** And select a **Database***	![img](/img/attachments/loyalty-programs/en/settings.png)

- **External identifier — an internal code for the program, used for integrations. For example, you can sp**
- **Database — the source from which members will be taken.**

INFO: Important

After selection, the database cannot be changed, and only one active loyalty program can exist in a sin

4. Click

Create

The loyalty program is created, and you will automatically be taken to its settings.

Step 2. Create points

Points are the internal currency of the program, awarded for purchases and other target actions, and ca

1. In the program settings, open the **Basic** tab	![img](/img/attachments/loyalty-programs/en/general_settings.png)

![img](/img/attachments/loyalty-programs/en/currency_types2.png)

Standard points

![img](/img/attachments/loyalty-programs/en/standard_currency1.png)
Standard points — the primary type for accrual and redemption.

2. Configure the point settings: - Name; - External identifier — for example, `bonus_point1`	![img](/img/attachments/loyalty-programs/en/currency-settings.png)

3. Configure: - Initial balance; - Points validity period; - Bonus-to-currency ratio — specify how many points equal 1 unit of currency (e.g., `1`)	![img](/img/attachments/loyalty-programs/en/currency-setting.png)

4. Click

Save

Standard points are ready to use.

Qualification points

Qualification points are used to evaluate customer activity and determine their tier in the loyalty prograr

![img](/img/attachments/loyalty-programs/en/qualification-currency.png)

2. Configure the point settings: - Name; - External identifier — a required field, used for API	![img](/img/attachments/loyalty-programs/en/qualification-currency

3. Set the **bonus to currency ratio**:	![img](/img/attachments/loyalty-programs/en/qualification-currency

4. Click

Save

Qualification points are created. To use them, you need to create a tier group and tiers.

Loyalty levels group

A loyalty tier group is a set of statuses (e.g., “Bronze”, “Silver”, “Gold”) that members receive as they ac

1. Below the points list, find the Loyalty tier groups block and click ![img](/img/attachments/loyalty-cases/en/creation-button.png)	**![img](/img/attachments/loyalty-programs/en/level-groups-create.pn**
2. Enter a Name and select a Tier group model	**![img](/img/attachments/loyalty-programs/en/choose-accumulative.**
3. Link the previously created Qualification points.	**![img](/img/attachments/loyalty-programs/en/choose-created-point.**

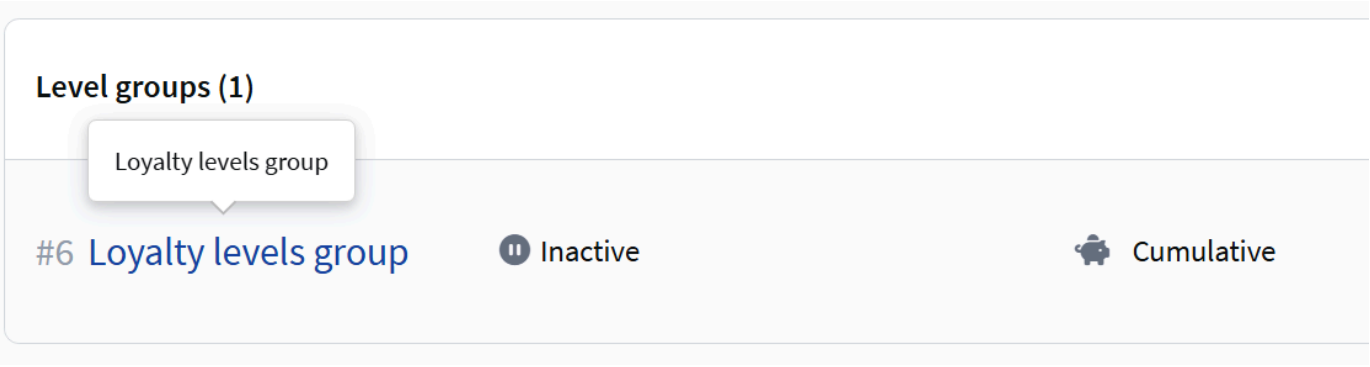
4. Click

Save

The tier group is created. Now you need to add tiers to it.

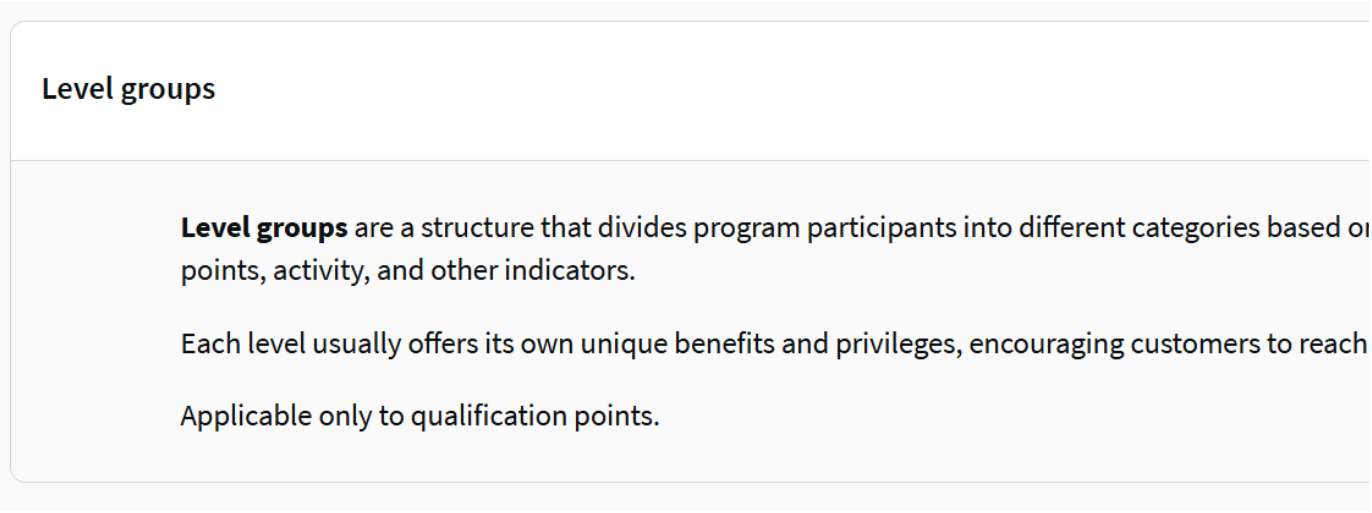
Create loyalty tiers

1. Click on the created group to open its settings



2. In the Manage tiers section, click

+ Create

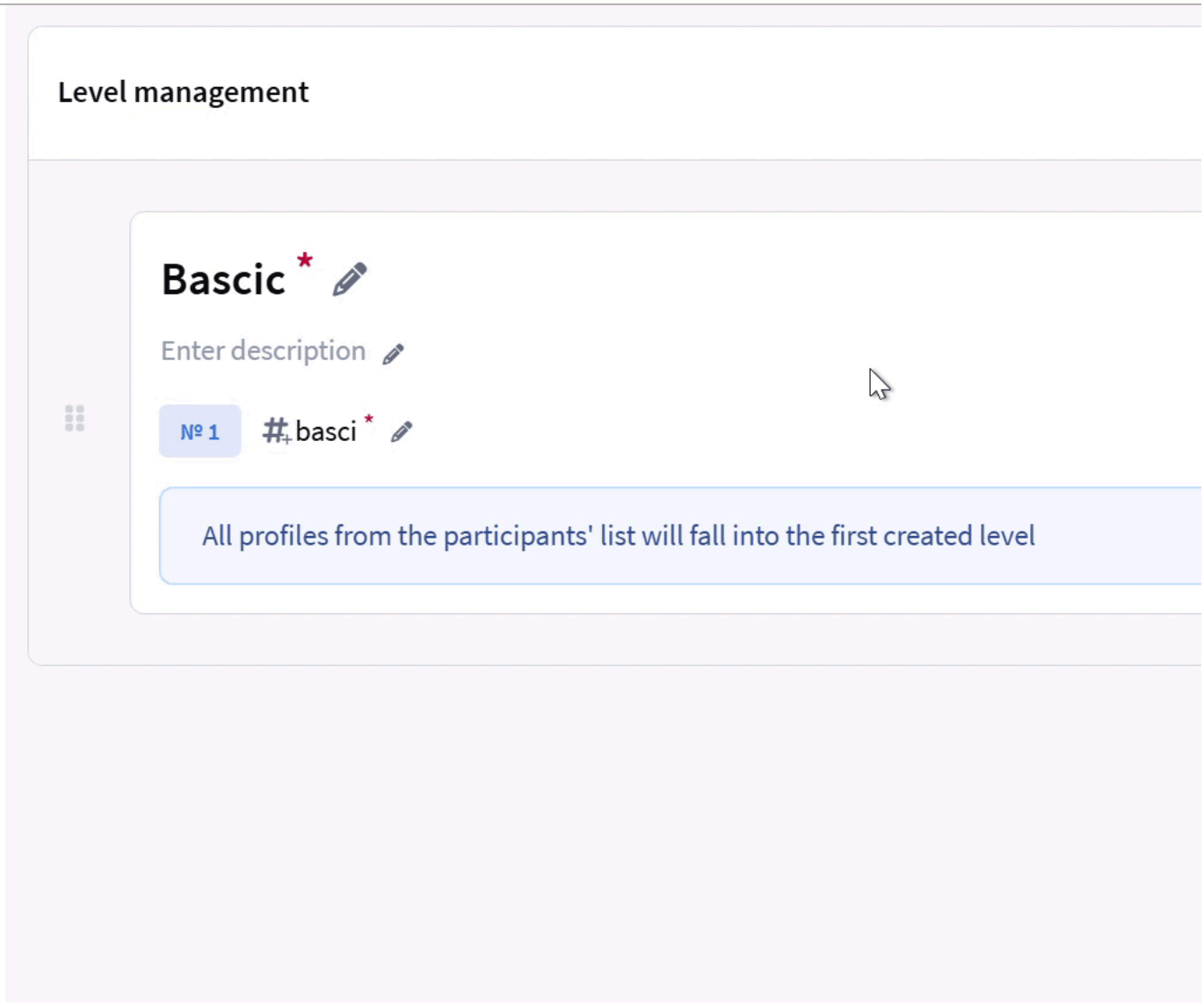


3. Create the first tier: **- Name — for example, "Basic" - External identifier — for example,** `basic`	**![img](/img/attachments/loyalty-programs/en/first-level-create.png)**

All program members automatically fall into the first tier.

INFO

To save a tier, click anywhere outside the tier card.



5. Create the required number of tiers. For each, specify: - name; - external identifier; - minimum balance.	![img](/img/attachments/loyalty-programs/en/next-level-create.png)

Minimum balance — the minimum amount of selected points in a profile's account required to reach the

6. After creating all tiers, click

Save

INFO

Members move to the next tier automatically when they reach the specified minimum balance of qualifi points through points promotions — just like bonus points.

Step 3. Add members

Members are the customers who will participate in the loyalty program.

1. Go to the Members and partners tab	![img](/img/attachments/loyalty-programs/en/participants_partners.
2. In the Loyalty program members block, click + Add	![img](/img/attachments/loyalty-programs/en/adding_profiles.png)
3. Select "Add all database profiles"	![img](/img/attachments/loyalty-programs/en/add-all-profiles.png)

All customers from the selected database are added to the program.

Step 4. Create a promotion

A promotion is a rule by which participants earn points. The platform offers two types of promotions: "Tr

For more details on promotions, see the article "Creating and configuring a loyalty program".

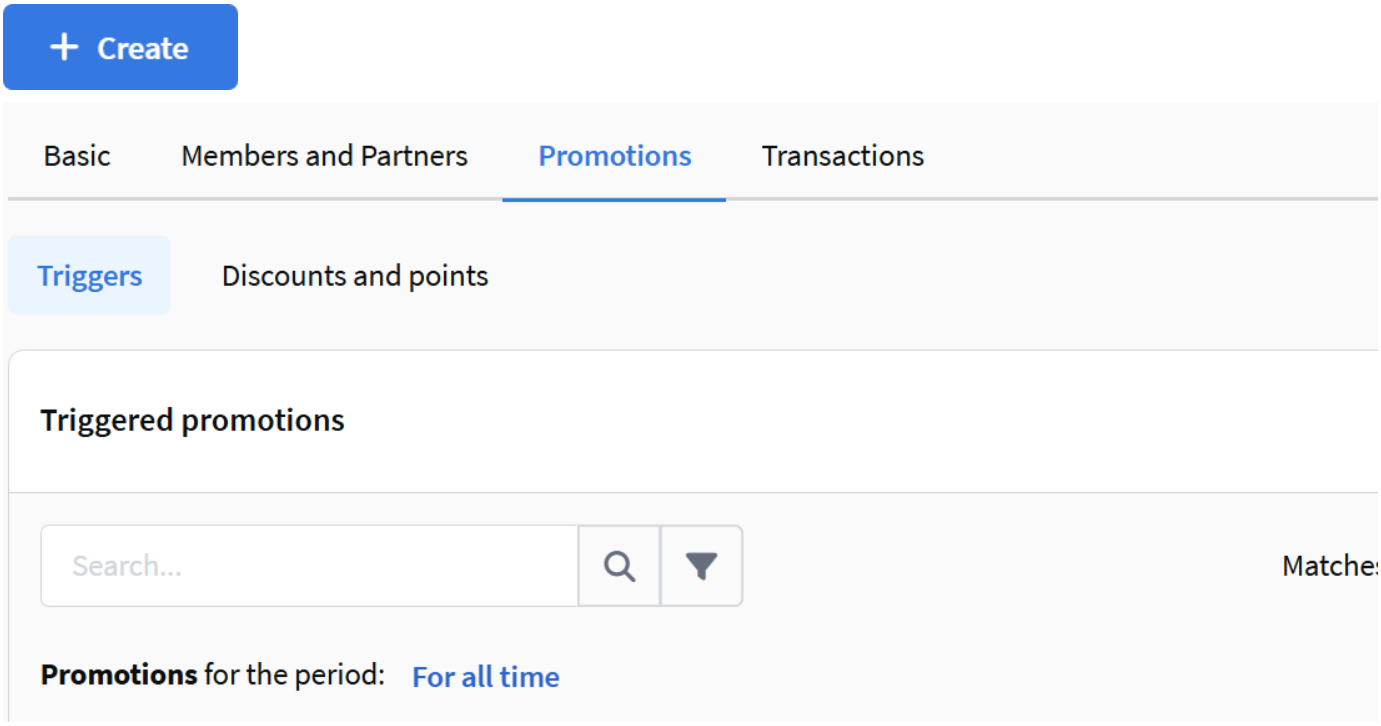
Trigger promotion

A trigger promotion defines rules for accruing and redeeming points that are not automatically applied t via API, or other integrations.

You can learn more about trigger promotions in this article

---

1. Go to the Promotions — Trigger promotions tab and click



2. Specify: - Name - External identifier - Active period	![img](/img/attachments/loyalty-programs/promos/en/triggered-name.png)

3. Configure Points accrual rules: - Select points — choose the previously created point type - Can accrue — configure the range:	![img](/img/attachments/loyalty-programs/promos/en/triggered-earn

Similarly, configure the Points redemption rules.

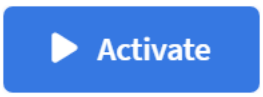
INFO

Instead of a range, you can specify a fixed value — useful if a specific action should award a fixed numl

---

4. Click

and



The trigger promotion is created.

Discount and Points promotion

Points promotions manage a participant's loyalty balance. They accrue or deduct points, both standard a conditions are met, such as purchasing products, reaching an order total, participating in a promotion, o

1. Go to the Promotions — Discount and points promotions tab	![img](/img/attachments/loyalty-programs/promos/en/promos1.png)

2. Click ![img](/img/attachments/loyalty-cases/en/creation-button.png) and select the Points promotion type	![img](/img/attachments/loyalty-programs/promos/en/choose1.png)

A points promotion manages the accrual or redemption of points. In this guide, we are creating a promot details on promotion types and their application, see the article "How the loyalty program works".

3. Fill in the basic settings: - Name; - External identifier Go to Products and orders	![img](/img/attachments/loyalty-programs/promos/en/promo-main-settings.png)

4. In the Products or Orders block, click + Conditions to add a rule.

INFO: Important

Conditions for accrual can be configured at both the product level and the entire order level. Choose the configuration mechanics are identical in both cases.

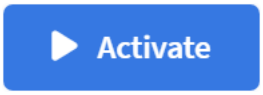
In the pop-up window, select the benefit Accrue points.

![img](/img/attachments/loyalty-programs/promos/en/choose-earn-points.png)

5. Configure accrual: **- Accrue — specify a % of the order/product price - To ... account — select the created point type**	**![img](/img/attachments/loyalty-programs/promos/en/benefit-setting**

6. Click

and



The promotion is created.

Step 5. Apply changes

All changes to promotions must be applied to take effect.

1. At the top of the promotions page, find the yellow notification	![img](/img/attachments/loyalty-programs/promos/en/changes.png)

2. Click the Apply button
Now the promotion is active.

Step 6. Test changes

Before waiting for real orders, you can test the promotion in test mode.

1. Click the "Test" button (next to the ![img](/img/attachments/loyalty-cases/en/creation-button.png) button)	![img](/img/attachments/loyalty-programs/promos/en/promos1.png)
2. In the Loyalty program member block, click "Select" and choose any profile from the list	![img](/img/attachments/loyalty-programs/promos/en/select-membe
3. In the Products for promotion block, create a test order.	![img](/img/attachments/loyalty-programs/promos/en/test-order.png

4. Click Test

INFO: About testing

Test mode works for all types of promotions: accrual, redemption, discounts. For more details on testin see the article [“Loyalty programs”](#).

5. Go to the Points tab	![img](/img/attachments/loyalty-programs/promos/en/test-points.png

The Points tab displays test transactions. This means the promotion is configured correctly.

Basic loyalty program use cases

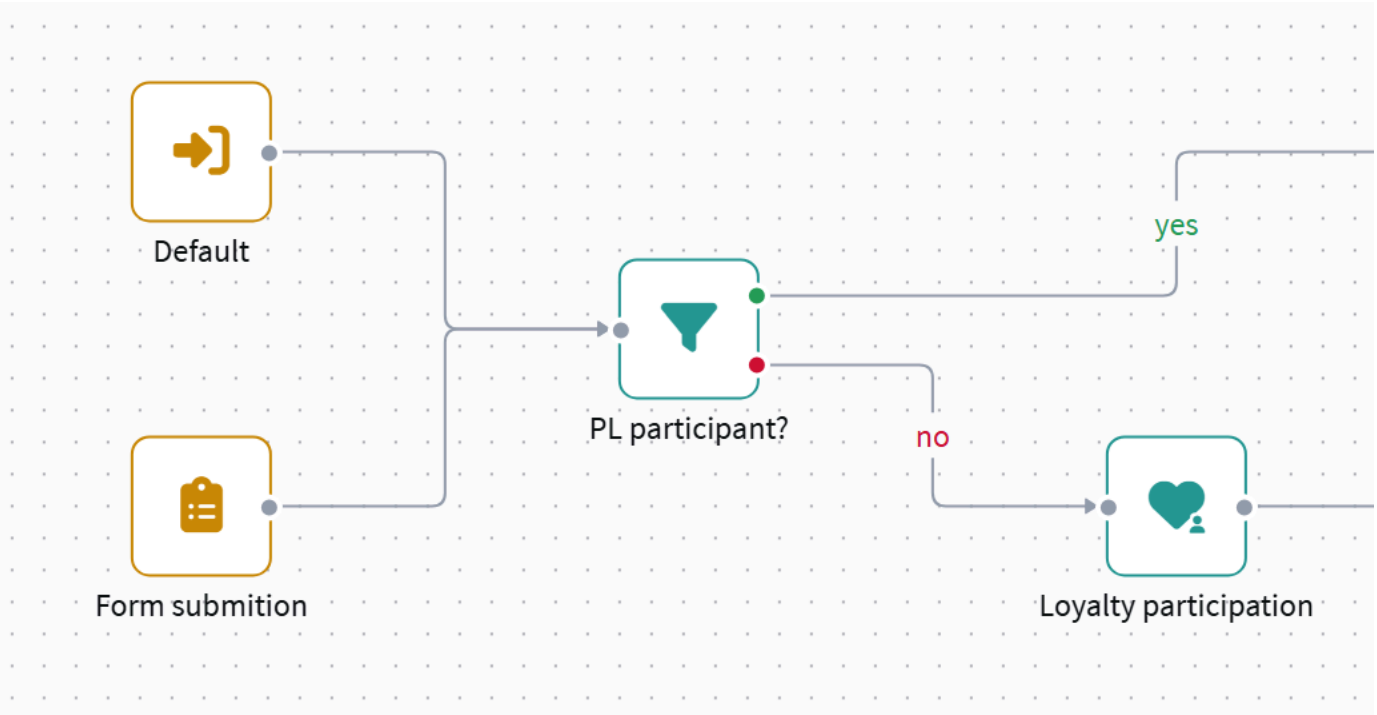
A loyalty program in Altcraft can be used for automatically rewarding users and for personalized commu interaction with the product. Below are basic practical cases that show how to configure point accrual, program, and sending notifications based on user actions and their balance status. All examples are des be adapted to specific business tasks.

CAUTION

Altcraft Platform provides technical tools for storing and processing data, including custom fields that participate in the loyalty program. The platform is not responsible for the legal correctness of the meth such consent. Issues of compliance with legal requirements (including those related to personal data a the user's responsibility, taking into account applicable regulations and the advice of legal professiona

Rewarding a profile for filling out a form

This case allows you to automatically add a user to the loyalty program immediately after they fill out a application. The logic works correctly for both new participants and those already registered in the loya filling out the form:



INFO: Please note

If the initial balance in the point settings is zero, after the "Loyalty participation" node, you must also a node with a trigger promotion to award the bonus for filling out the form.

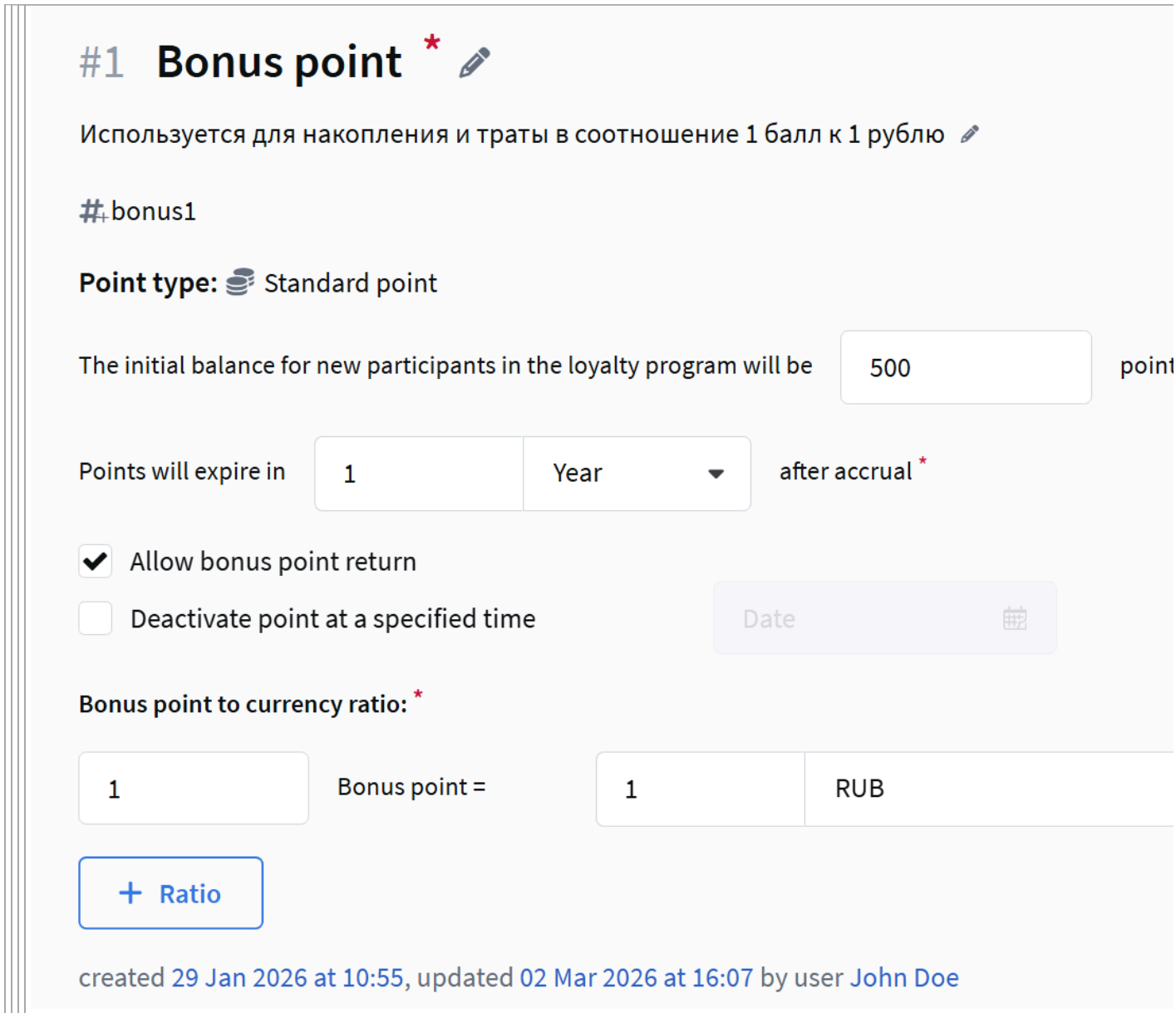
Loyalty

Before creating a scenario, you need to prepare the basic configuration of the loyalty program. In the Lo create the program itself.

Configure standard points. When creating standard points, you must fill in the required parameters:

- **point name and external identifier;**
- **initial balance for a new loyalty program member;**
- **points expiration period (after a specified number of days, months, or years after accrual);**
- **point-to-currency ratio.**

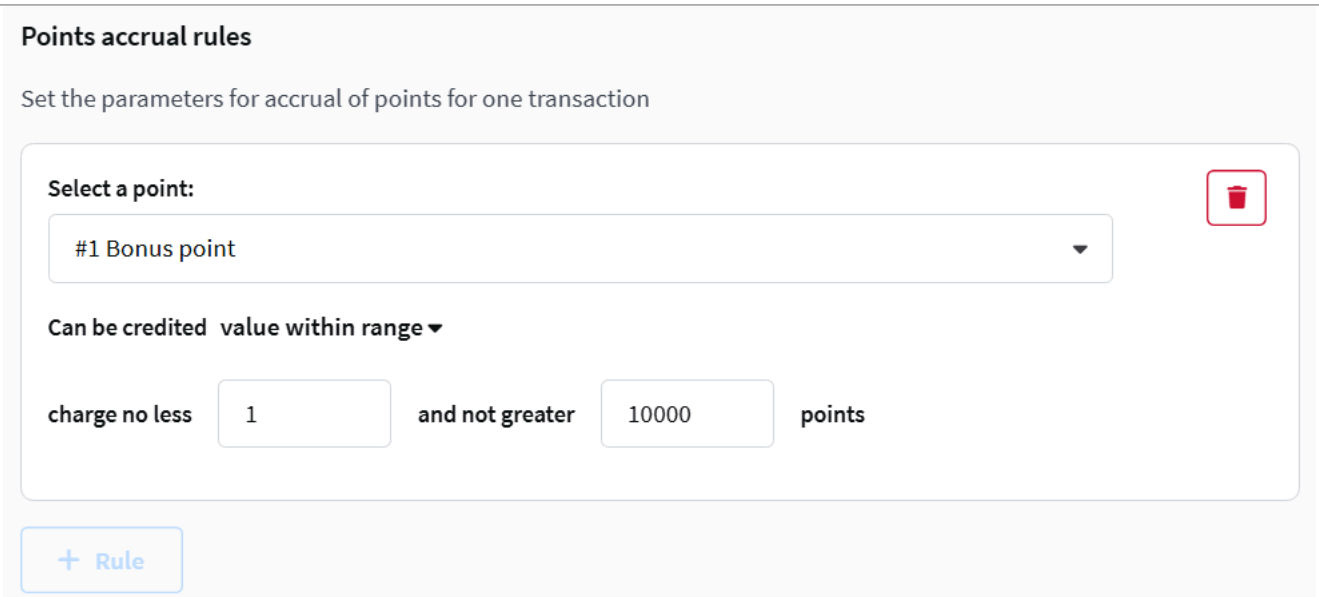
These parameters determine the balance with which a profile will be added to the program and how the for in the future:



A trigger promotion is used as a universal mechanism for accruing or redeeming points. Despite the nam own trigger — the moment of its execution is determined by the scenario.

Configure the point accrual rules.

- **Select the created point type;**
- **Set a range or a fixed amount:**



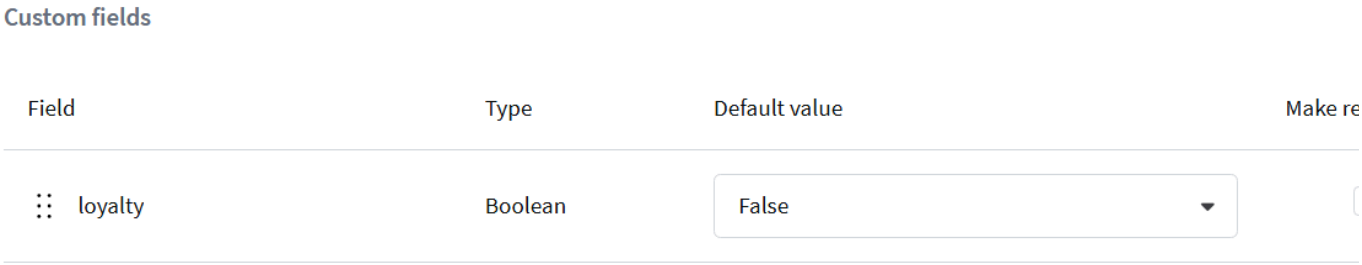
INFO

If a range is specified, the exact number of points to be accrued is determined in the scenario node set same trigger promotion for different cases, for example, awarding 500 points for downloading a mobile form, without creating separate promotions for each case.

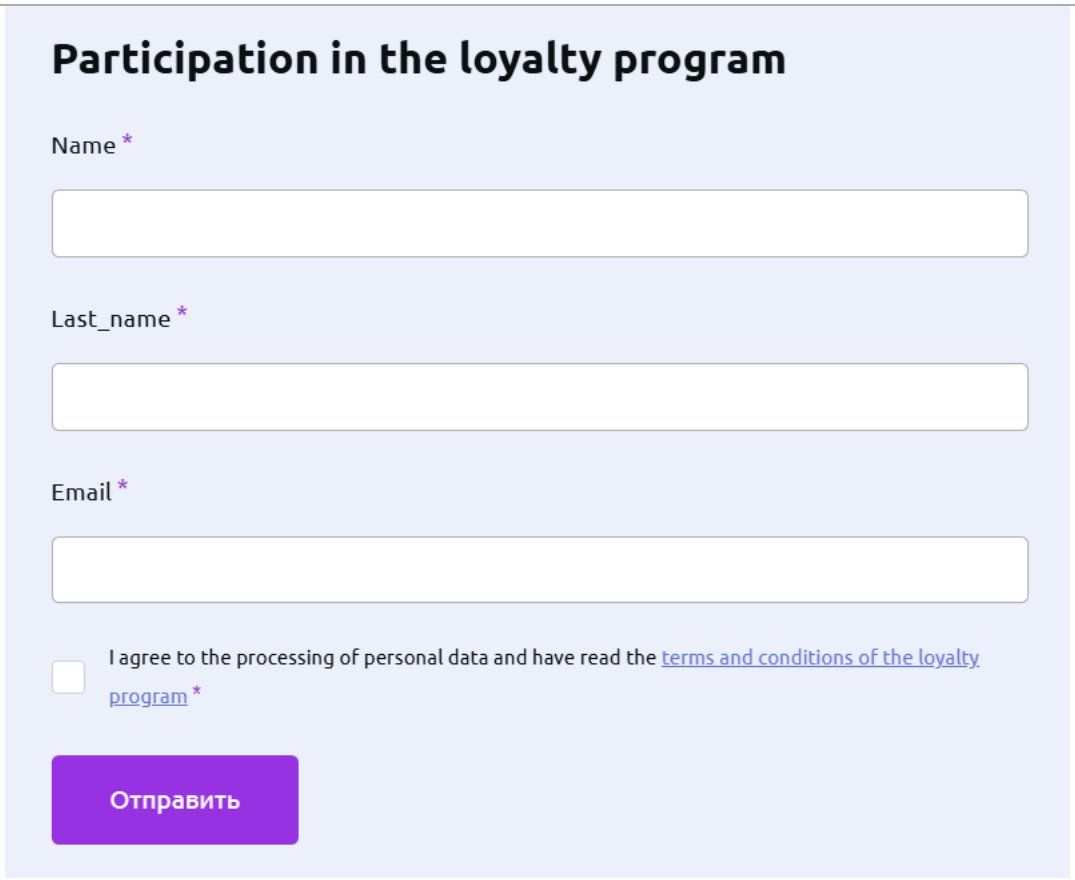
After saving and activating the loyalty program, the trigger promotion becomes available in automation s as a Loyalty points management node, and it is there that the exact moment of execution and the precis redeemed are determined.

Database and Form

In the Profile database, create an additional Boolean field to record the user's consent to participate in t

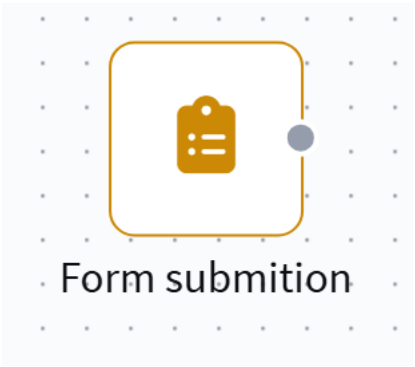


Then, create a Form and add this boolean field to it. It is recommended to make the field required so tha without confirming consent. After publishing the form, data from it will be saved to the user's profile, inc

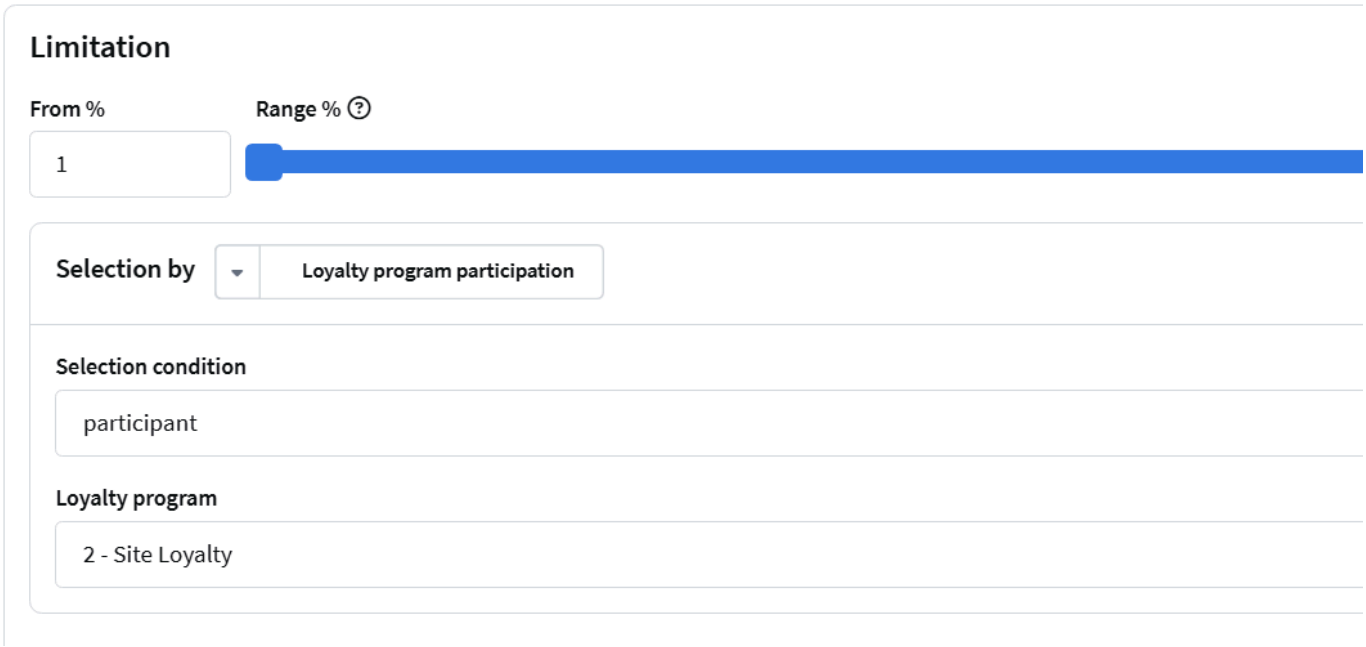


Scenario

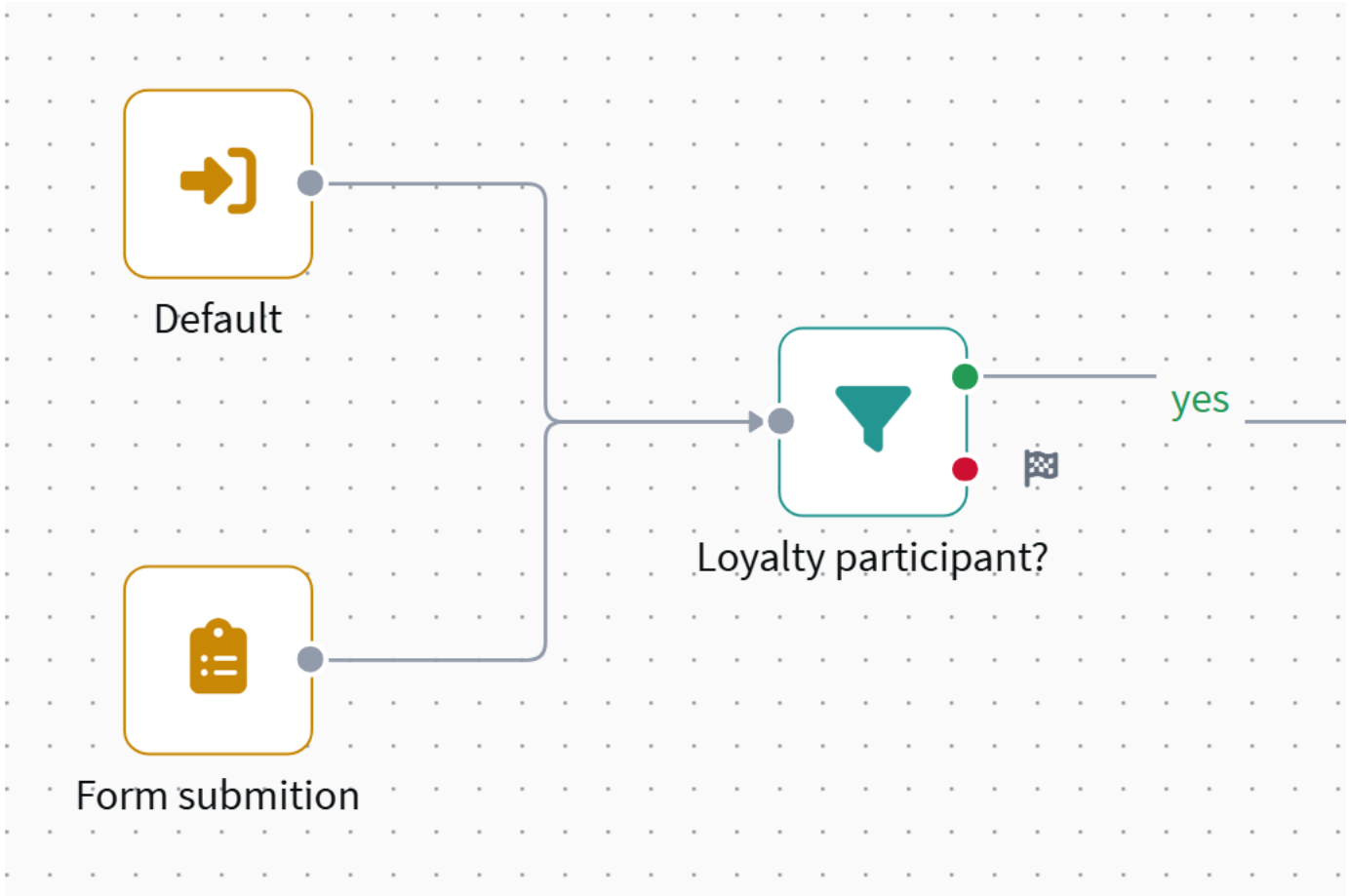
Once the loyalty program and trigger promotion are configured, you can proceed to create a scenario. In profiles to prevent the same profile from entering the scenario again. Select Form submission as the ent



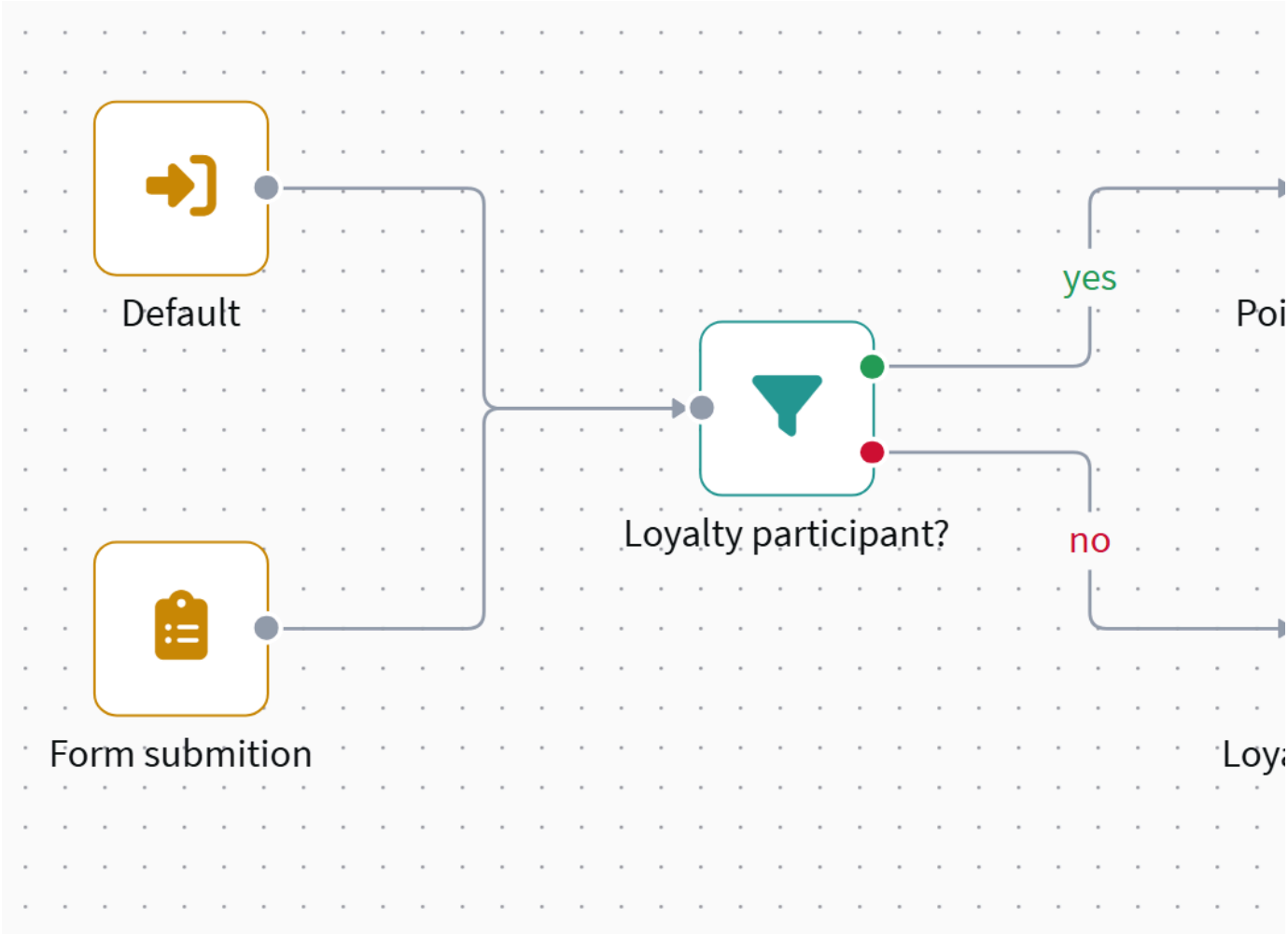
In the scenario, add a condition check element to verify the profile's participation in the loyalty program:



If the profile is already a member of the program, the “Loyalty participation” node is not used, and the tr accrue, for example, 500 bonus points for filling out the form:



If the profile is not yet a member of the loyalty program, the scenario adds the profile to the program. Si required, no additional consent check is needed:



When a profile is added to the program, points are accrued automatically — according to the set initial b

INFO: Please note

If the point settings specify a zero initial balance upon registration, you must add a Loyalty points mana previous one) to manually accrue the required number of points.

Automatically rewarding a profile for installing a mobile app

This case allows you to automatically add a user to the loyalty program or accrue points immediately aft This uses a pixel that triggers upon app installation.

INFO: Please note

If the initial balance in the point settings is zero, after the “Loyalty participation” node, you must also a node with a trigger promotion to award the bonus for downloading the app.

Loyalty

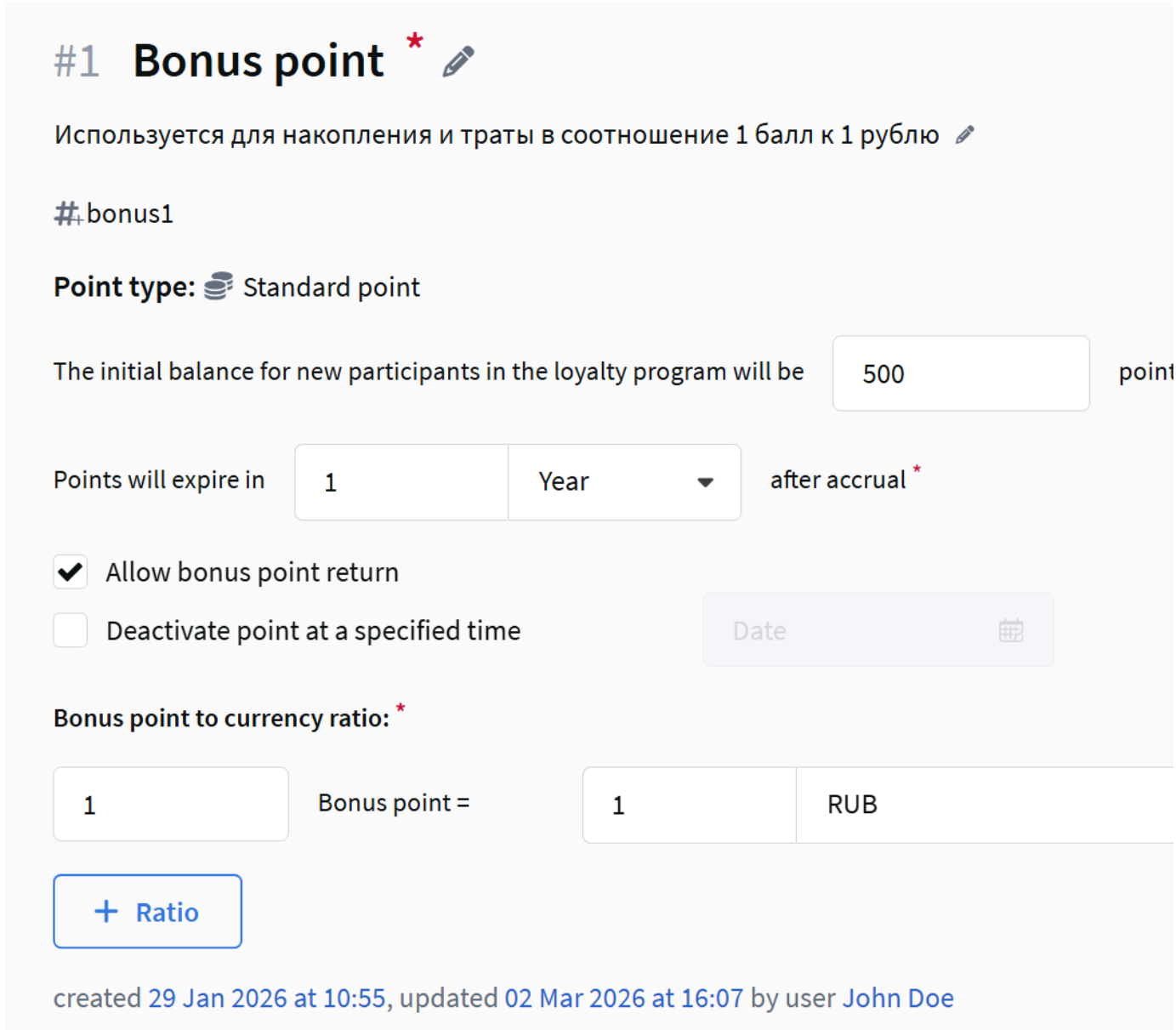
Before creating a scenario, you need to prepare the basic configuration of the loyalty program. In the Lo create the program itself.

Configure standard points. When creating standard points, you must fill in the required parameters:

- **point name and external identifier;**
- **initial balance for a new loyalty program member;**
- **points expiration period (after a specified number of days, months, or years after accrual);**

- **point-to-currency ratio.**

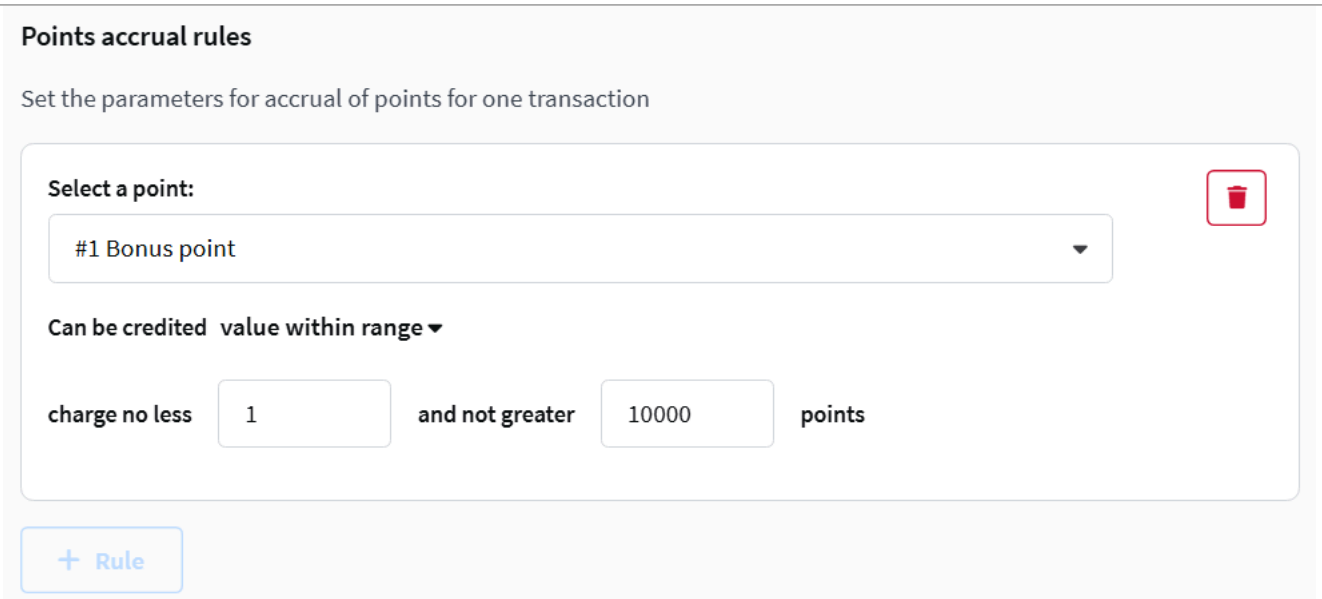
These parameters determine how many points a profile will receive after installing the app:



A trigger promotion is used as a universal mechanism for accruing or redeeming points. The moment of i scenario.

Configure the point accrual rules:

- **Select the created point type;**
- **Set a range or a fixed amount.**



INFO

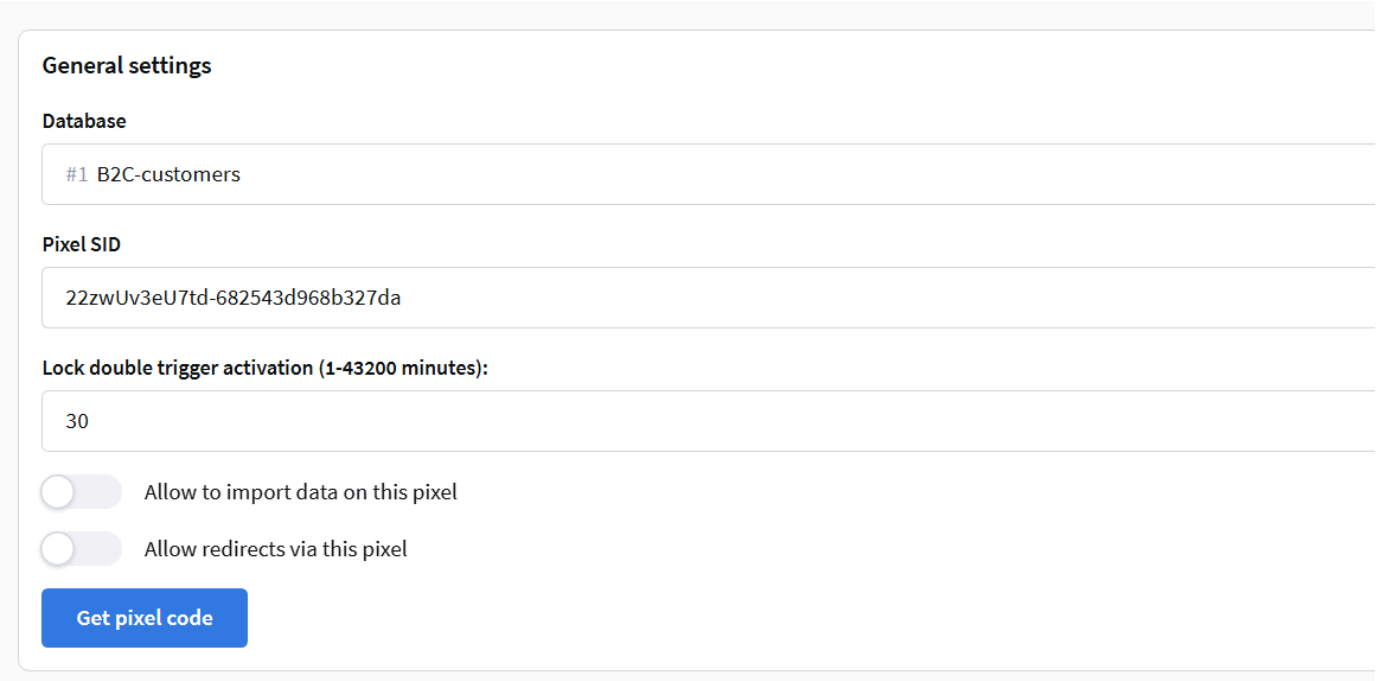
If a range is specified, the exact number of points to be accrued is determined in the scenario at the m allows you to use the same trigger promotion for different cases: for example, 300 points for installing i Android.

After saving and activating the loyalty program, the trigger promotion becomes available in automation s as a Loyalty points management node, where the moment of point accrual is determined.

Pixel

Create a pixel for the mobile application. The pixel is used for a single purpose — to record the fact of ap

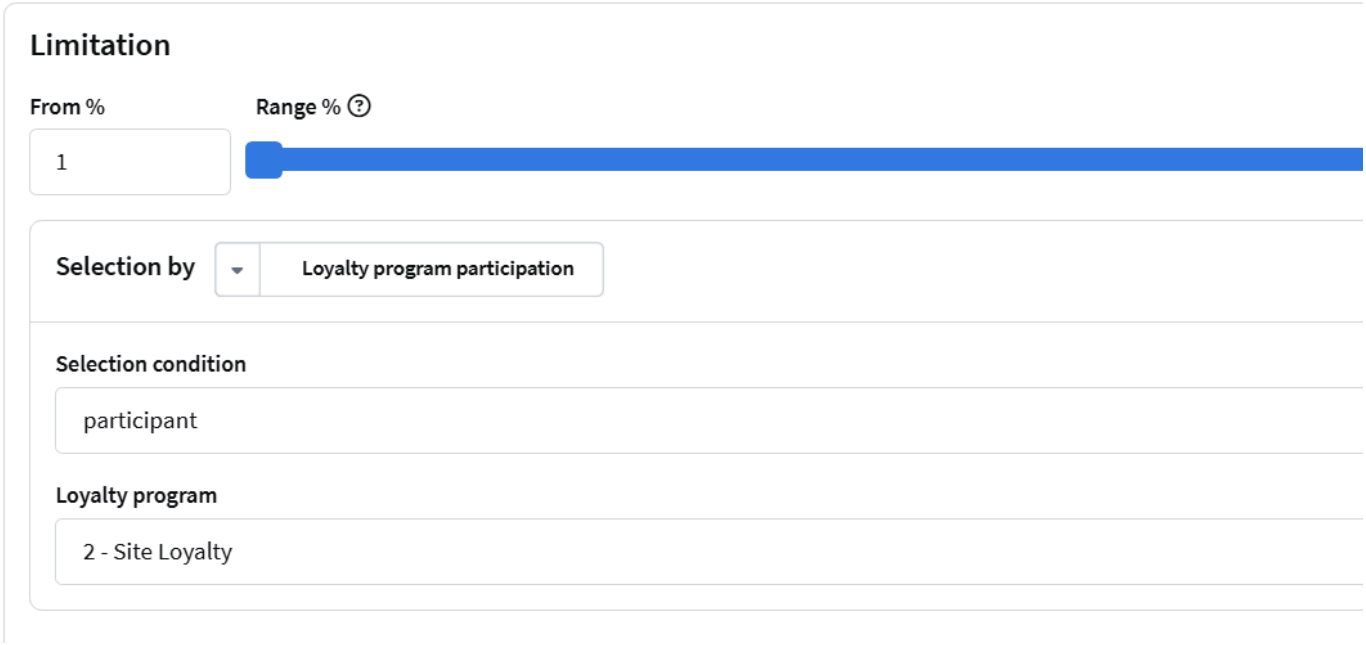
- **Event type: App install goal.**
- **It is activated automatically upon installation.**
- **For correct operation, ensure the pixel is integrated into the mobile application via mSDK.**



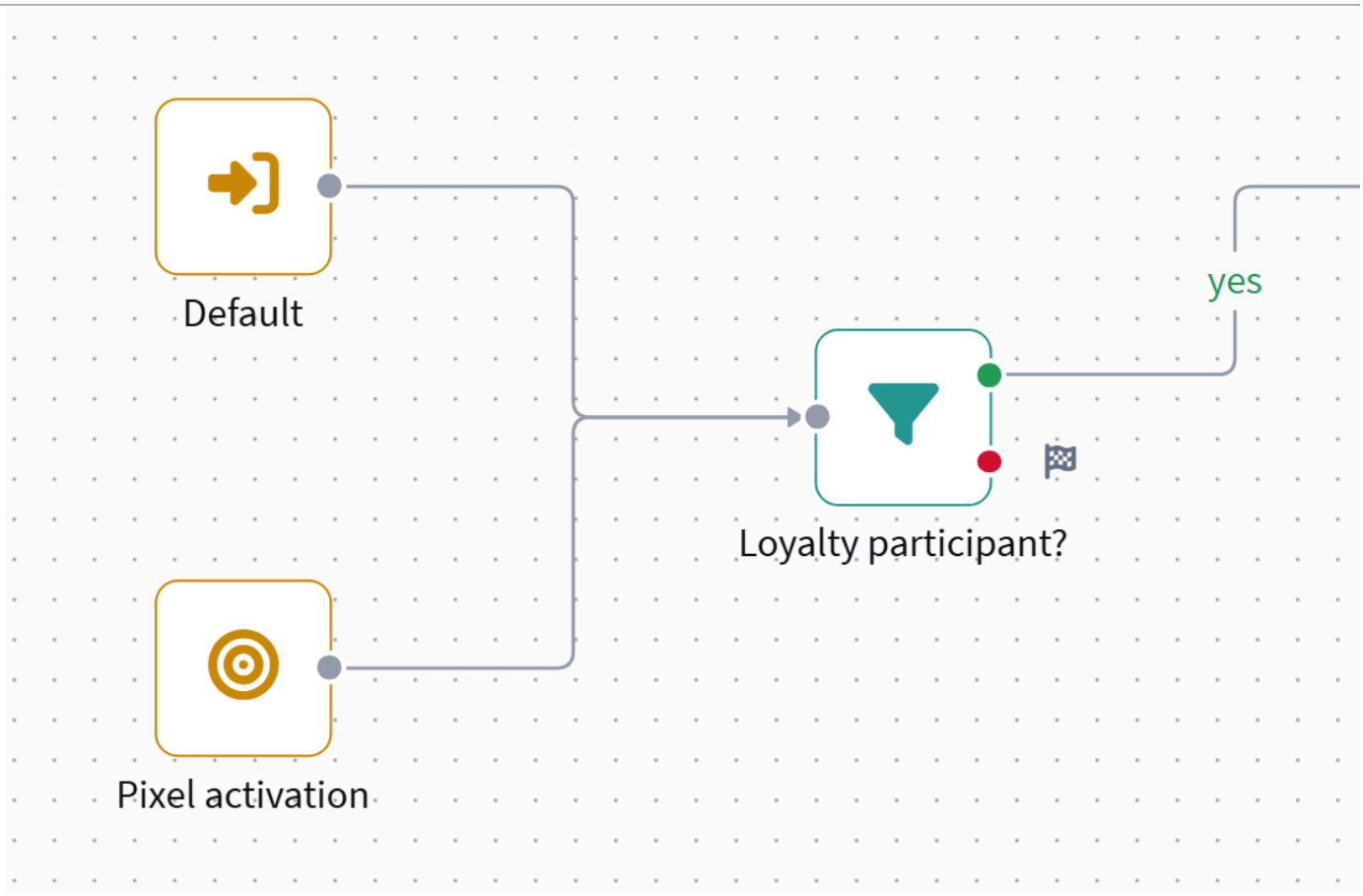
Scenario

Once the loyalty program and trigger promotion are configured, create a scenario. In the scenario, disabl same profile from entering the scenario again.
Select Pixel event — app install as the entry trigger:

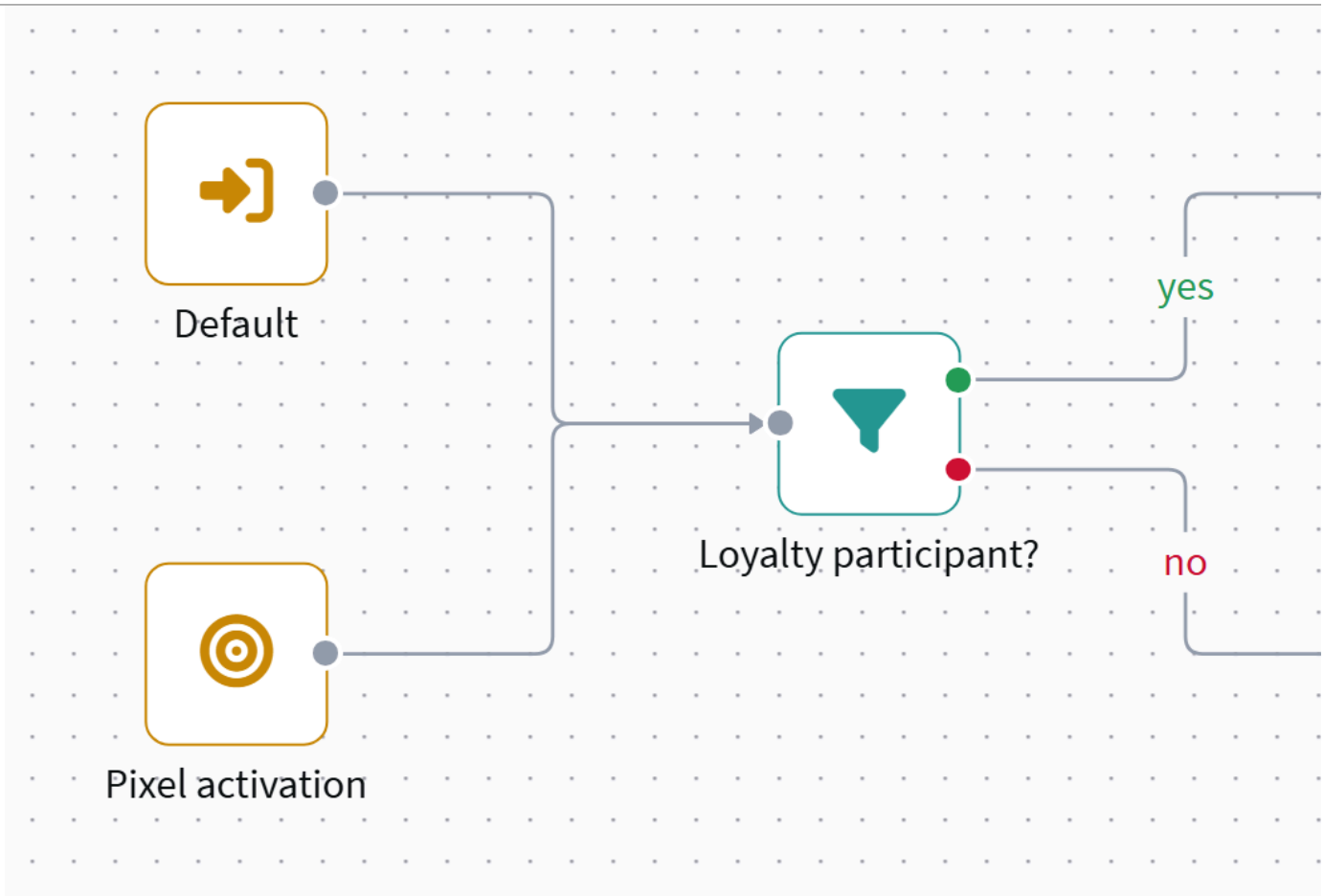
In the scenario, add a condition check element to verify if the profile is already a member of the loyalty p



If the profile is already a member of the program, the "Loyalty participation" node is not used, and the tr accrue points (for example, 500 points for installing the app):



If the profile is not yet a member of the loyalty program, the scenario adds the profile to the program. Af automatically accrued due to the configured initial balance of the standard points.



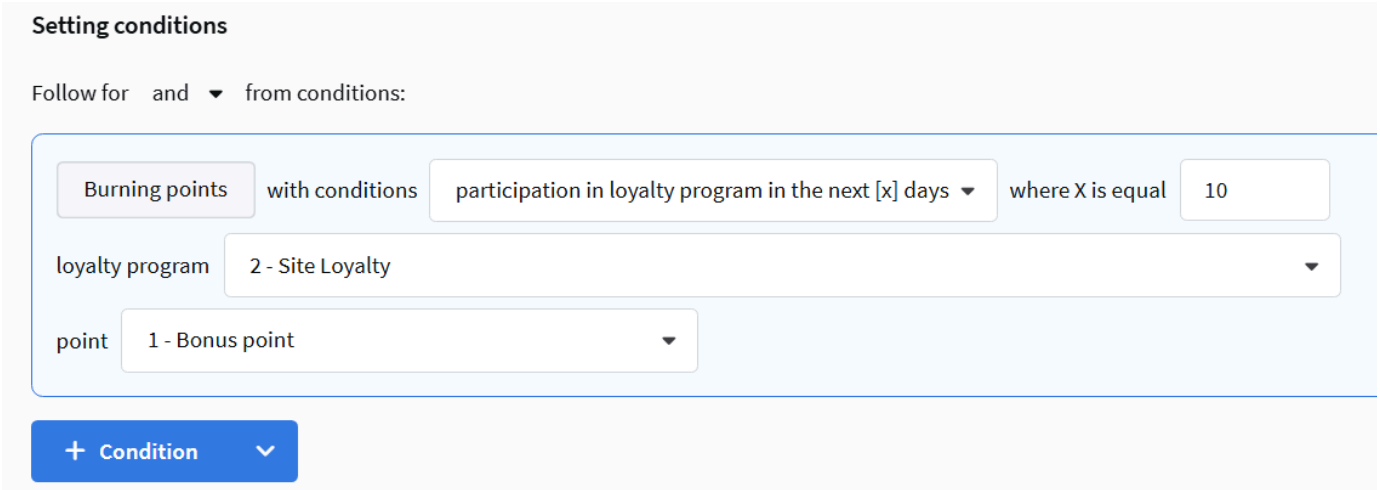
Notification about imminent loyalty points expiration

In this case, users receive a Notification that their loyalty program points are about to expire. This uses profiles with the soonest point expiration, and a regular campaign executed on a schedule.

Segment

Create a dynamic segment for profiles whose points are about to expire soon.
In the segment conditions, use the filter Interaction with entities — Loyalty — Points expiration

- **Condition: in the next [X] days**
- **Value X:** `10`
- **Loyalty program: select the desired program**
- **Points: specify the required point type**



Such a segment automatically includes profiles that have points of the specified type expiring within the

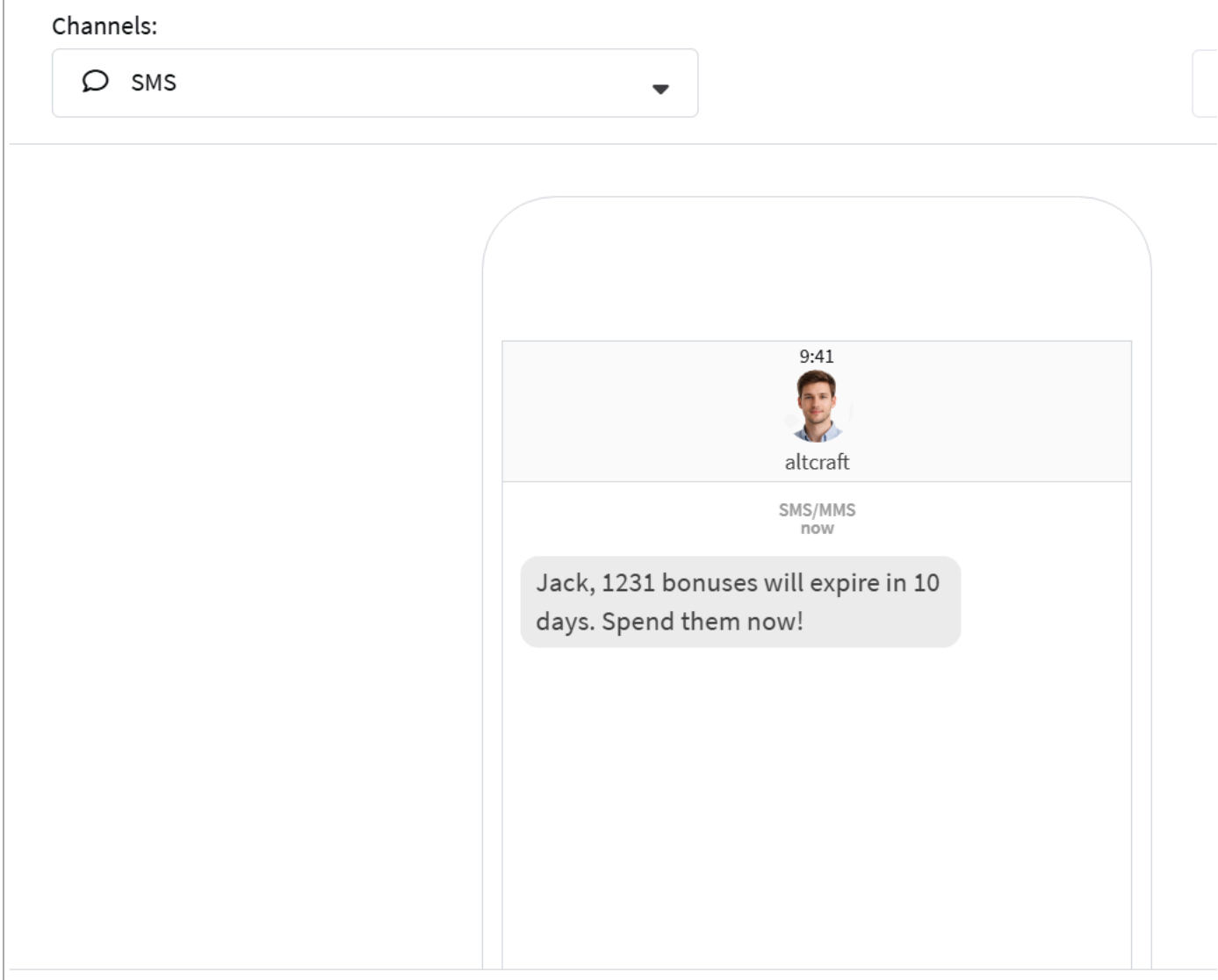
SMS Template

Create an SMS message template that will display information about the imminent points expiration.
Use the variable `{loyalty_expiration_points(<Loyalty_Program_ID> "<Point_SID>" x y "desc")}` **to output the data.**
The variable returns information about the nearest expiration of points for the selected currency:

- `<Loyalty_Program_ID>` **— loyalty program identifier;**
- `<Point_SID>` **— point SID;**
- `x` **— points validity period;**
- `y` **— transaction count limit;**
- `"desc"` **— sort by transaction date (descending).**

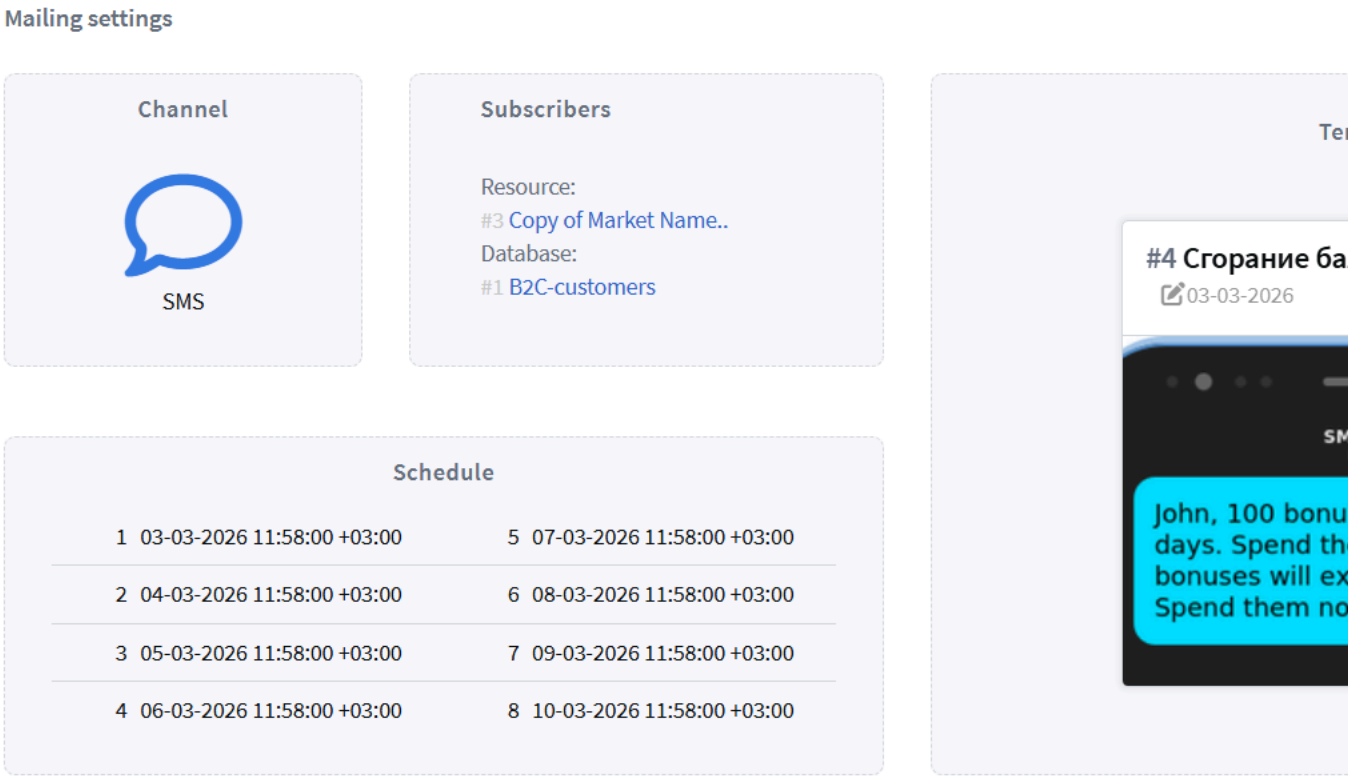
Example SMS text:

```
{for $i $p = loyalty_expiration_points(2 "bonus1" 10 1 "desc")}{lead._fname}, {$p.Amount} bonus points will expire in 10 days. Spend them now
{end}
```



Mailing

Create a regular maling, select the created dynamic segment as the audience. Choose the SMS template information. Configure the schedule (e.g., “Every day at selected time” “Send in the profile’s timezone if



After activating the campaign, the system will send notifications daily only to those profiles whose point

INFO

Using a dynamic segment and a regular campaign allows you to notify users about points expiration wit Notifications are sent only to the current audience and automatically stop when the conditions change.

Order Segments

Order segments are a tool for fine-tuning the conditions under which benefits are applied in discount and Programs module.

Order segments work similarly to dynamic segments: you define a set of conditions, and compliance wit time when benefits are applied to an order. This makes it possible to build flexible marketing mechanics number of items in the cart or special offers for specific regions or sales channels.

TIP: Use case example

For example, a marketer wants to launch a promotion that gives a 10% discount only for orders placed they create an order segment where in the Sales channels section they select the is in list operator, sp the Website cart contact point.

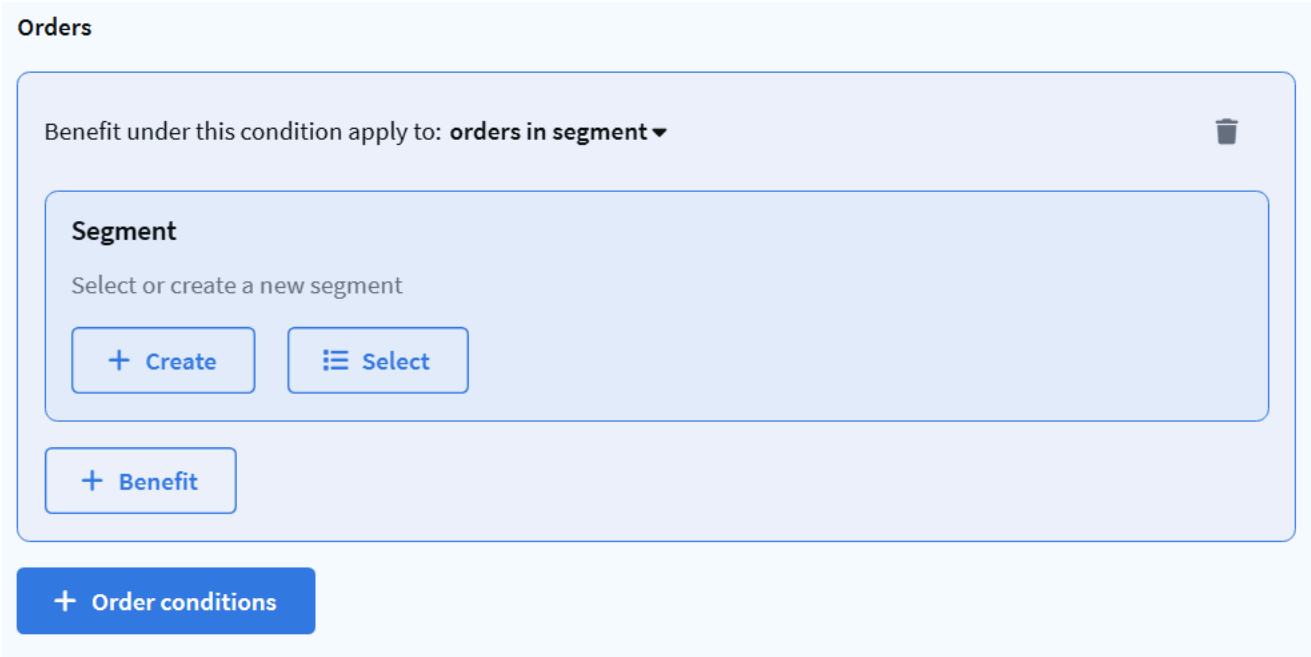
Then, in the discount promotion settings, in the Orders section, they select orders in segment from the segment.

As a result, the 10% discount will be applied only to orders placed through the website cart. Orders fro stores or phone orders) will not receive this discount.

Configuring an order segment

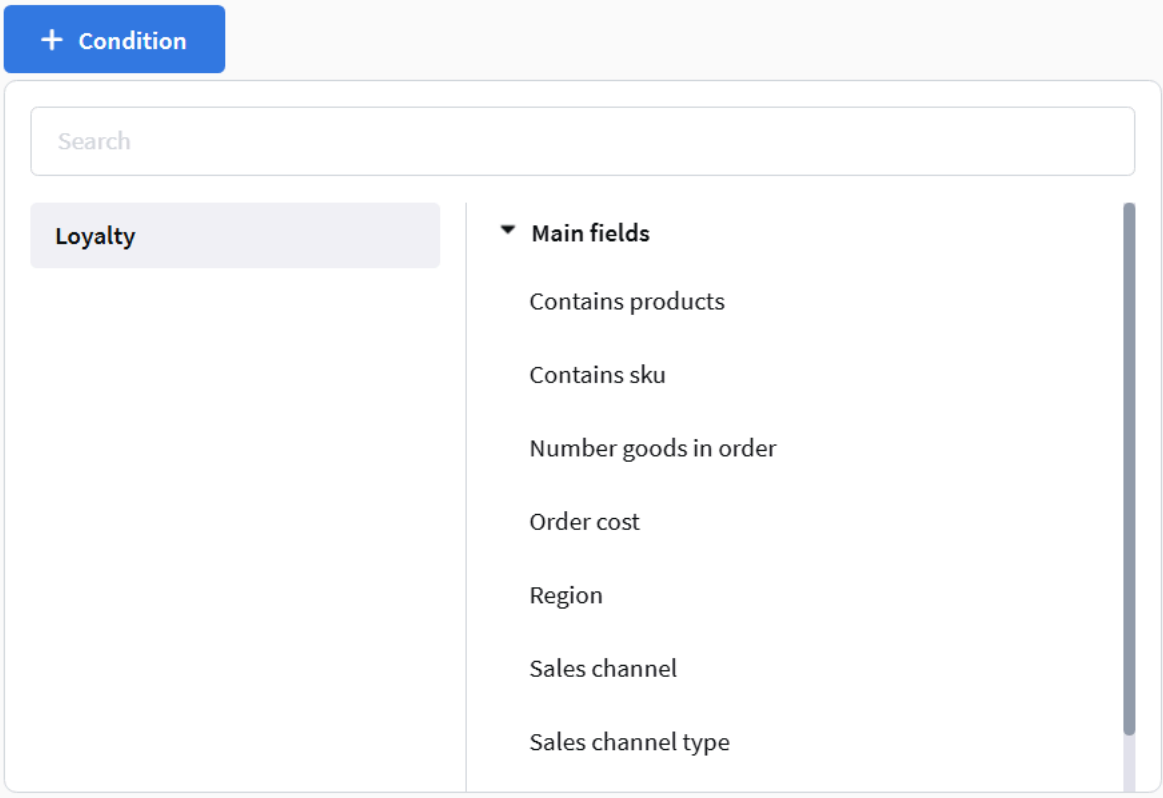
To use an order segment in a promotion, go to the Orders section during promotion setup and select ord Then click Select segment.

In the window that opens, you can select an existing segment or create a new one by clicking + Create.



Creating a new segment

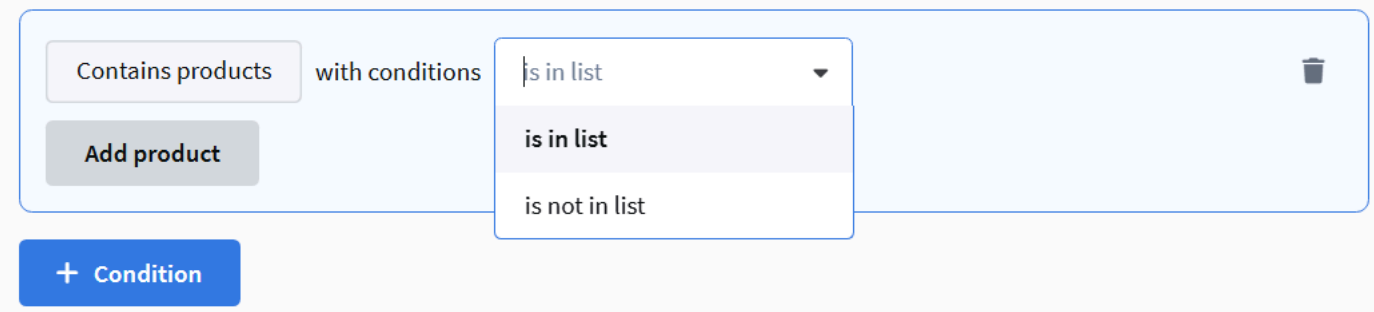
When creating a segment, specify its name and, if needed, a description. Then configure the conditions



The following conditions are available:

Contains products

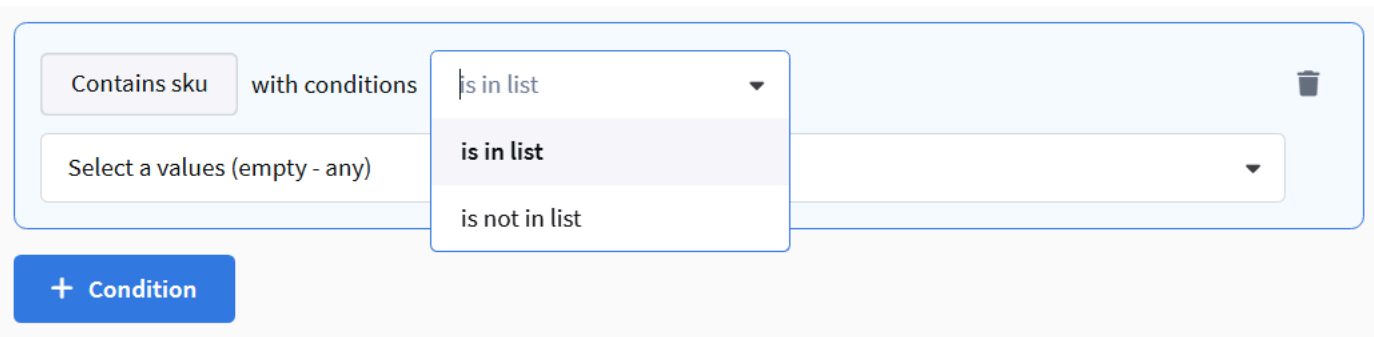
Checks whether the order contains specific products from a list.



- **is in list — benefits are applied if the order contains products that are included in the selected list;**
- **is not in list — benefits are applied if the products in the order are not included in the selected list.**

Contains SKU

Checks whether the order contains SKUs from a list.

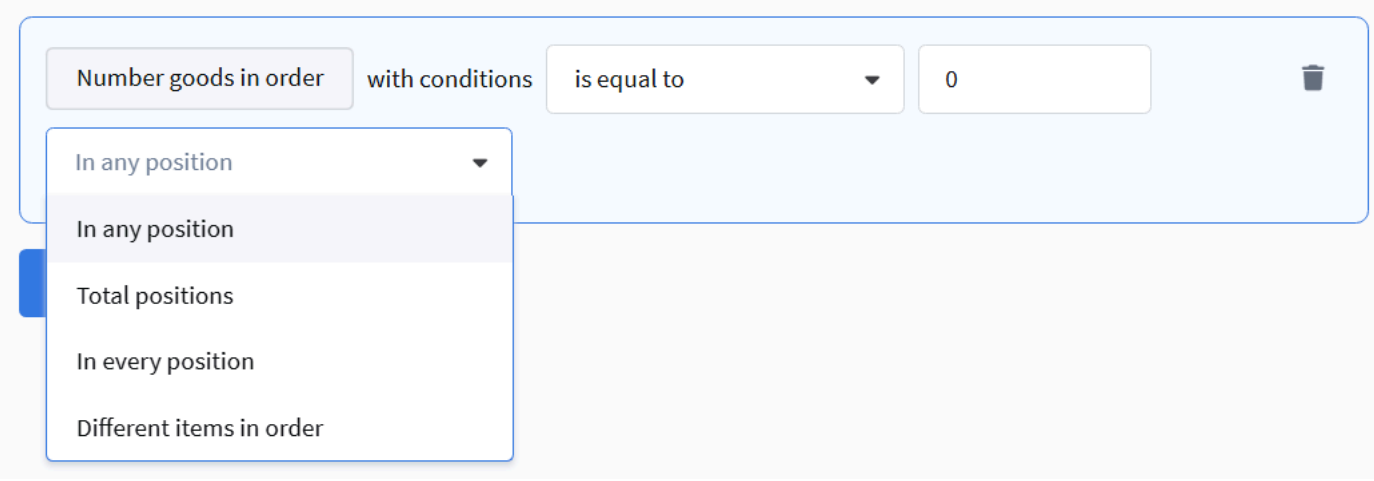


- **is in list — benefits are applied if the order contains SKUs included in the selected list;**
- **is not in list — benefits are applied if the order does not contain SKUs included in the selected list.**

Number of goods in order

Checks the number of goods in the order. Available calculation methods:

- **In any position — checks if there is at least one line item with the specified quantity.**
- **Total positions — checks the total quantity of goods across all line items.**
- **In every position — checks that each line item has the specified quantity.**
- **Different items in order — checks the number of unique products in the order.**

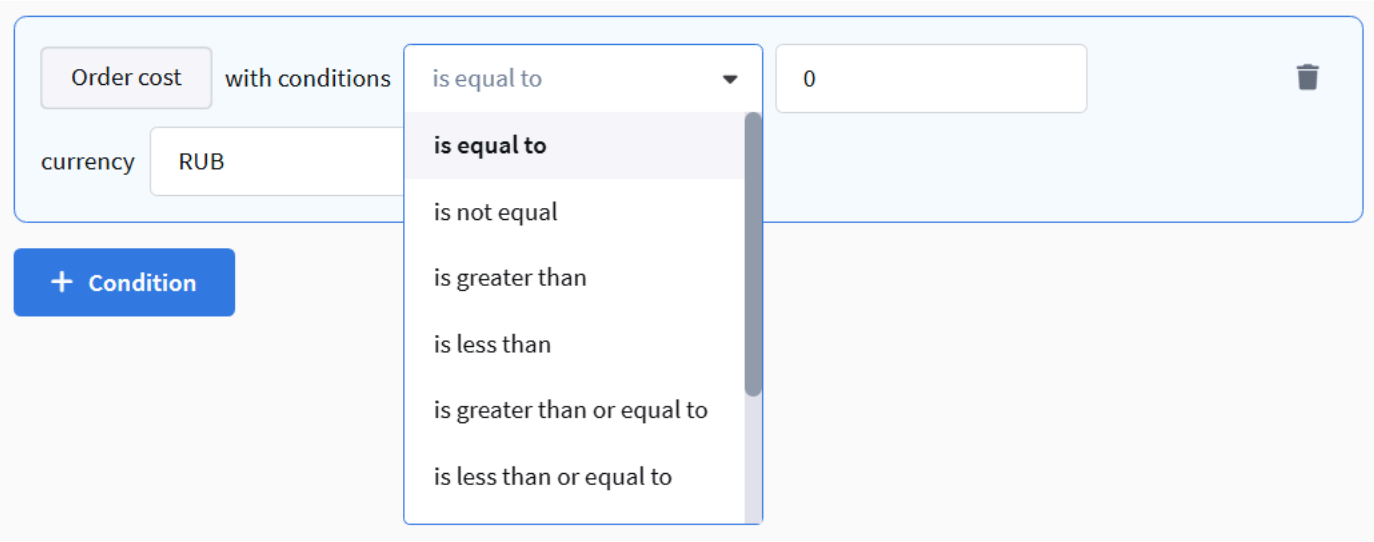


For the selected calculation method, specify an operator and a value:

- **is equal to — the quantity matches the specified number.**
- **is not equal — the quantity does not match the specified number.**
- **is greater than — the quantity exceeds the specified number.**
- **is less than — the quantity is less than the specified number.**
- **is greater than or equal to — the quantity exceeds or equals the specified number.**
- **is less than or equal to — the quantity is less than or equals the specified number.**
- **in the range — the quantity is within the specified range (inclusive).**

Order cost

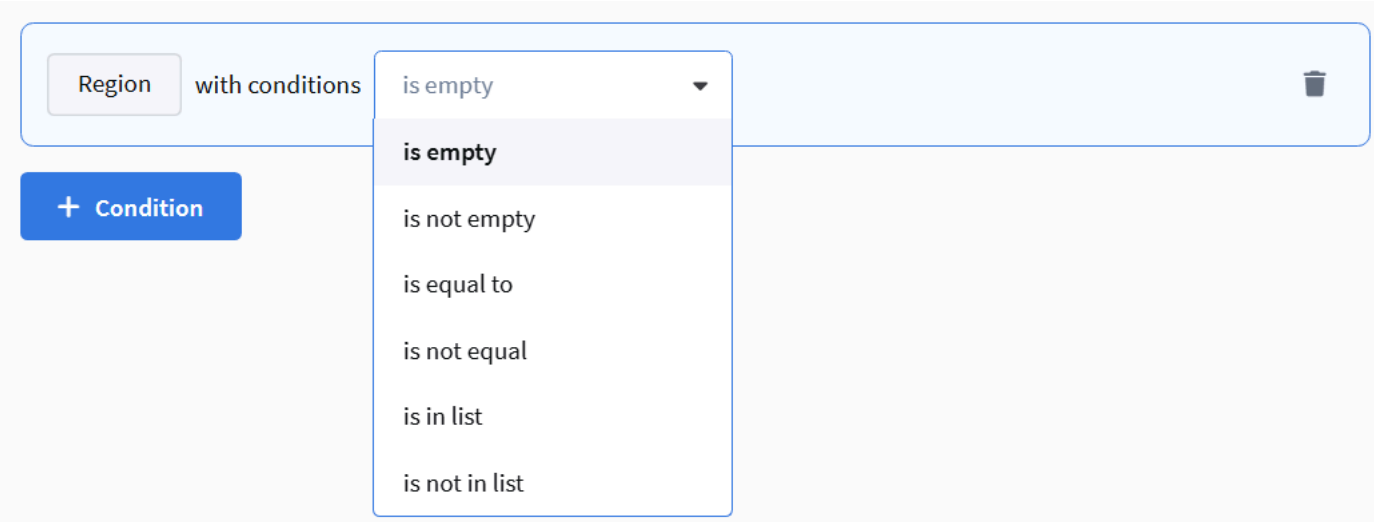
Checks the order cost. Specify an operator, amount, and currency:



- **is equal to — the order cost exactly matches the specified amount.**
- **is not equal — the order cost does not match the specified amount.**
- **is greater than — the order cost exceeds the specified amount.**
- **is less than — the order cost is less than the specified amount.**
- **is greater than or equal to — the order cost exceeds or equals the specified amount.**
- **is less than or equal to — the order cost is less than or equals the specified amount.**
- **is in list — the order cost matches one of the values in the list.**
- **is not in list — the order cost does not match any value in the list.**
- **in the range — the order cost is within the specified range (inclusive).**

Order region

Checks the order region. Available operators:

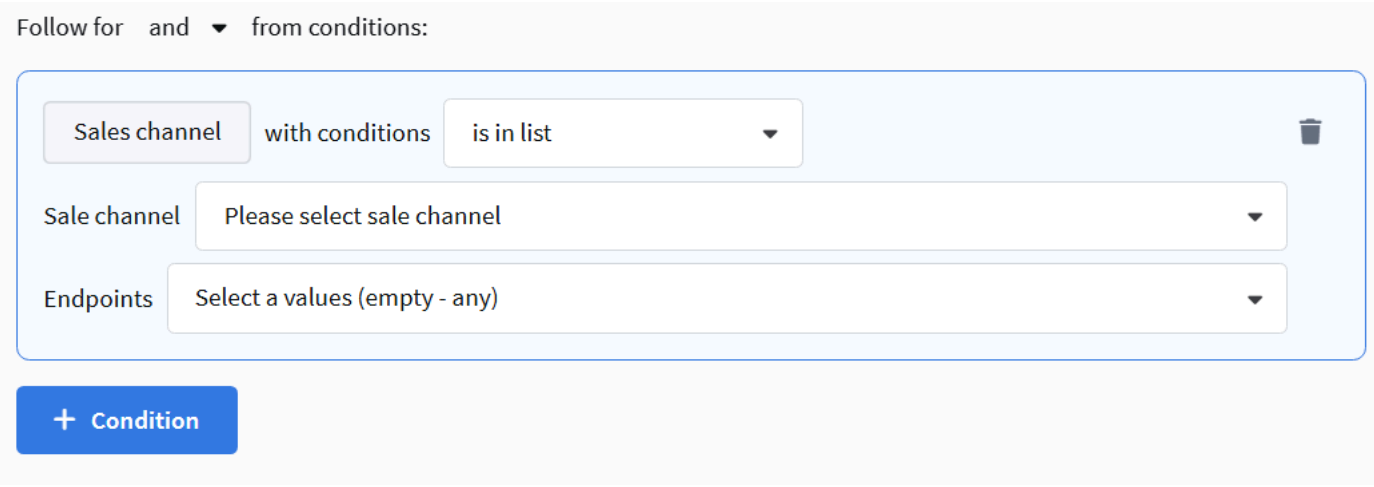


- **is empty — no region is specified for the order.**
- **is not empty — any region is specified.**
- **is equal to — the region matches the selected value.**
- **is not equal — the region does not match the selected value.**
- **is in list — the region is included in the selected list.**
- **is not in list — the region is not included in the selected list.**

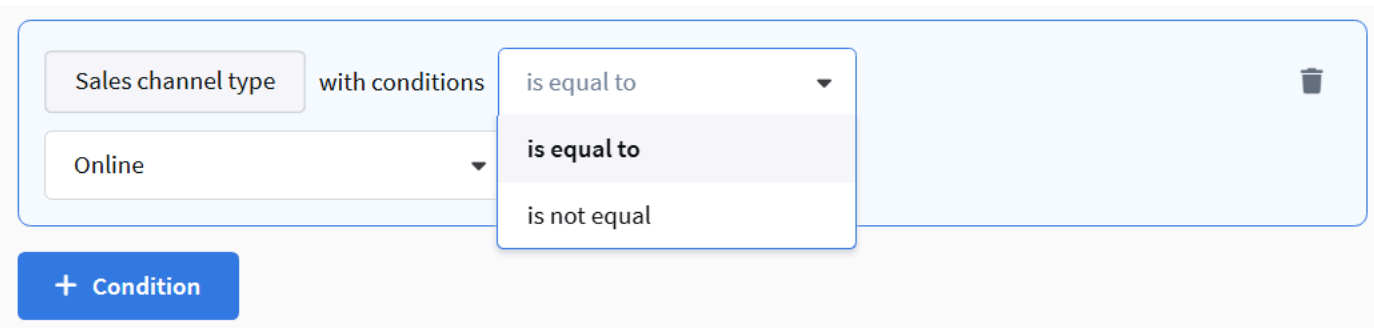
Sales channels

Checks through which sales channel and contact point the order was placed.

- **is in list — benefits are applied if the order was placed through one of the selected sales channels a**
- **is not in list — benefits are applied if the order was not placed through the selected sales channels a**



Sales channel type

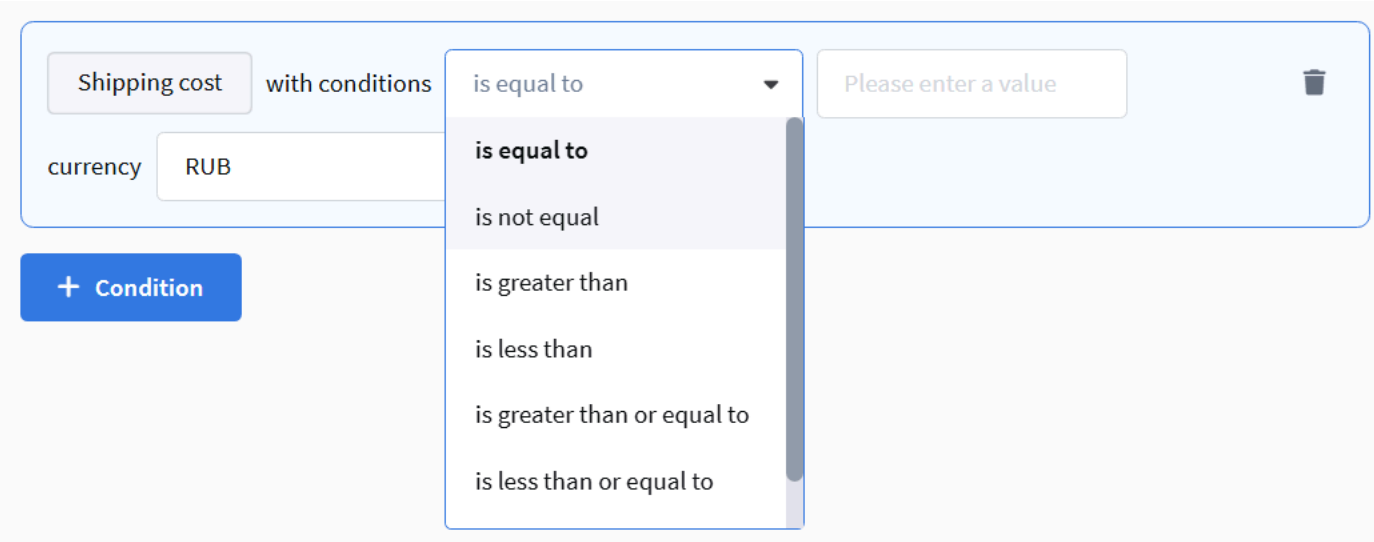


Checks the sales channel type (Online / Retail). Available operators:

- **is equal to — the sales channel type matches the selected value.**
- **is not equal — the sales channel type does not match the selected value.**

Shipping cost

Checks the order shipping cost. Specify an operator, amount, and currency:



- **is equal to — the shipping cost exactly matches the specified amount.**
- **is not equal — the shipping cost does not match the specified amount.**
- **is greater than — the shipping cost exceeds the specified amount.**
- **is less than — the shipping cost is less than the specified amount.**
- **is greater than or equal to — the shipping cost exceeds or equals the specified amount.**
- **is less than or equal to — the shipping cost is less than or equals the specified amount.**
- **in the range — the shipping cost is within the specified range (inclusive).**

Promotion codes

Description

You can create and manage customizable promotion codes to lead efficient loyalty programs using Altcr can be used in variety of cases where you want to give your customers personal offers or set limited acc

- **Personalpromo code can be attached to certain customer profile for personal offers.**
- **Universal promo code can be attached to several profiles, It can be used to organize massive market**

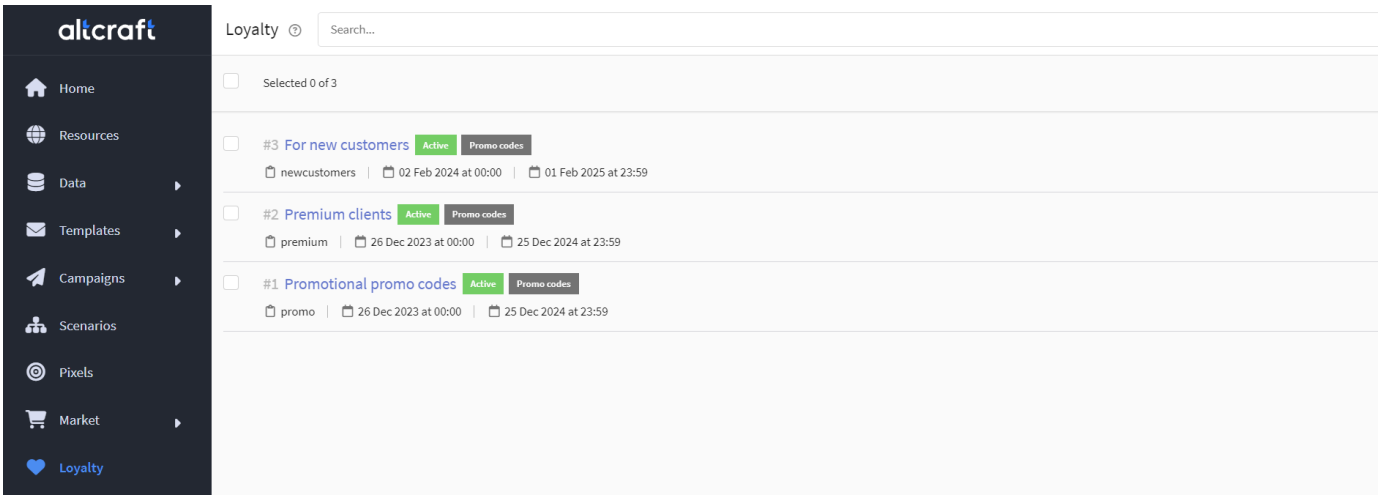
Promo codes can be attached to customers manually or automatically within mailings and workflows. Pr program interface, or they can be imported from file.

> **TIP**
>
> **You can use API for advanced promo codes configuration. For example, you can track customers who u documentation is available there.**

Managing promo codes

Go to the Loyalty section in the main menu to open the list of existing loyalty programs. Using context m or deactivate, and view the list of promo codes created in a loyalty program. To edit a program, click on

To create a new program, click + Create on the top panel.



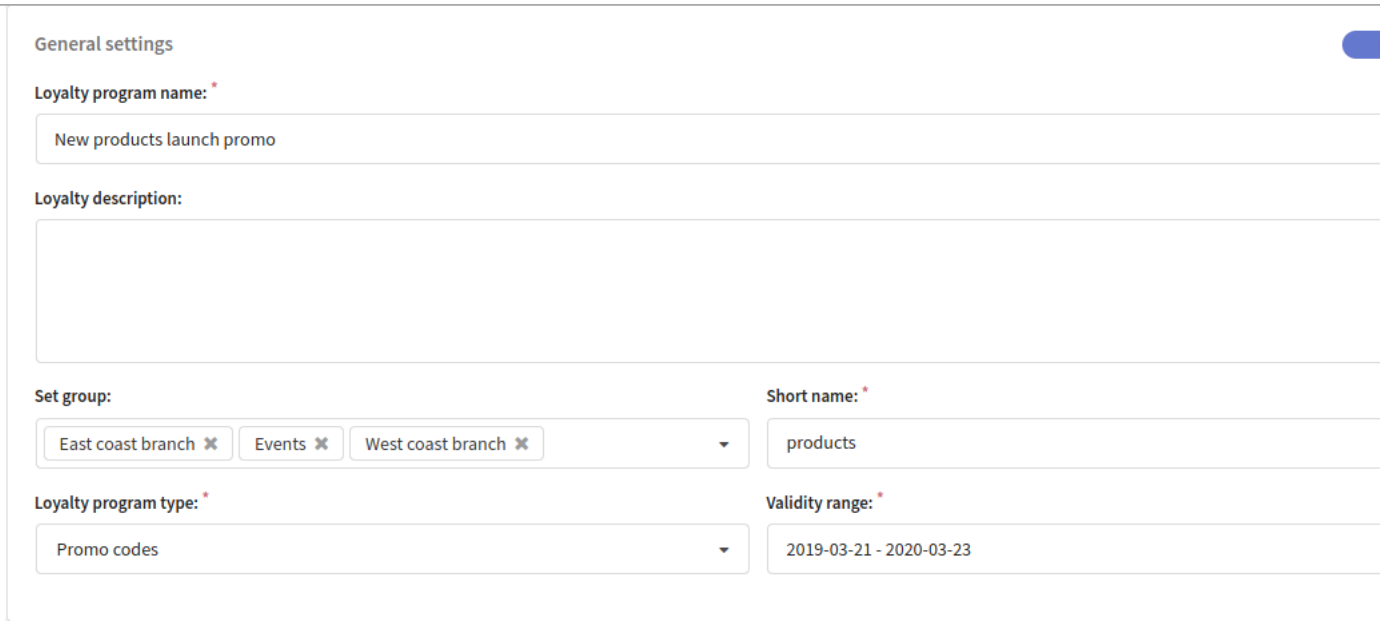
Promo code data

Each promo code contains the following information:

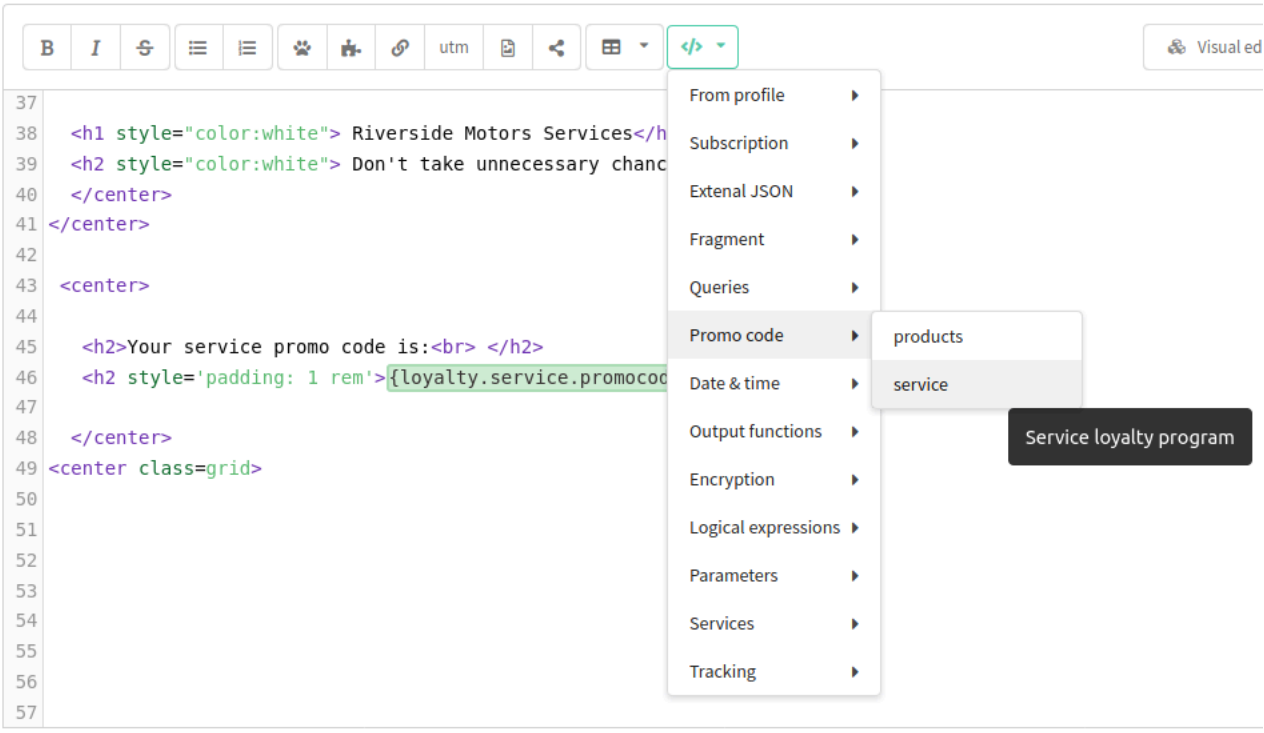
- **Promo code creation date, loyalty program ID**
- **Promo code attachment sign and date**
- **Promo code life time**
- **Promo code activation sign and date**
- **Additional data fields**

General settings

In general settings, enter loyalty program name, its short name, and set access group if necessary.



Loyalty program can be used in templates by its short name in variables menu or macro `{loyalty.name.prom`



Loyalty program validity range defines time when promotion codes can be sent to your customers. If a c messages containing it won't be sent to recipients. This parameter does not affect promo codes that hav customers. Select dates range and press Apply button.

> **TIP**
>
> **At the moment, only one loyalty program type is available - promo codes.**

You can import your own custom promotion codes by pressing Import from file button or via a special AP promo codes right away in Altcraft platform.

Once imported or generated the are processed in a same manner. In case you generate promo codes - se available.

Generating promo codes

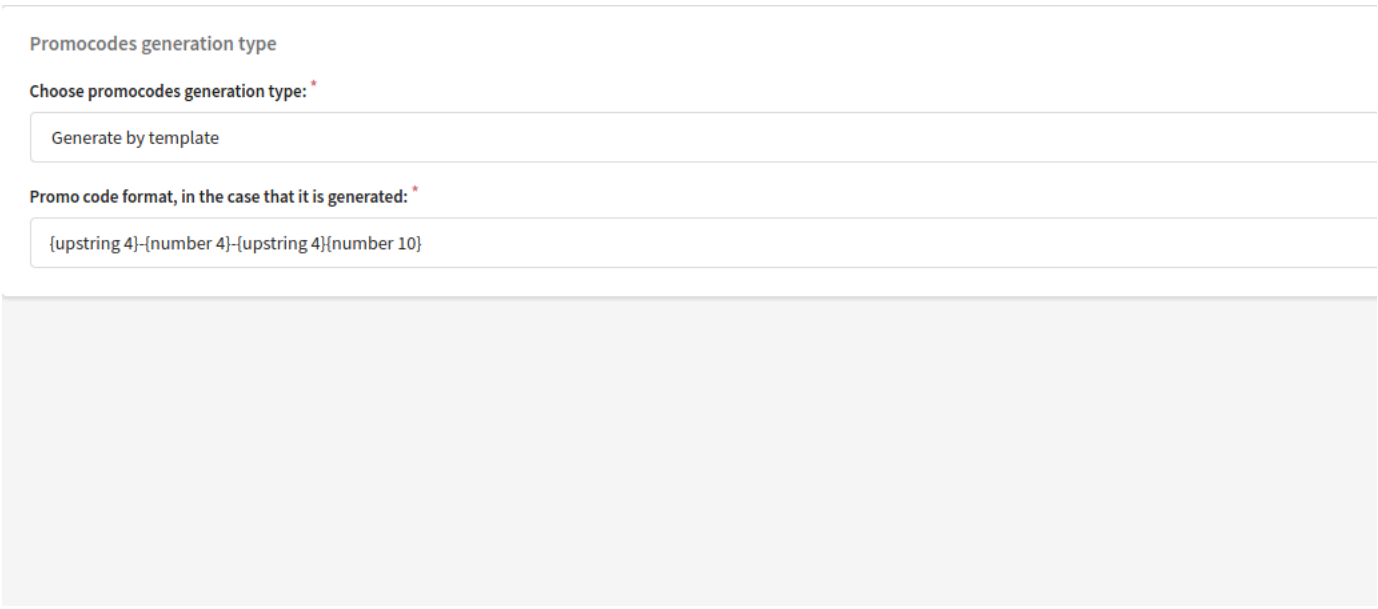
> **CAUTION**
>
> **If you plan to import promo codes from file, you can skip this paragraph.**

You can configure promo codes generation in Promocodes generation type section.

Generating by template

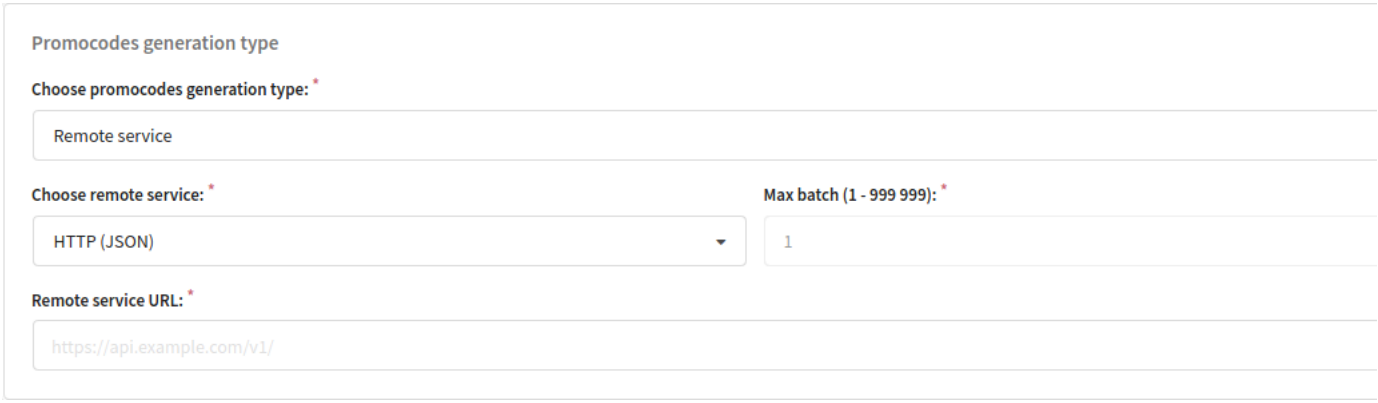
Altcraft Platform has its own tool to generate promo codes, which you can use by selecting Generate by useful if you do not have your own system for configuring promo codes, or promo codes list.

Select symbols combination you want to use from the `</code>` drop-down menu:



Generating by remote service

An alternate way to generate promo codes is a request to external service, which will create promo cod option in the drop-down menu:



Enter service URL and max promo codes batch in one query. POST query with JSON data of the followin

```
{
    account_id: Number (Account identifier),
    loyalty_id: Number (Loyalty program identifier),
    loyalty_name: Line (Loyalty program name),
    loyalty_shortname: Line (Loyalty program short name),
    campaign_id: Number (Mailing identifier, if it requests promo codes),
    campaign_name: Line (Mailing name, if it requests promo codes),
    workflow_id: Number (Workflow identifier, if it requests promo codes),
    workflow_name: Line (Workflow name, if it requests promo codes),
    count: Number (Amount of promo codes to be generated),
    is_test: Boolean (Test condition),
    fields: {
        <Field name>: <Value>
    }
}
```

If promo codes are requested by mailing or workflow, their identifiers will be given in request. If `is_test` used only for preview. All loyalty program fields will be given with the request.

Promo codes list in JSON format will be returned:

```
{
    codes: Lines array (promo codes)
}
```

If it is necessary to return promo codes with custom fields values, promo codes must be returned as obj

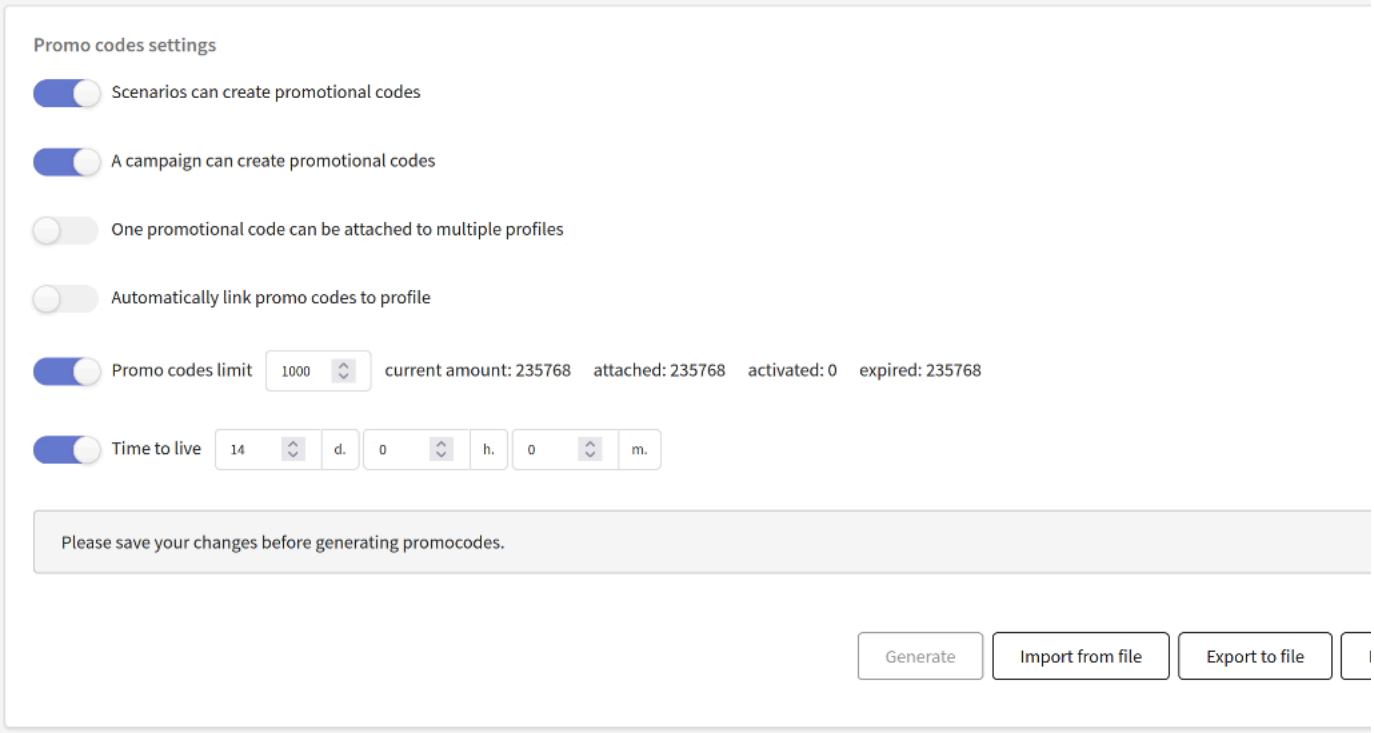
```
{
    codes: Objects array {
        code: Line (promo code),
        fields: {
            <Field name>: <Value>
        }
    }
}
```

TIP

You can use Generate button in Promo code settings block to check generating service. Here, you can a view promo codes.

Promo codes settings

You can configure the loyalty program to match your business logic in the Promo Code settings block. In provides commands for generating, importing, exporting, and viewing promo codes.



The following settings are available:

- **A workflow can create promotional codes — Workflow can create promo codes if there is not enough generated according to formatting scheme.**
- **A mailing can create promotional codes — Mailing can create promo codes if there is not enough co generated according to formatting scheme.**

CAUTION

When testing a mailing, the loyalty program generates a new promo code instead of using an existing o

- **One promotional code can be attached to multiple profiles — With this option, you can use one prom mailing or scenario in this case will assign the first active promotional code of the loyalty program.**
- **Automatically link promo codes to profile — When generating or importing from a file, promotional co profiles.**
- **Promo codes limit — Limit can be set for total promo code count**
- **Time to live — Time can be set for promo codes to be alive. When a specified time after generation r automatically get expired status.**

CAUTION

Promo code lifetime can be more than loyalty program validity range. If loyalty program is finished, pro sent to customers, but previously sent codes can be used by customers until their lifetime is not finishe

Automatically link promo codes to profile

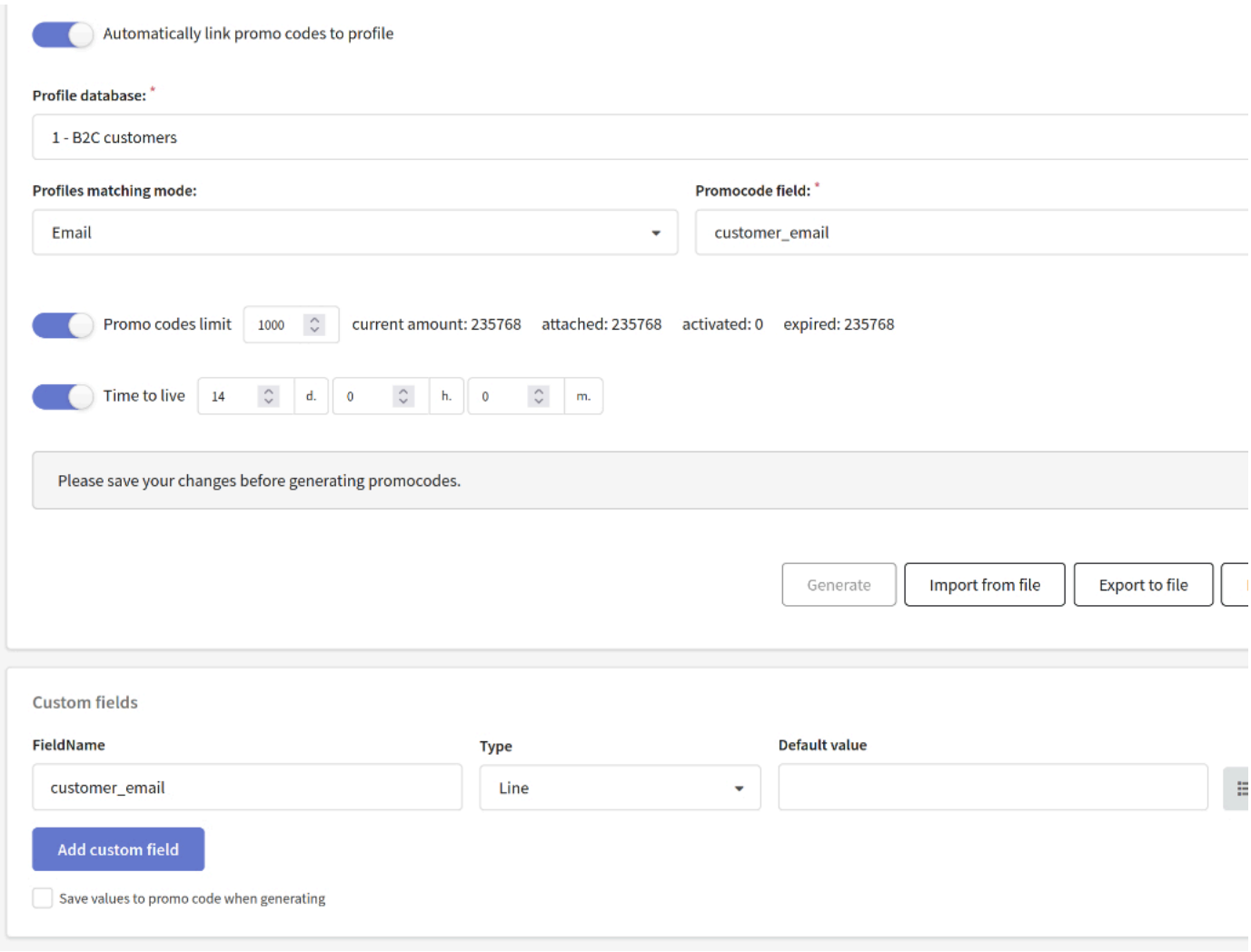
Select the database with customers to whom you want to assign promotional codes. Next, specify the pr promocode field. The matching mode determines what data the platform will use to search for profiles. T

- **Email**
- **Phone**
- **By custom field**

If the value in the promo code field matches the value of the field specified as the matching mode, the pl the found profile.

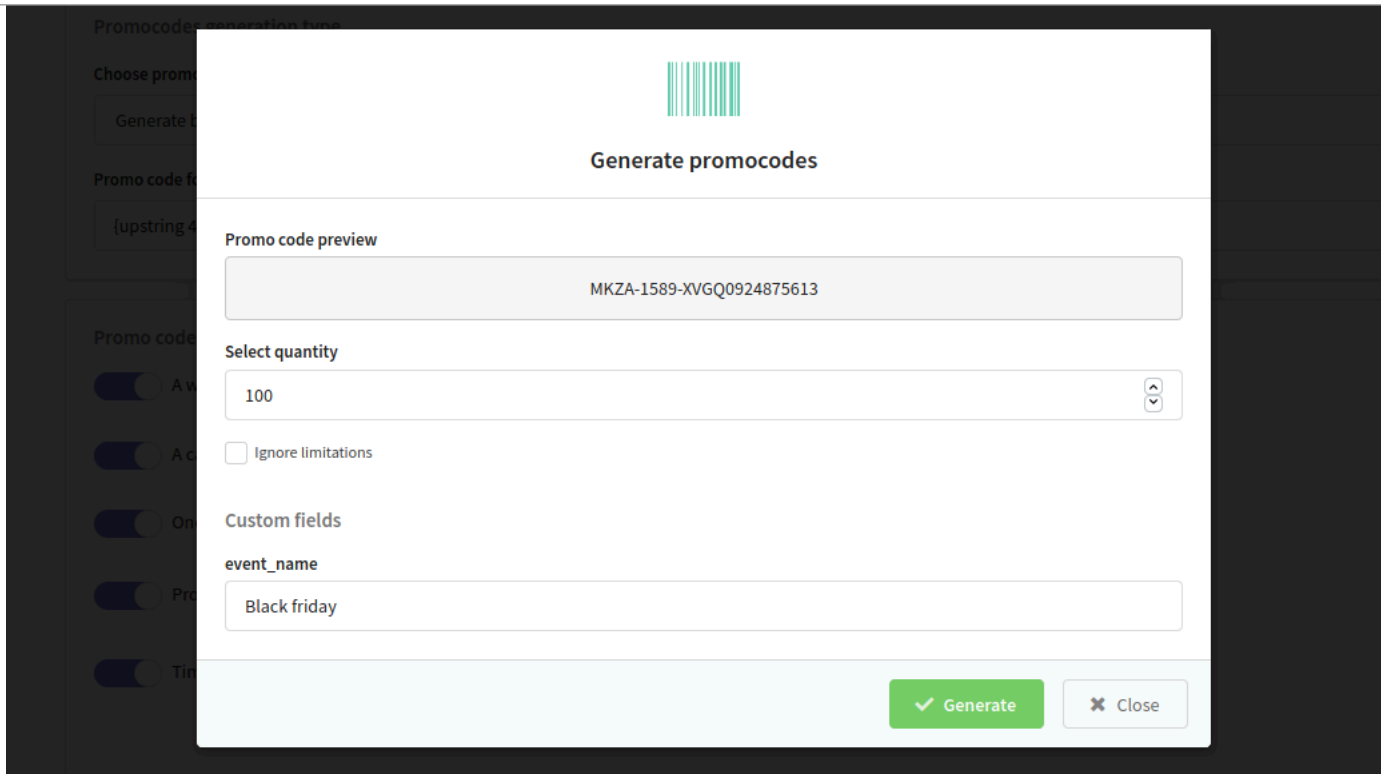
TIP

The promotional code field is created in the Custom fields section at the bottom of the page.



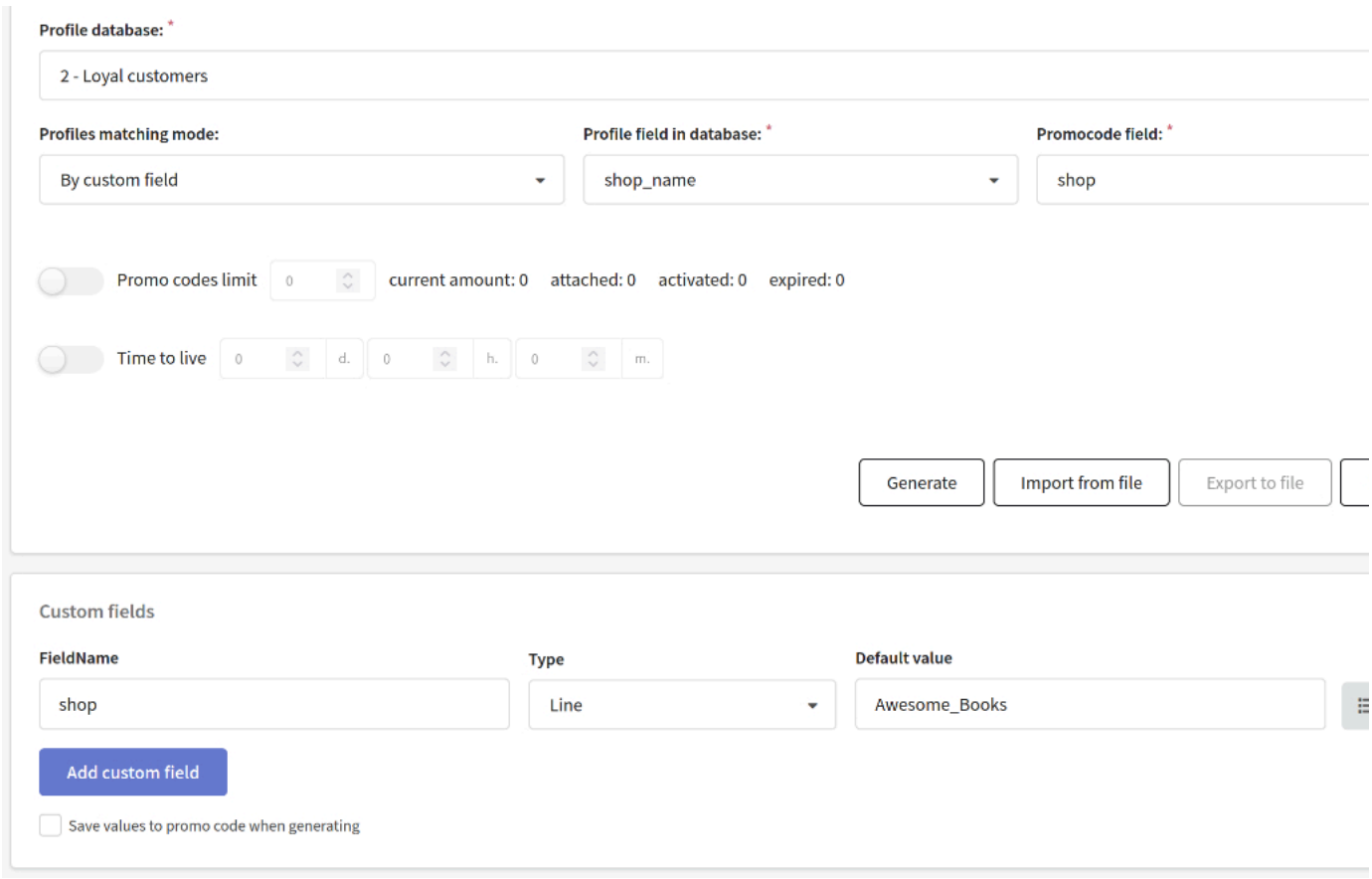
Generation from a template

After configuring generation options, press Generate button. Modal window with promo code preview wil quantity and select values for custom fields, if they are used in loyalty program. After confirming promo updated in the interface at once, or a task for generating promo codes will be created.



To automatically link promo codes to profiles during generation, use the matching mode by a custom fiel field and select it in the matching settings. Specify a default value. If the value of the custom field in the custom field of the promo code, the promo code will be assigned to the found profile.

In the example below, promo codes will be assigned to profiles that have a custom field “`shop_name`” cont

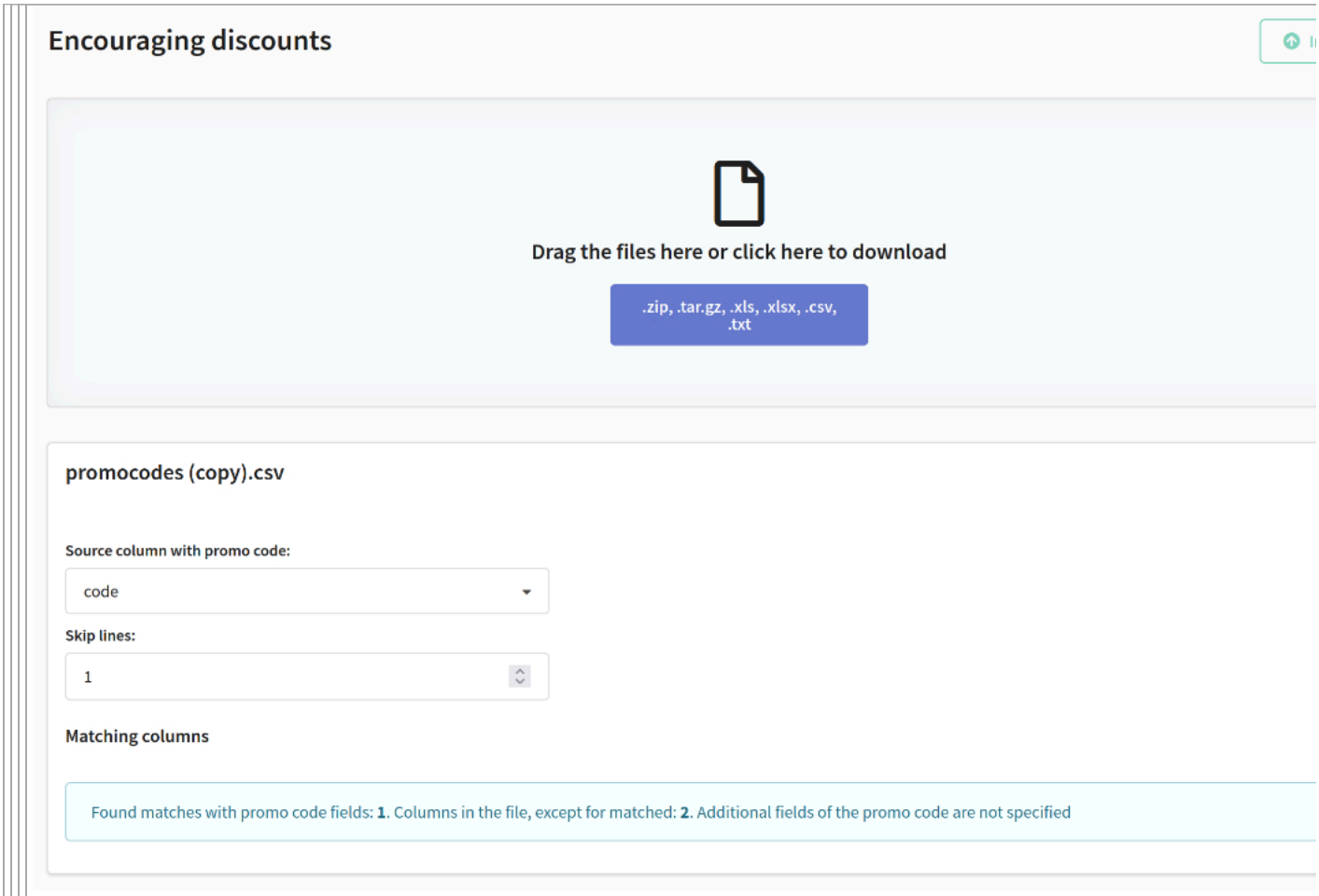


Import from file

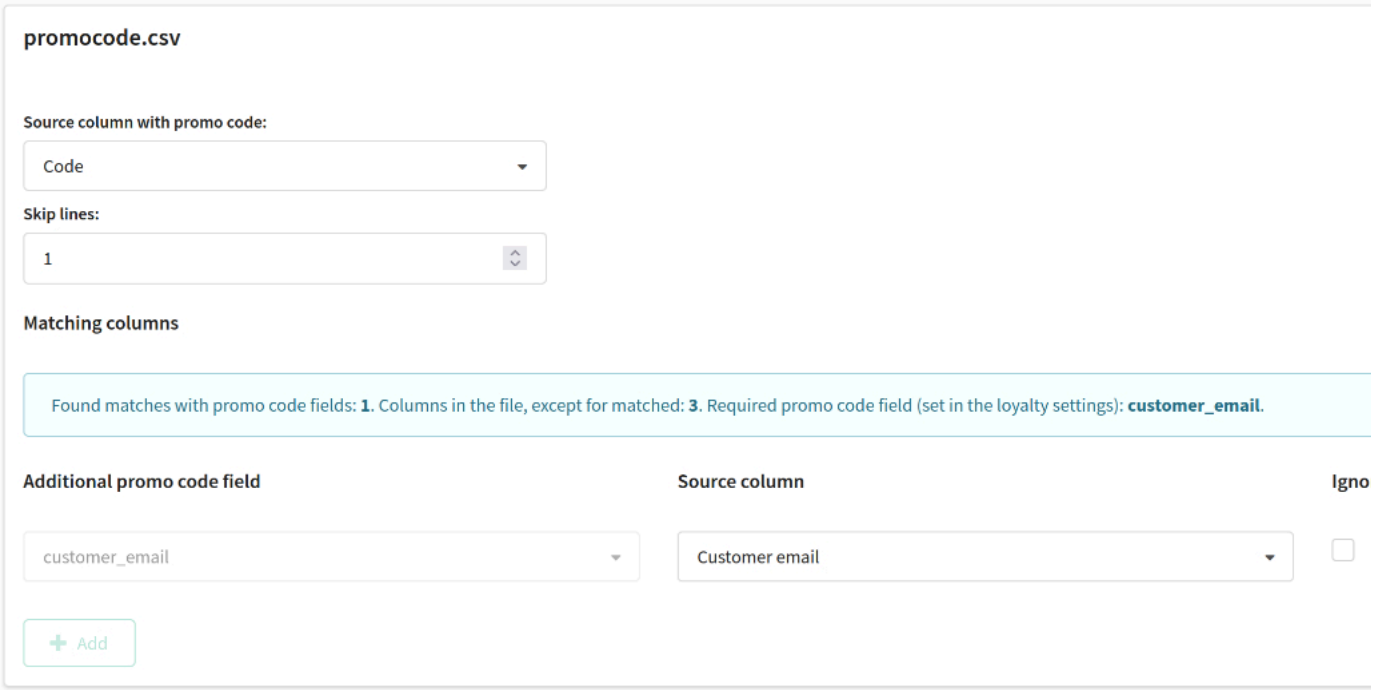
TIP

The import button will be available only after you save the loyalty program.

Press Import from file button to upload XLS or CSV file into the platform, select column with promo code of the page to launch import:

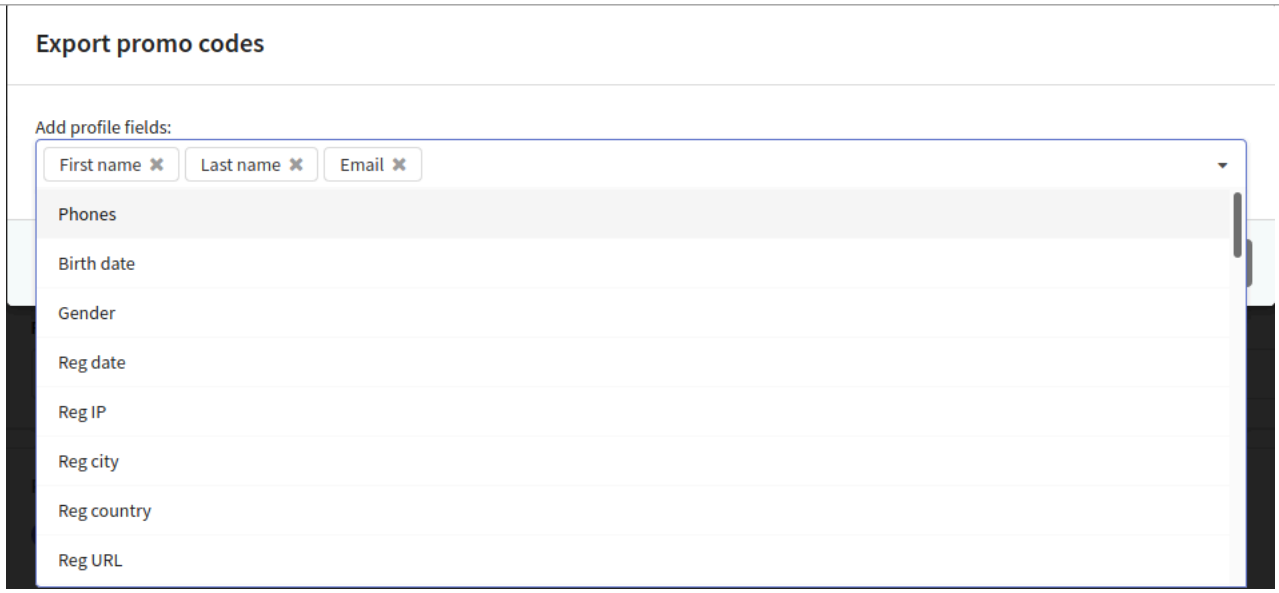


To automatically link promo codes to profiles during import from a file, set up columns matching. As the the file that will be used to search for the required profile:



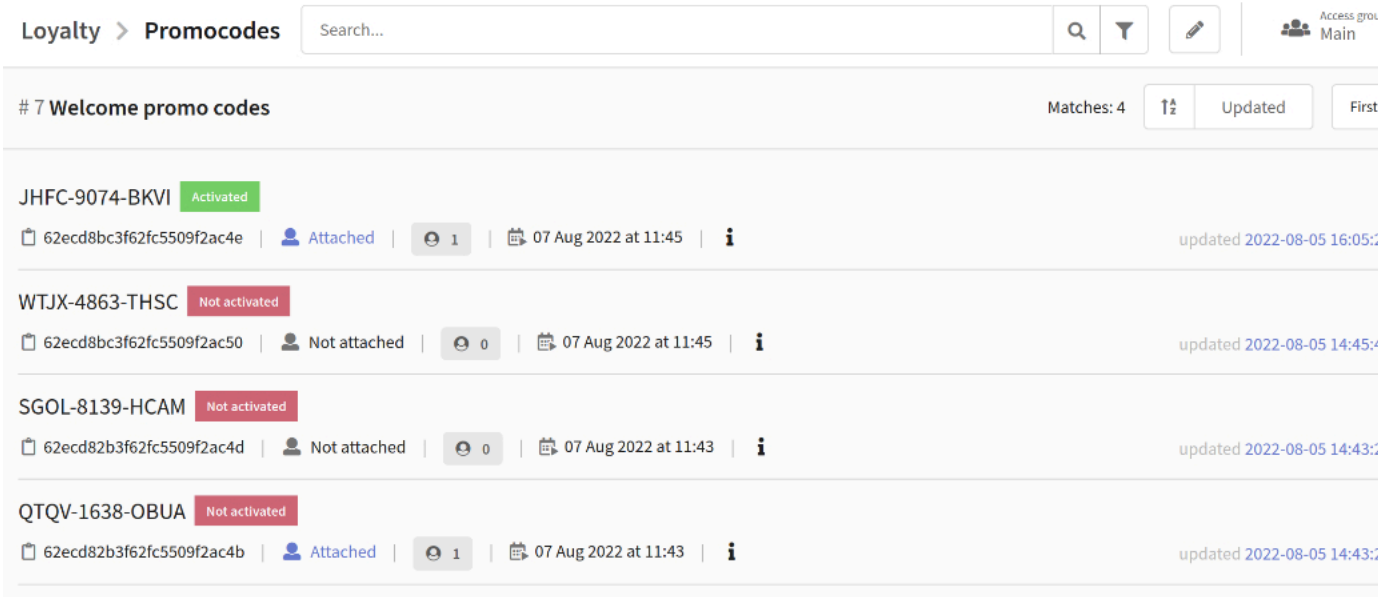
Export to file

Using Export to file, you can download promo codes and fields of profiles, which promo codes were atta drop-down menu and press OK button to export file in CSV format:



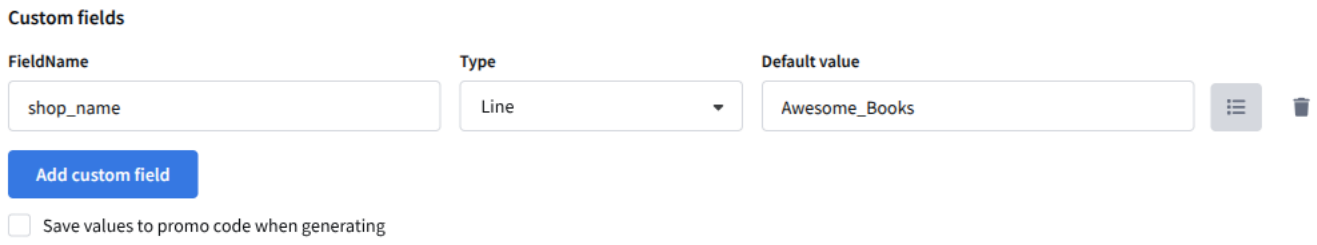
Promo codes browsing

Press Browse button to open list of promo codes created in loyalty program. Brief information will be dis remove any promo code manually from the list:



Custom fields

Custom fields can be requested during promo codes generating and they will be sent to external service fields can be saved in the promo code. If external service returns fields with the same name, they will be

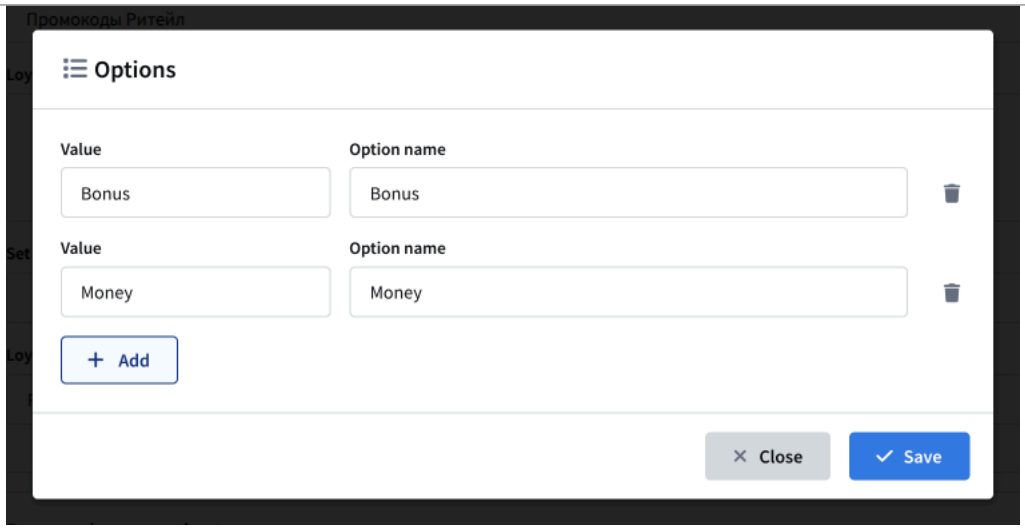


The following field types are available: “Text”, “Number”, “Yes/No”, “String list”, “File”, “Date”. Each field

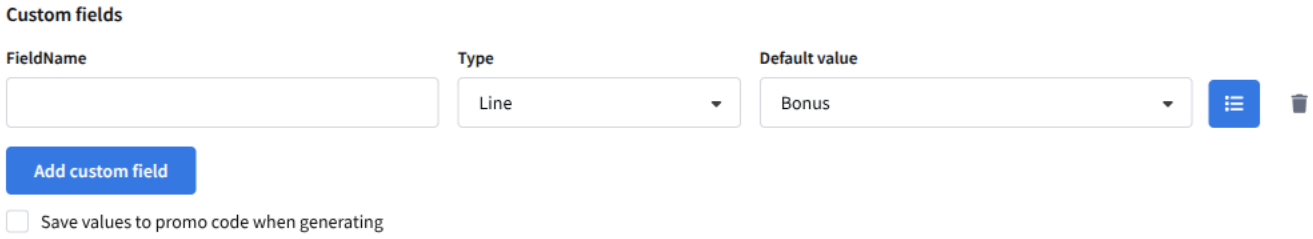
For fields of type “Text” and “Number”, you can define a list of options. Instead of free-form input during select a value from a drop-down list. Each option consists of two parameters:

- **Value — the text displayed in the interface. This is what the user sees when selecting from the list.**
- **Option name — the unique identifier of the option. Saved in the promo code data. Must be unique am be empty, maximum length — 255 characters.**

To configure options, click the list icon button next to the “Value” field. In the dialog that opens, add the and save. After saving, the input field will be replaced with a drop-down list:



During promo code generation, a drop-down list is displayed for a field with options. The selected value i code. The Option name is saved in the promo code data.



Reports and analytics

NOTE

All reports can be downloaded in CSV and XLSX formats by clicking the Export Report button in the act

Channel report

Description

The channel report provides data on all events that are tracked in campaigns: sends, deliveries, opens, c either for one channel or for all of them at once. It is possible to group data according to various attribut filters to get only the information you need. You can also compare metrics, for example, for different tim

Get started by selecting Analytics —> Channels in the main menu.

> INFO
>
> **The Altcraft platform allows an inaccuracy of 0.02% when generating reports.**

Report settings

Channel report settings include:

- **grouping data,**
- **adding filters**
- **saving the report.**

On the right you can see the following buttons:

![img](/img/attachments/76809136/76809177.png) **List of reports — open the list of ready reports;**

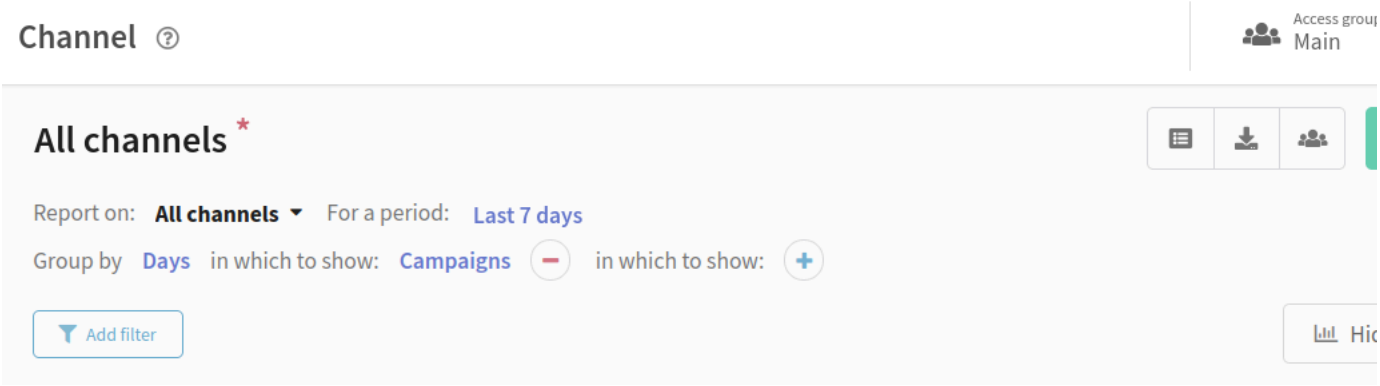
![img](/img/attachments/76809136/76809178.png) **Export report — download the report in csv or xlsx**

![img](/img/attachments/76809136/76809179.png) **Assigned groups — assign access groups**

Grouping data

By default, the report is built for all channels — Email, SMS, Push, Telegram, WhatsApp*, Viber™, Notify, data for a specific channel, select it in the "Report by field.

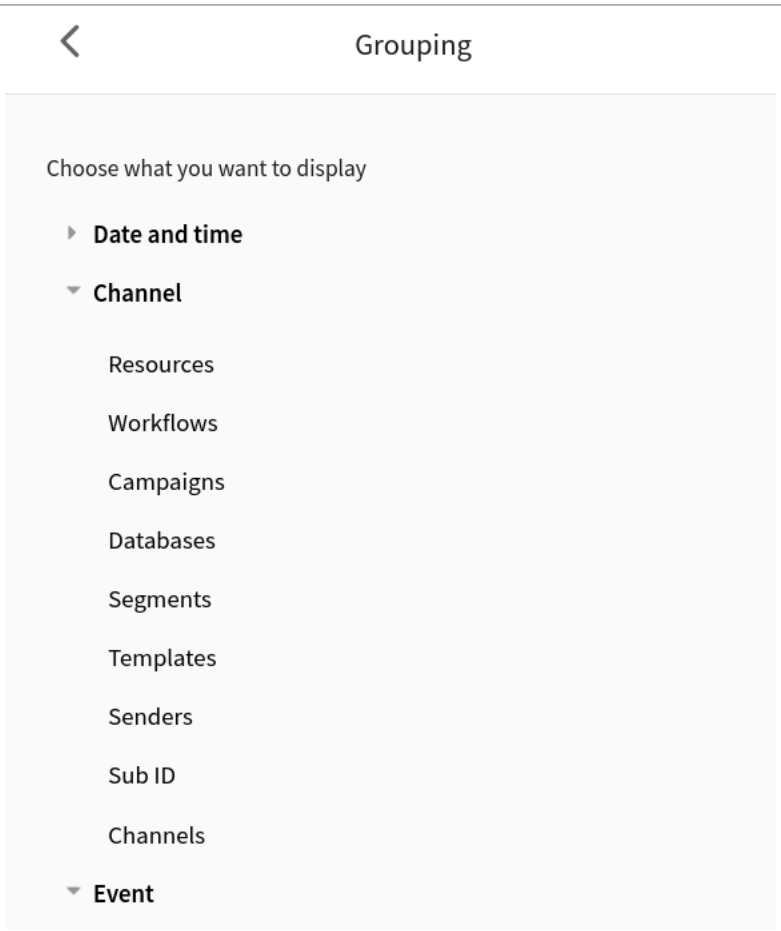
Next, set a period. You can set the date range in the calendar manually or choose one of the presets.



Select the attribute by which you want to group the data in the report. After that, you can detail the grou by one in the in which to show" field. Grouping is possible up to three levels.

You will see the attributes for grouping in the window on the right. They are divided into three categories

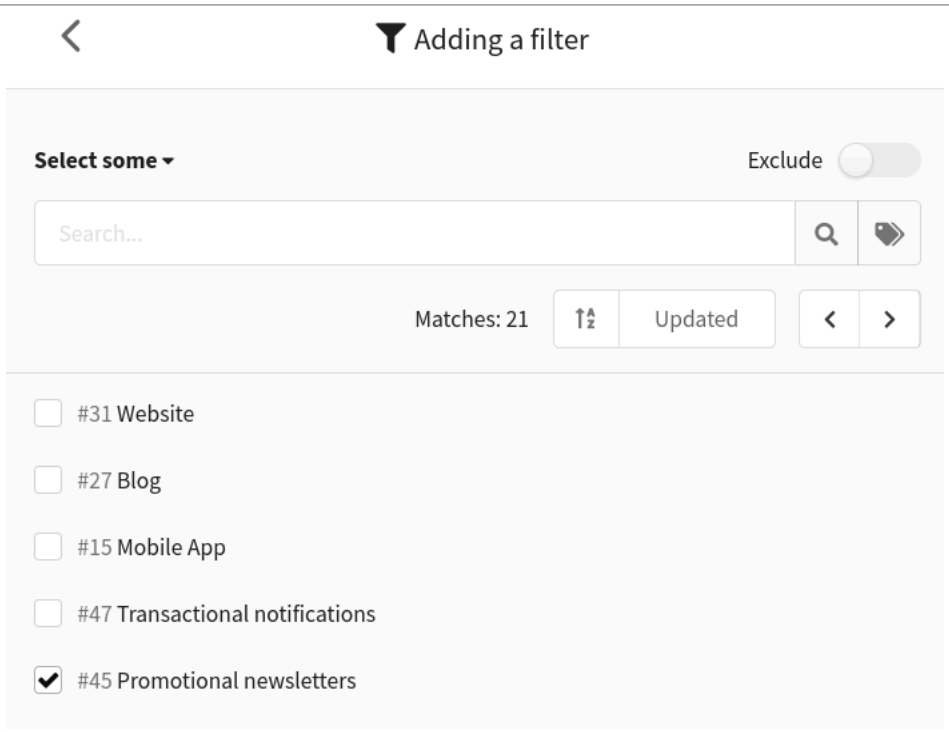
- **Date and time. Group data by hours, days, weeks, months, quarters and years.**
- **Channel. Group by platform objects (resources, scenarios, campaigns, mailings, segments, etc.).**
- **Event. Group by link, domain, domain group, or provider.**



Adding filters

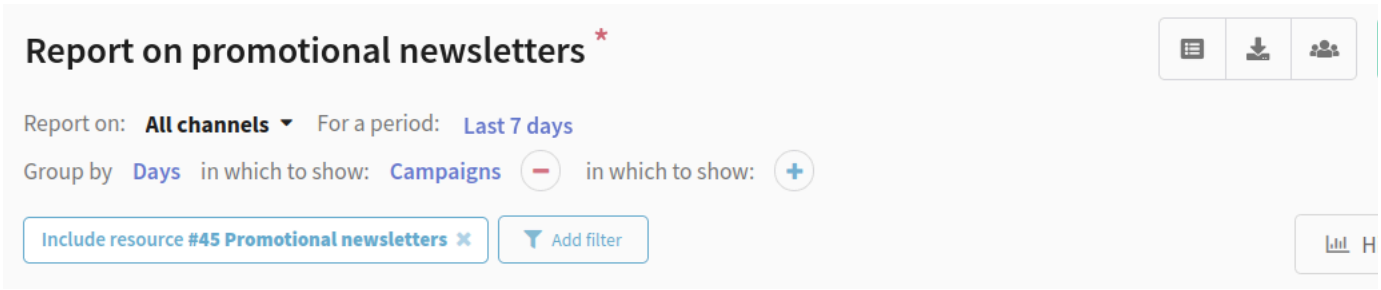
Categories	Available filters	
Date and time	• **Hours (24)** • **Days** • **Weekdays** • **Year and weeks** • **Months** • **Quarters** • **Years** • **Year and quarters**	**Using filters you ca period of time. Suc days or remove sta You can also exclu being done on your**
Channel	• **Campaigns — statistics for all mailings and workflows included in selected campaigns** • **Channels — filter used to display data for a specific channel when the report was built for all channels** • **Resources — statistics for all mailings launched within the resource** • **Workflows — statistics for all mailings launched as part of an automation scenario** • **Mailings — statistics for specific mailings** • **Databases — statistics for mailings whose recipients are included in the selected database** • **Segments — statistics for mailings whose recipients are included in the selected segments** • **Templates — statistics for all mailings that use the selected templates** • **Senders — virtual senders**	**Mailings filter helps you want to analyz from the list. To se the search bar or s** To view metrics for mail tag. A similar op </td>
Event	• **Vendor — add or exclude mailing statistics depending on the Vendor field value in the profile database** • **Domain group — data for emails sent to a specific domain group (Gmail, Hotmail, Yandex)** • **Domain — data for emails sent to a specific domain (gmail.com, example.com, altcraft.com, etc.)** • **Link — mailing events associated with specific links in the template. Links are selected by the `name` attribute**	**In case of deliverat setting up a domai If you don't want th email, select the do**

To filter data, click the “Add filter” button. A window will appear on the right where you can choose whic
If you need to remove some data from the report, click on the Exclude switch in the upper right corner.
After setting the filter, click Apply at the bottom of the page.



The added filters will be displayed in the report. There is no limit to the number of filters you can use. Cl

to remove the filter.



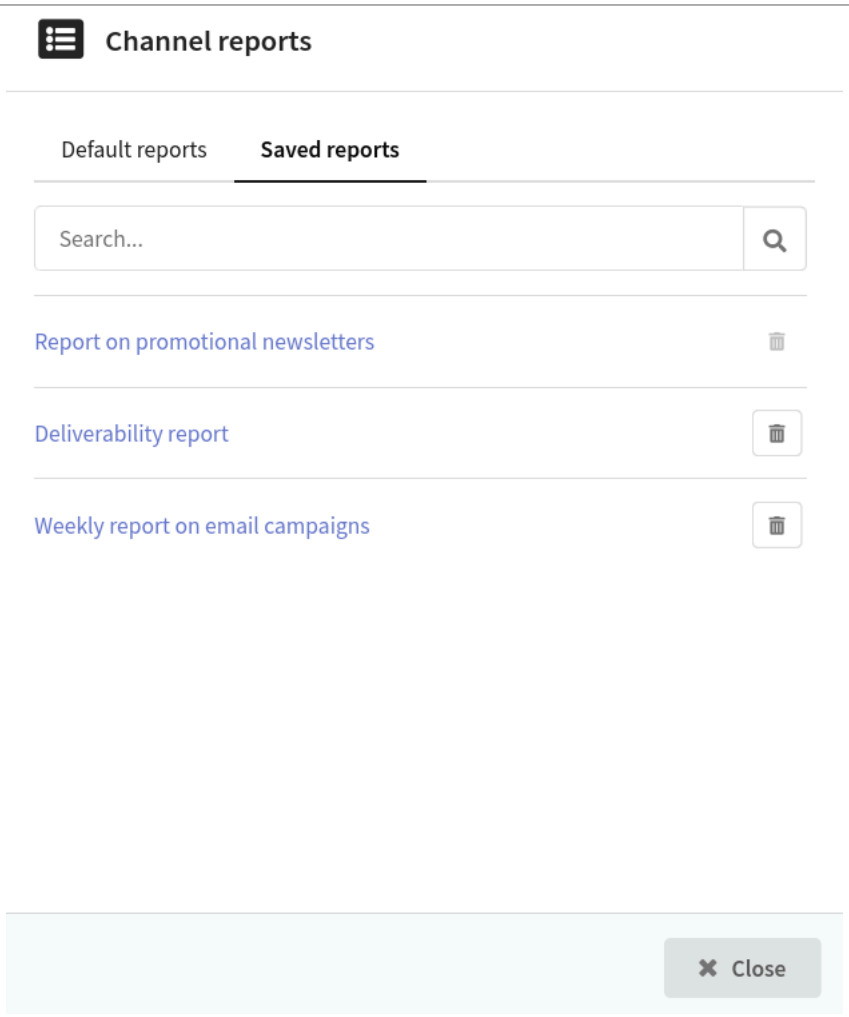
Saving a report

You can save the current report settings. To do this, edit the name of the report (the default is "All Chan
This is useful if you frequently query data using the same groupings and filters.

The saved report will be available to all users within group visibility.

Click

to open the list of reports. In a new window you will find default (system) and saved reports. Click on th



To delete a saved report, click

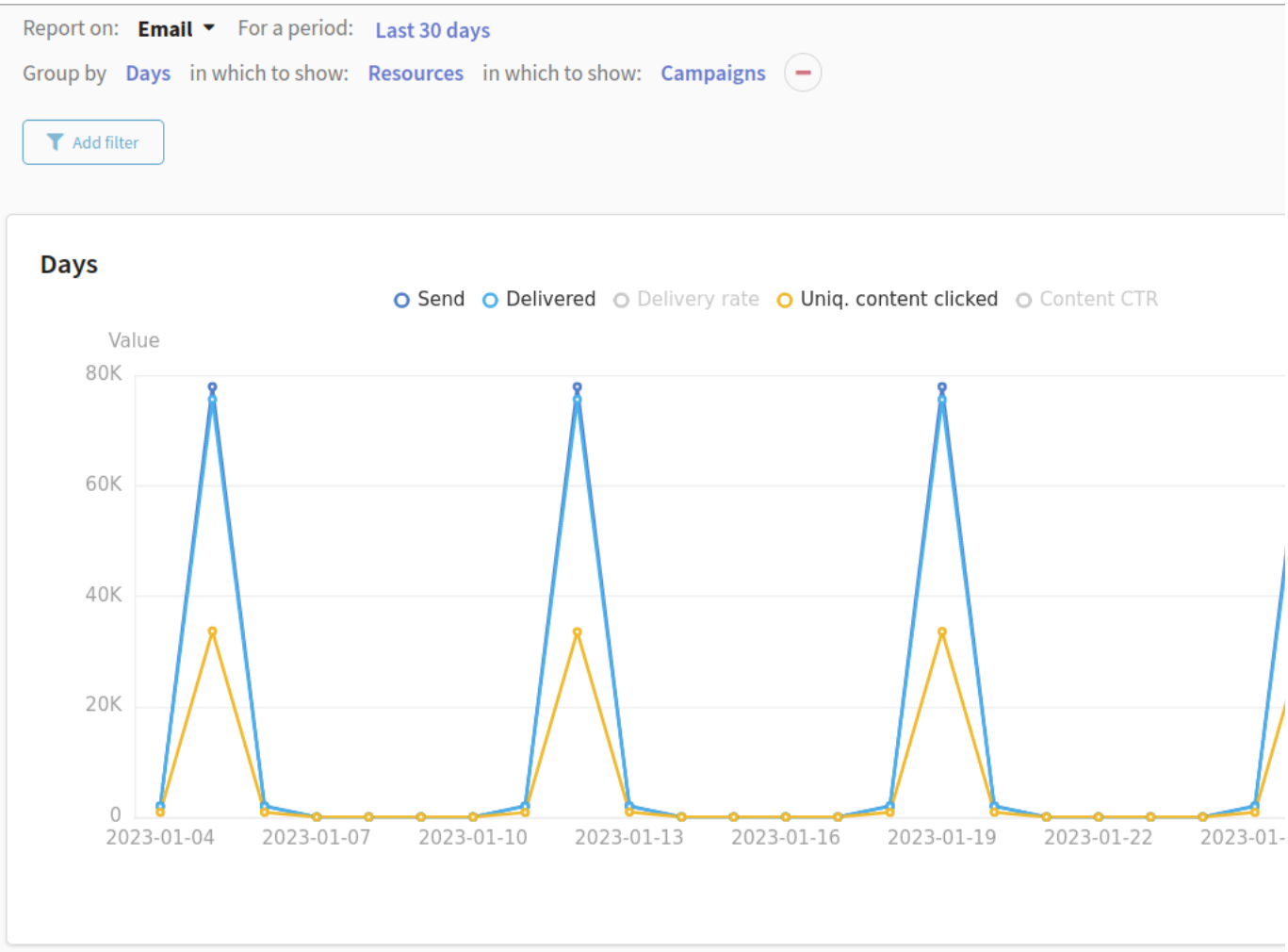
. However, you cannot delete the report that you are currently viewing.

Viewing a report

The report includes a table with data and a corresponding diagram. The chart can be collapsed by clicki

Diagram

The chart displays data based on grouping. When the report is grouped by days, weeks and quarters, the changes in the metrics for the selected period:



When grouped by hour, day of the week, month, year, or channel and event, the chart shows the ratio of the selected period:

When you hover over the chart, you can see the exact figures.

The header contains the metrics displayed on the chart. Any column of the table can be displayed on the of indicators is 7. Click the metric to hide it.

Data table

The table contains quantitative and percentage metrics of campaigns (sends, deliveries, clicks, unsubsc of metrics can be found below.

Metrics are grouped based on the report settings. Depending on the details of the grouping, the table str
For example, with the setting “Group by Campaigns, in which to show Mailings, in which to show Channe
hierarchical structure of marketing strategy effectiveness.
With the setting “Group by days, in which to show Resources, in which to show Mailings” the table will d

Name	Send	Delivered	Delivery rate	Uniq. content clic
Total	8 303 211	7 287 553	87,77 %	2 150 570
2022-11-11	3 270	746	22,81 %	368
2022-11-10	74 032	71 739	96,9 %	31 980
#4 Service messaging (no opt-out)	863	838	97,1 %	442
#28 Welcome trigger	328	320	97,56 %	139
#6 Welcome trigger	349	337	96,56 %	144
#5 SMS opt-in reminder	186	181	97,31 %	159
#2 Blog subscription	73 169	70 901	96,9 %	31 538
2022-11-09	5 214	5 050	96,85 %	2 365
2022-11-08	1 984	1 931	97,33 %	1 275

To add a new metric to the table, hover over the column name and click

. Select a metric from the list. If you want to remove a column, click

.
To move a column to another part of the table, you need to hover over the column name, hold down the le
to the desired position.

Name	Send	Delivered	Delivery rate	Uniq. content clic
Total	8 303 211	7 287 553	87,77 %	2 150 570
2022-11-11	3 270	746	22,81 %	368

Other options:

— switch quantitative metrics to percentages;

— display a column on a diagram.

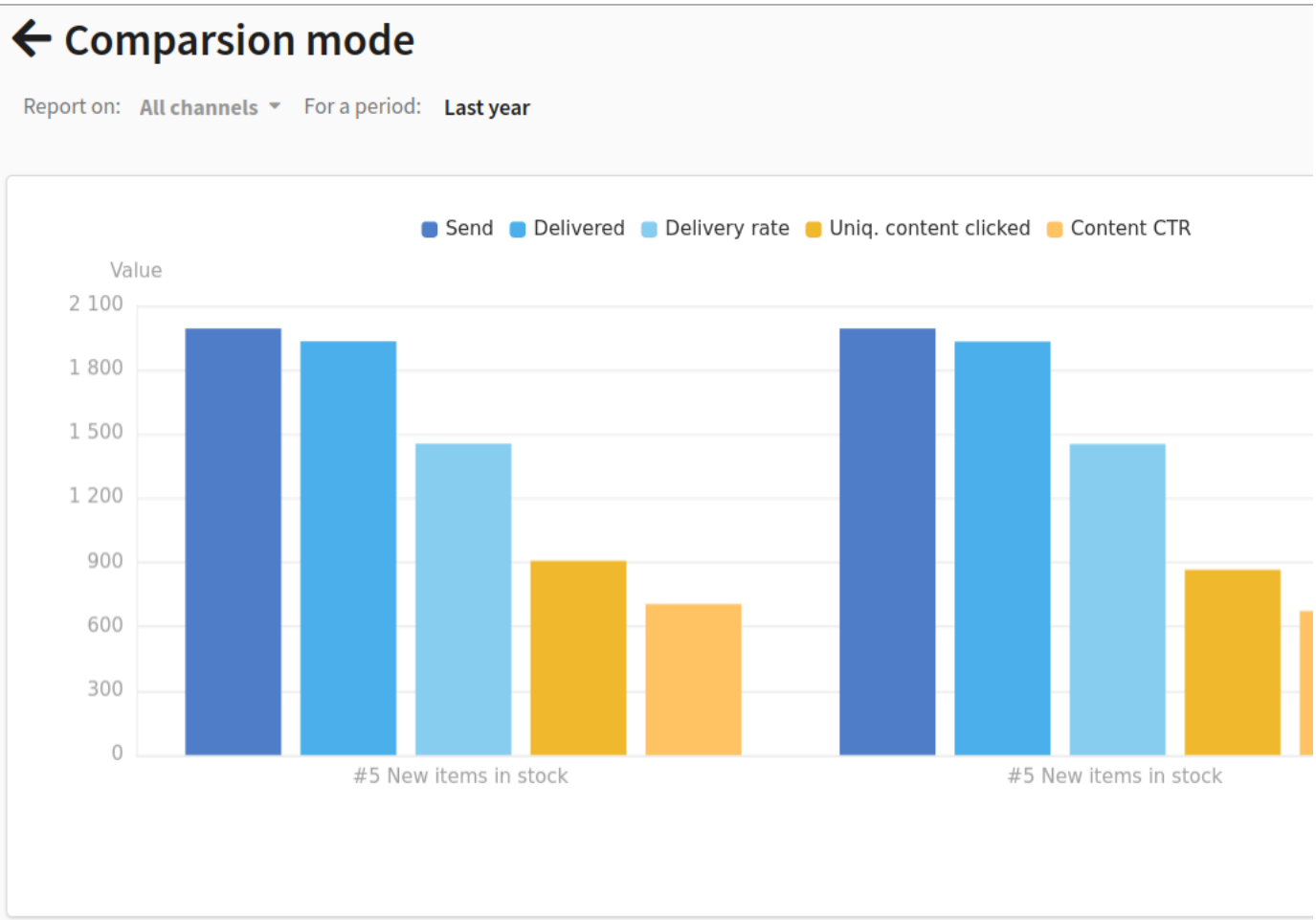
Comparing data

The comparison mode allows you to see data on several objects at the same time. For example, you can templates or evaluate the change in campaign metrics from period to period.
To switch to the comparison mode, select the objects in the table that you want to compare. Click “Com objects at different nesting levels.

Name	Compare	Send	Delivered	Delivery rate	Uniq. content clic
Total		8 303 211	7 287 553	87,77 %	2 150 570
07 November - 13 November 2022		90 287	80 588	89,26 %	36 583
#9 Happy to meet you - join in!		2 352	1 517	64,5 %	687
#5 New items in stock		1 993	1 932	96,94 %	908
#3 Blog digest		74 833	70 578	94,31 %	31 391
#2 New items in stock		7 189	4 269	59,38 %	2 230
#1 Welcome message		3 920	2 292	58,47 %	1 367
31 October - 06 November 2022		87 776	81 327	92,65 %	36 780
#9 Happy to meet you - join in!		1 818	1 461	80,36 %	622
#5 New items in stock		1 993	1 931	96,89 %	865
#3 Blog digest		72 417	70 269	97,03 %	31 511
#2 New items in stock		8 671	5 341	61,6 %	2 356
#1 Welcome message		2 872	2 325	80,95 %	1 426

The comparison mode also contains a chart and a table with metrics. In the diagram, the metrics for eac
The first object in the table is the comparison column. Other objects are compared with this object. The the metric is calculated relative to this object. To change the comparison column, click

and select a different object.



Metrics	07 November - 13 November 2022 #5 New items in stock	31 October - #5 New
Send	1 993	
Delivered	1 932	
Delivery rate	96,94 %	
Uniq. content clicked	908	
Content CTR	47 %	
Uniq. unsubscribed	0	

To go back to the general mode, click ← Comparison mode.

Report metrics

INFO: Unique event

The uniqueness of any event is limited by the time specified in the platform configuration (`UNIQUE_ACTIONS`
1440 minutes. It means that a repeated event in a day will be counted as unique again. Check with your
uniqueness value set.

All indicators of unique events are marked in the table.

All metrics

Metric	Channel	Description

Sent to queue	**All**	**The total number of messages placed in the send queue across all channels.**
Undelivered	**All**	**The total number of messages sent but not delivered through all channels. It consists of Hardbounced and Softbounced me delivered due to undefined reason.**
Undelivery rate	**All**	**The percentage of undelivered messages through all channels.**
Delivered	**All**	**The total number of delivered messages through all channels.**
Delivery rate	**All**	**The percentage of messages delivered through all channels.**
Opened	**Email** **Push**	**The number of open messages.** **If the same profile opens a message multiple times, each open is counted.**
Uniq. opened	**Email** **Push**	**The total number of unique message opens through all channels.** **If the same profile opens a message multiple times, only one open is counted.**
Open rate	**Email** **Push**	**The percentage of profiles who opened the email or push.** **(Uniq. opened / Delivered) * 100%**
Replied	**Email**	**The number of replies to an email.**
Read	**Email** **WhatsApp*** **Viber** **Notify**	**The number of emails read.** **An event is logged if the recipient opens the message and views it for at least 8 seconds.**
Uniq. read	**Email**	**The number of unique reads of an email.**
Read rate	**Email**	**The number of unique reads of an email.** **(Uniq. read / Delivered) * 100%**
Glanced	**Email**	**An event is logged if the recipient opens the email and views it for less than 4 seconds.**
Skimmed	**Email**	**The event is logged if the recipient views the message for 4 to 8 seconds.**
Content clicked	**All**	**The number of clicks on links. When a recipient clicks multiple times on the same link, each click event is counted. For exa clicks each link twice, 6 clicks are counted.**
Uniq. content clicked	**All, but Telegram Group and MAX Group**	**Number of profiles that clicked any link in the message at least once. Counts profile engagement with message links. If the times, only one click on the content is counted. Used to calculate content CTOR and CTR.**
Uniq. link clicked	**Email** **Telegram Bot** **WhatsApp*** **Viber** **Notify** **MAX bot**	**The total number of unique clicks on links in messages. Tracks only the first engagement per link per profile. Used to calcu**
Content CTR	**All but Telegram Group and MAX Group**	**The percentage of users who click on at least one link within a piece of content out of the total number of users who recei** **(Uniq. content clicked / Delivered) * 100%**
Content CTOR	**Email** **Push**	**The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than j** **(Uniq. content clicked / Uniq. opened) * 100%**
Link CTR	**Email** **Telegram bot** **MAX bot** **Whatsapp** **Viber** **Notify**	**The percentage of clicks on the link within a message out of the number of delivered messages.** **(Uniq. link clicked / Delivered) * 100%**
Link CTOR	**Email**	**The percentage of clicks on the link within a message out of the number of opens.** **(Uniq. link clicked / Uniq. opened) * 100%**
Camp. errors	**All**	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selecte**
Deliv. errors	**All**	**Problems with protocol during sending a message.**
Suppressed	**All**	**The number of messages that were not sent because the recipient's email address or phone number has previously been in**
Policy offenced	**All**	**The number of violations of the sending policy.**
Softbounced	**Email**	**The number of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Softbounce rate	**Email**	**The percentage of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailal**
Hardbounced	**Email** **SMS** **Push**	**The number of messages that were not delivered due to an invalid email address, phone number, or push subscription ID.**
Hardbounce rate	**Email** **SMS** **Push**	**The percentage of messages that were not delivered due to an invalid email address, phone number, or push subscription I**
Complained	**Email** **SMS** **Push**	**The number of messages marked as spam by recipients.**
Complains rate	**Email**	**The percentage of messages marked as spam by profiles.** **(Complained / Delivered)*100**
Subscribed	**All but Telegram Group and MAX Group**	**The number of subscriptions to resources in any channel.**
Unsubscribed	**Email** **SMS** **Push** **Viber** **Notify**	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.**
Uniq. unsubscribed	**Email** **SMS** **Push**	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**Email** **SMS** **Push**	**(Unsubscribed / Delivered) * 100%**

Unsub to open rate	**Email Push**	**(Unsubscribed / Opened) * 100%**
Undefined	**-**	**Unable to determine reason for non-delivery.**

Email channel

Metric	Description
Sent to queue	**The number of email messages placed in the send queue.**
Undelivered	**The number of undelivered email messages. It consists of Hardbounced and Softbounced messages and messages which are not delivered due to undef**
Undelivery rate	**The percentage of undelivered email messages.**
Delivered	**The number of delivered email messages.**
Delivery rate	**The percentage of delivered email messages.**
Opened	**The number of opened email messages. If the same profile opens an email message multiple times, each open is counted.**
Uniq. opened	**The number of unique email message openings. If the same profile opens a message multiple times, only one open is counted.**
Open rate	**The percentage of recipients who opened the email. (Uniq. opened / Delivered) * 100%**
Replied	**The number of replies to an email.**
Read	**The number of emails read. An event is logged if the recipient opens the message and views it for at least 8 seconds.**
Uniq. read	**The number of unique reads of an email.**
Read rate	**The number of unique reads of an email. (Uniq. read / Delivered) * 100%**
Glanced	**An event is logged if the recipient opens the email and views it for less than 4 seconds.**
Skimmed	**The event is logged if the recipient views the message for 4 to 8 seconds.**
Content clicked	**The number of clicks on links. When a recipient clicks multiple times on the same link, each click event is counted. For example, if there are 3 links in a are counted.**
Uniq. content clicked	**Number of profiles that clicked any link in the message at least once. Counts profile engagement with message links. If the same profile clicks on any li links, it allows tracking which links users select first when engaging with the message.**
Uniq. link clicked	**The total number of unique clicks on links in email messages. Tracks only the first engagement per link per profile. If the recipient clicks on two differer counted.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked /**
Content CTOR	**The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than just received it). (Uniq. conten**
Link CTR	**The percentage of clicks on the link within a message relative to the number of delivered messages. (Uniq. link clicked / Delivered) * 100%**
Link CTOR	**The percentage of clicks on the link within a message relative to the number of opens. (Uniq. link clicked / Uniq. opened) * 100%**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Suppressed	**The number of messages that were not sent because the recipient's email address was included in the suppression list.**
Policy offenced	**The number of violations of the sending policy.**
Softbounced	**The number of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Softbounce rate	**The percentage of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Hardbounced	**The number of messages that were not delivered due to an invalid email address.**
Hardbounce rate	**The percentage of messages that were not delivered due to an invalid email address.**
Complained	**The number of messages marked as spam by recipients.**
Complains rate	**The percentage of messages marked as spam by recipients. (Complained / Delivered) * 100**
Subscribed	**The number of subscriptions to the email channel in resources.**
Unsubscribed	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the email message.**
Uniq. unsubscribed	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**(Unsubscribed / Delivered) * 100%**
Unsub to open rate	**(Unsubscribed / Opened) * 100%**
Undefined	**Unable to determine reason for non-delivery.**

NOTE: Email tracking limitations

Open, glance, skim, and read metrics depend on the loading of tracking pixels in the email. Some mail c or proxy images, which affects the accuracy of these metrics:

- **Image blocking (Outlook, Thunderbird by default) — pixels do not load, opens are not recorded. Solu exceptions list.**
- **Proxy servers (Gmail Image Proxy, Apple Mail Privacy Protection, corporate proxies) — load pixels t open but does not allow measuring viewing time (glanced/skimmed/read).**
- **Gmail prefetching — may load images before the email is actually opened (preview pane).**

For accurate engagement assessment, it is recommended to use click metrics (CTR), which do not dep

SMS channel

Metric	Description
Sent to queue	**The total number of SMS messages placed in the send queue.**
Undelivered	**The number of undelivered SMS messages.**
Undelivery rate	**The percentage of undelivered SMS messages.**
Delivered	**The total number of delivered SMS messages.**
Delivery rate	**The percentage of delivered SMS messages.**

Content clicked	**The number of clicks on links.**
Uniq. content clicked	**Number of profiles that clicked on any link in the message at least once. Counts profile engagement with message links. If the same profile clicks on counted.**
Content CTR	**The percentage of users who click on at least one link within the SMS message out of the total number of delivered messages. (Uniq. content clicked**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Suppressed	**The number of messages that were not sent because the recipient's email address or phone number has previously been included in the suppression l**
Policy offenced	**The number of violations of the sending policy.**
Hardbounced	**The number of messages that were not delivered due to an invalid phone number**
Hardbounce rate	**The percentage of messages that were not delivered due to an invalid phone number.**
Complained	**The number of messages marked as spam by recipients.**
Subscribed	**The number of subscriptions to the SMS channel in resources.**
Unsubscribed	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the SMS message.**
Uniq. unsubscribed	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**(Unsubscribed / Delivered) * 100%**
Undefined	**Unable to determine reason for non-delivery (unknown network error).**

Push channel

Metric	Description
Sent to queue	**The number of push notifications placed in the send queue.**
Undelivered	**The number of undelivered push notifications.**
Undelivery rate	**The percentage of undelivered push notifications.**
Delivered	**The number of delivered push notifications.**
Delivery rate	**The percentage of delivered push notifications.**
Opened	**The number of opened push notifications.** ****Web push:**** **Number of opens equals number of deliveries.** ****Mobile push:**** **Open tracking depends on client-side implementation of event handling for links. Opens are recorded by the application itself and don't depend on whether t**

Uniq. opened	**The number of delivered push notifications openings.** If the same profile opens an notifications multiple times, only one open is counted. </td> </tr> <tr> <td>Open rate</td> <td> The percentage of recipients who opened the push. (Uniq. opened / Delivered) * 100% </td> </tr> <tr> <td>Read</td> <td> The number of read push notifications. The event is registered when a recipient reads a push notification in the notification center (e.g. in cascaded communication, when notification content is duplicated f </td> </tr> <tr> <td>Uniq. read</td> <td> The number of unique push notification reads. If the same profile reads a notification multiple times, only one read is counted. </td> </tr> <tr> <td>Read rate</td> <td> The percentage of recipients who read the push. (Uniq. read / Delivered) * 100% </td> </tr> <tr> <td>Content clicked</td> <td>The number of clicks in the push notification (inluding links and buttons). For mobile pushes, clicks are counted only if the link includes tracking. Used to track redirects to websites.</td> </tr> <tr> <td>Uniq. content clicked</td> <td> If the same profile clicks the push notification multiple times, only one unique content click is counted. </td> </tr> <tr> <td>Content CTR</td> <td> The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Content CTOR</td> <td> The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than just received it). (Uniq. content clicked / Uniq. opened) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Policy offenced</td> <td> The number of violations of the sending policy. </td> </tr> <tr> <td>Hardbounced</td> <td> The number of messages that were not delivered due to an invalid push subscription ID. </td> </tr> <tr> <td>Hardbounce rate</td> <td> The percentage of messages that were not delivered due to an invalid push subscription ID. </td> </tr> <tr> <td>Complained</td> <td>The number of messages marked as spam by recipients.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to the Push channel in resources.</td> </tr> <tr> <td>Unsubscribed</td> <td> The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message. </td> </tr> <tr> <td>Uniq. unsubscribed</td> <td>The percentage of unique unsubscribes from resources.</td> </tr> <tr> <td>Unsub to deliv rate</td> <td>(Unsubscribed / Delivered) * 100%</td> </tr> <tr> <td>Unsub to open rate</td> <td>(Unsubscribed / Opened) * 100%</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Telegram Bot channel

Metric	Description
Sent to queue	**The number of messages placed in the send queue for delivery to the subscriber in the Telegram bot.**
Undelivered	**The number of messages sent but not delivered in the Telegram bot.**
Undelivery rate	**The percentage of undelivered messages in the Telegram bot.**

Delivered	**The number of delivered messages in the Telegram bot.**
Delivery rate	**The percentage of messages delivered in the Telegram bot.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any link multiple times, only one unique content click is counted.**
Uniq. link clicked	**The number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message relative to the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Policy offenced</td> <td>The number of violations of the sending policy.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to resources in the Telegram bot.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Telegram Group channel

Metric	Description
Sent to queue	**The number of messages placed in the send queue for the Telegram group.**
Undelivered	**The number of undelivered messages in the Telegram group.**
Undelivery rate	**Percentage of undelivered messages to the Telegram group.**
Delivered	**The number of messages delivered to the Telegram group.**
Delivery rate	**Percentage of messages delivered to the Telegram group.**
Content clicked	**The number of click on links.**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Policy offenced	**The number of violations of the sending policy.**
Undefined	**Unable to determine reason for non-delivery (unknown network error).**

MAX Bot channel

Metric	Description
Undelivered	**Number of messages not delivered to the MAX group**
% Undelivered	**Percentage of messages not delivered to the MAX group**
Delivered	**Number of messages delivered to the MAX group**
% Delivered	**Percentage of messages delivered to the MAX group**
Content clicks	**If a recipient clicks the same link multiple times, each click event is recorded. For example, if a message contains 3 links and the recipient clicks e**
Generation errors	**An error occurred while generating the template. It happens when the profile does not have a subscription to the resource selected in the Campaig**
Transport errors	**An error occurred in the message delivery protocols.**
Policy limit exceeded	**Number of messages not sent because the limit set in the sending policy was exceeded.**
Unsubscribes	**The profile unsubscribed from the MAX Bot channel.**
Reads	**Number of messages read by the subscriber.**
Undefined	**The reason for undelivery could not be determined (for example, an unknown network error occurred)**

MAX Group channel

Metric	Description
Undelivered	**Number of messages not delivered to the chat with the MAX bot**
% Undelivered	**Percentage of messages not delivered to the chat with the MAX bot**
Delivered	**Number of messages delivered to the chat with the MAX bot**
% Delivered	**Percentage of messages delivered to the chat with the MAX bot**
Content clicks	**If a recipient clicks the same link multiple times, each click event is recorded. For example, if a message contains 3 links and the recipient clicks e**
Generation errors	**An error occurred while generating the template. It happens when the profile does not have a subscription to the resource selected in the Campaig**
Transport errors	**An error occurred in the message delivery protocols.**
Policy limit exceeded	**Number of messages not sent because the limit set in the sending policy was exceeded.**
Unsubscribes	**The profile unsubscribed from the MAX Group channel.**
Reads	**Number of messages read by the subscriber.**
Undefined	**The reason for undelivery could not be determined (for example, an unknown network error occurred)**

WhatsApp channel

Metric	Description
Sent to queue	**Number of messages placed in the send queue on WhatsApp*.**
Undelivered	**Number of undelivered WhatsApp* messages.**
Undelivery rate	**Percentage of undelivered messages on WhatsApp*.**
Delivered	**Number of messages delivered on WhatsApp*.**
Delivery rate	**Percentage of messages delivered on WhatsApp*.**
Read	**The number of messages read by the user.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any part of the message multiple times, only one click on the content is counted.**
Uniq. link clicked	**The total number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Suppressed</td> <td> The number of messages that were not sent because the recipient's phone number has previously been included in the suppression list. </td> </tr> <tr> <td>Policy offenced</td> <td> The number of violations of the sending policy. </td> </tr> <tr> <td>Subscribed</td> <td>The profile subscribed to the WhatsApp* channel.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Viber channel

Metric	Description
Sent to queue	**The total number of messages placed in the send queue on Viber.**
Undelivered	**The total number of messages sent but not delivered on Viber.**
Undelivery rate	**The percentage of undelivered on Viber messages.**
Delivered	**The total number of delivered messages on Viber.**
Delivery rate	**The percentage of messages delivered on Viber.**
Read	**The number of messages read by the user.**
Uniq. read	**The number of unique reads in Viber chat.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any link multiple times, only one click on the content is counted.**
Uniq. link clicked	**The number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Suppressed</td> <td>The number of messages that were not sent because the recipient's email address or phone number has previously been included in the <a href="/en/user-guide/profiles-and-databases/66534">suppression li </tr> <tr> <td>Policy offenced</td> <td>The number of violations of the sending policy.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to resources in the channel.</td> </tr> <tr> <td>Unsubscribed</td> <td>The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Notify channel

Metric	Description
Sent to queue	The total number of messages placed in the send queue through Notify.
Undelivered	The total number of messages sent but not delivered on Notify.
Undelivery rate	The percentage of undelivered on Notify messages.
Delivered	The total number of delivered messages on Notify.
Delivery rate	The percentage of messages delivered on Notify.
Read	The number of messages read by the user.
Content clicked	The number of clicks on links.
Uniq. content clicked	If the same profile clicks on any part of the message multiple times, only one click on the content is counted.
Uniq. link clicked	The number of unique clicks on links in messages.
Content CTR	The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100%
Link CTR	The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100%
Camp. errors	Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.
Deliv. errors	Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.
Suppressed	The number of messages that were not sent because the recipient's email address or phone number has previously been included in the [suppressio
Policy offenced	The number of violations of the sending policy.
Subscribed	The number of subscriptions to resources in the channel.
Unsubscribed	The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.
Undefined	Unable to determine reason for non-delivery (unknown network error).

* ***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a***

* **Viber™ is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.**

Traffic report

Description

**The traffic report provides data on profile traffic from sources such as channels, web forms, pixels, and
according to certain attributes, drill down into the report, or use filters to exclude redundant information
data to see how switching to a different strategy have affected your performance metrics.**

**The report is based on hit. It is a unit of traffic measurement. A hit is logged when a user action on a we
request to the server. The action can be a goal registration or a click in a message.**

Report settings

Traffic report settings include:

- **grouping data,**
- **adding filters**
- **saving the report.**

On the right you can see the following buttons:

![img](/img/attachments/76809136/76809177.png) **List of reports — open the list of ready reports;**

![img](/img/attachments/76809136/76809178.png) **Export report — download the report in csv or xlsx**

![img](/img/attachments/76809136/76809179.png) **Assigned groups — assign access groups**

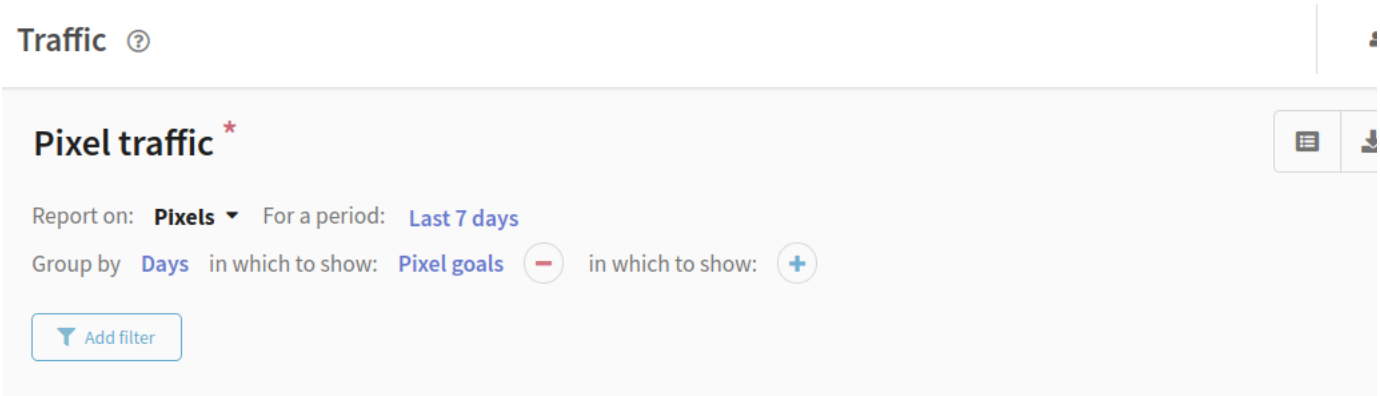
Grouping data

The report provides data on the following traffic sources:

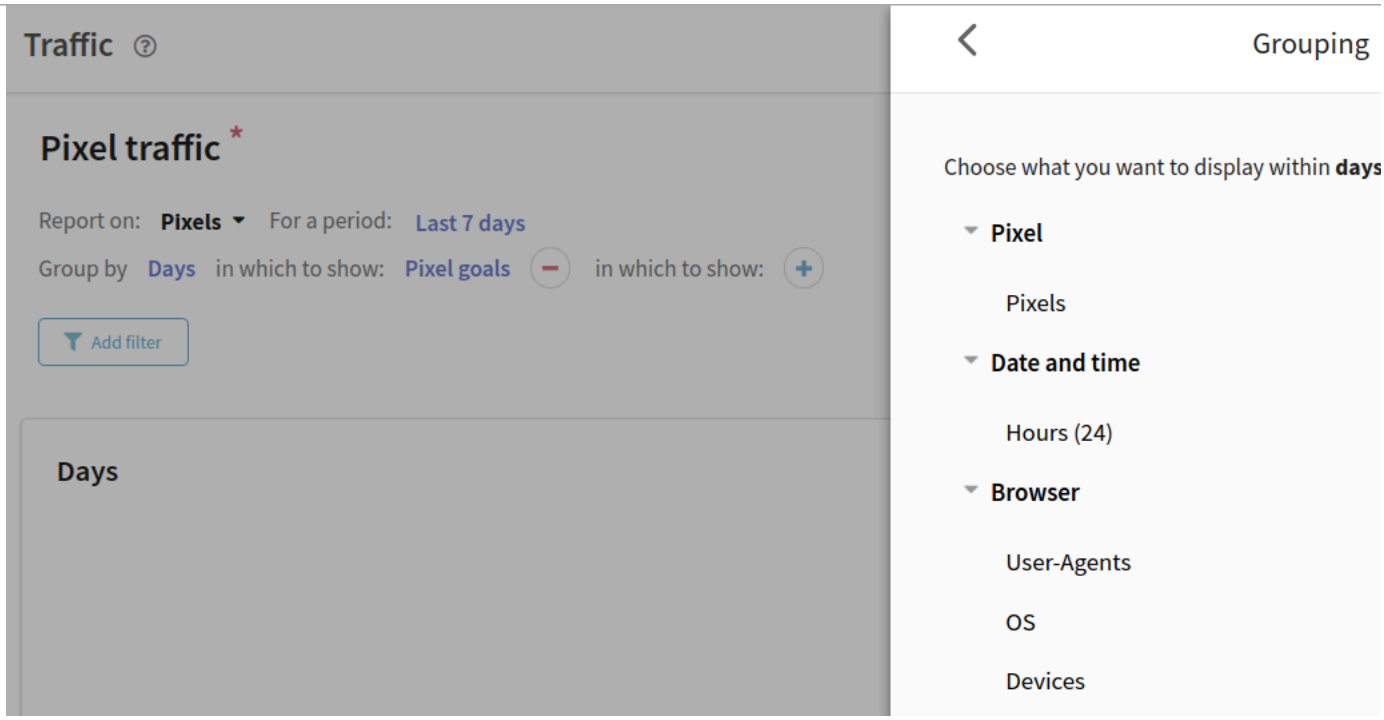
- **Channels. Statistics on subscriptions and unsubscribes, opens and clicks in messages.**
- **Pixels. Statistics on goals that are registered by pixels.**
- **Forms. Statistics onthe number of loads and submissionsof web forms.**
- **Pop-ups. Statistics on pop-ups shows and fills.**

To get data for a specific traffic source, select it in the Report by field.

Next, set a period. You can specify the date range in the calendar manually or choose one of the presets



**Select the attribute by which you want to group the data in the report. After that, you can detail the grou
by one in the “in which to show” field. Grouping is possible up to three levels. You will see the grouping c**



TIP

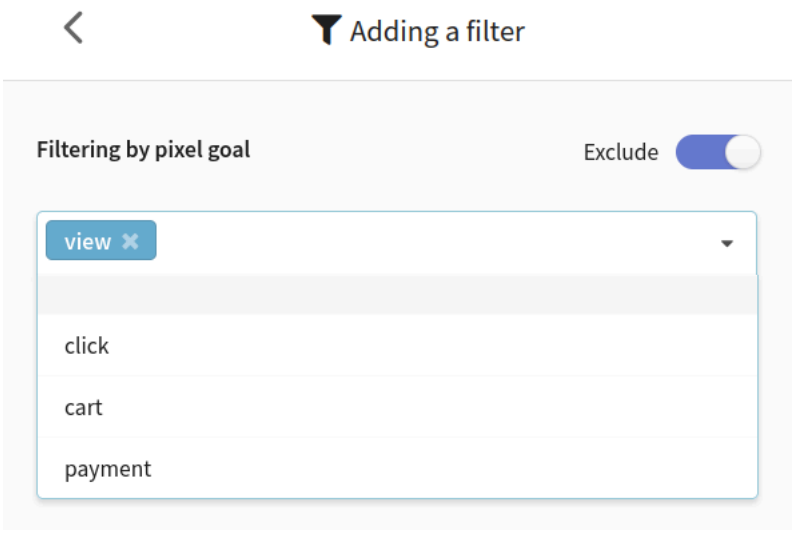
Not all browsers send responses specifying the device, browser, and operating system in the header. Gi Agent" contain "Other" for such cases. Also if a header was transmitted, but the system could not read will be recorded in "Other".

Adding filters

With the help of filters, you can leave only the necessary information in the report. Use filters if you wan goals, resources, etc. Filter categories are the same as grouping categories.

Click Add filter to filter the data. In the window on the right, you can choose which data to include in the

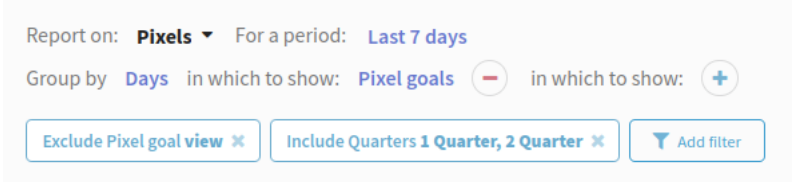
If you need to remove some data from the report, click on the Exclude switch in the upper right corner.



After setting the filter, click Apply at the bottom of the page.

The added filters will be displayed in the report. There is no limit to the number of filters you can use. Cl

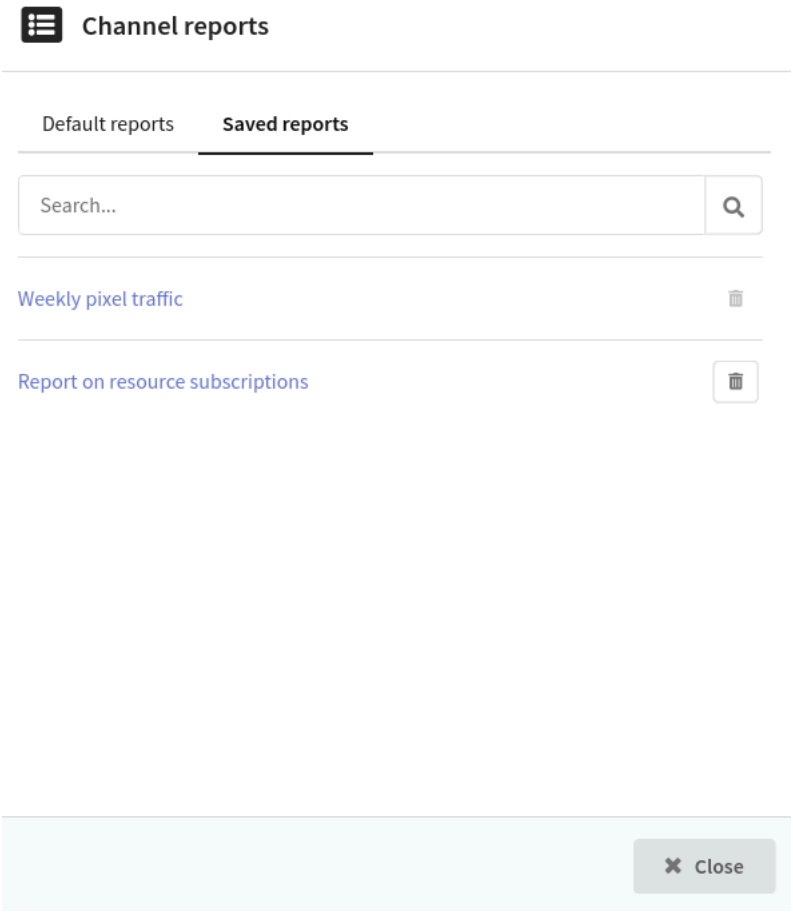
to remove the filter.



Saving a report

You can save the current report settings. To do this, edit the name of the report (the default is “All Chan
This is useful if you frequently query data using the same groupings and filters.
The saved report will be available to all users within group visibility.
Click

to open the list of reports. In a new window you will find default (system) and saved reports. Click on th



To delete a saved report, click

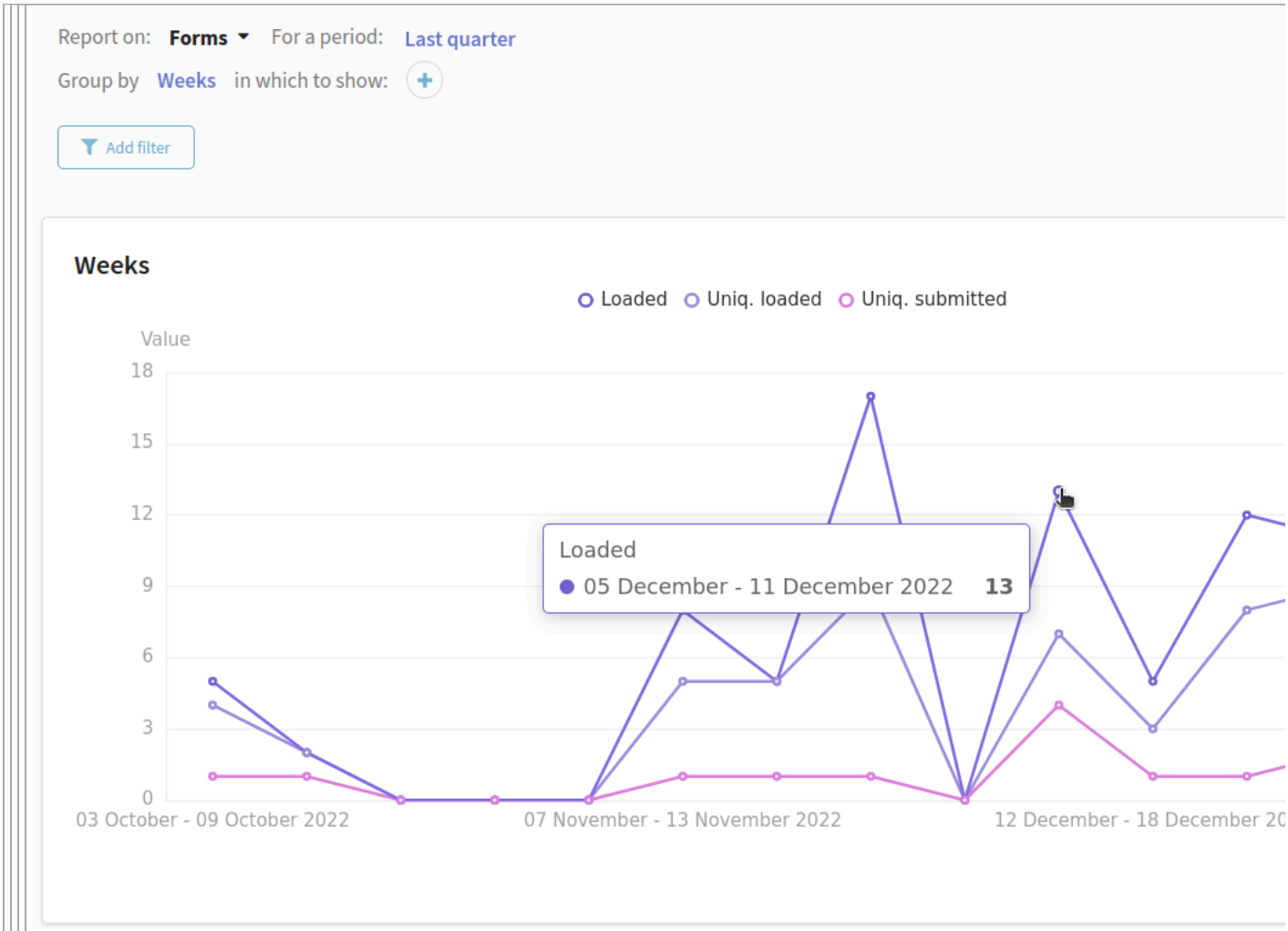
. However, you cannot delete the report that you are currently viewing.

Viewing a report

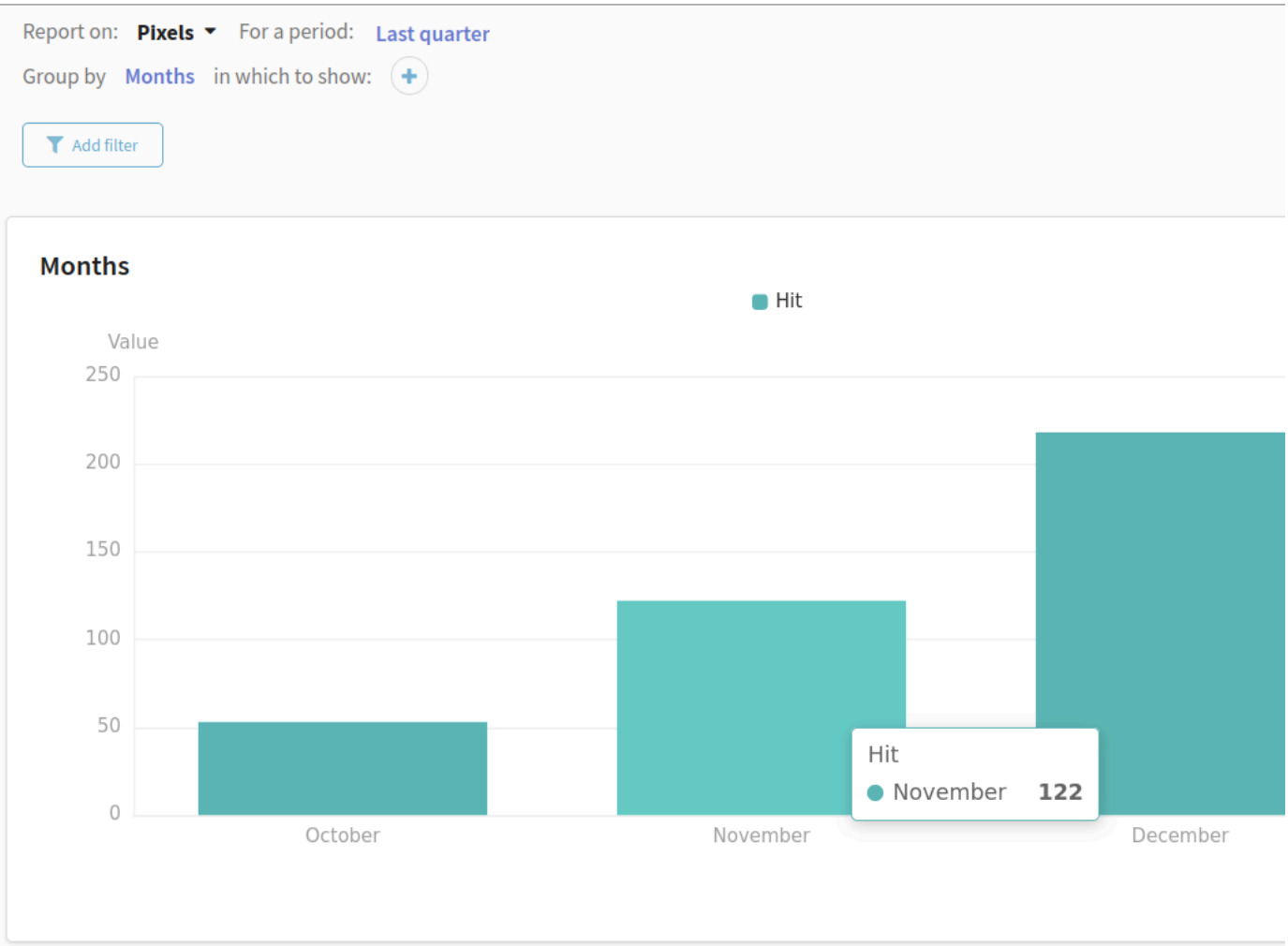
The report includes a table with data and a corresponding diagram. The chart can be collapsed by clicki

Diagram

The chart displays data based on grouping. When the report is grouped by days, weeks and quarters, the changes in the metrics for the selected period:



In other cases, the diagram shows the ratio of quantitative or percentage indicators in the selected peri



When you hover over the chart, you can see the exact figures.
The header contains the metrics displayed on the chart. Any column of the table can be displayed on the of indicators is 7. Click the metric to hide it.

Data table

The table contains quantitative and percentage indicators by traffic sources. Description of metrics can
Metrics are grouped based on the report settings. Depending on the details of the grouping, the table str
For example, if you set `"Group by days, in which to to show Pixels, in which to show* Pixel goals"` **, the table will d**

Name	Hit	Uniq. hit	Goal value
Total	16	16	2 800
2022-10-14	1	1	50
2022-10-11	15	15	2 750
#18 Test pixel	2	2	150
#17 Online store pixel	6	6	2 200
view	2	2	200
payment	3	3	1 500
card	1	1	500
#16 Website pixel	4	4	400

To add a new metric to the table, hover over the column name and click

. Select a metric from the list. If you want to remove a column, click

.

To move a column to another part of the table, you need to hover over the column name, hold down the le to the desired position.

Name	Send	Delivered	Delivery rate	Uniq. content clicl
Total	8 303 211	7 287 553	87,77 %	2 150 570
2022-11-11	3 270	746	22,81 %	368

Other options:

— switch quantitative metrics to percentages;

— display a column on a chart.

Comparing data

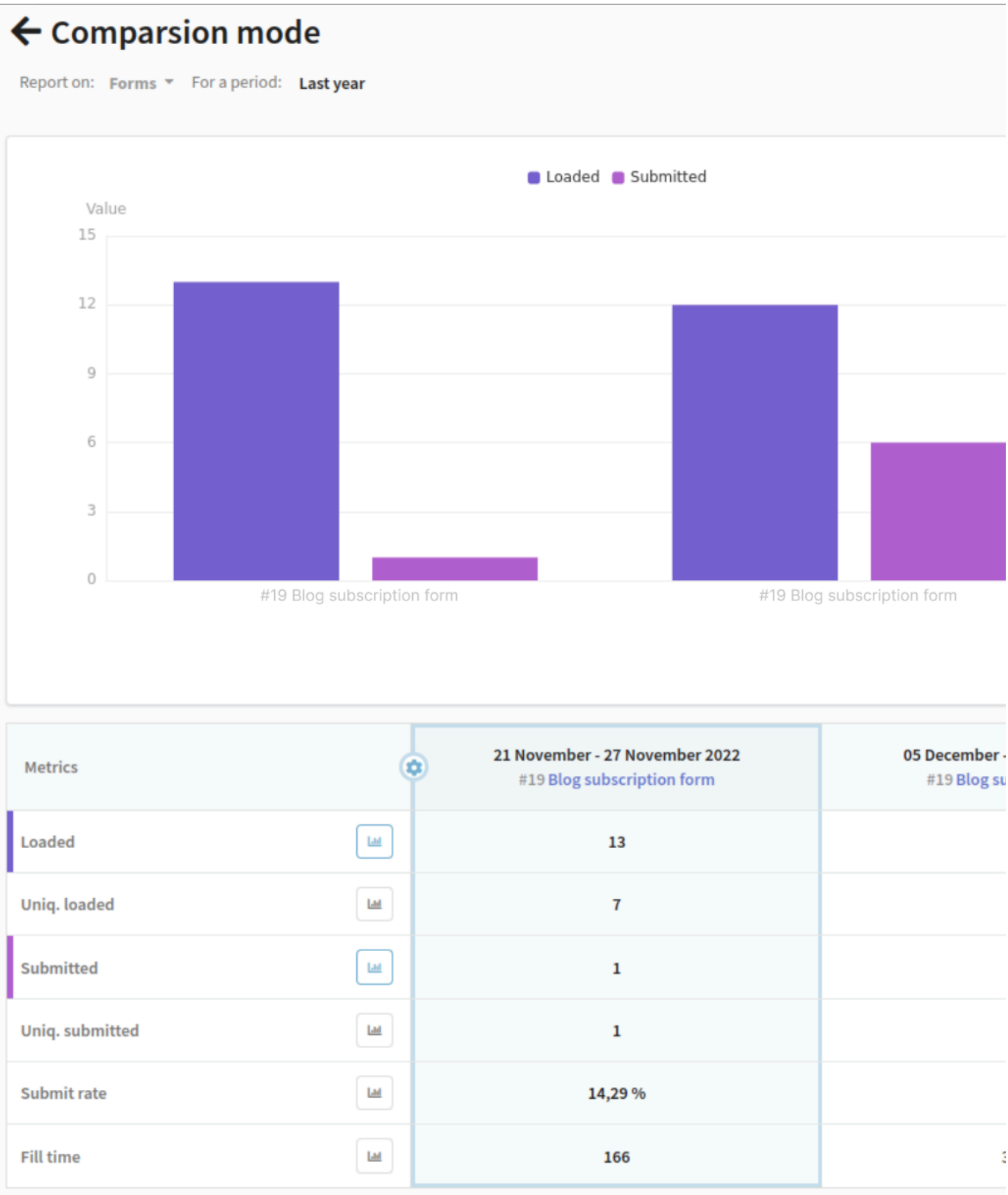
The comparison mode allows you to see data on several objects at the same time. For example, you can templates or evaluate the change in mailing metrics from period to period.

To switch to the comparison mode, select the objects in the table that you want to compare. Click “Com objects at different nesting levels.

Name	Compare	Loaded	Uniq. loaded	Submitted	Uniq. submitted
Total		265	155	42	36
26 December - 01 January 2022		11	9	2	2
19 December - 25 December 2022		12	8	2	1
12 December - 18 December 2022		5	3	1	1
05 December - 11 December 2022		13	7	7	4
#23 Webinar registration		1	1	1	1
#19 Blog subscription form		12	6	6	3
21 November - 27 November 2022		17	9	1	1
#23 Webinar registration		4	2	0	0
#19 Blog subscription form		13	7	1	1

The comparison mode also contains a chart and a table with metrics. In the diagram, the metrics for eac
The first object in the table is the comparison column. Other objects are compared with this object. The the metric is calculated relative to this object. To change the comparison column, click

and select a different object.



Metrics	21 November - 27 November 2022 #19 Blog subscription form	05 December - #19 Blog su
Loaded	13	
Uniq. loaded	7	
Submitted	1	
Uniq. submitted	1	
Submit rate	14,29 %	
Fill time	166	

To go back to the general mode, click ← Comparison mode.

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Report metrics

Report by	Metric	Description
Channels	**Opened**	**The number of messages opened by profiles.**
	Uniq. opened	**The number of unique message opens.** **If the same profile opens a message multiple times, only one unique open is counted.**
	Unsubscribed	**The number of unsubscribes from resources in all channels.**
	Subscribed	**The number of subscriptions to resources in all channels.**
	Confirmed	**The number of clicks on the confirmation link (link with the `confirm-link` attribute).**
	Hit	**The number of clicks on a link within a message.**
	Uniq. hit	**The number of unique clicks on a link in messages.** **If the same profile opens a message multiple times, only one unique hit is counted.**
Pixels	**Hit**	**The number of goals registered by pixels.**
	Uniq. hit	**The number of unique goal registrations.** **If a target is registered multiple times for the same profile, only one unique hit is recorded.**
	Goal value	**The sum of the values of all goals registered by the pixels.**
	Value per hit	**(Goal value / Hit) * 100%**
Forms	**Loaded**	**The number of times the form page was loaded.**
	Uniq. loaded	**The number of times the form page was loaded by a profile.** **If the same profile loads the form multiple times, only one unique load is logged.**
	Submitted	**The number of successfully submitted forms.**
	Uniq. submitted	**The number of unique submissions.** **If one profile submits the form multiple times, one unique posting is recorded.**
	Submit rate	**(Uniq. submitted / Uniq. loaded) * 100%**
	Fill time	**Duration of interaction with forms (in seconds). The time spent by each profile filling out the form is summed up.**
Pop-ups	**Hits to targeting**	**The number of times the conditions for the popup to appear, specified in the trigger, were met. However, the popup might not be shown to the us "Limits" blocks.**
	Shows	**The number of times the popup has been shown to users.**
	Uniq. shows	**If a popup is shown to a user multiple times within the same session, only one unique show is recorded.**
	Uniq. CTA	**The number of clicks as long as there are no fills, or the number of fills if there have been any.**
	Closed	**The number of clicks on the close button (closeButton).**
	Uniq. closed	**If a user closes a popup multiple times within the same session, one unique closure is recorded.**

Cohorts report

Description

A cohort in Altcraft analytics is a group of subscriber profiles from a specific database imported at the s
data empowers you with over time customer behavior statistics. You can use it to define how different g
lead providers for instance - interact with your marketing services and thus measure the real impact of I

> **TIP**
>
> **You can predict efficiency of future or recent marketing actions comparing cohorts and actions dates.**

Subscribers are divided into groups, which behavior is analyzed, in this report.

Comparing different subscriber cohorts you can retrieve actionable data on your lead engagement mailin
improve your customer engagement strategy.

Another option is to analyze activity rate of customer profiles from one database, imported at different t
minor marketing changes affect over time profile activity - especially changes that have no instant impac

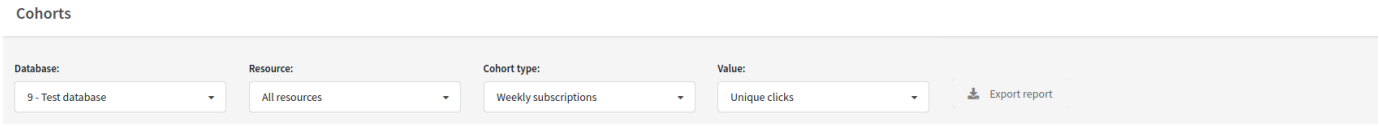
Managing cohorts

Diagram

> **TIP**
>
> **Start working with cohorts from selecting time period, database, resource, cohort type, and its value.**

Cohorts report can track changes in subscribers groups by next values:

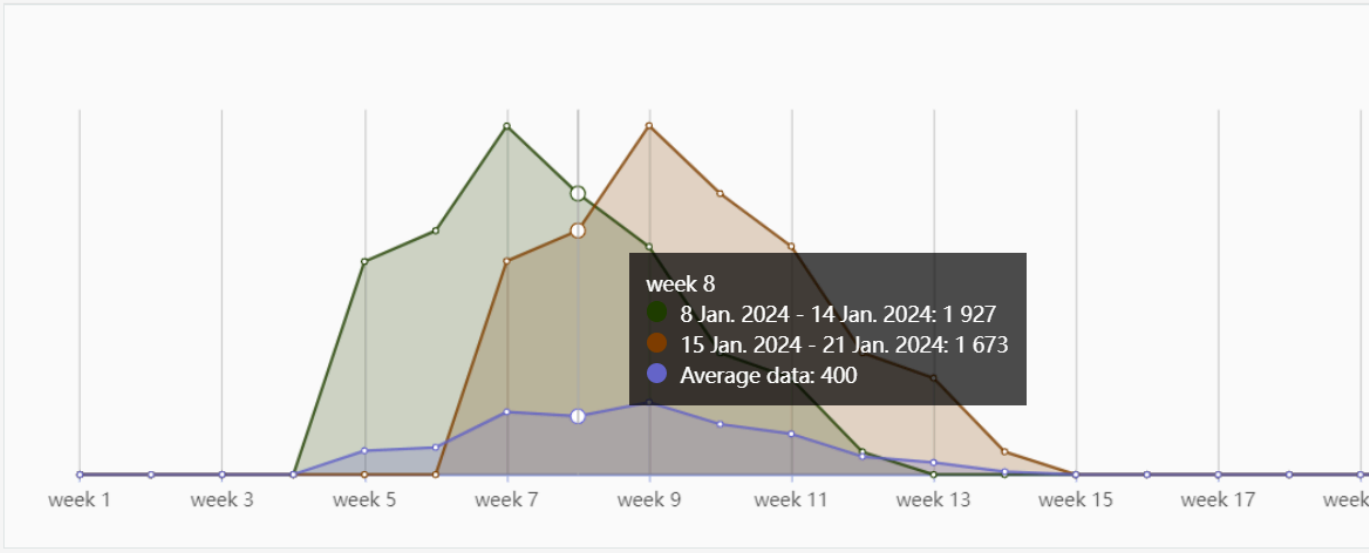
- **Unique clicks**
- **Unique opens**
- **Unique clicks to unique opens**
- **Unique opens to sends**
- **Unique clicks to sends**



Cohort consists of subscribers within certain period of time (week or month).

Cohorts in diagram can be grouped by days, weeks or months.

When you place cursor on the diagram, list of appropriate information will appear.



> **TIP**
>
> **Cohorts report can be downloaded in XLS or CSV formats.**

Table

The following information is displayed in the table: amount of actions selected in Value menu, and avera

Cohorts / Intervals	week 5	week 6	wee
1 Jan. 2024 - 7 Jan. 2024	0	0	0
8 Jan. 2024 - 14 Jan. 2024	1 461	1 672	2 39
15 Jan. 2024 - 21 Jan. 2024	0	0	1 46
22 Jan. 2024 - 28 Jan. 2024	3	1	3
29 Jan. 2024 - 4 Feb. 2024	0	0	0

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Lifetime report

Description

Customer lifetime report is an accurate tool to analyze customer cohorts lifetime. Lifetime here indicate or link clicks.

Graph columns represent average subscriber activity time for the specified subscribers cohort.

- **A maximum clicks per day amount for a cohort is retrieved.**
- **Then all other days results are divided by this maximum number. Thus relative value for each day is**
- **Then the relative results are summed to form an average customer lifetime, measured in days.**
- **If a day's result is less than 5% of a cohort maximum it will be skipped.**

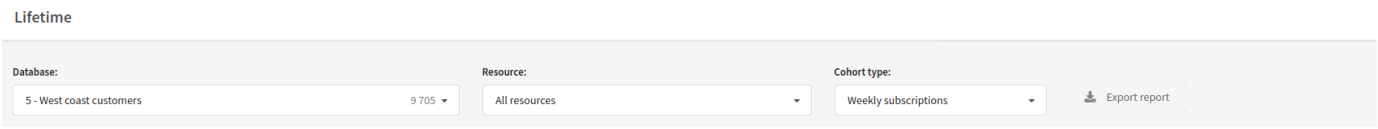
Cohort activity values in lifetime report table represent a relation between a specific time period and the

General settings

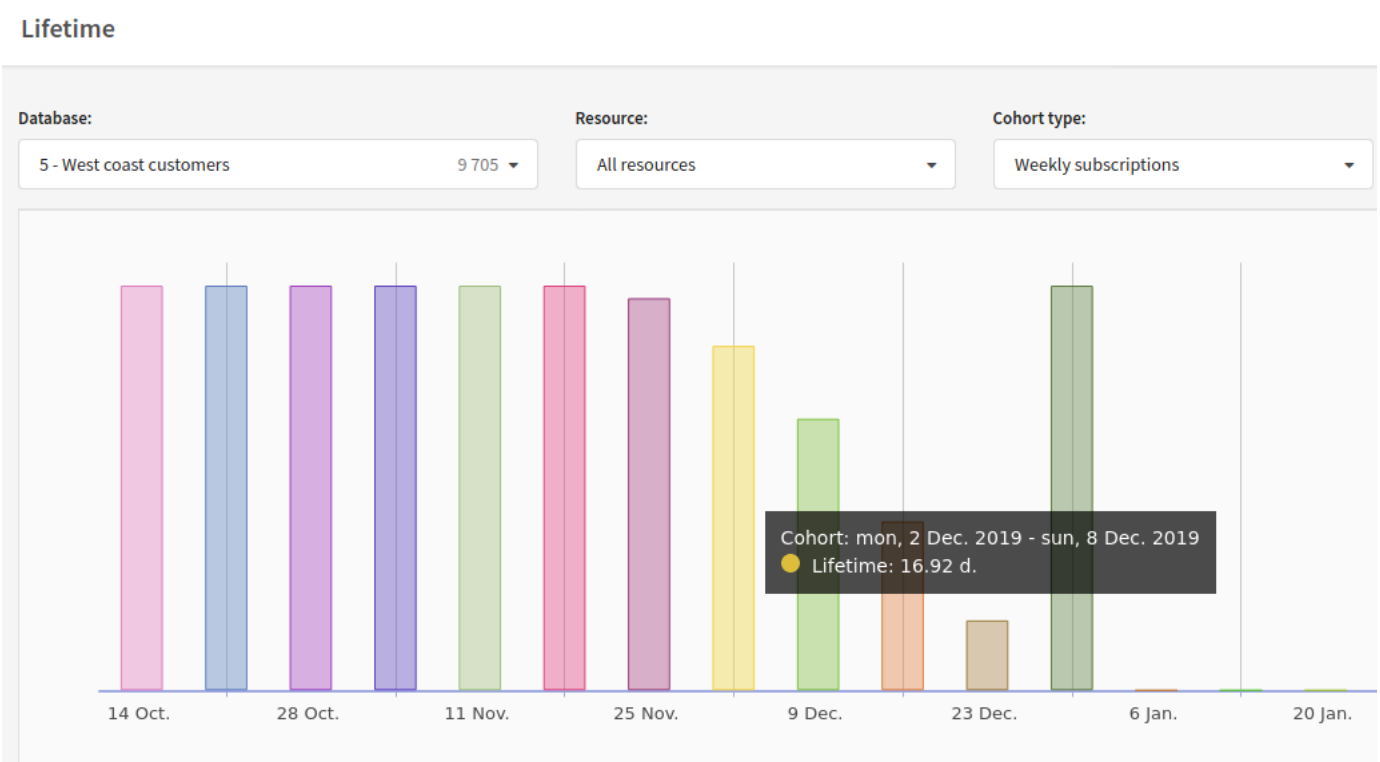
To calculate certain cohort lifetime, select database first.

After this, you can select resource. Also, you can select no resource, and lifetime report will be counted

You can also select cohort size - monthly or weekly.



In cohorts columns, you can see sequence of decreasing clicks amount by each week and decreasing in



INFO

To the right of the settings is the Export report button, with which you can download data in XLSX or C

Table

You can see ratio of clicks amount in one cohort to general amount of subscribers in cohort in the table.

Cohorts / Intervals	week 1	week 2	week 3	week 4
9 Dec. 2019 - 15 Dec. 2019	39.17%	100.00%	84.73%	0%
16 Dec. 2019 - 22 Dec. 2019	39.17%	100.00%	0%	0%
23 Dec. 2019 - 29 Dec. 2019	100.00%	0%	0%	0%
30 Dec. 2019 - 5 Jan. 2020	39.17%	100.00%	84.73%	36.28%
6 Jan. 2020 - 12 Jan. 2020	0%	0%	0%	0%
13 Jan. 2020 - 19 Jan. 2020	0%	0%	0%	0%

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

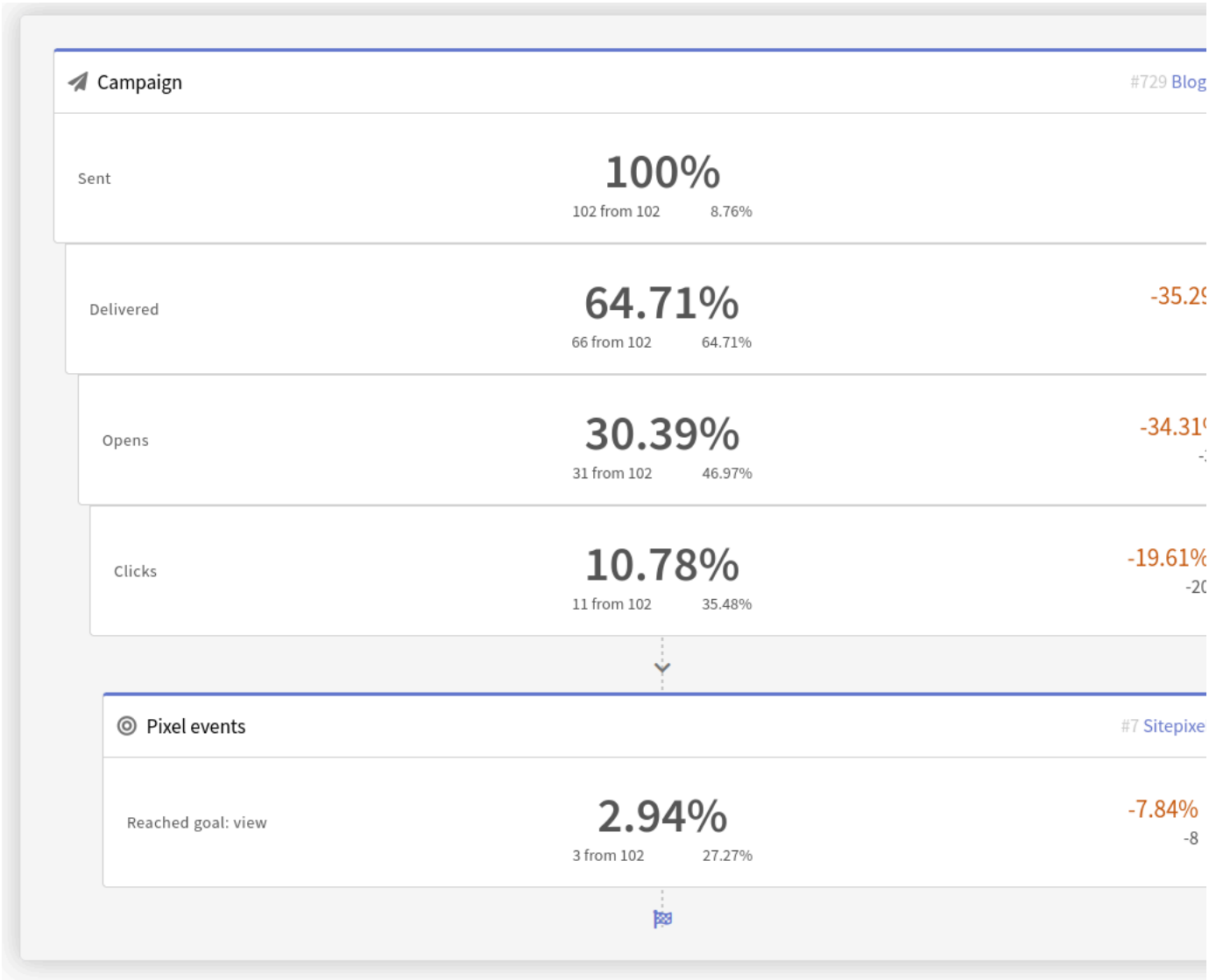
Funnels report

Description

Conversion funnel is a term that describes customers quantity changes as they go through the whole tra visitors to buyers or subscribers.

From the start number, only some part of the audience will go to registration form, and less of them will of this process is similar to the funnel.

Using a variety of available conversion funnel steps you won't miss a part of your customers' journey in r detect steps where conversion rate is high and perform appropriate actions.



What can be conversion funnel used for?

Conversion funnels can become your trusted providers of actionable marketing analytics. Each funnel st engaged in a certain activity (profile import, message reads and link clicks, web page visits or purchase Analyzing conversion funnel can help you to understand reasons for unsubscribing and test key elements of available conversion funnel steps you won't miss a part of your customers' journey in need of improve where conversion rate is high and act accordingly and precisely upon this data.

Calculation of the conversion period

Since the funnel report collects data from different sources, the calculation can take a long time in some working with the report, the date periods for it are calculated and stored in the platform.

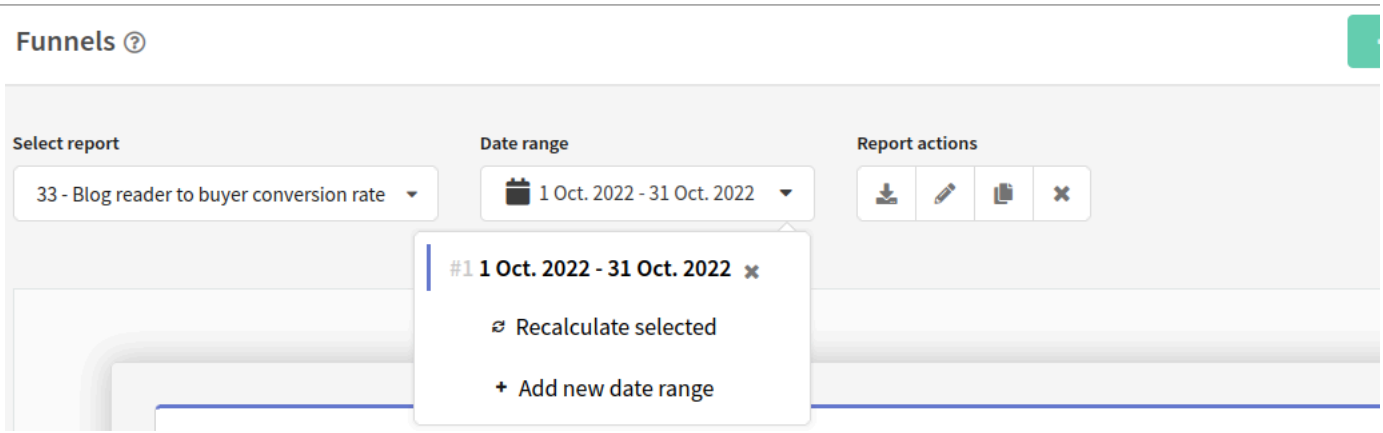
In the menu, select a ready-made report and the time period you are interested in. You can also add a ne the current one.

The following buttons can be found on the right side:

![img](/img/attachments/76809136/76809178.png) **Export report —**

![img](/img/attachments/9406257/edit_icon.png) **Edit report — modify the report;**

![img](/img/attachments/9406257/clone_icon.png) **Clone report — duplicate the report**



How to create a funnel?

To add a new conversion report, click Create in the top right corner of the page.

General settings

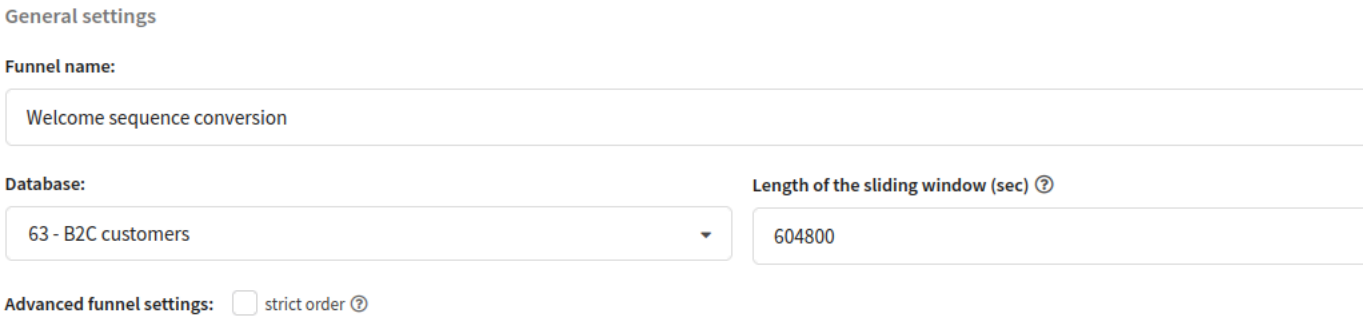
Enter the name and description of the new funnel, and select the database that stores the data of the cu in.

Next, specify the length of the sliding window. This is the time period between the first and last conditio specified in seconds and defaults to 604800 seconds (7 days).

For example, you leave the default length of the sliding window (604800 seconds) and add the following s *delivered —> Template #1 opened —> Achievement of the "purchase" goal*. In this case, the first conditio delivered to the profile, and the last condition will be met when the profile activates the pixel by his acti step of the funnel only if no more than 7 days pass from the moment the message is delivered to the mo same time, the order of actions specified in the funnel should be preserved.

TIP

30 days — 2592000 seconds
14 days — 1209600 seconds
1 day — 86400 seconds
1 hour — 3600 seconds



Additionally, you can enable strict order in the funnel. This setting does not allow other events to interve For example, you are building a funnel: *Mailing #1 delivered → Mailing #1 opened → Achievement of the "* second and third steps the platform recorded another event in the profile history, the profile will not proc Use a strict order if it is important to you that the profile perform a strict sequence of actions in a short

Building a funnel

The following blocks can be used in custom order in funnels:

- **Information by segment**
- **Information by template actions**
- **Information by mailing actions**
- **Information by goals reached**
- **Information by subscriptions**

CAUTION

The funnel counts the number of profiles, not the number of events. For example, if the platform record the funnel will only show one profile at that step.

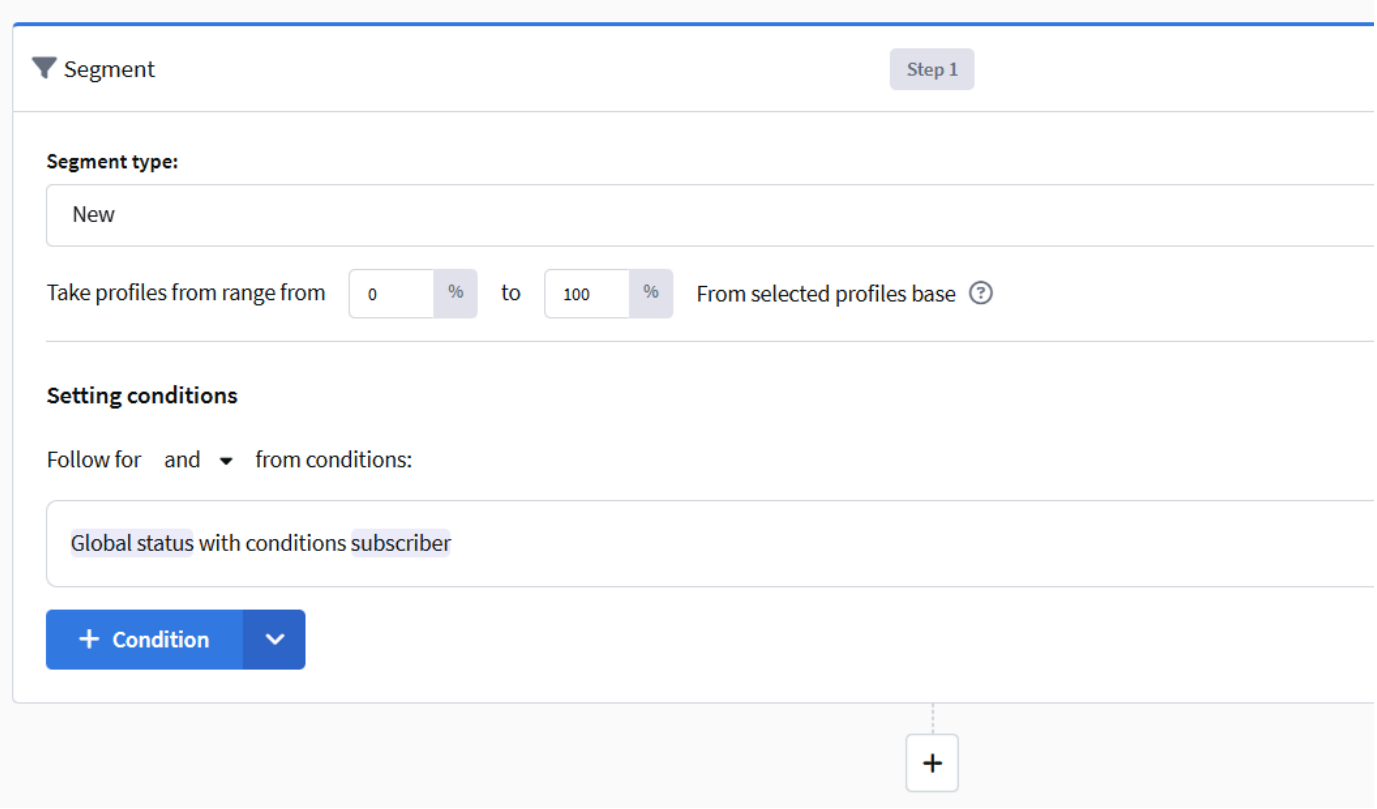
Add a step: segment

A segment as a funnel condition can only be added in the first step. This allows you to narrow the select

To get information about segment, you need to select segment type (new or existing) —> select amount be fixed number or range —> add rule or rules group:

- **"AND" condition — subscribers that fit all rules and rules groups in query, will be selected.**
- **"OR" condition — subscribers that fit at least one rule or rules group, will be selected.**

—> find necessary segment. After this, you can add one more rule or rules group, or go to the next step.



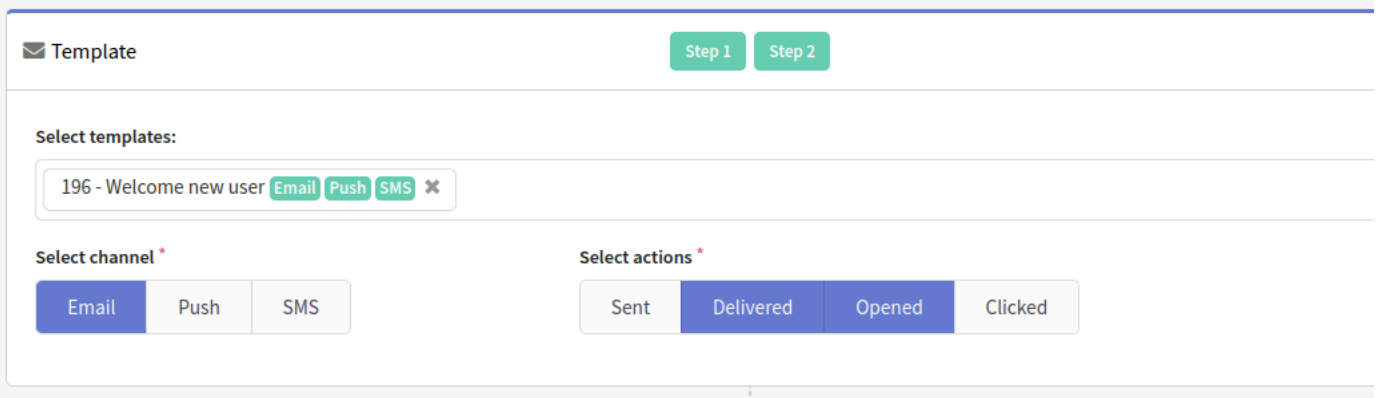
CAUTION

“Segment” step does not use selected time period for selection. It works only as filter.

Add a step: template

This block shows amount of profiles which performed selected actions in selected actions.

To get information about actions in the template, select “Add a step: template” from the options —> sele communication channel or channels (email; push; sms) —> select action or actions in message, that you select several actions, amount of steps will be increased.



CAUTION

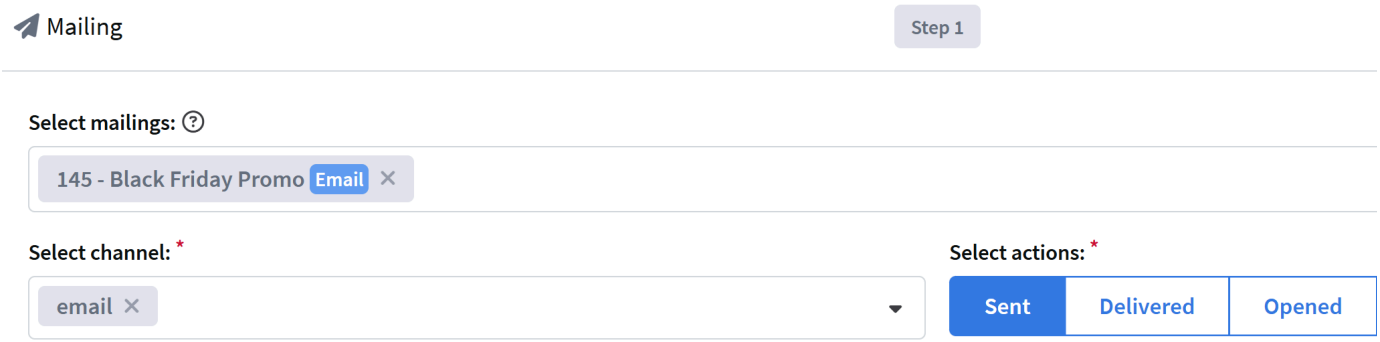
If a click event is registered in the SMS channel but there is no delivery event, the delivery event will be milliseconds before the click event.

Add a step: mailings

This blocks allows you to count profiles by amount of certain actions by selected channels in the mailing

To get information by mailings, select the mailing, then select action or actions in message, that you wa

select several actions, amount of steps will be increased.



CAUTION

If a click event is registered in the SMS channel but there is no delivery event, the delivery event will be milliseconds before the click event.

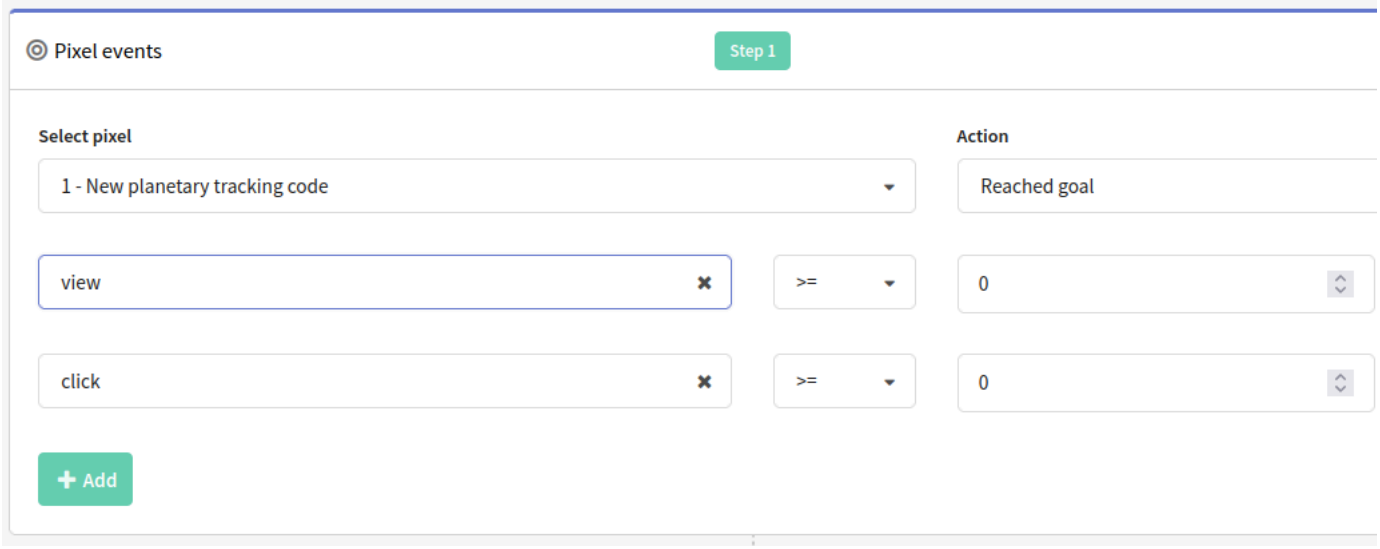
Add a step: goal reached

This block shows amount of subscribers that reached pixel goal or performed target actions. Available e

- **pixel visit — any event of a selected pixel;**
- **reached goal — an event that a client achieves one or more goals.**

To get information about the number of subscriber who reached a certain goal, you need to select a pixe and its value. Next, specify the pixel value that must be fixed for the profile to get to this funnel step. Th available:

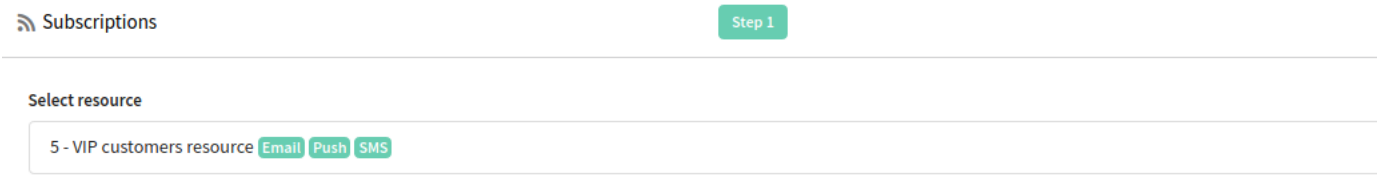
- **more or equal**
- **less or equal**
- **more**
- **less**
- **equals**



Add a step: subscriptions

This block shows information about profiles who subscribed to any channel of a certain resource. Inform subscriptions quantity.

To get information about completed subscriptions amount, select the resource.



INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Goals report

Description

Goals report are a report of how your clients have completed targeted actions and the value of taking th displays amount of pixels activated by website visitors depending on goals defined.

Goals can be attached to customer profiles using such Altcraft tools as:

- **Tracking pixel scripts hosted on site pages and other resources**
- **Node "Goal registration" in scenario workflow**
- **Goal Registration API**

> INFO
>
> **The report automatically adds up the value of actions with positive and negative values and displays it**

Managing report

At the top of the page, select the period for generating the report.

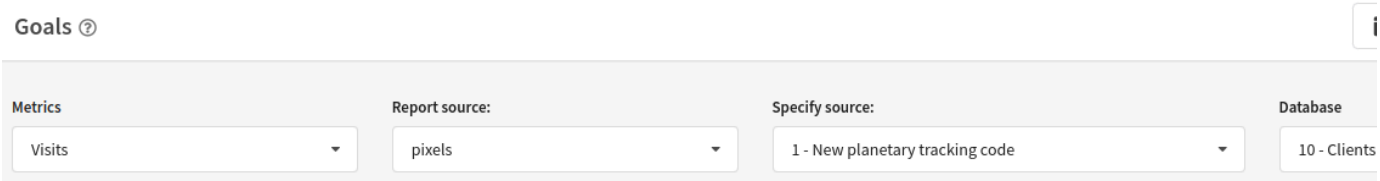
In goals report, you can select Report source and Metrics.

There are two available metrics:

- **visits — how many goal actions the user performed;**
- **profit — the sum of the action value.**

The report sources:

- **pixels — analytics for pixels for which goals are registered;**
- **goals — analytics for goals registered for different pixels.**



You can specify the source: select a pixel or goal to analyze. When choosing a specific pixel or goal, you for well-known clients:

> INFO
>
> **You can download the resulting report in MS Excel and CSV format by clicking the Export report button.**

Profit dynamics and goals achievement

The goal report graph shows the total number of visits as well as the total profit from the actions perfor

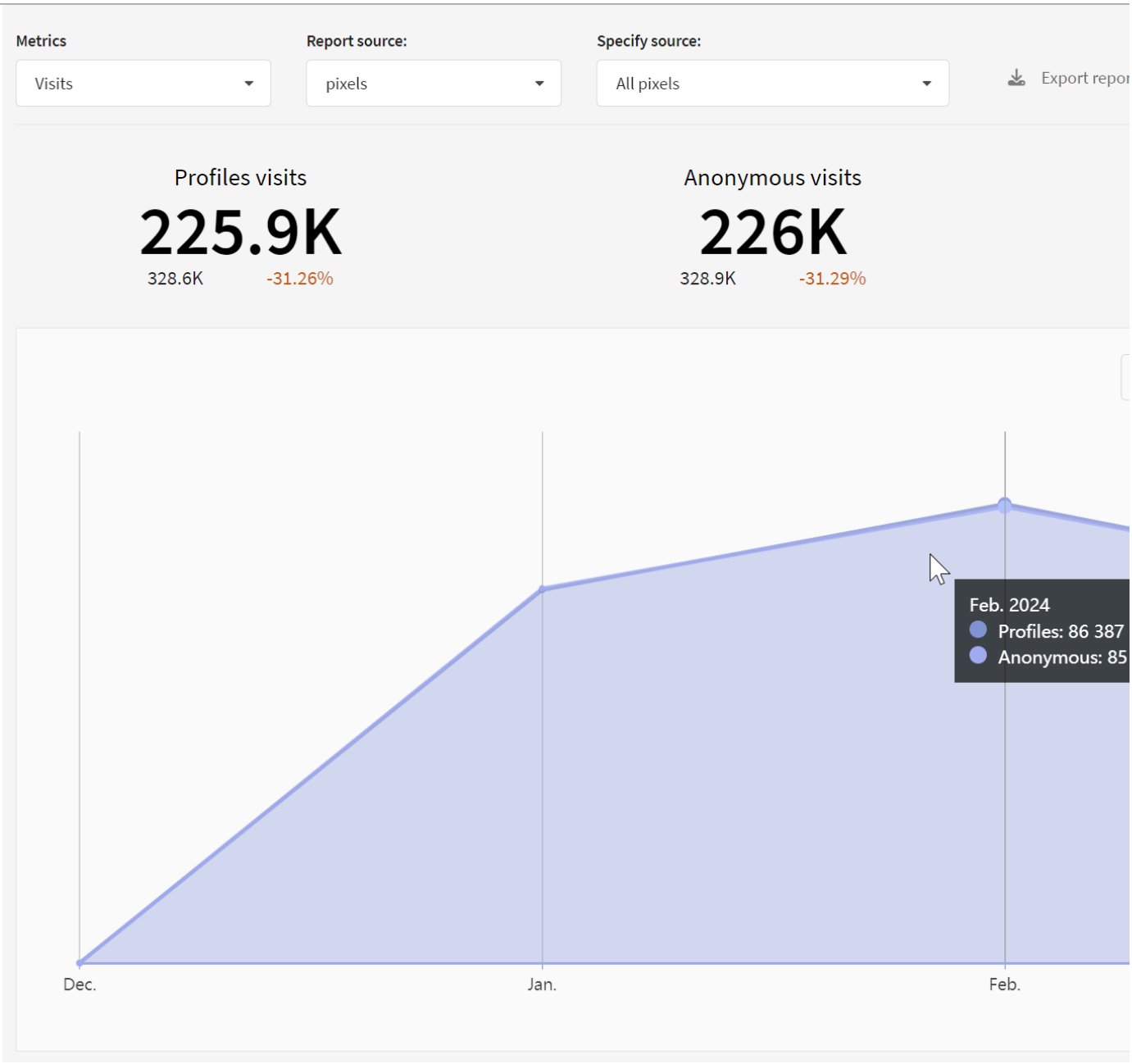
Profile visits give information about pixel activation by subscribers from databases. Anonymous visits gi from databases.

> INFO
>
> **Goals can be registered for clients whose contact and personal information you do not yet have. Such e Altcraft distinguishes them from each other based on connection data (cookies). If a client later registe data is added to the already existing profile.**

Using pixel, you can register certain income or another value at the moment of page visit or completing anonymous and profiles visits is summary profit.

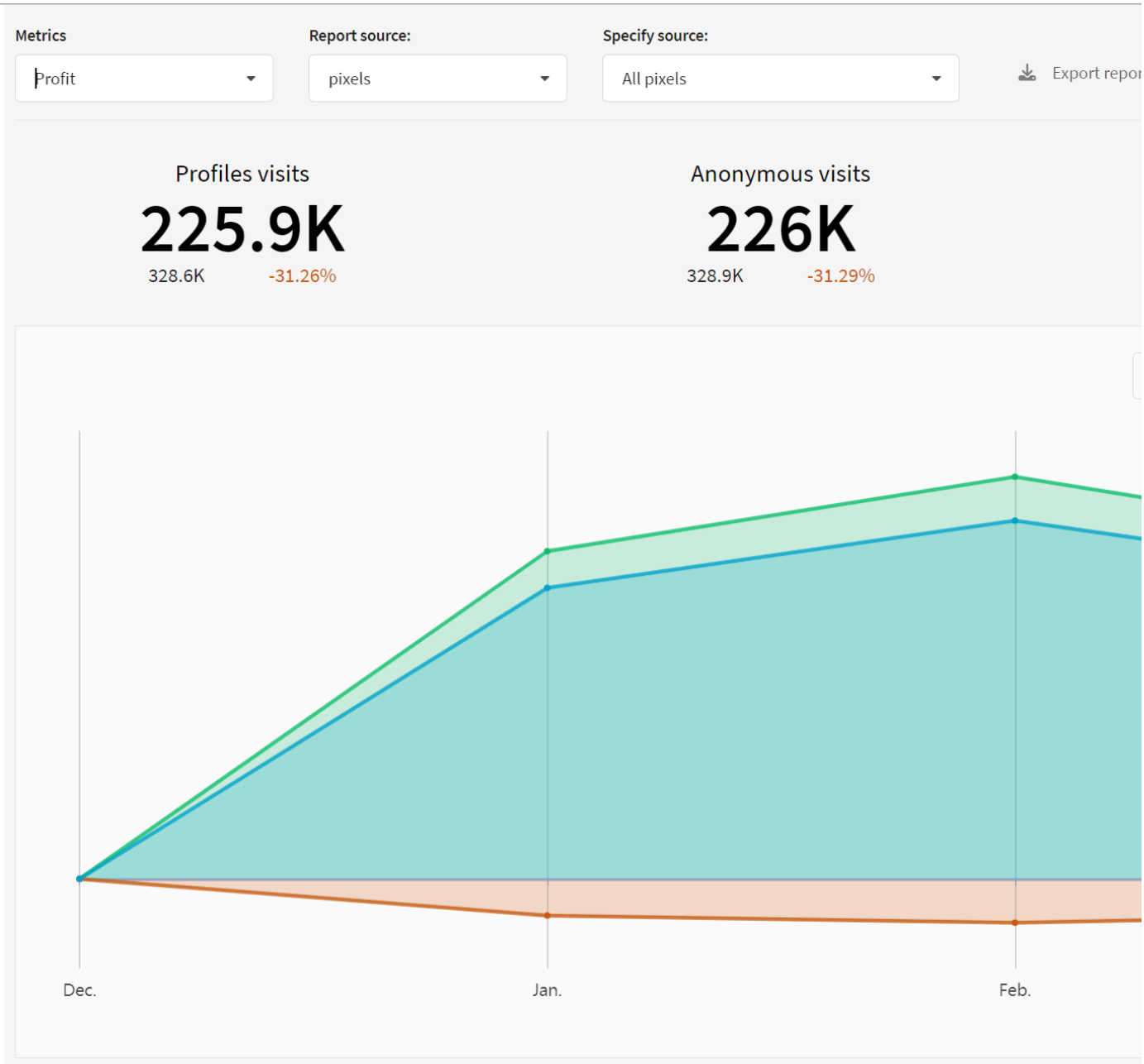
Visits metric

If you select Visits metric, on the graph you can see such information as the number of profiles visits an



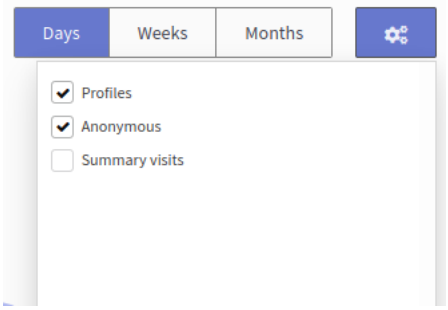
Profit Metric

If you select Profit metric, on the graph you can see such information as summary income, summary exp



The graph is built by days, weeks or months of the report period. Using the button

, you can remove unnecessary indicators from the chart or, conversely, add them.

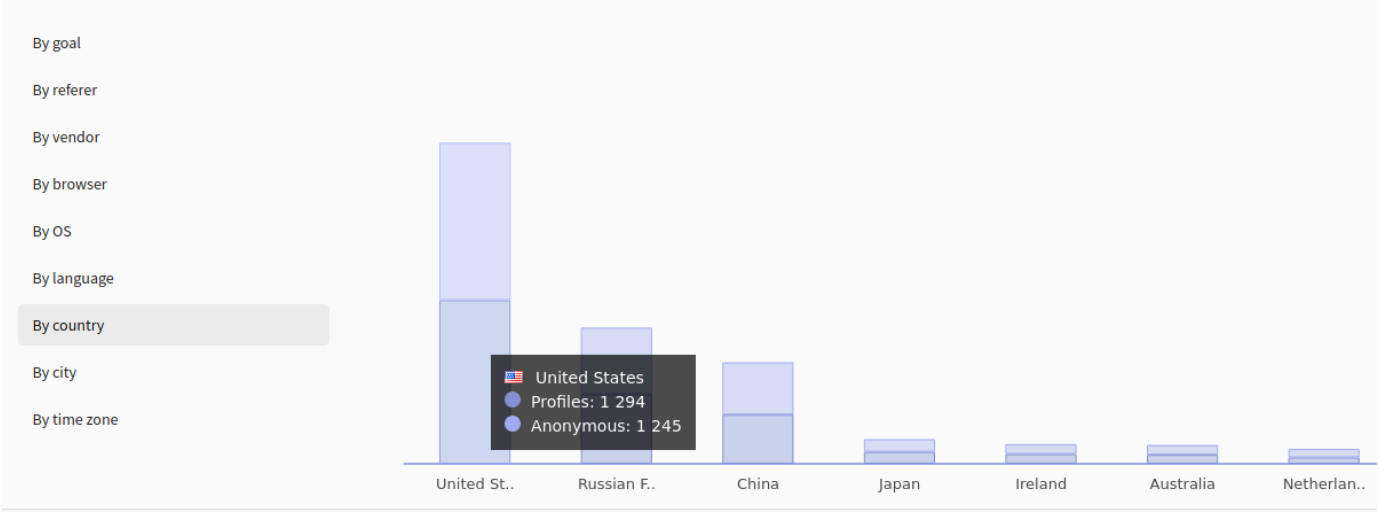


Analytical report slices

When registering a goal action, Altcraft collects information about the device used by the client, as well referer. Based on this data, you can additionally group the resulting audience:

- **By goal**
- **By referer**
- **By vendor**
- **By browser**
- **By OS**

- **By language**
- **By country**
- **By city**
- **By time zone**



Report indicators

The table at the bottom of the page contains a summary of the report based on the settings, slicers, and that will be uploaded during export. Click

to add columns to the table or hide unnecessary ones:

Profiles	Anonymous	Profiles income	Anonymous income	Profiles expenses	Anonymous expenses	Profiles profit	Anonymous profit
68	74	11 411.2	13 017.5	-7 801.9	-3 447.5	3 609.3	9 570
26	24	4 404.2	3 604.3	-3 400.7	-1 223.5	1 003.5	2 380.8
21	28	3 803.4	2 605.3	-1 600.3	-2 201.1	2 203.1	404.2
35	27	4 003	5 206.6	-2 801.4	-1 600.3	1 201.6	3 606.3
27	33	2 805.4	9 005.5	-3 046.3	-1 600.8	-240.9	7 404.7
346	367	66 658.3	56 667.3	-26 699.4	-29 770.1	39 958.9	26 897.2
39	40	6 607.8	5 206.1	-2 224	-3 601.3	4 383.8	1 604.8
30	34	5 604.8	5 006	-1 400.7	-2 600.8	4 204.1	2 405.2
31	40	4 003.5	6 607.8	-2 424.1	-3 401.2	1 579.4	3 206.6
15	32	3 600.8	4 205.1	-1 000.5	-1 600.8	2 600.3	2 604.3
2 779	**2 837**	**476 868.2**	**450 325.4**	**-233 764**	**-227 302.1**	**243 104.2**	**223 023.3**

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

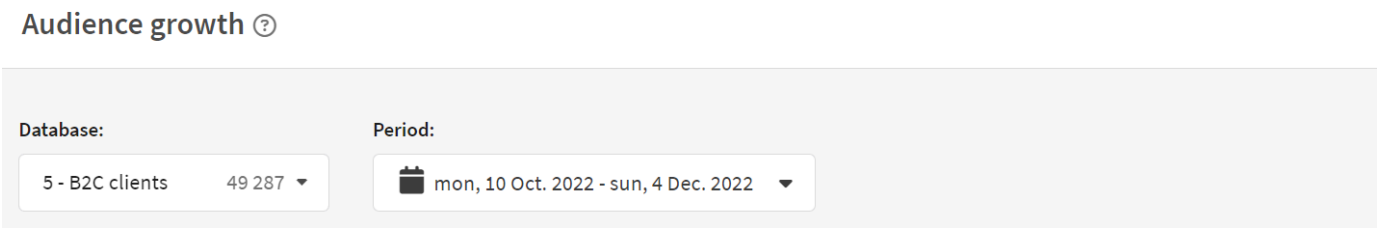
Audience growth report

Description

The audience growth report allows you to track the increase in the number of profiles in your database.
new subscriber profile additions and does not account for unsubscribes or deletions.

Report settings

At the top of the page, select the period for generating the report.
By default, the report is generated for all available profile databases. You can use the dropdown menu to

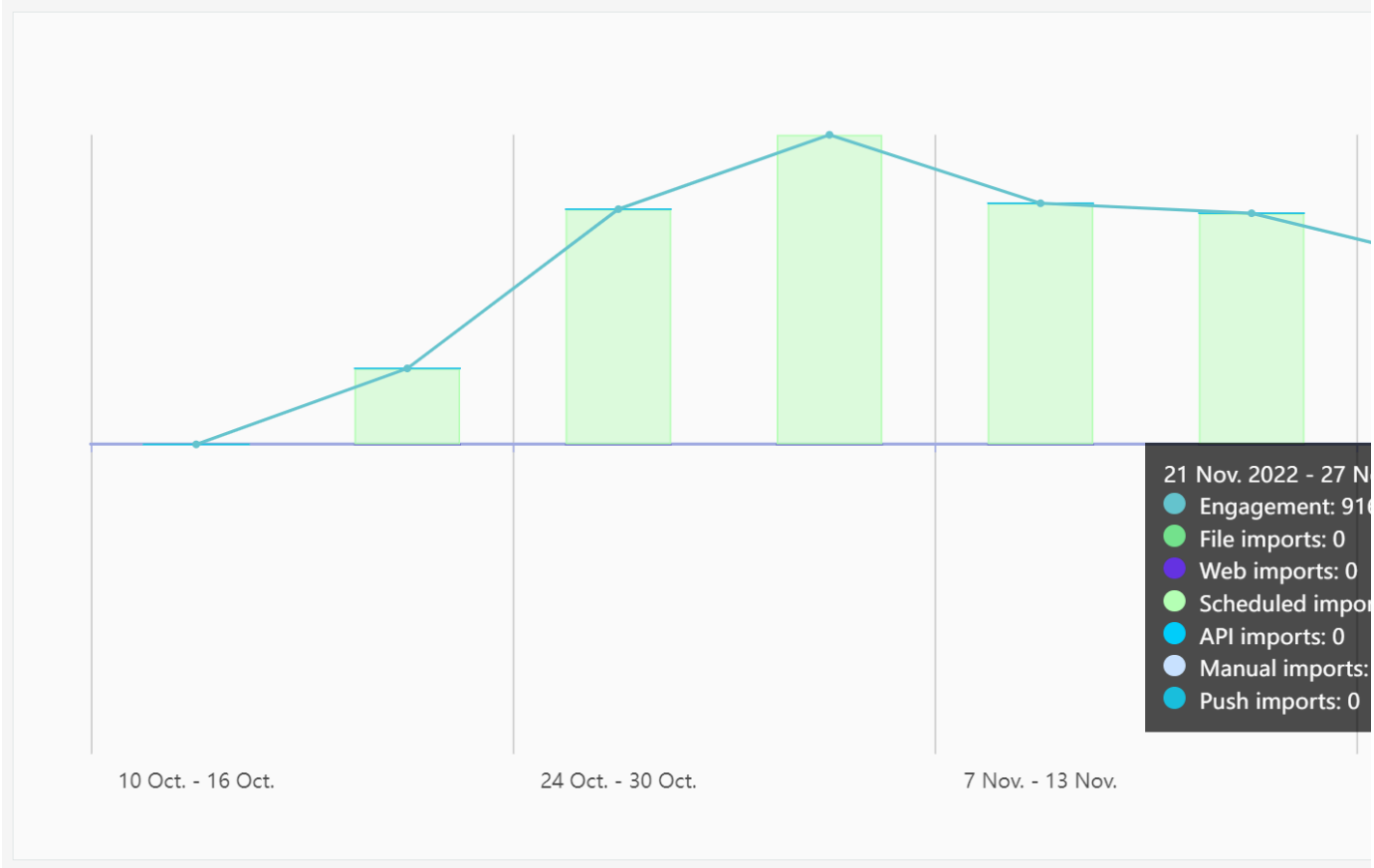


TIP

You can download report in MS Excel or CSV formats.

Growth graph

The visual part of the report displays the dynamics of audience growth. The data can be presented in th
weeks, or months of the selected period.



Audience dynamics metrics

Audience growth is represented by the following import metrics:

- **Engagements — net increase in the number of profiles in the database (excluding unsubscribes or ot profiles).**
- **File imports — adding multiple profiles from a table file.**
- **Web imports — creating new profiles at the moment of subscribing through the form or pop-up publis**
- **Scheduled imports — creating of new profiles on a schedule from an external SQL database.**

- **API imports — creating new profiles via API.**
- **Manual import — creating profiles through the platform interface.**
- **Push imports — creating anonymous and personal profiles at the moment of subscribing to push cha**

Date	Engagement	File imports	Web imports
28 Nov. 2022 - 4 Dec. 2022	+1	0	1
21 Nov. 2022 - 27 Nov. 2022	+916	0	0
14 Nov. 2022 - 20 Nov. 2022	+1 178	0	0
7 Nov. 2022 - 13 Nov. 2022	+1 229	0	0
31 Oct. 2022 - 6 Nov. 2022	+1 578	0	0
24 Oct. 2022 - 30 Oct. 2022	+1 199	0	0
17 Oct. 2022 - 23 Oct. 2022	+387	0	0
10 Oct. 2022 - 16 Oct. 2022	0	0	0
Totals:	6 488	0	1

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

TIP

Audience growth report displays the increase in the number of profiles in the database. If you need to t subscriptions to a resource, use Traffic report and apply the filter.

Click map report

Description

Click map is a report on your customers' clicks on links in a sent Email message. The click map displays Email message, including unsubscribe links. The click map helps to determine the quality of the templat successful location and design of links.

INFO

Click map is available only for Email messages. This feature is not supported for other sending channel

TIP

It should be noted that the report is based on a template, not on mailings. If the template was used in s summarized.

Statistics Collection Features

When analyzing click map data, consider:

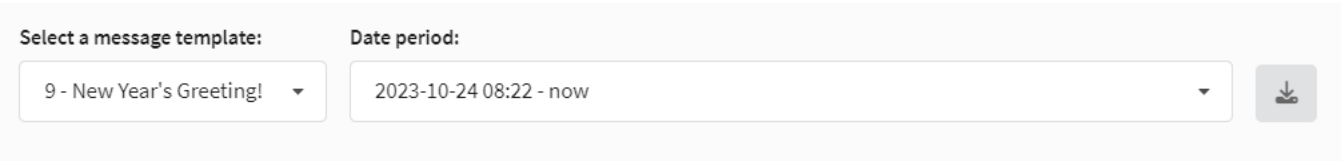
- **Total unique clicks (displayed at the top) - unique clicks across the entire message for the selected**
- **Clicks on individual links (in the table) - unique clicks for each specific link**
- **Click uniqueness is determined within 24 hours - repeated clicks from the same user within 24 hours**
- **All clicks are counted, including unsubscribe link clicks that may have been performed through emai**
- **Dynamic links - for personalized URLs, statistics are collected by the final domain, not by the origina**

CAUTION

Numbers in overall statistics and detailed table may differ for several reasons: different uniqueness cal of clicks (including unsubscribe links), dynamic link processing features, and platform data processing

Managing click map

To start working on click map report, you need to select the template that you want to analyze. If you ha example, for different mailings, select the required version from the drop-down menu Date period:



TIP

You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Click map report displays only unique clicks for all links in email message, including unsubscription links

Additional information is displayed on the right:

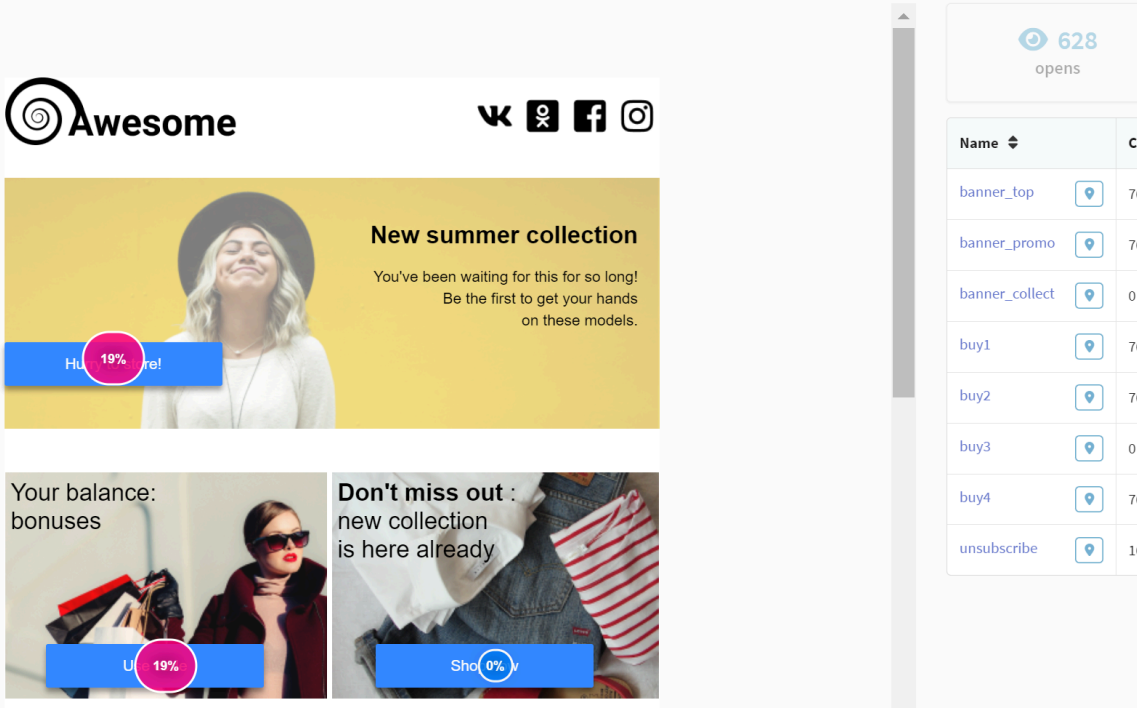
- **The number of unique openings of the template**
- **The total number of unique clicks**
- **The number of unsubscribed customers**

Below is a click chart and a table with the number of clicks on a particular link, percentage of total click the message:

Click map

Select a message template: New items in stock

Date period: 2024-08-16 09:34 - now



There is a geo-location button next to each link. When you click on it, you are transported to the area of

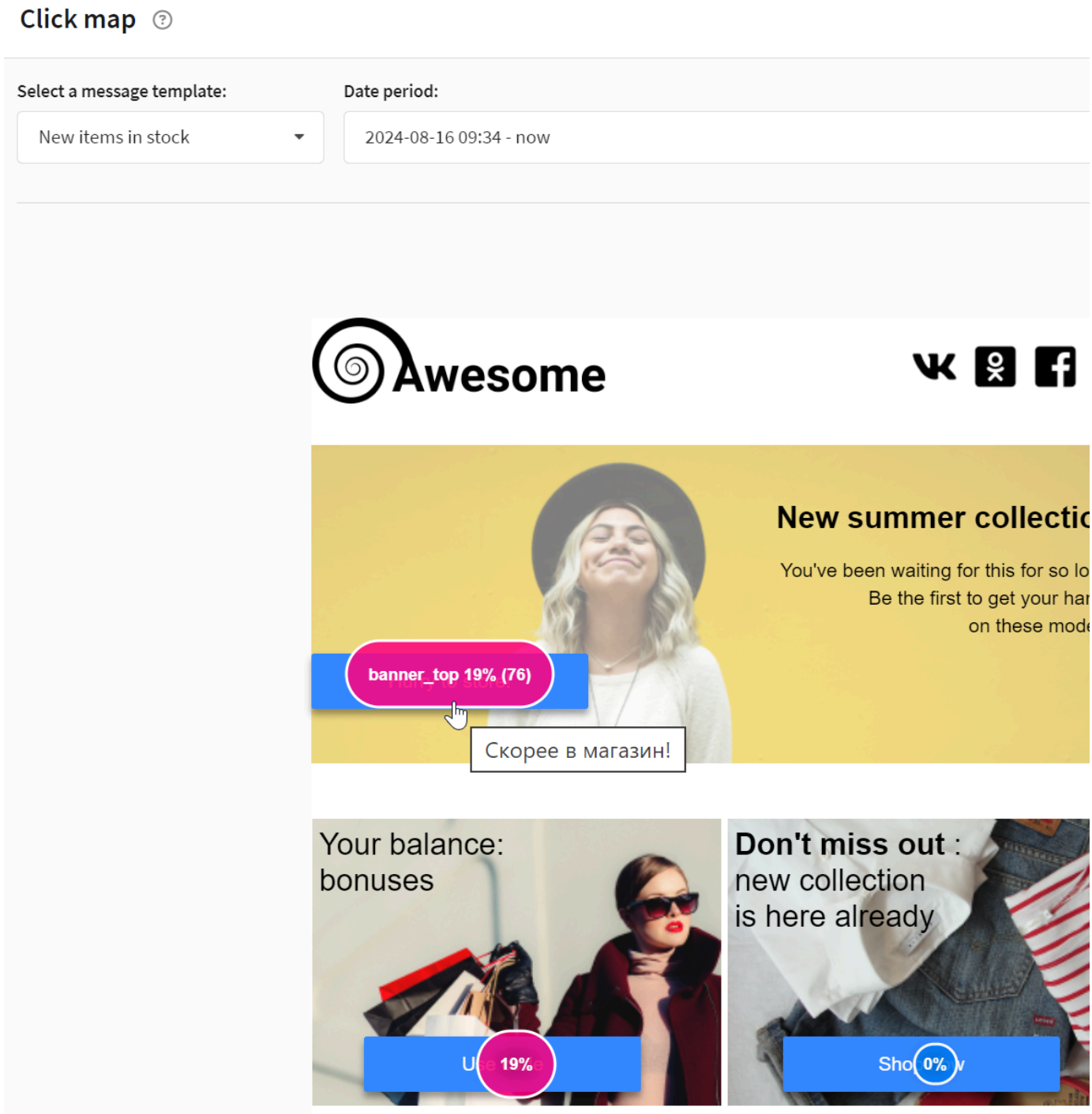
628 opens

396 clicks

Name	Click	%	CRT
banner_top	76	19%	12%
banner_promo	76	19%	12%
banner_collect	0	0%	0%
buy1	76	19%	12%
buy2		19%	12%
buy3	0	0%	0%
buy4	76	19%	12%
unsubscribe	16	4%	3%

Show on the map

This makes the search for links more convenient, especially if there are a lot of them in the template or links scattered around.



INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Loyalty programs report

**The loyalty program report shows key loyalty program metrics: points usage, transactions, purchases, a
a clear view of customer engagement and overall program effectiveness.**
To open the report, go to Analytics → Loyalty in the main menu.

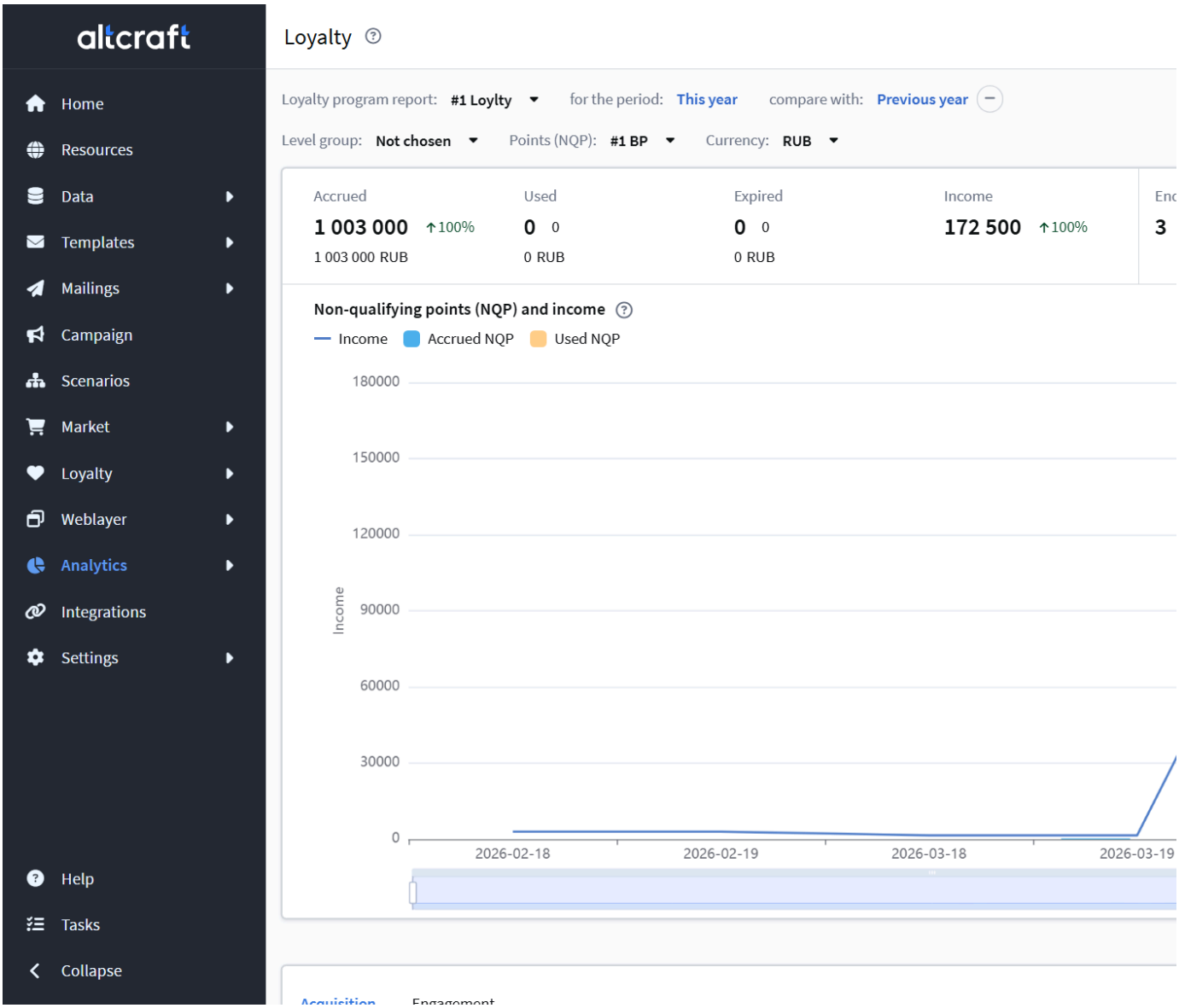
Main report metrics

> **TIP: What is NQP**
>
> **One of the key metrics used in analytics is NQP points. NQP are points that do not participate in loyalty activity. In other words, these are bonus points that a member earns and spends.**

Select a loyalty program, the reporting period, and a comparison period. You can also choose a tiers gro
Main report metrics:

- **Accrued — amount of NQP accrued to members during the period;**
- **Redeemed — amount of NQP redeemed by members during the period;**
- **Expired — amount of NQP that expired during the period;**
- **Revenue — total value of orders placed by loyalty members during the period;**
- **Members at period end — number of loyalty program members (recalculated at the end of the day);**
- **NQP balance at period end — total amount of NQP held by loyalty program members (recalculated at**

Below the metrics, charts are displayed: member distribution by tiers and a chart of revenue, accrued, a

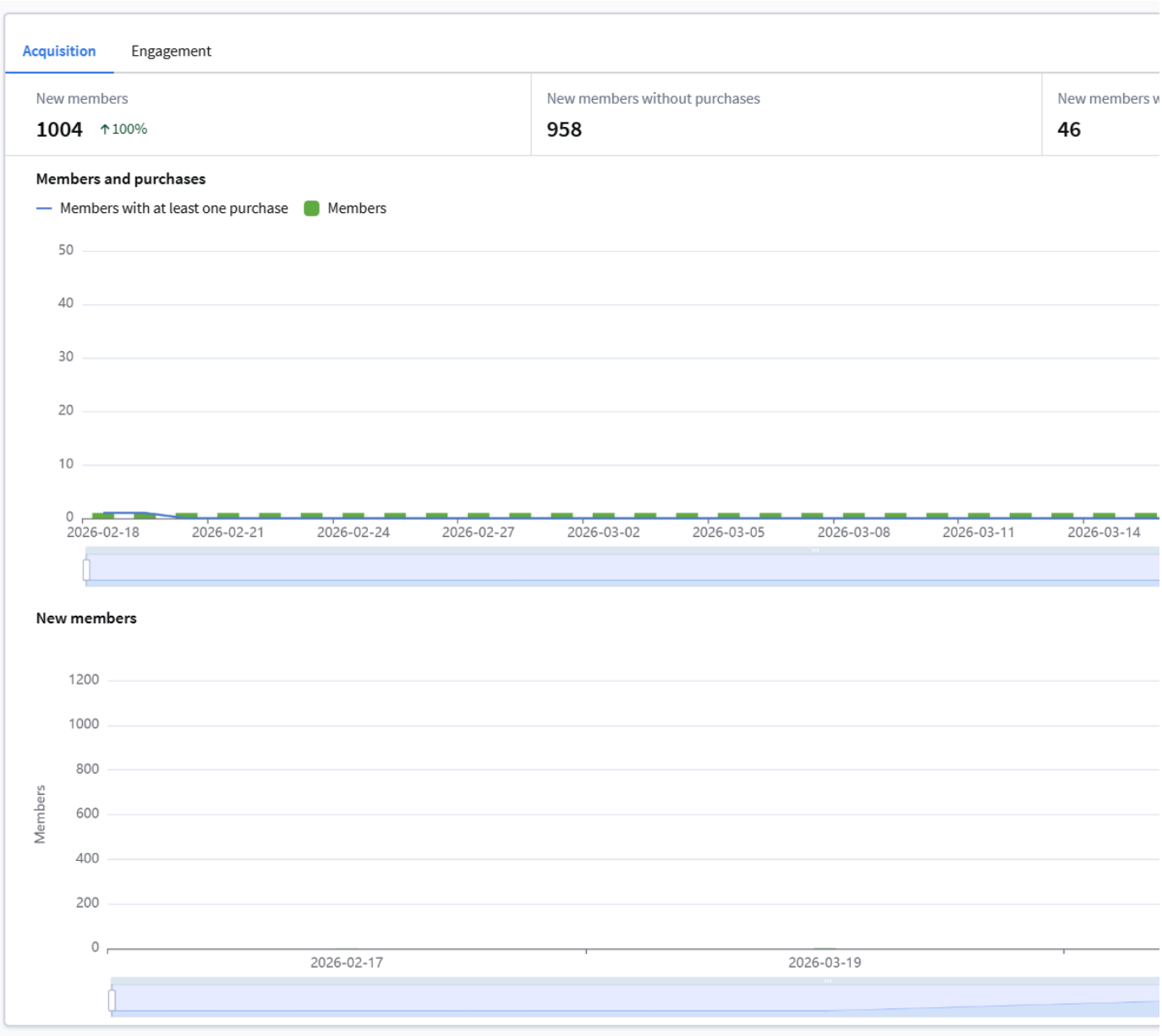


Acquisition

This section contains data on acquiring new loyalty program members:

- **Number of new members, including members without purchases and members with at least one purc**

- **New members chart.**



Engagement

This section shows member engagement based on activity. It includes the Points, Purchases, Member ac
Points (NQP)
This tab contains:

- **Share of redeemed points relative to accrued points (for the period), including new members with an**
- **Charts for program members and their transactions;**
- **Expired NQP chart.**

Acquisition | Engagement

Points (NQP) | Purchases | Members activity | Membership time

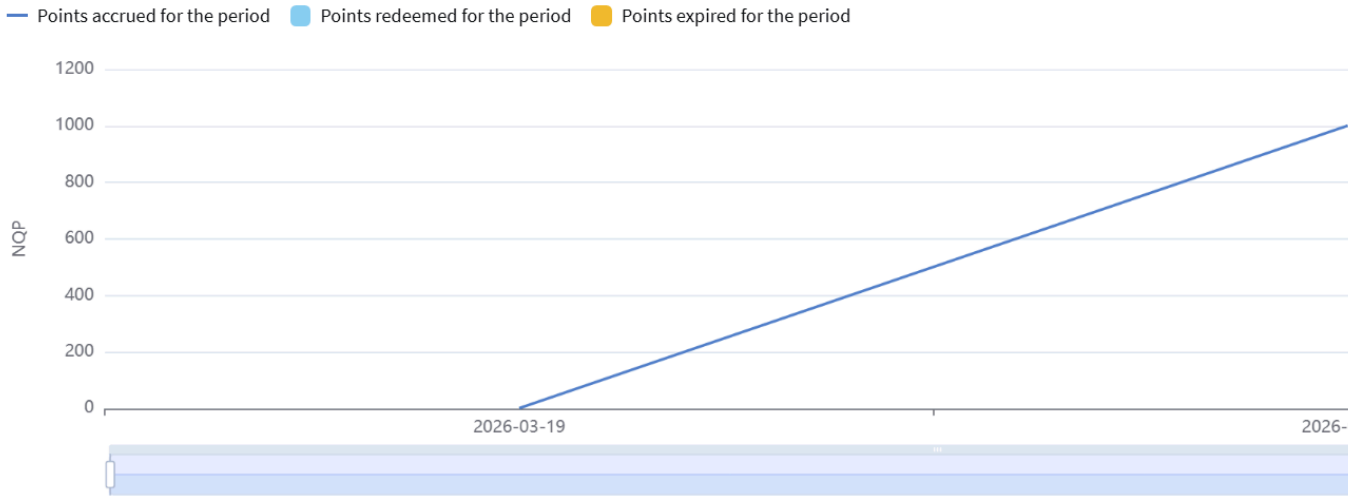
The share of points spent from those accrued for the period (Redeeming rate)
0%

Members with one transaction
1 003

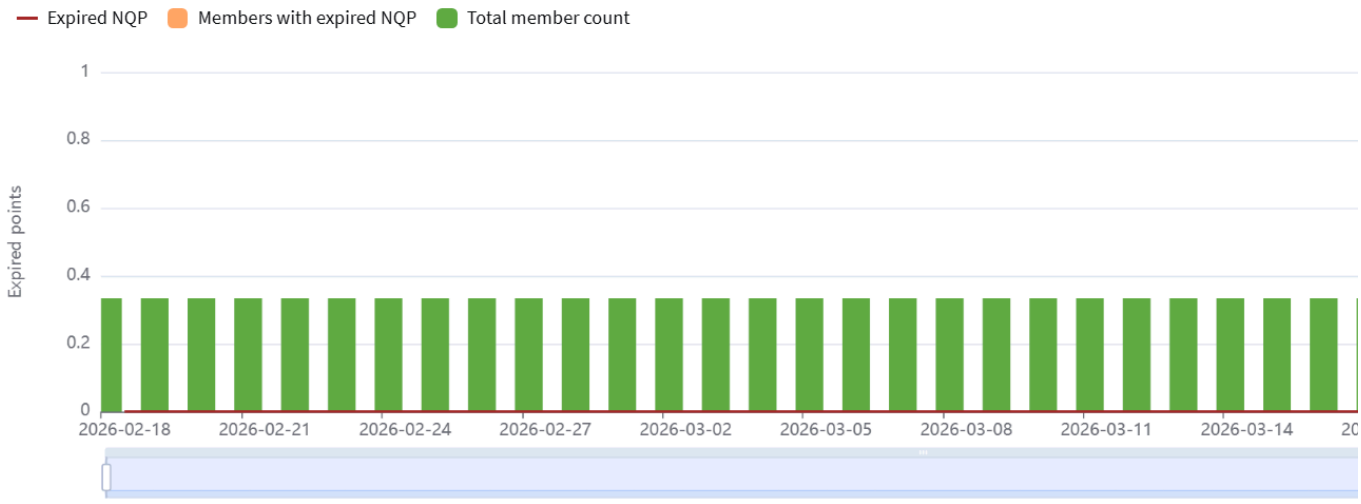
Members
1

Members and point transactions

Number of transactions

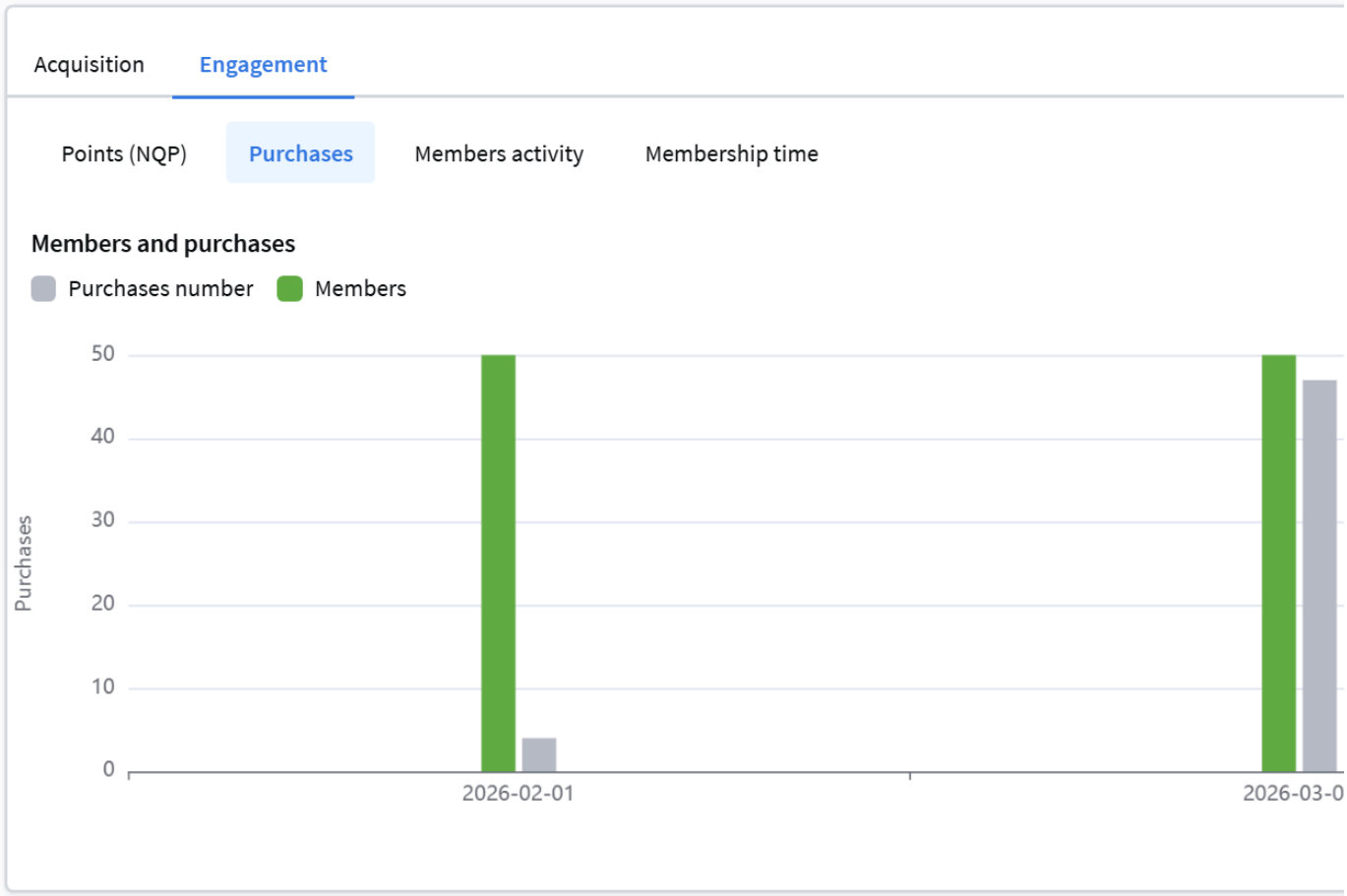


Expired NQP



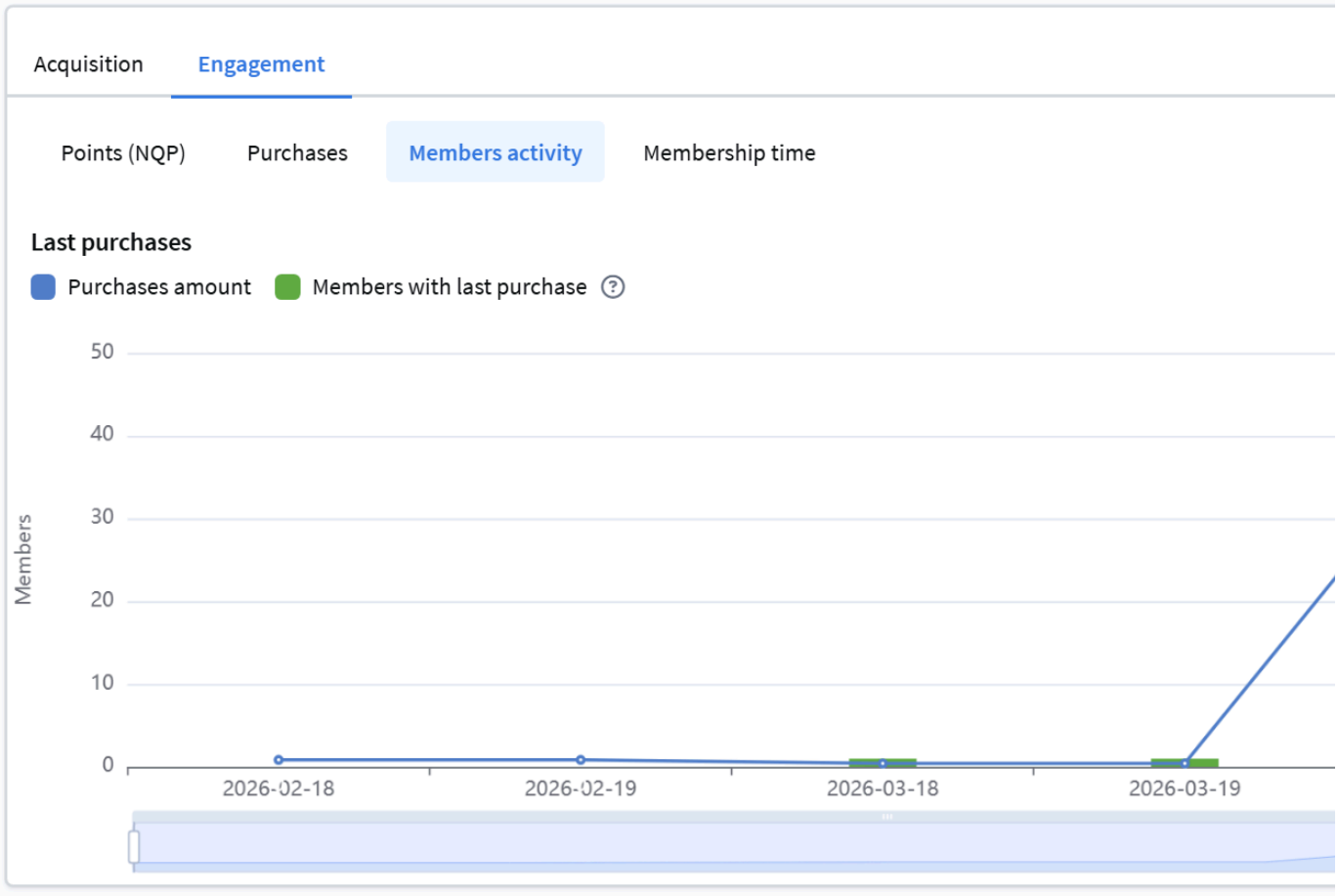
Purchases

This section displays purchase data for loyalty program members.



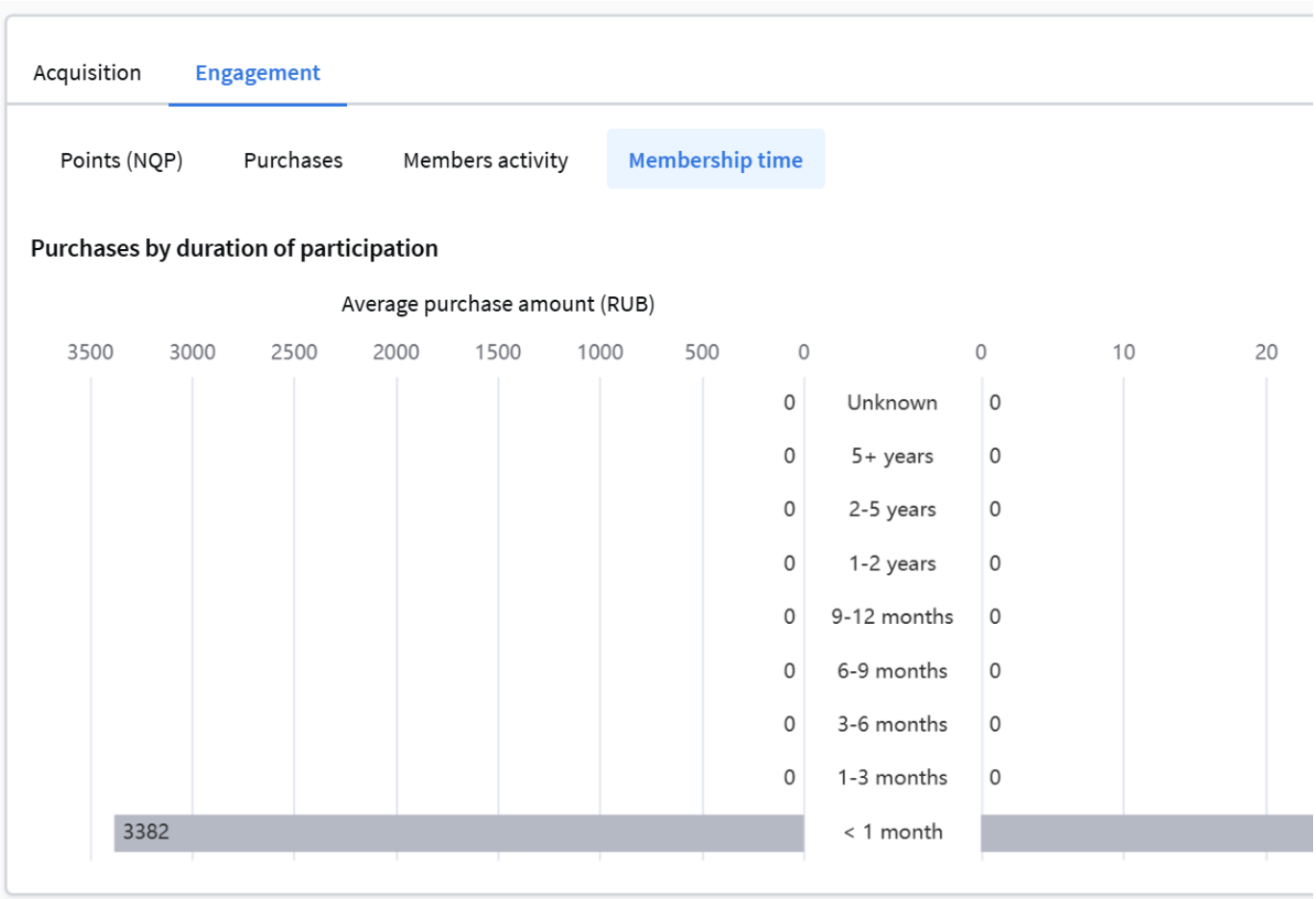
Member activity

This chart shows purchase totals by month and members whose last purchase occurred in the correspo



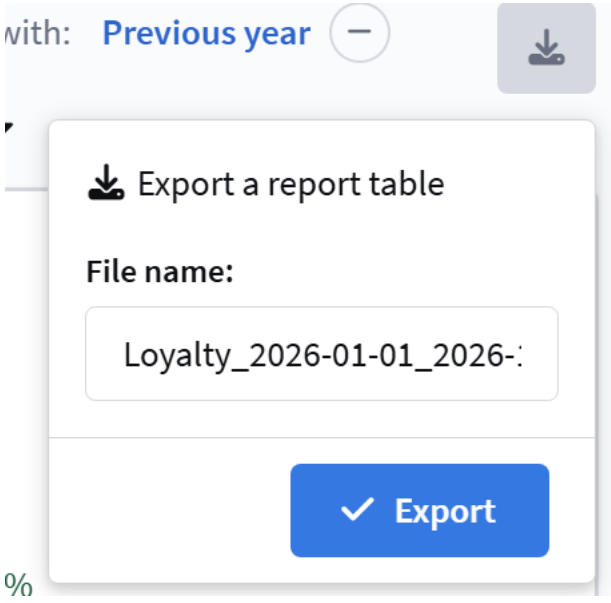
Membership duration

This section contains a histogram with data on members, their average purchase amount, and the length program.



Report export

To export the report, click the corresponding icon in the upper-right corner of the screen. Enter a report



An export task for the loyalty program report will be created. You can download it from the Tasks sectio

Bounces report

Hard and Soft bounces

In such cases, when the message cannot be delivered to the recipient, the mail server sends back an er these errors: Hard bounce and Soft bounce.

Hard bounces are server class errors (5xx). Such errors occur in cases, when sending to the mail server does not exist. If the server returns a Hard bounce error, the platform changes the profile status to Hard to the global supression list. As long as the address is in the global supression list, you will not be able t remove an address from the global supression list only in the administration panel in the **Supression lis

> CAUTION
>
> **Email addresses with Hard bounce errors are collected in the supression list located in the administrat confused with the Global suppress list in the user panel.**

Soft bounces are sending client errors (4xx). Unlike Hard bounce, Soft bounce does not affect customer bounce is received, the platform tries to resend the message according to resend rules. If the message record a non-delivery event.

The reasons of Soft bounce can be:

- **Subscriber inbox is full**
- **Letter attachments are too big for subscriber mail server**
- **Email service is temporarily unavailable**

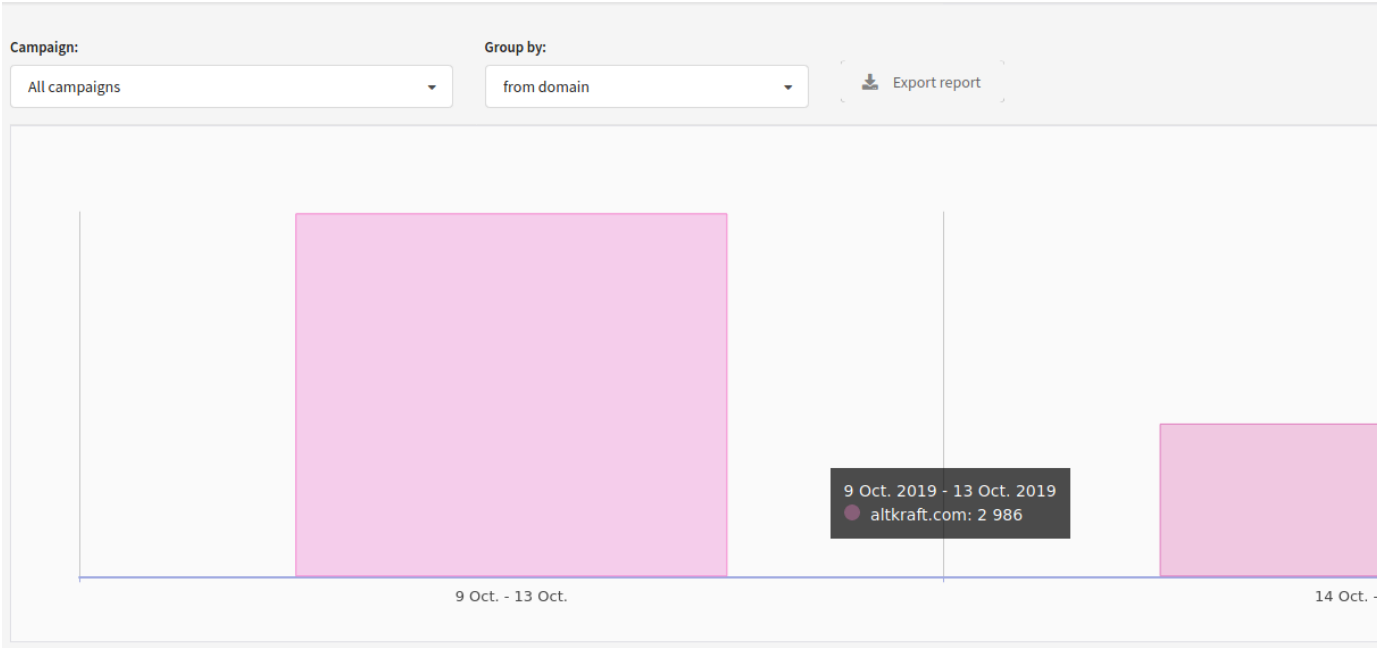
This report will allow you to track the number of Email messages that didn't get delivered to recipients t bounces are combined in this report.

Report settings

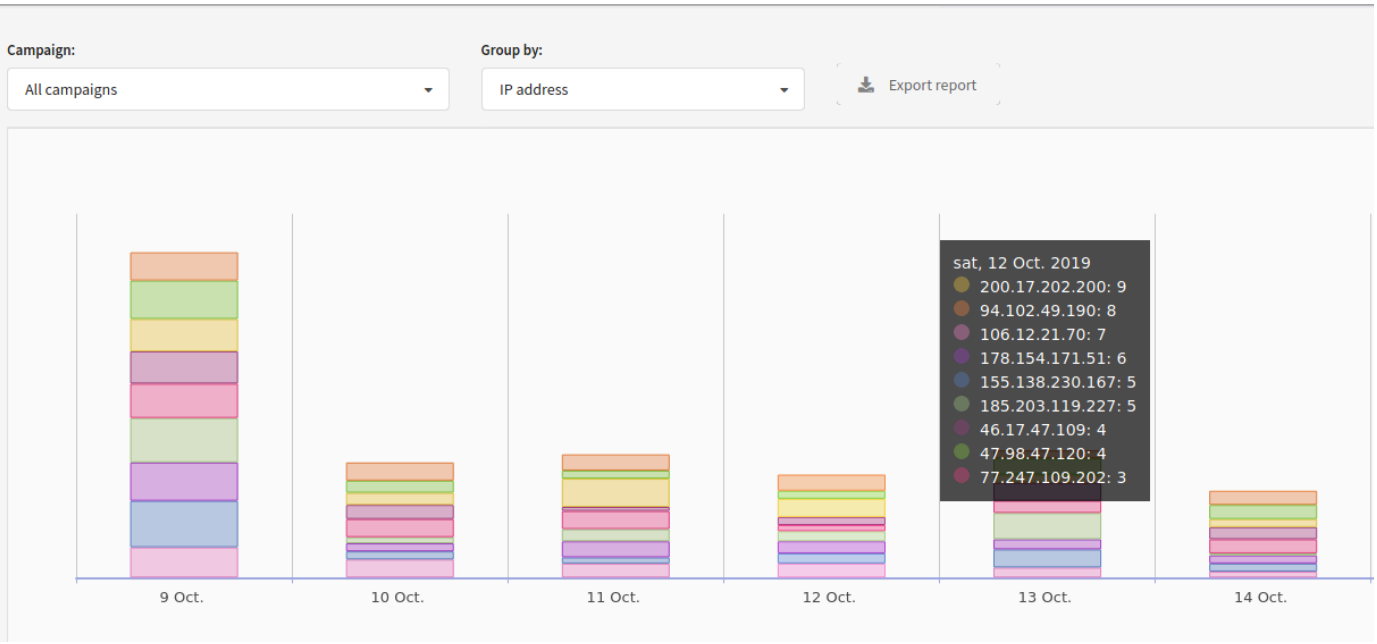
To start working on bounces report, you need to select time period and mailing (or all mailings).

Also, you can use grouping by from domain or IP address that was used for mailing.

Grouping by from domain:



Grouping by IP address



INFO

You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Table

The following information is displayed in the table:

- **email domain that was used for sending mailings;**
- **bounce text (reply from provider about customer email temporary unavailable);**
- **amount - number of message bounces if one message was sent several times.**

Group by	Bounce text	To domain	Amount
95.163.255.37	Your message was temporarily def..	wikipedia.org	4
94.102.49.190	This response is sent when the m..	ft.com	3
106.12.105.45	Your message was temporarily def..	ameblo.jp	3
172.104.108.109	Your message was temporarily def..	vk.com	3
185.234.219.238	This kind of response is sent ba..	ustream.tv	3
186.43.201.151	Your message was temporarily def..	com.com	3
134.209.163.72	Your message was not delivered b..	paginegialle.it	3
106.12.105.45	Your message was temporarily def..	vinaora.com	3
62.173.138.240	This response is sent when the m..	tamu.edu	3
200.17.202.200	This response is sent when the m..	tuttocitta.it	3
Totals:			**1 478**

To view the full error message, hover over the bounce text. A list of common soft bounce reasons and th table below:

Bounce text	Description
`421 Message source blocked temporarily. One or more sender authentication rules matched.`	**Messages from this sender are temporarily blocked due to a failure to meet email authentication rules. This can occur The sender's From-domain authentication settings should be checked.**
`421 Service Temporarily Unavailable`	**The server is temporarily unavailable, potentially due to overload or technical maintenance.**
`451 Please try again later`	**A temporary issue on the server side, likely due to overload or maintenance.**
`451 <<EMAIL>>: Recipient address rejected: Intentional policy rejection, please try again later`	**The recipient's email address was rejected based on server policy. This may happen due to incorrect settings on the re spam.**

Bounce text	Description
`450 <<EMAIL>>: Sender address rejected: Greylisted for 120 seconds! Please, see <URL>`	**The sender’s address is temporarily rejected due to greylisting. The server temporarily refuses messages from the sen The platform will automatically retry after 120 seconds. If the issue persists after retrying, the sender’s settings should**

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Undeliveries report

Description

Undeliveries report displays information about number of Email, SMS, Push, Telegram Bot, MAX Bot, Vib not reach subscribers.

Most recent undelivery reasons are:

Channel	Reasons
Email	**Maximum number of attempts to send message has been reached** **Message has not been sent until end of its lifetime (24 hours)** **Message has been removed from messages queue by user** **Message has been sent to mailbox which does not exists**
SMS	**Study your provider's documentation**
Push	**Invalid push token in a resource subscription**
Telegram Bot	**Invalid chat_id.**
Viber	**Study your provider's documentation (the reasons may vary)**
MAX Bot	**Invalid user_id**
WhatsApp*	**Study your provider's documentation (the reasons may vary)**
Notify	**Study your provider's documentation (the reasons may vary)**

In some cases it's impossible to determine a reason of undelivery. These messages delivery error marke

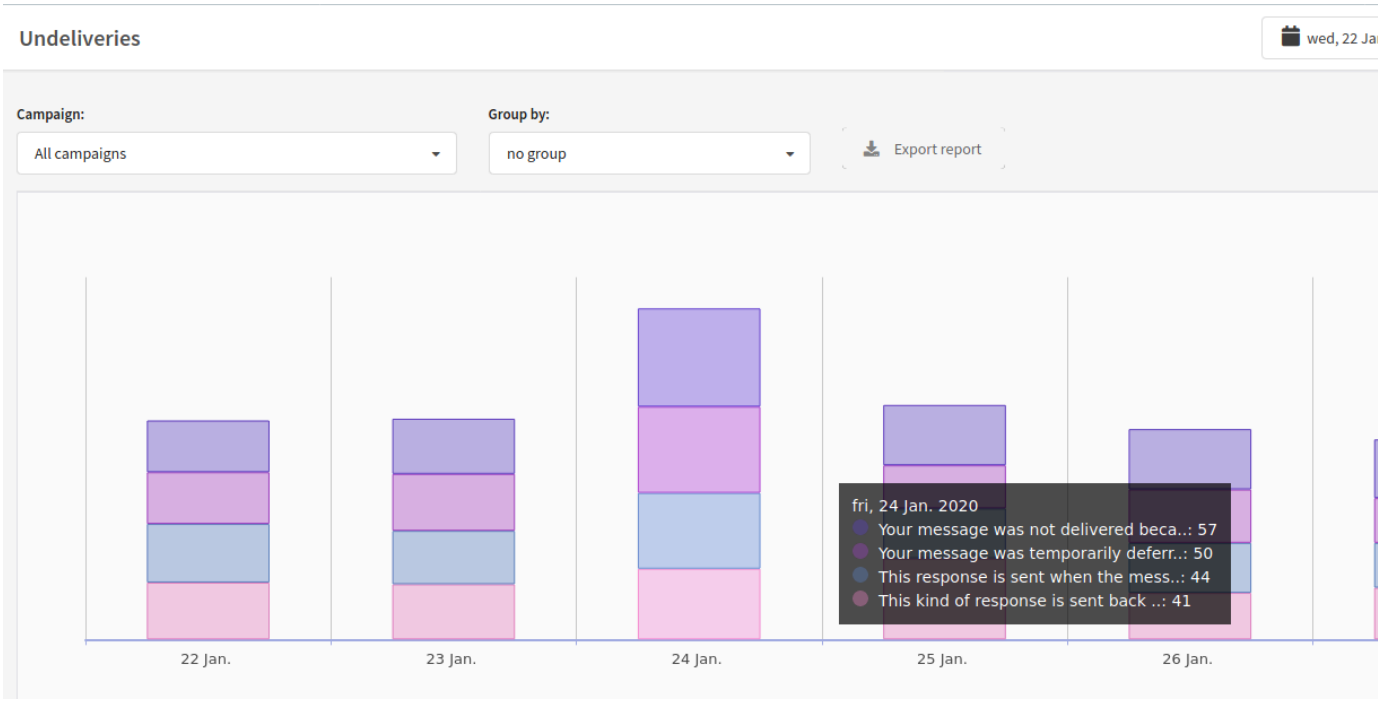
INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Report settings

At the top of the page, select the period for the report.

You can group information by one or all mailings, also you can use grouping by from domain or by IP addr



INFO

You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Table

The following information is displayed in the table:

- **Group by — selected mailing (or "all mailings");**
- **To domain — receiving email domain;**
- **Bounce text — message from email provider about temporary unavailability of the recipient mailbox;**
- **Amount — number of replies.**

Group by	To domain	Bounce text	Amount
All campaigns	freewebs.com	Your message was not delivered b..	4
All campaigns	aol.com	Your message was not delivered b..	4
All campaigns	rambler.ru	Your message was not delivered b..	3
All campaigns	furl.net	Your message was temporarily def..	3
All campaigns	scientificamerican.com	Your message was temporarily def..	3
All campaigns	smh.com.au	This response is sent when the m..	3
All campaigns	fotki.com	This kind of response is sent ba..	3
All campaigns	nps.gov	This response is sent when the m..	3
All campaigns	odnoklassniki.ru	This kind of response is sent ba..	3
All campaigns	merriam-webster.com	This response is sent when the m..	3
Totals:			**955**

To view the full error text, hover over the bounce message. Common delivery errors and their description

Bounce text	Description
`500 dial tcp <IP4:PORT>-><IP4:PORT>: i/o timeout`	**Network issue, failed to connect to the receiving server.**
`500 lookup <DOMAIN> on <IP4:PORT>: no such host for domain:<DOMAIN>`	**The domain does not exist.**
`540 <<EMAIL>>: recipient address rejected: Inactive`	**The recipient address exists, but it has been deactivated (e.g., disabled or ter**
`550 relay not permitted`	**Incorrect forwarding configuration on the recipient's mailbox.**
`550 No such user!`	**The email address does not exist on the server (either the account was delete**
`550 Unrouteable address`	**Unroutable address (e.g., if the recipient's domain lacks A/PTR records).**
`550 Message was not accepted -- invalid mailbox. Local mailbox <EMAIL> is unavailable: user is terminated`	**The user's mailbox has been deleted.**
`550 Message was not accepted -- invalid mailbox. Local mailbox <EMAIL> is unavailable: account is disabled`	**The user's mailbox is disabled.**
`552 5.2.2 <<EMAIL>>: user is over quota`	**The recipient's mailbox is full.**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Report on global control groups

Global control groups (GCG) - a part of the customer base that temporarily does not receive marketing c the effectiveness of marketing strategies. You can learn more about control groups on this page.

The Global Control Groups report is available on the corresponding tab of the Analytics section:

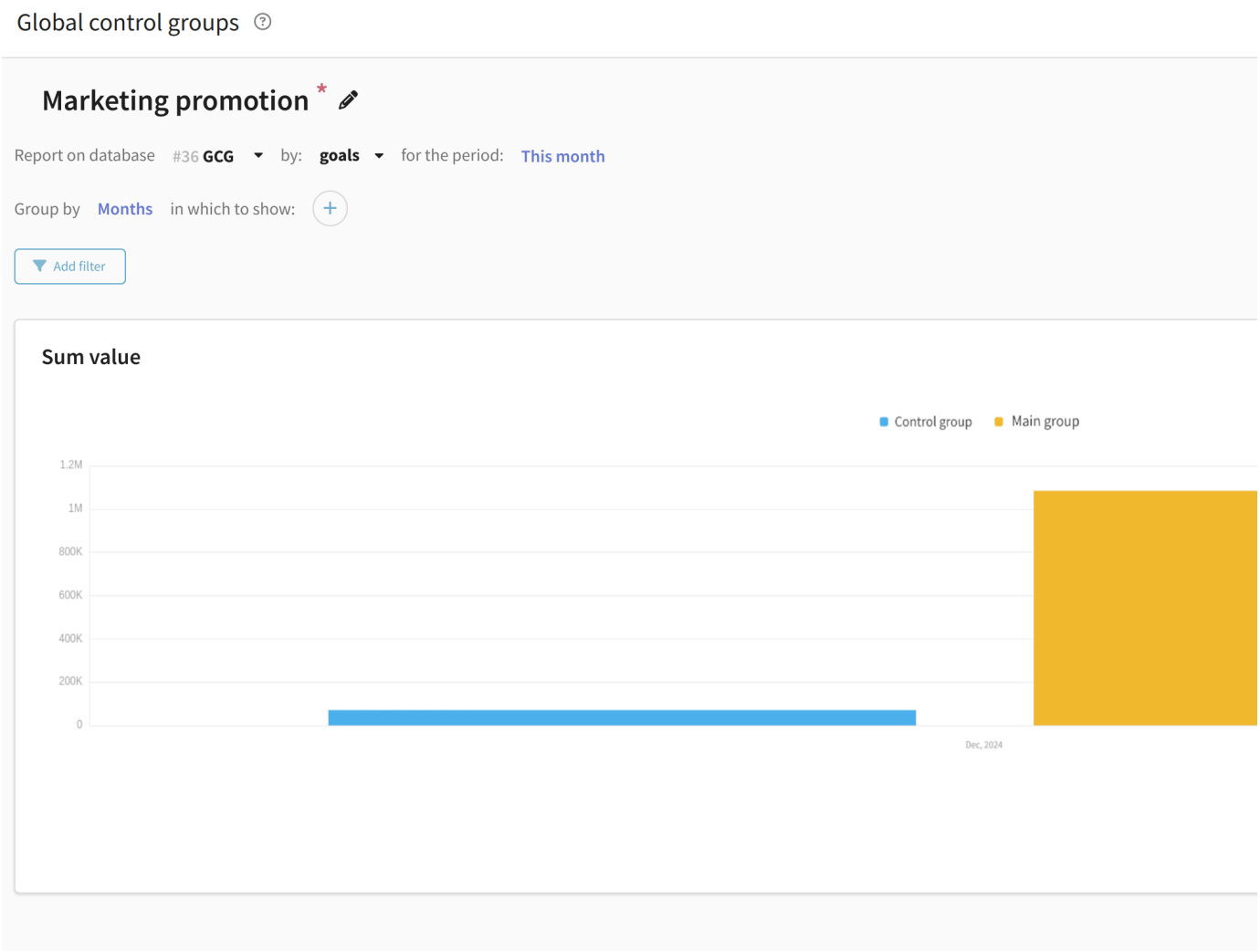
![img](/img/attachments/gcg/en/8-analytics.png)	

TIP

Global control group analytics are based on metrics collected from either Market data or pixel targets. in the platform beforehand. You can read more about the Market here, and this section contains inform

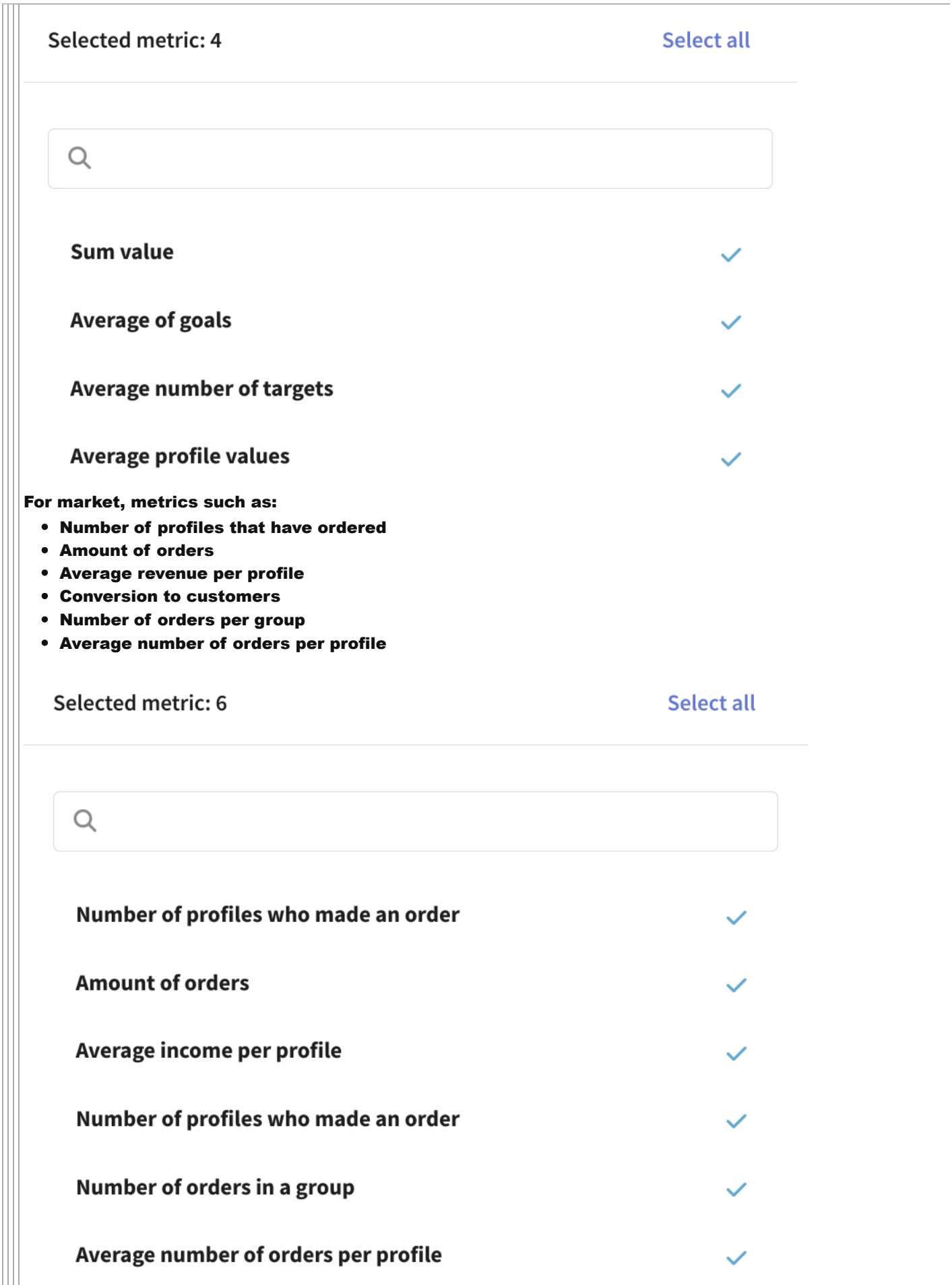
The report presents a comparison of data for the control and main groups. The system calculates report day.

Information can be retrieved by market and by targets. For market metrics, you can select sales channel and pixel targets is available. A graph based on the specified metrics of both groups will be displayed on



In the table below you can see the values of the metrics. The following metrics are available for targets:

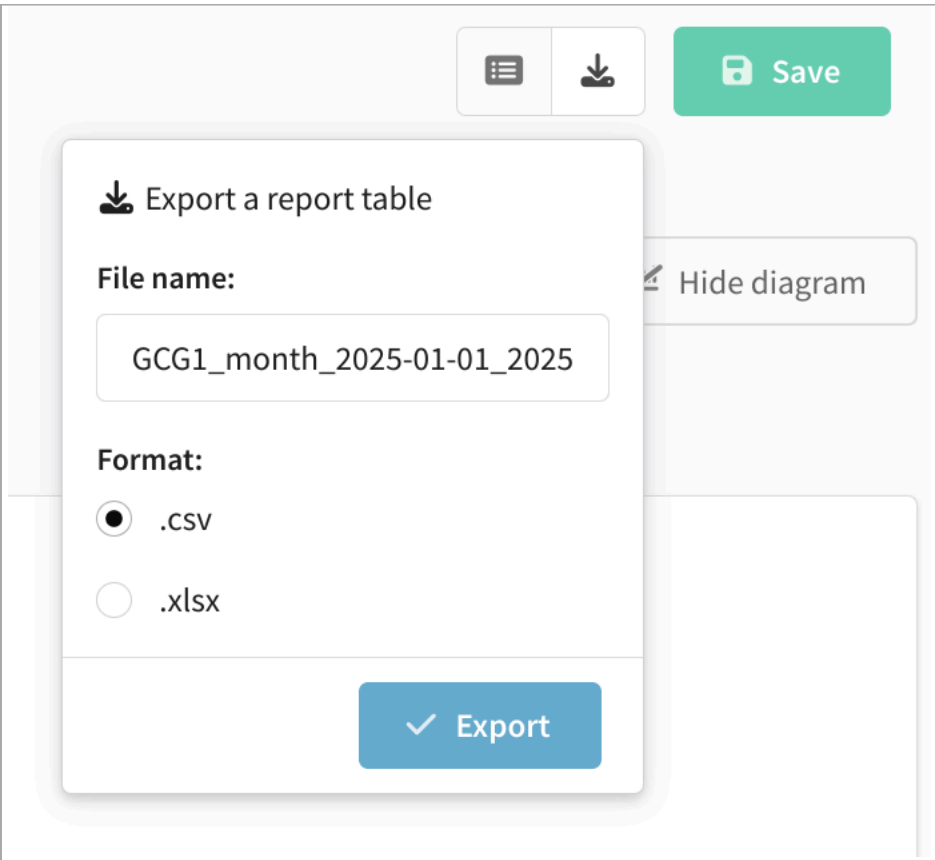
- **Sum Value**
- **Average value of goals**
- **Average number of goals**
- **Average profile value**



For market, metrics such as:

- **Number of profiles that have ordered**
- **Amount of orders**
- **Average revenue per profile**
- **Conversion to customers**
- **Number of orders per group**
- **Average number of orders per profile**

You can save the report or upload it in any convenient format by clicking on the corresponding button at



The saved report will be available in the report list available next to the upload and save buttons:

Channel reports

Saved reports

Search...

GCG

Summary report

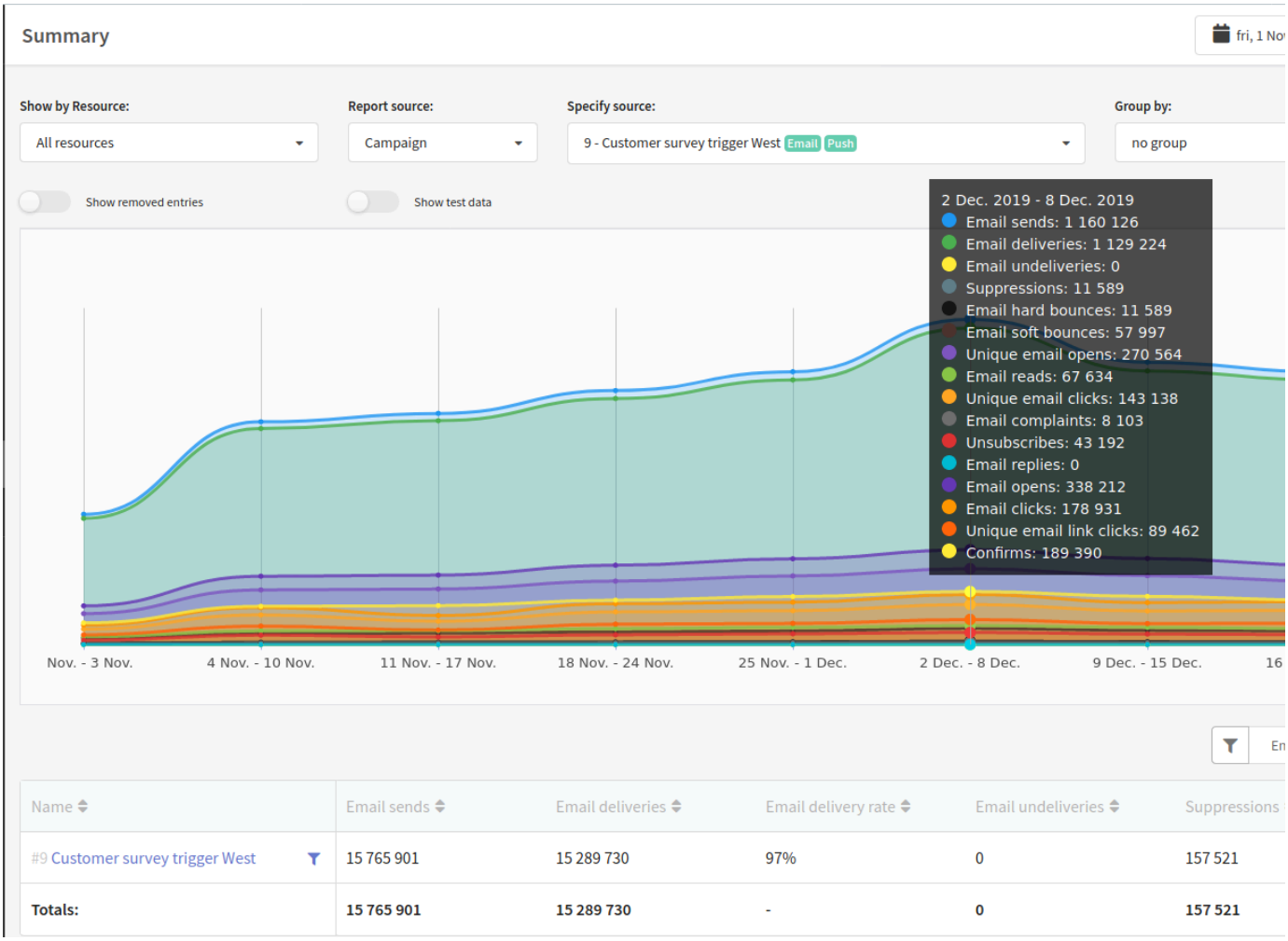
CAUTION

The summary report is no longer supported. It is not available in versions 65 and higher.

Description

A summary report is the basic tool for detailed marketing mailing analytics. It gathers mailing statistics are sent, including absolute numbers of opens, reads or clicks as well as relative parameters like deliver rate.

Moreover the summary report then merges selected mailings to provide you with a summary of your mar interactive graph, where you can see your over-time messaging dynamics and get insights on possible c

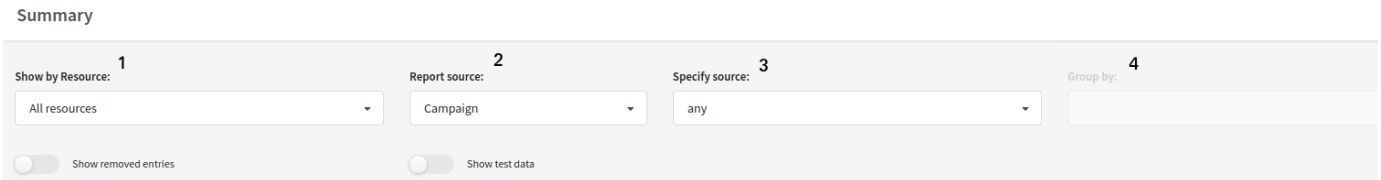


INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Managing summary report

You can start working on summary report from resource selection in Show by Resource menu (1), or disp



Next, you can select Report source(2). The following options are available:

- **Mailing**
- **Scenario**
- **Database**

- **Segment**
- **Message template**
- **Resource**
- **Domain group**
- **Domain**
- **Vendor**
- **Day**
- **Week**
- **Month**
- **Environment**
- **Sender**

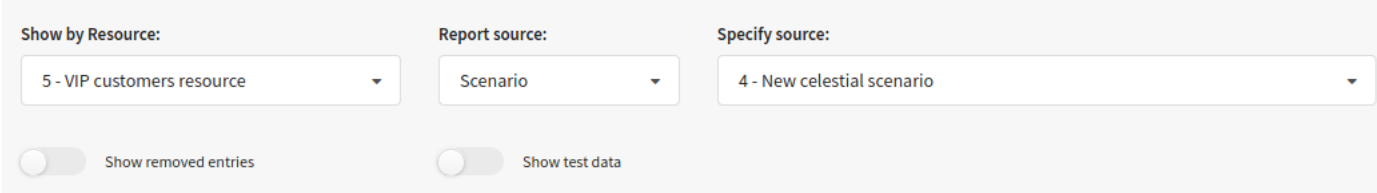
Using Specify source menu (3), you can build diagram using information from the menu 2, except domain
Using next Group by menu (4), you can group report by:

- **Domain**
- **Domain group**
- **Database**
- **Segment**
- **Message**
- **Resource**
- **Push provider**
- **Vendor**
- **By day**
- **By week**
- **By month**
- **Environment**
- **Sender**

Next, you can select time period for the report.
You can download configured report in XLS or CSV formats.
Also, there are two options — Show removed entries and Show test data.
Show removed entries — includes removed information into the report
Show test — includes test mailings into the report

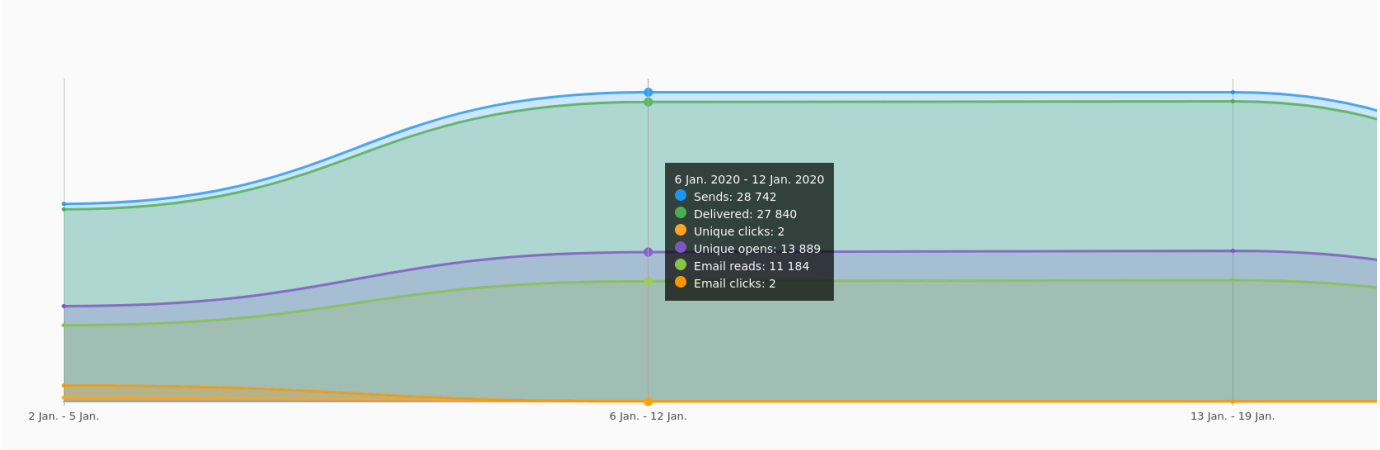
Summary report by scenarios triggers

Select report by Scenario to see detailed statistics by all mailings within automation scenario.

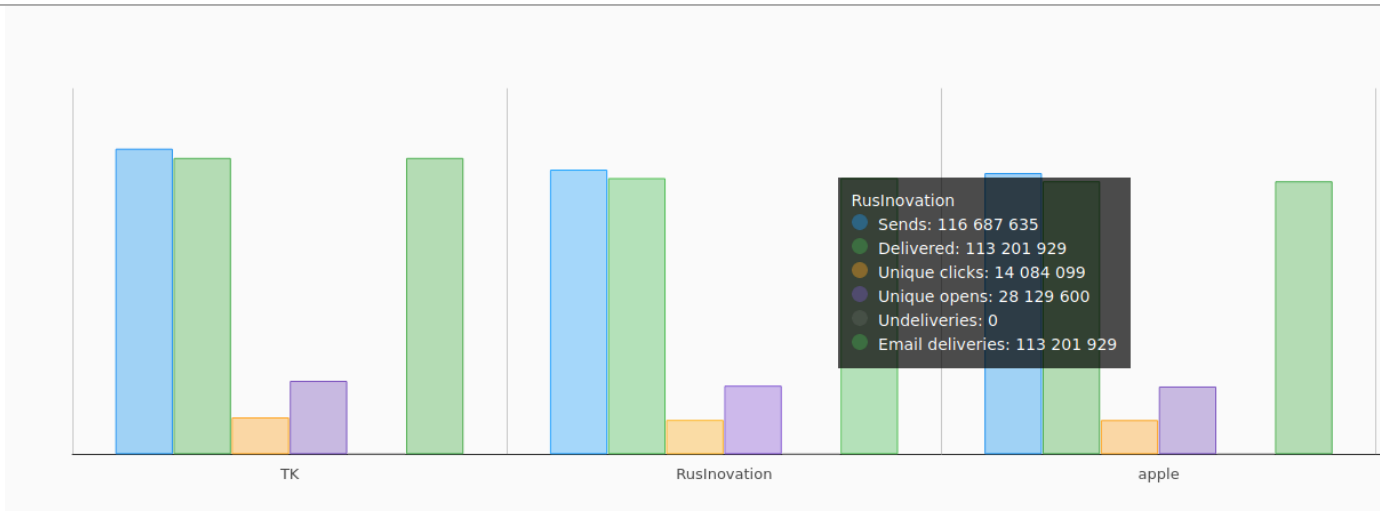


Diagram

You can select information for display in the diagram by “gear” buttons. Also, you can see diagram for da



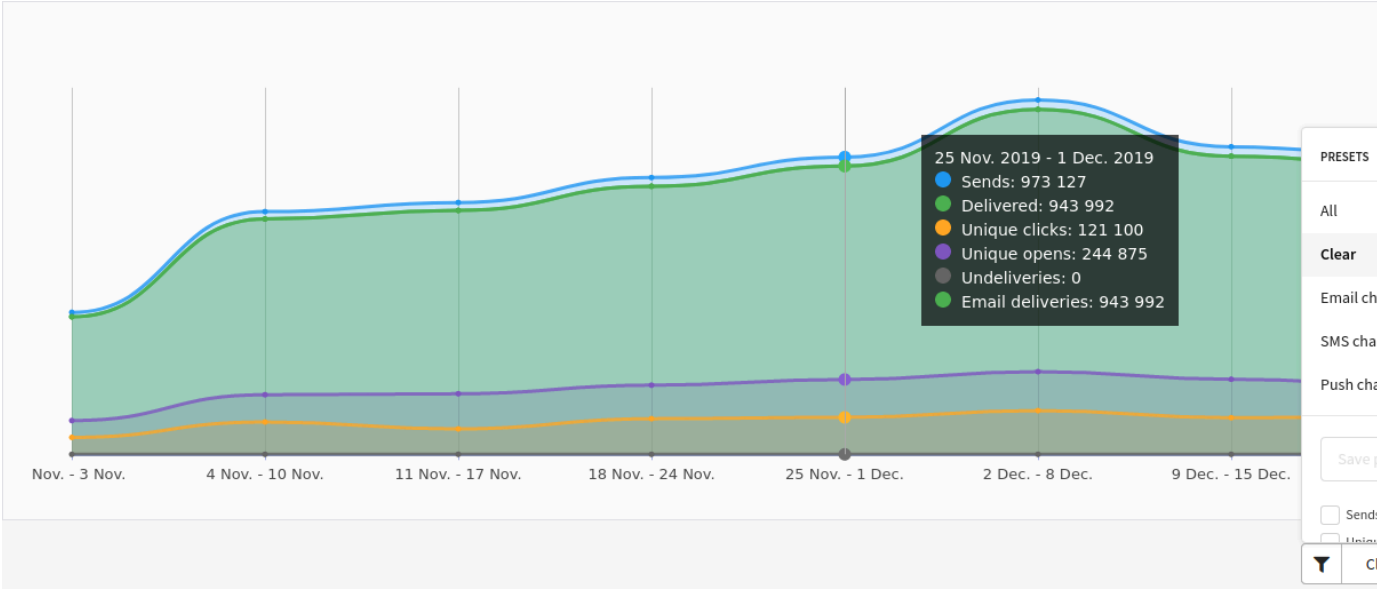
For *domain groups, domains, vendors and environment*, diagram is displayed in columns.



Performance indicators

You can select values for display in table in Select column settings menu. You can use presets or select table show amount of actions.

If you need more information about values of the summary report, you can read [“Summary report metric



Name	
#9 Customer survey trigger West	
Totals:	

TIP

You can filter data in the table by tag, which allows to improve search.

Summary report metrics

Channel	Indicator	API name	Description
All	Sends	sends_total	Total number of sent messages in all the channels
All	Deliveries	delivs_total	Total number of delivered messages in all the channels
All	Unique clicks	uclicks_total	Total number of unique clicks in messages in all the channels
All	Unique opens	uopens_total	Total number of unique opened messages in all the channels
All	Undeliveries	undelivs_total	Total number of unique undelivered messages in all channels
All	Offence	offence	Number of communications policy violations
Email	Email sends	sends	Number of sent emails
Email	Email deliveries	delivs	Number of delivered emails
Email	Email opens	opens	Number of opened emails
Email	Email clicks	clicks	Number of clicks in emails
Email	Unique email opens	uopens	Number of unique email opening
Email	Unique email clicks	uclicks	Number of unique clicks on any clickable item of the email
Email	Unique email link clicks	ulclicks	Uniqueclick on each link of each email by the same profile
Email	Email reads	reads	Number of read emails. Email read event is logged if the recipient has opened the message and left it open for at least 12 seconds.
Email	Email replies	replies	Number of emailreplies
Email	Email complaint	complains	Number of emails followed by spam complaints (FBL)
Email	Email hard bounces	hbounces	Number of emails that have failed to deliver because of recipient'sinvalid address or a sudden mistake during tl sending
Email	Email soft bounces	sbounces	Number of emails that has failed to deliver because the mailbox is full or the server is temporarily down
Email	Confirms	confirms	Number of confirmed emails
Email	Email unsubscribe	unsubs	Number of unsubscribers. Recipient opted out by clicking a message link (`{unsubscribe} or {globalunsubscribe}`).
Email	Suppression list	suppresses	Number of emails that hasn't been delivered because the recipient's address is in the suppression list
Email	Email undeliveries	undelivs	Number of undelivered emails. It consists of soft and hard bounces and undefined delivery errors
Email	% Email delivery rate	delivery_rate	The percentage of the emails that are successfullydelivered
Email	% Email open rate	u_open_rate	The percentage of the recipients who opened an email
Email	% Email CTR	u_click_rate	The percentage of unique recipients who clicked on a link after receiving the email
Email	% Email CTOR	u_open_click_rate	The percentage of unique recipients who clicked on a link after opening the email
Email	% Email read rate	email_read_rate	The percentage of recipients who read the email, i.e. those who opened the email and did not close it for at lea
Email	% Email reply rate	email_reply_rate	The percentage of the recipients who replied to an email
Email	% Email complaint rate	complain_rate	The percentage of the emails that are followed by spamcomplaints
Email	% Email hard bounce rate	hb_rate	The percentage of emails that have failed to deliver because of recipient'sinvalid address or a sudden mistake sending
Email	% Email soft bounce rate	sb_rate	The percentage of emails that has failed to deliver because the mailbox is full or the server is temporarily down
Email	% Email unsubscribe rate	email_unsub_rate	The percentage of people who unsubscribed from emails
SMS	SMS sends	sends_sms	The number of sent SMS
SMS	SMS deliveries	delivs_sms	The number of delivered SMS
SMS	SMS clicks	clicks_sms	The number of clicks in SMS
SMS	Unique SMS clicks	uclicks_sms	The number of unique clicks in SMS
SMS	% SMS delivery rate	sms_deliv_rate	The percentage of SMS that has been successfully delivered
SMS	SMS undeliveries	undelivs_sms	The number of undelivered SMS
SMS	% SMS click-though rate	sms_ctr	The percentage of unique clicks in the SMS in relation to the total number of delivered SMS
SMS	SMS unsubscribes	unsubs_sms	Number of unsubscribers. Recipient opted out by clicking a message link `{unsubscribe} or {globalunsubscribe}` .
Push	Push sends	sends_push	The number of sent push notifications
Push	Push opens	opens_push	The number of opened push notifications
Push	Unique push opens	uopens_push	The number of opened push notifications
Push	Push deliveries	delivs_push	The number of delivered push notifications
Push	Push undeliveries	undelivs_push	The number of undelivered push notifications
Push	Push clicks	clicks_push	The number of clicks on any clickable element of the push notification
Push	Uniquepush clicks	uclicks_push	The number of clicks on any clickable element of the push notification
Push	% Push open rate	push_open_rate	The percentage of opened push notifications
Push	% Push send rate	push_deliv_rate	The percentage of push notifications that has been successfully delivered
Push	% Push click-though rate	push_ctr	The percentage of clicks in the push notification in relation to the push openings
WhatsApp*	Delivered	whatsapp_deliv	Number of messages delivered via WhatsApp*
WhatsApp*	undelivered	whatsapp_undeliv	Number of messages send via WhatsApp*
WhatsApp*	Send	whatsapp_send	Number of messages send via WhatsApp*
WhatsApp*	WhatsApp* subscription	whatsapp_subscribe	The number of new subscriptions to the WhatsApp* channel

INFO

Unique event

The uniqueness of any event is limited by the time specified in the platform configuration (`UNIQUE_ACTIONS` **1440 minutes. It means that a repeated event in a day will be counted as unique again. Check with your uniqueness value set.**

All indicators of unique events are marked in the table.

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a*

Data analytics

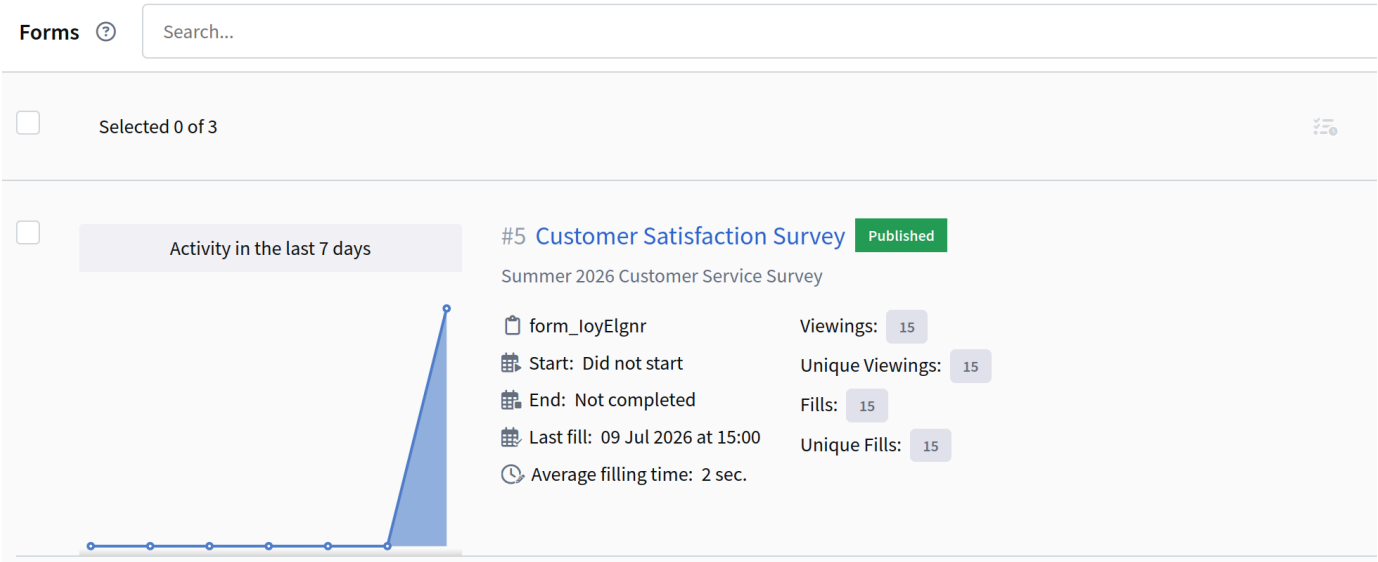
Description

You can analyze the results obtained after filling out the form in the “Collected data” section or in the “Ir

Collected data shows the responses of a specific user.

Insights provides a general summary of the form (views, completions) and statistics on a specific quest

To go to one of the sections, find the required form in the general list and click the corresponding icon o



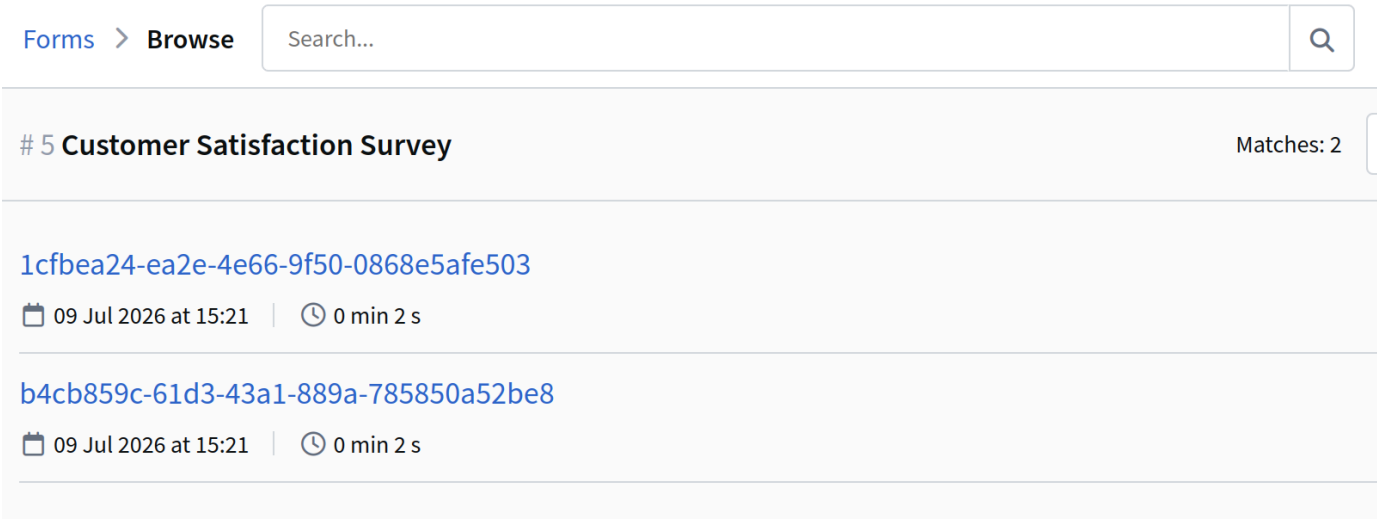
You can view data on views, unique views, fills, unique fills, and average fill time directly in the form list, section.

TIP

When a user interacts with a form on the website, the platform sets a cookie. If the cookie is not alread submission is recorded. The cookie’s lifespan is determined by the `TRK_WEB_CTRL_COOKIE_TIME_SEC` configurat days.

Collected data

Open the form and go to the Collected data section. Here you will see a list of IDs of all completed forms information.



INFO

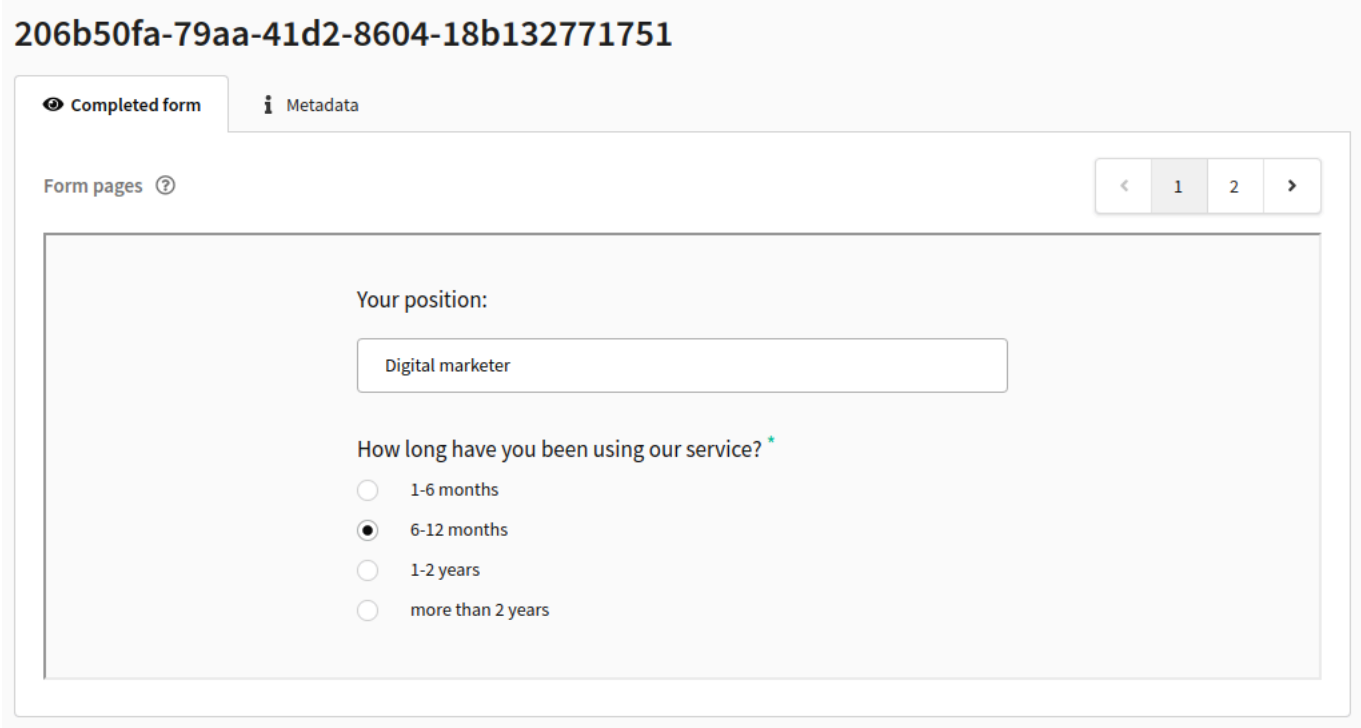
The following buttons are located in the upper right corner:

— go to the form editing window.

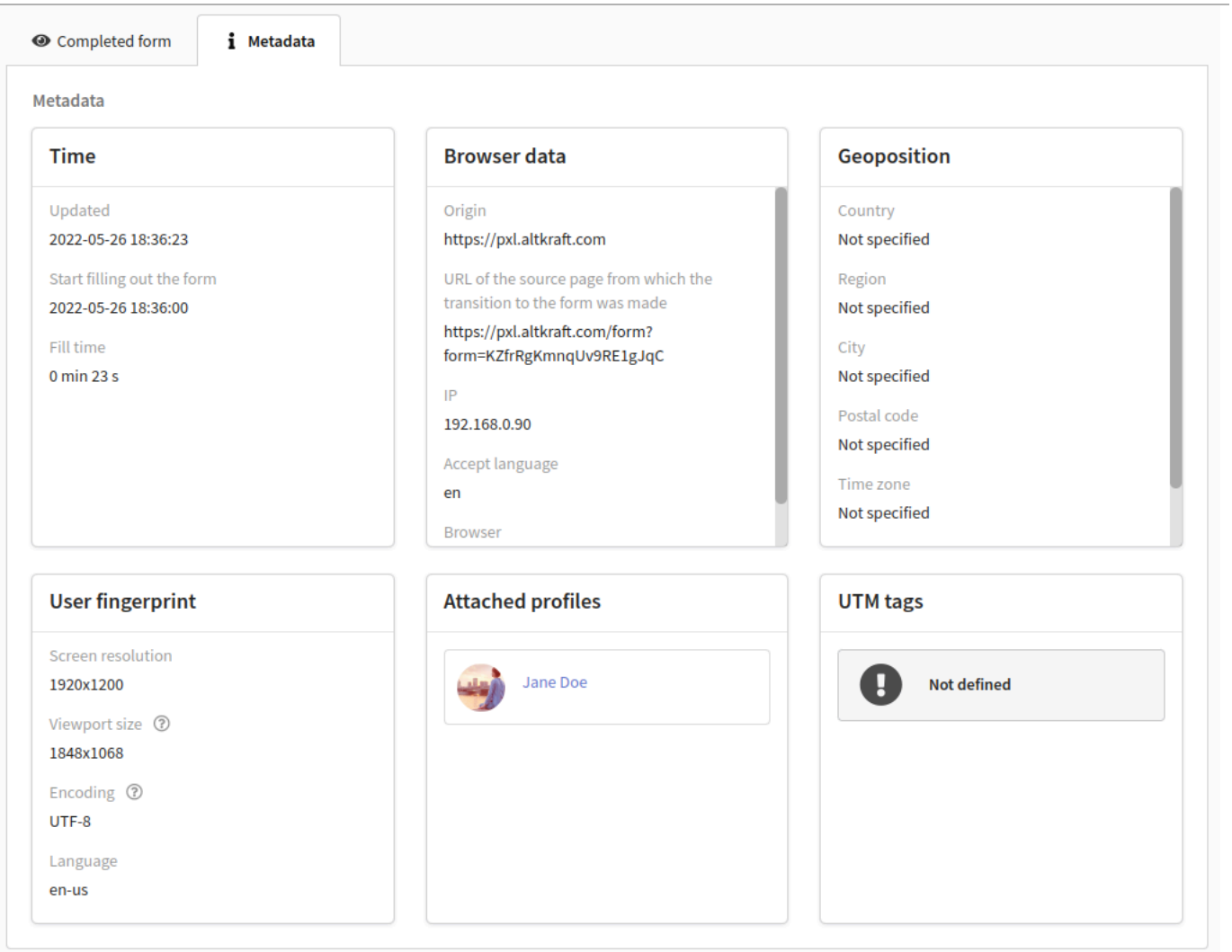
— open form insights.

— export data on form completions for a certain period.

On the Completed form tab, you can find out exactly how the user filled out each field:



The Metadata tab displays detailed information about filling out the form: time, browser data, geolocatio



Form insights

Open the form and go to the "Insight section. On the top panel, open the calendar and select the period f
Analytics provides the following information:

- **Views (how many times the form page was loaded).**
- **Fills (how many times the form was filled out and submitted).**
- **Average filling time .**

INFO

These metrics are not unique. If the same user has viewed or completed the form multiple times, all vie counted.

There are two tabs available in Form insights:

- **Results analysis — statistics on a specific field in the form.**
- **Statistics of survey — a general summary of the form.**

Results analysis

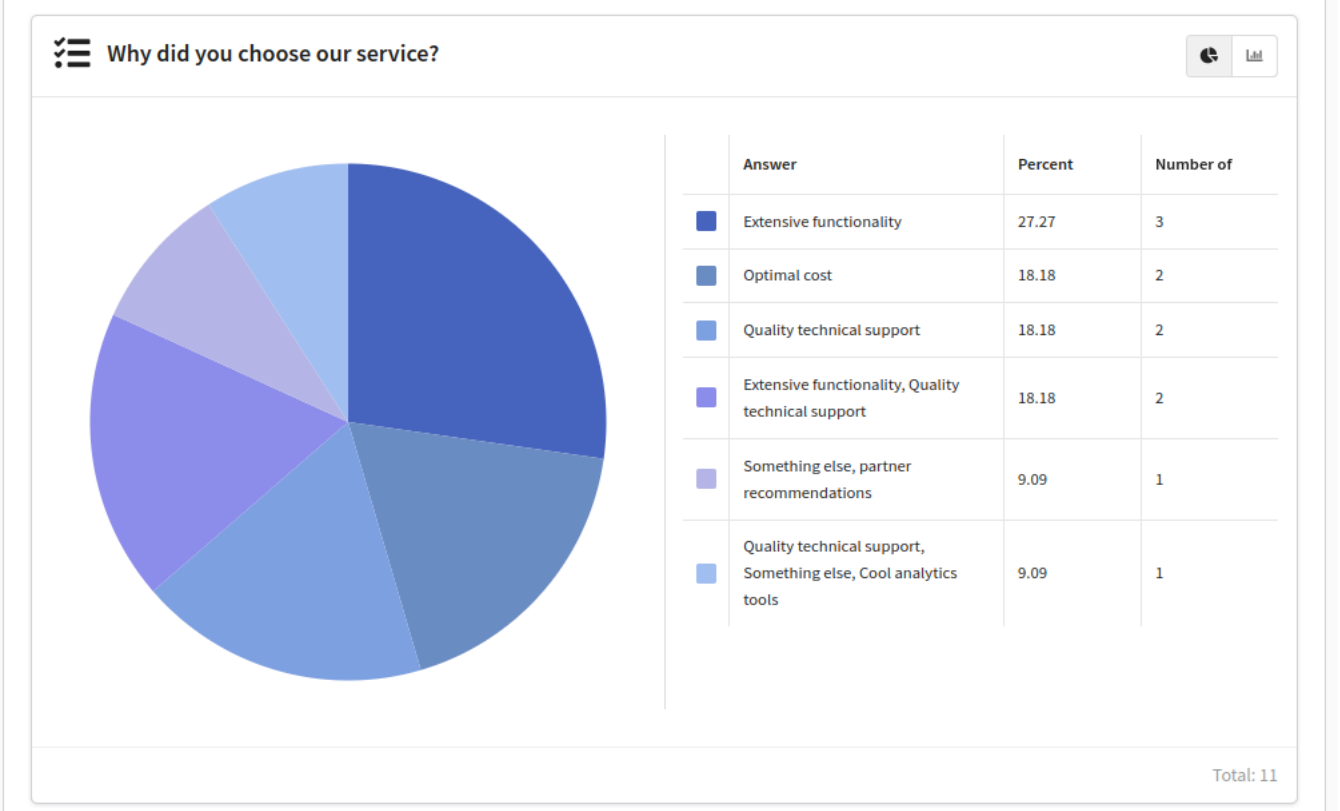
This tab displays statistics on the completion of a particular field in the form. To switch between pages upper right corner.

Data for each field is collected in a separate block. Here you can see the question and answers given by shows how many times the field was filled in total.

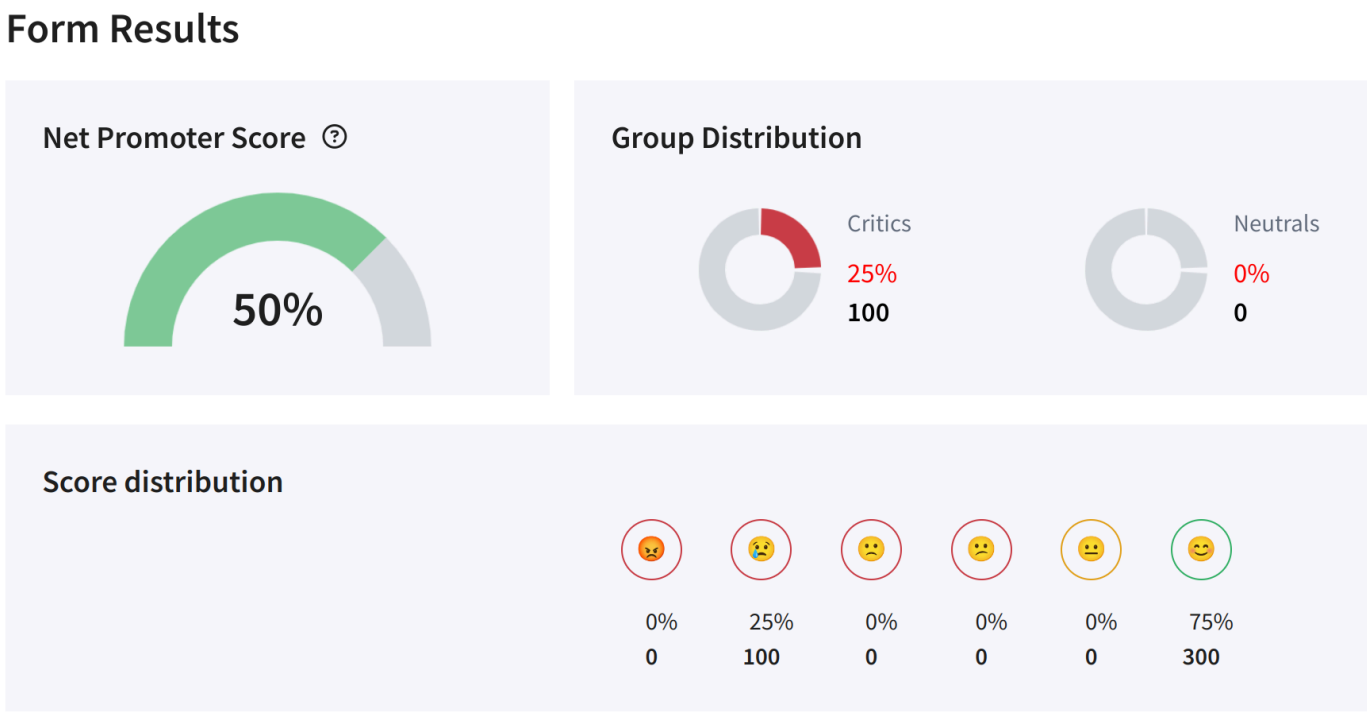
Most of the data are presented in percentage and quantitative terms. Additionally, it is possible to visual

or pychart

.



For forms containing NPS survey, the Net Promoter Score and the distribution of responses by group will



Statistics of survey

On this tab you can find the following information:

Page views **Views will show how many times each page has been loaded. With this metric, you can find out on which page users most often abandon the form, and further optimize it.**	**![img](/img/attachments/37848463/56258576.png)**
Reasons for refusal **1. Abandoned fills — the user partially or completely fills out the form, but does not submit it.** **2. Bounces — there were technical problems when filling out the form (for example, the server stopped).** **3. Repeat fills — the same user fills out the form several times. Although such fillings appear in the rejection reasons, they are also included in the overall summary.**	**![img](/img/attachments/37848463/56258577.png)**

Analytical slices of the report

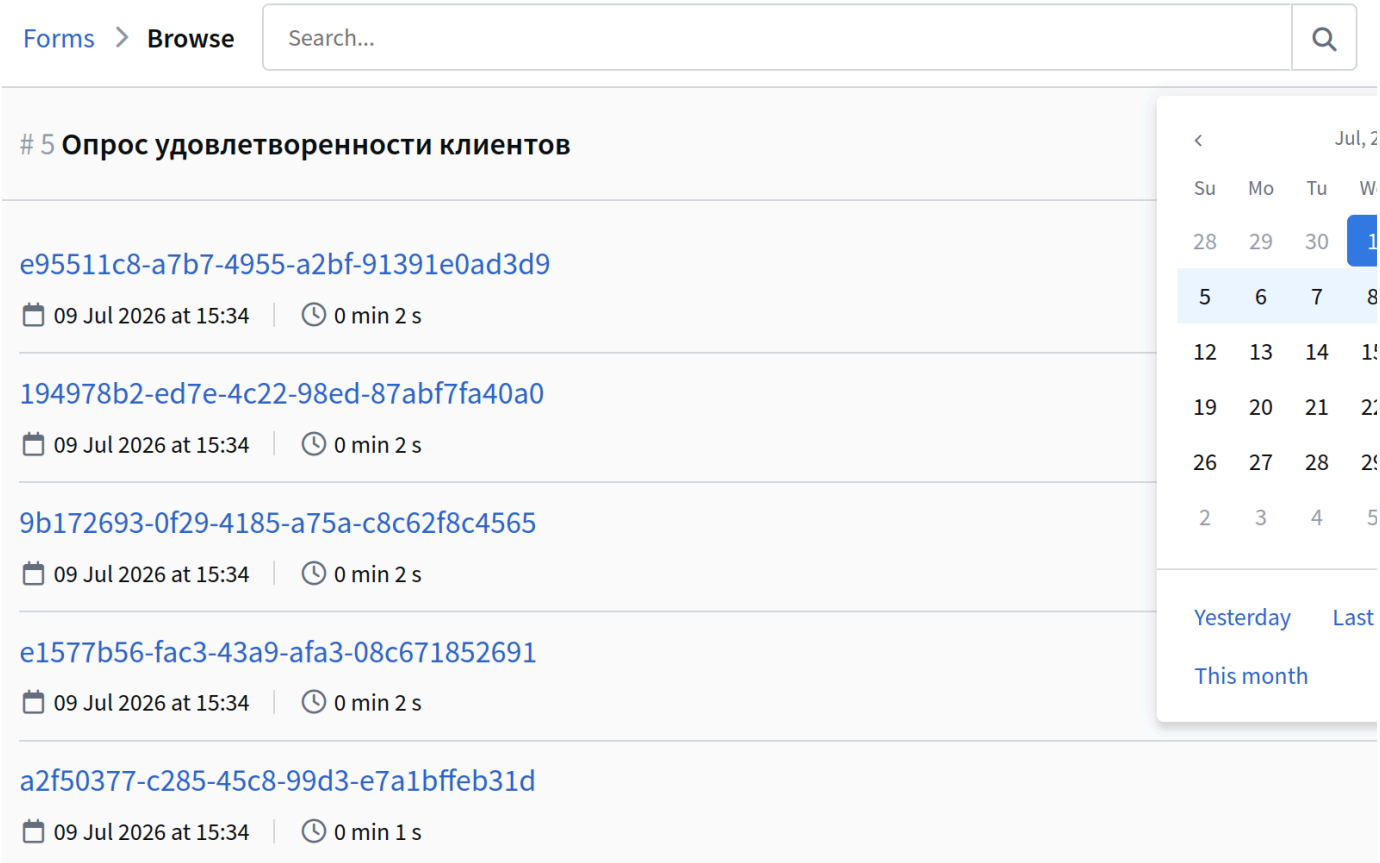
Below you can group respondents by additional parameters:

- **referrer**
- **browser**
- **operation system**
- **language**
- **time zone**
- **UTMs**

Form export

Choose a form. Go to the “Collected data” section. There is an export button

at the top of the page. Click on it.
In the calendar, specify for which period to export data. Click OK. Go to Tasks and download the file.



Binding data channel and forms

Description

In most cases, the form is intended for website visitors. We create forms to invite new visitors to subscr does it work? The client goes to the site, sees the form and, if he has both strength and desire, fills it ou user left is sent to one of the databases on the platform and we get a new digital profile. As you may hav form usually don't subscribe to any of our resources yet.

But what if we'd like to get feedback from the client who is already subscribed to one of the resources? additional information about the profile in order to decide whether it is worth subscribing to a new resou to the person whose digital profile is already in our database. Will the Altcraft platform be able to recogr

The answer is yes. If the profile that decided to fill out the form is already in one of your databases, the the person opens the form.

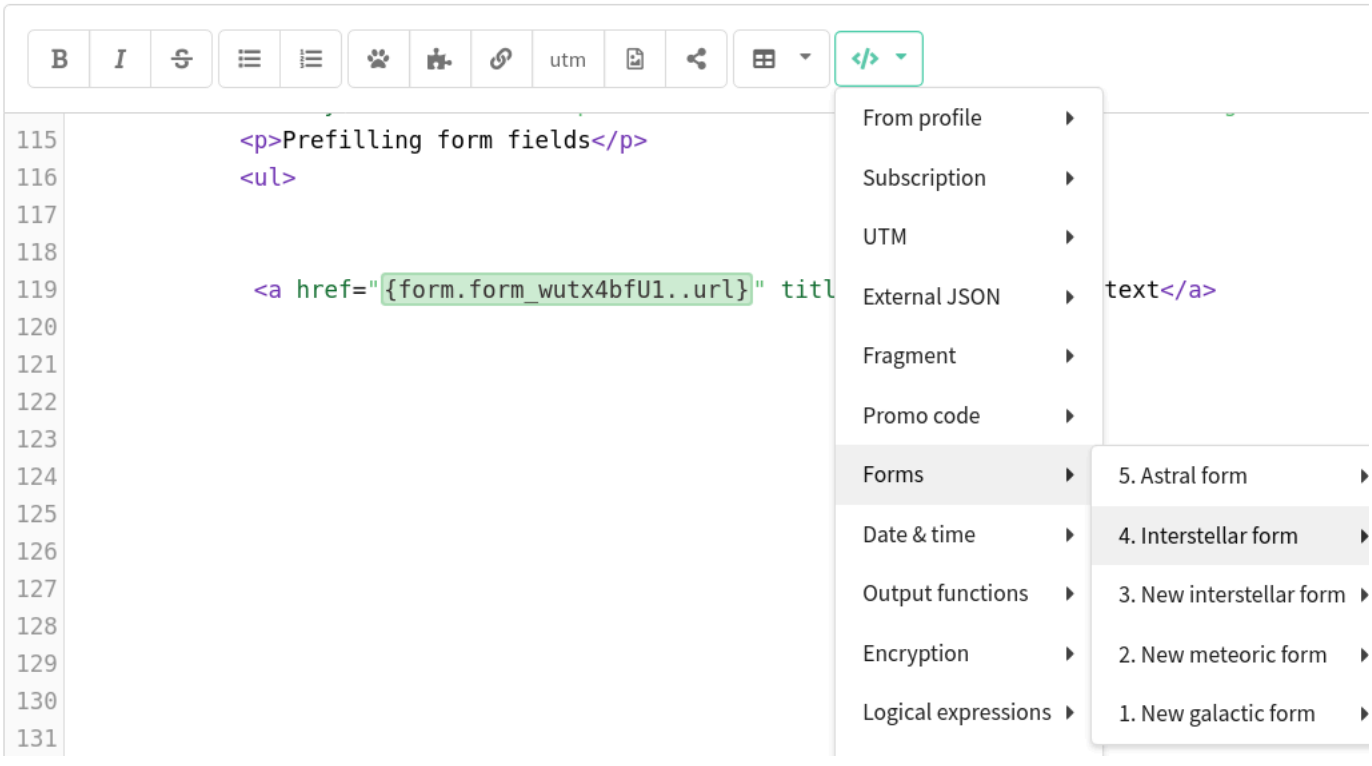
What are the advantages? You can set up automatic prefilling of form fields with data from the profile. F where you ask the customer to enter his email and phone number, but you already know all this informat and just send him a link to the template, where both his email and his phone number will be filled out?

The only thing is, all this is possible only if you use one of the communication channels to send the form form in the website structure, you can't prefill the form. Setting up automatic identification of a profile w time of creating a template for the mailing — in the HTML version of the letter.

How to add a form to a template

First of all, let's add a link to our template.

Now in the HTML version of the letter, click `</>` and select "Forms" from the drop-down list. You'll see al one of them.



Then, move the form variable you've just added to the value of the href attribute.

```
<a href="{form.form_eyyVPnQTE.url}" name="link1">Link to the form "form_eyyVPnQTE"</a>
```

```
115            <p>Prefilling form fields</p>
116            <ul>
117
118
119             <a href="{form.form_wutx4bfU1..url}" title="Link text">Link to the form</a>
120
121
```

TIP

By the way, if there are several pages in the form, you can add a link to the page you need.

```
<a href="{form.form_eyyVPnQTE.page_eq55Lrs_J.url}" name="link2">Link to the first page of the form "form_eyyVPnQTE"</a>
```

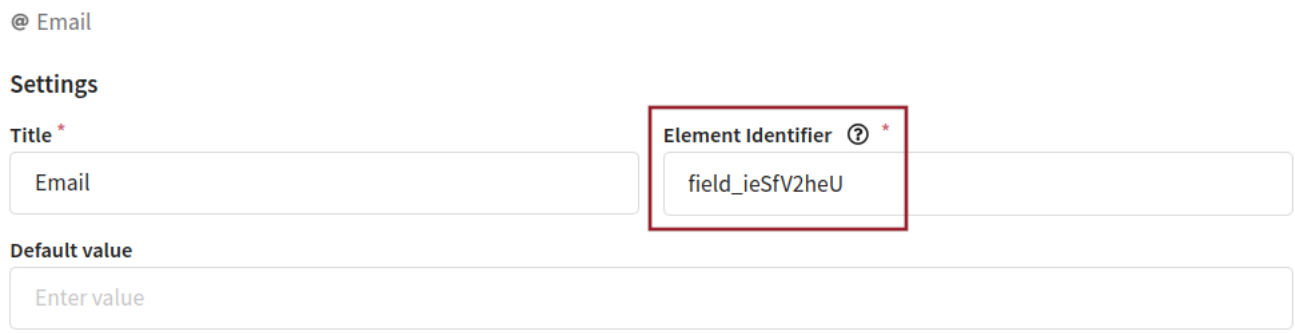
```
115            <p>Prefilling form fields</p>
116            <ul>
117
118
119             <a href="{form.form_wutx4bfU1.page_z5lZwM26BO.url}" title="Link text">Link to t
120
121
```

How to set up auto-complete fields

First, add a link to the form. We have described how to do this above.

After that, you need to pass the parameters and their values. Parameters are the fields of our form. For e cannot pass such general names — it is necessary to find the identifier of each field.

Where can you find it? Go back to your form. Each element you add has its own ID.



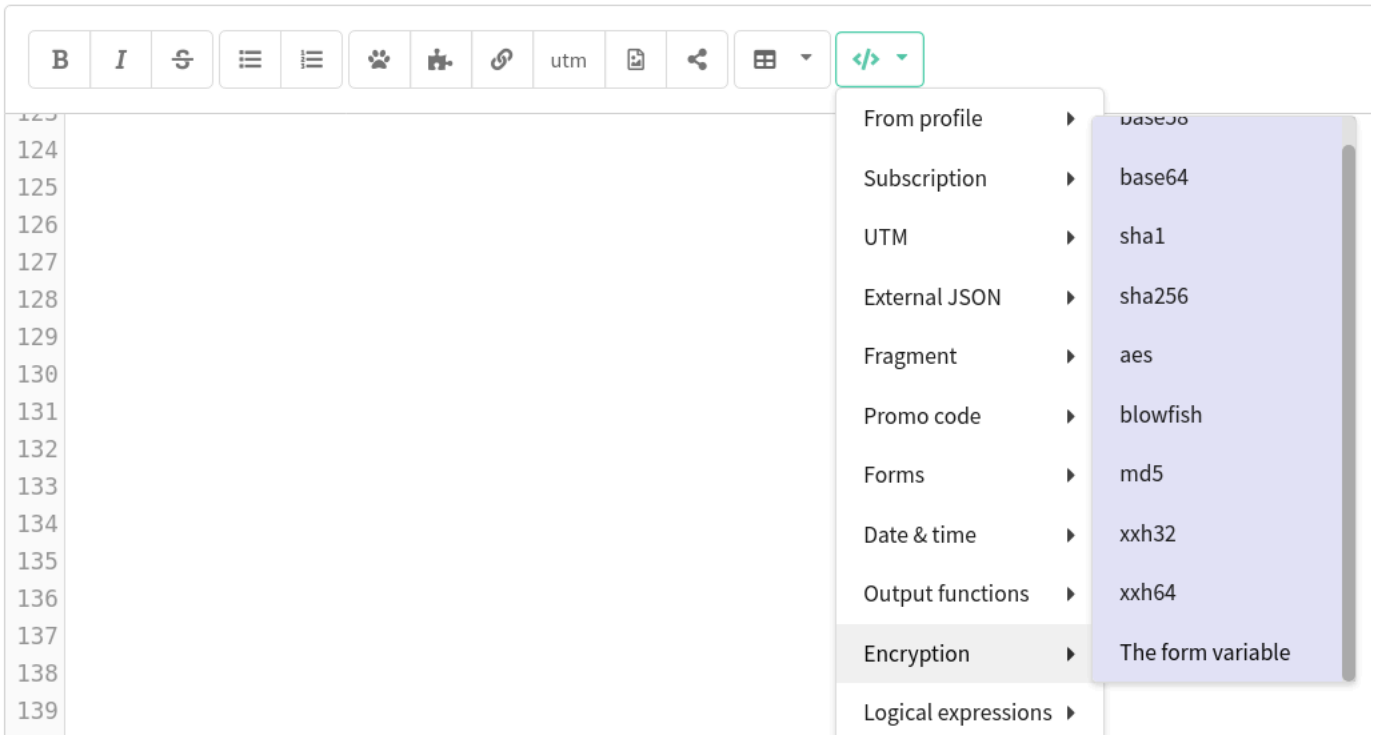
Copy it. Go back to the template and paste the parameter, separated by the ampersand &.

```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC" name="link3">Link to the form with automatic prefilling of fields</a>
```

Now it"s time to figure out the values of the parameters. The simplest option is to fill them on your own. number that will be displayed the same for each customer.

```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC={formcrypt("Свое значение")}" name="link3">Link to the form with auto-filled fields</a>
```

However, if you want to personalize your content, use variables. To do this, in the top panel, click "`</>`", variable".



The variable `{formcrypt(lead.email)}` will be added to the template.

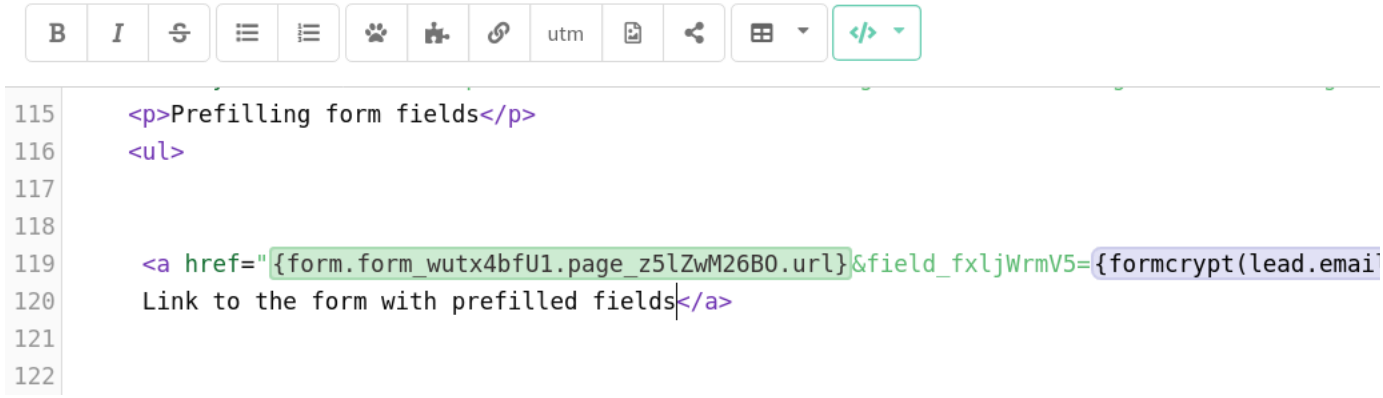
`lead.email` is the email of the subscriber, who is already in the Altcraft profile database. You can change available variables.

The variable must be added to the parameter after `=`.

```
field_54vkfiZJC={formcrypt(lead.email)}

field_fxljWrmV5={formcrypt("Your text")}
```

You can add as many parameters as you'd like to by listing them separated by `&`.




```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC={formcrypt(lead.email)}&field_fxljWrmV5={formcrypt("Custom value")}" name="link3">Link to
```

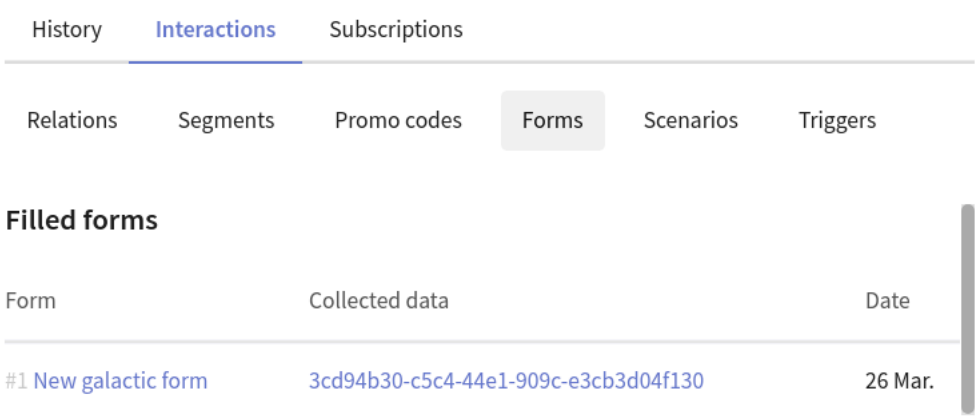
Matching by ID

When you create a form, you can use an additional matching mode in the Actions section: By message ID need to pass the smid parameter with the send_message_id value in the link.

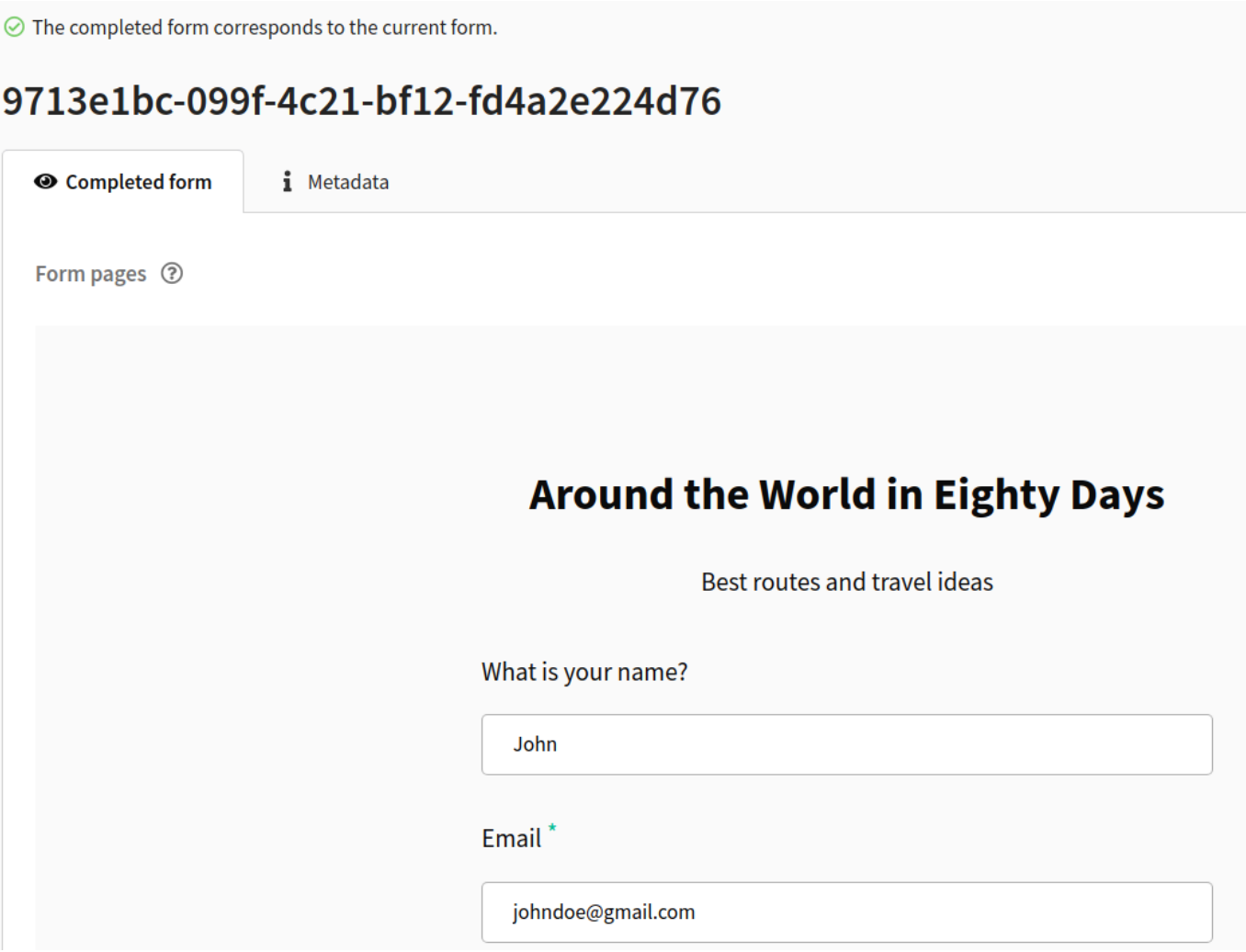
```
<a href="https://example.com/?smid={send_message_id}&field_54vkfiZJC={formcrypt(lead.email)}&field_5S0nwR_AD={formcrypt(lead.phone)}" name="l
fills in the fields</a>
```

Collected data

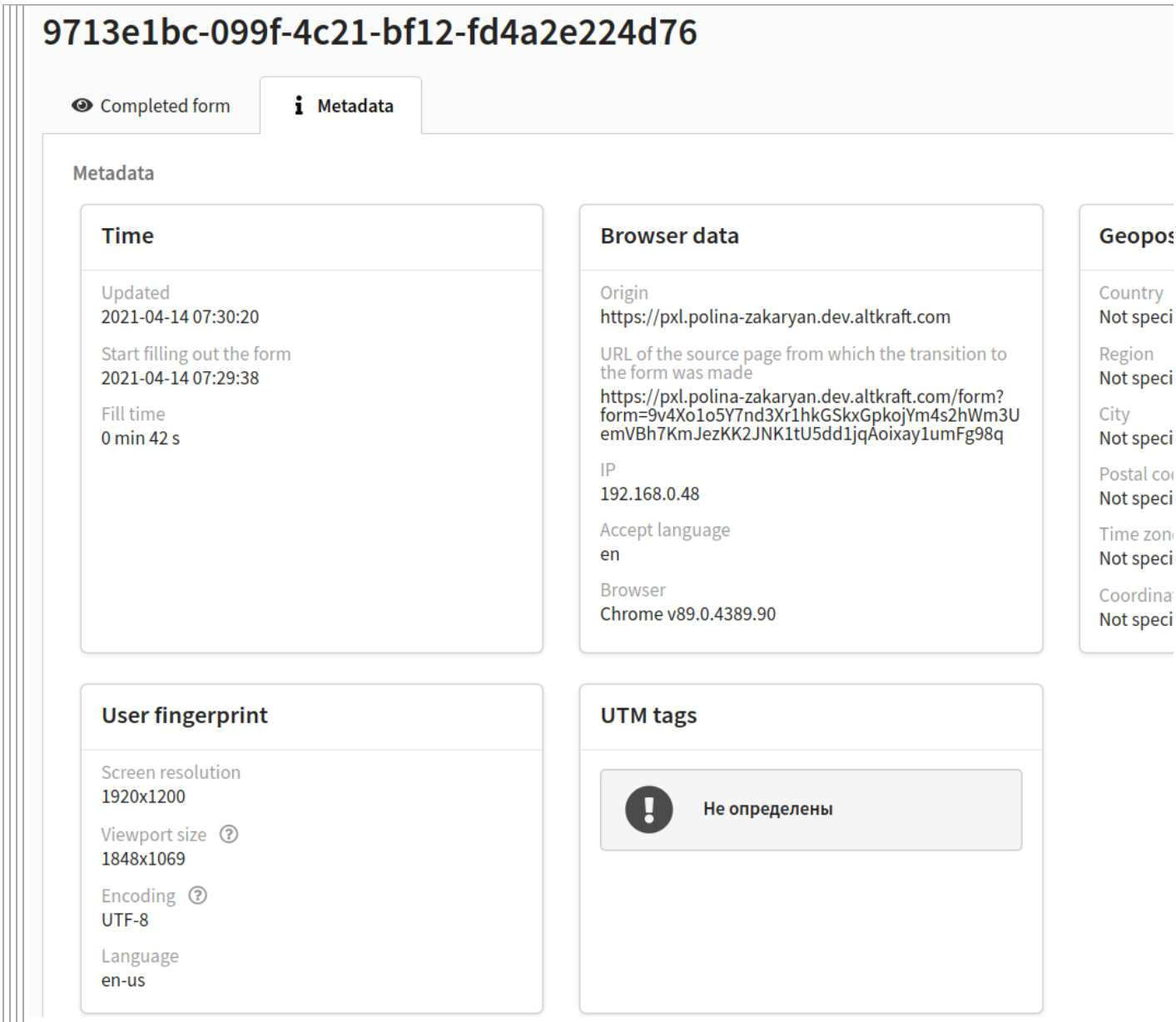
In the profile history on the "Interactions tab, you can find information about the subscriber's interaction what forms he filled out.



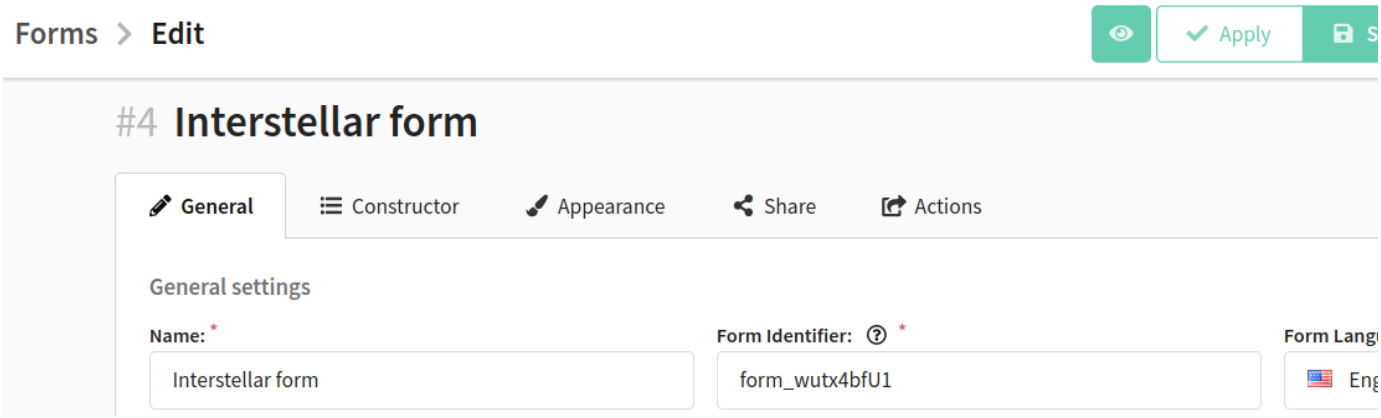
If you click on the identifier in the Collected data" column, you will be taken to a new page, where you ca about the interaction of the subscriber with a specific form. For example, see how exactly he filled in ea



Go to the next tab - “Metadata”. Here you will find detailed information: from the time of filling and the br at that moment to his geolocation and UTM tags.



By the way, you can go to the same panel with data for a specific subscriber directly from the form. Click corner.



Here you will see a list of the IDs of all users who filled out the form. Click “Details” to get detailed infor

Forms > **Browse**

#4 Interstellar form

Search by name or id

Sort by

9713e1bc-099f-4c21-bf12-fd4a2e224d76

Updated: 2021-04-14 10:30:20
Fill time: 0 min 43 c
Attached: unknown
IP: 192.168.0.48

Country:
City:
Time zone: Unknown time zone

NPS testing

The Net Promoter Score (or NPS for short) is a metric that shows customer loyalty to a company or prod reflects the willingness of customers to recommend your product or services. This index is one of the m loyalty. NPS allows you to analyze the strengths and weaknesses of your product and adjust your market

Usually NPS testing is a form with a question How ready are you to recommend us on a scale from 1 to 1 the answer, several types of respondents are distinguished:

- **Promoters - loyal customers who are ready to leave positive feedback and recommend the product**
- **Neutrals - neutral customers who use your product now, but will stop if they find a better offer.**
- **Critics (or detractors) - customers who have had a negative experience with you and are now ready t company.**

NPS values of 30% and above are considered acceptable. A value between 0% and 30% indicates that th product. Negative index values indicate serious problems that require immediate intervention.

NPS calculation

In mathematical terms, NPS is the difference between the percentages of promoters and critics. The exp

$$NPS = \frac{P-D}{T} \times 100\%$$

Where P are promoters, D are critics, T is total number of respondents.

The platform can use either average or aggregated NPS index when analyzing the resulting data. Averag NPS indices within the form. Aggregated NPS is the overall NPS index derived from the responses of all r can be calculated using the formulas:

$$NPS_{avg} = \frac{NPS_1 + NPS_2 + NPS_3}{n}$$

$$NPS_{agr} = \frac{\Sigma(P_i - D_i)}{\Sigma T_i} \times 100\%$$

Let's look at an example of calculating the NPS index. Let's say we ran 3 tests with the following results

- **Test #1 - 6 critics, 9 neutrals and 27 promoters for a total of 42 respondents**
- **Test #2 - 10 critics, 15 neutrals and 25 promoters for a total of 50 respondents.**
- **Test #3 - 21 critics, 35 neutrals and 14 promoters for a total of 70 respondents.**

The NPS calculation for these tests will look like this:

$$NPS_1 = \frac{27-6}{42} \times 100\% =$$

$$NPS_2 = \frac{25-10}{50} \times 100\% =$$

$$NPS_3 = \frac{14-21}{70} \times 100\% =$$

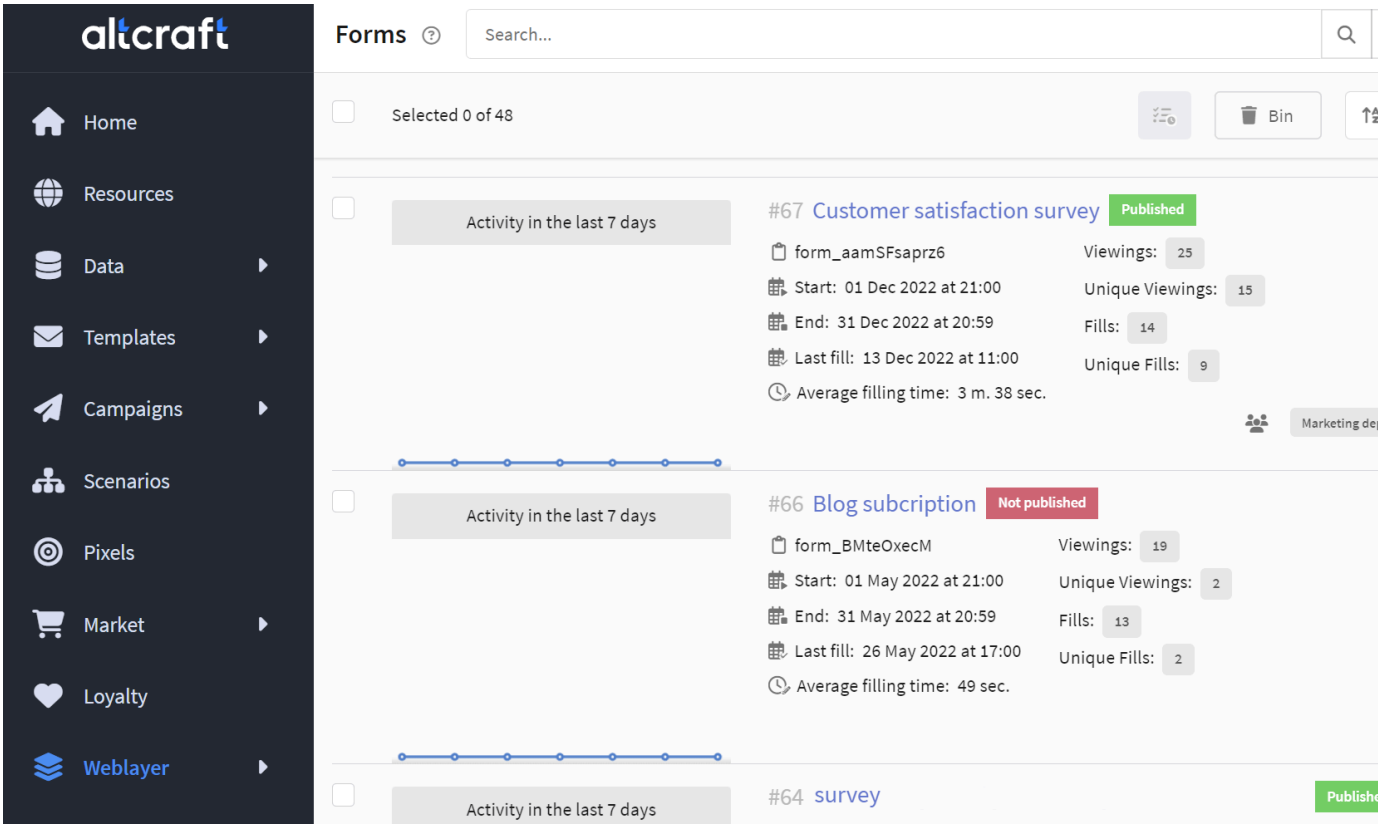
From this data, you can find the average and aggregated NPS:

$$NPS_{avg} = \frac{50\%+30\%-10\%}{3} = 23.3\%$$

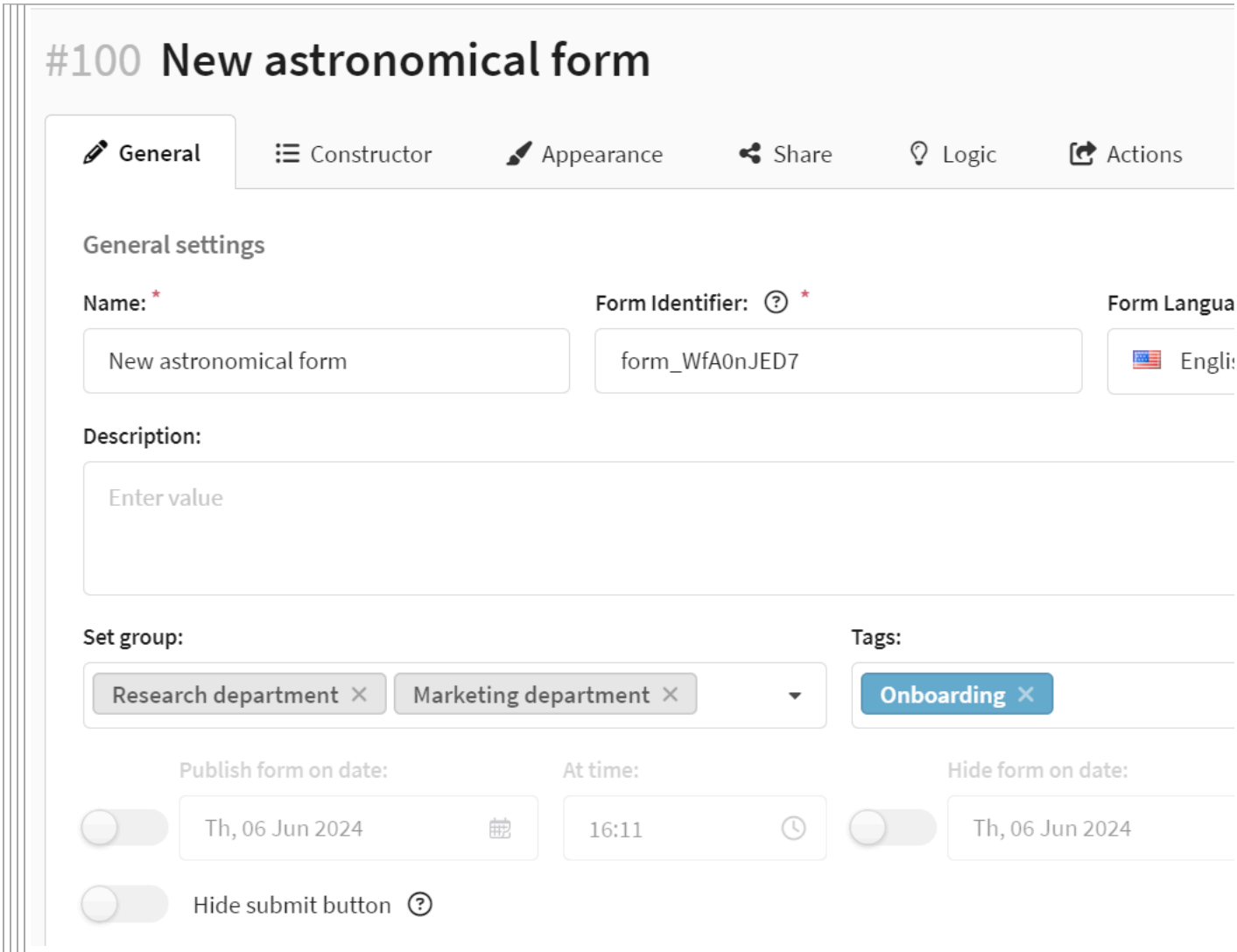
$$NPS_{agr} = \frac{(27+25+14)-(6+10+21)}{42+50+70} \times 100\%$$

Creating NPS testing in the platform

NPS testing is implemented in the Altcraft platform using webforms. To create one, go to “Weblayer” – “F Create:



On the first tab “General” the main form settings are available: name, description, identifier, language. T as an iframe element, you need to activate the Published switch in the upper right corner. Please note th cannot be deleted:



Below you can customize settings such as publishing schedule, final form page and set a refill restrictio

Publish form on date: | At time: | Hide form on date:

Mo, 11 Nov 2024 | 18:31 | Mo, 11 Nov 2024

Hide "Next" and "Submit" buttons

Final page

Redirect to the specified link bypassing the final page

Title:

Thanks!

Template:

Your form has been submitted successfully.

Refill restriction

Restrict form refill

Restrict within days

1

Redirect to the specified link if the form has been previously filled out

Title:

Restriction!

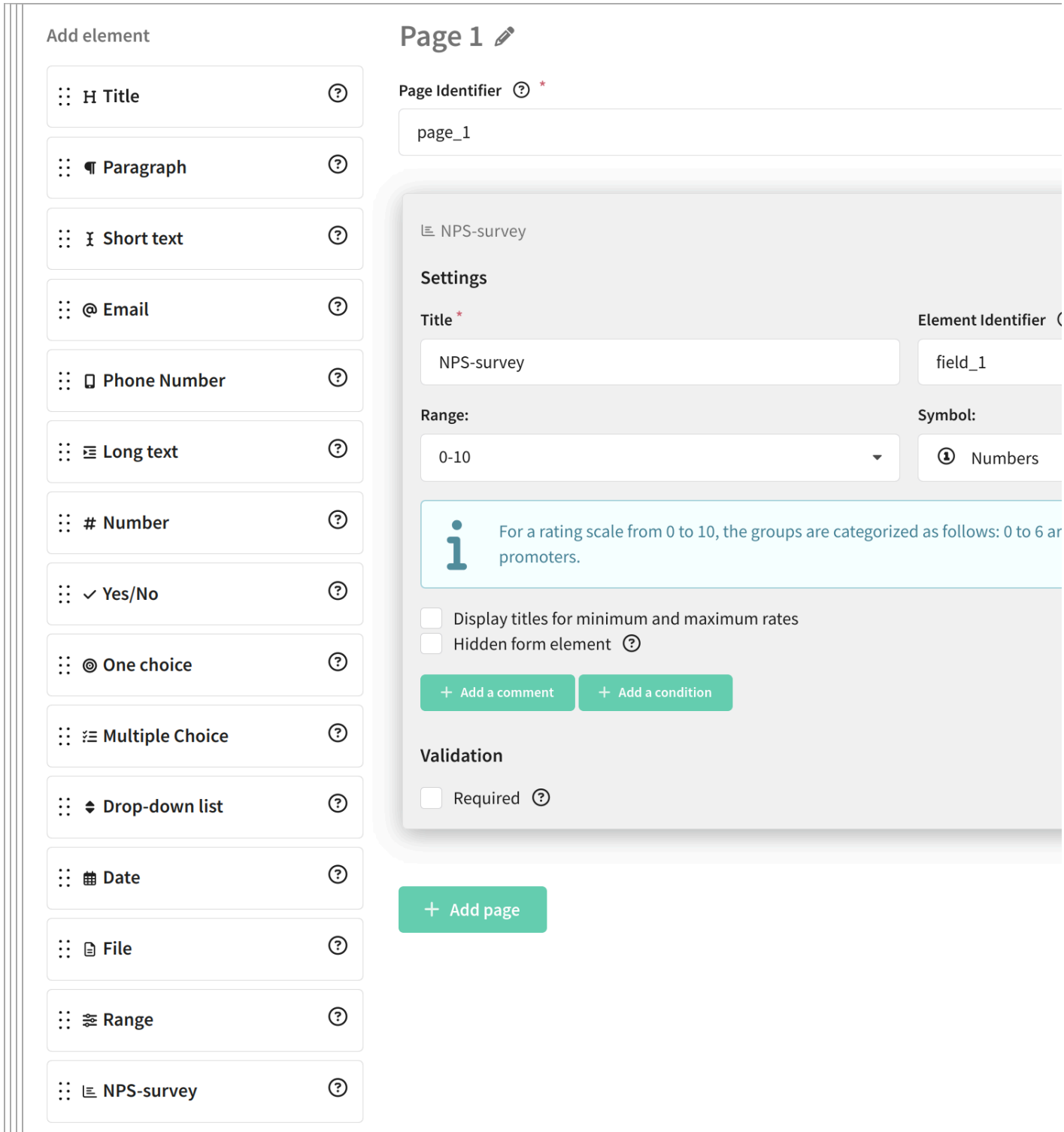
Template:

You have already completed this form.

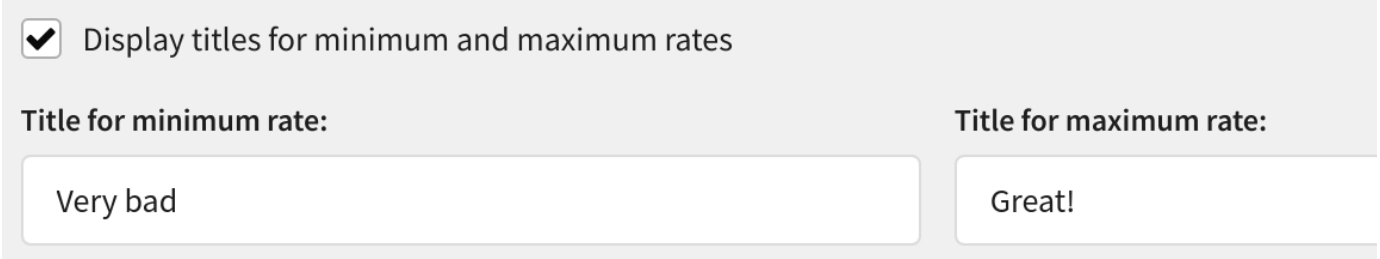
Insert additional code

- Additional code in <head /> tag
- Additional code at the beginning of the <body> tag
- Additional code at the end of <body> tag

To add an NPS survey to the form, go to the Constructor tab and add the NPS survey element:



Here you can change the title, scale and symbols for the scale. You can also make the item mandatory a maximum score:



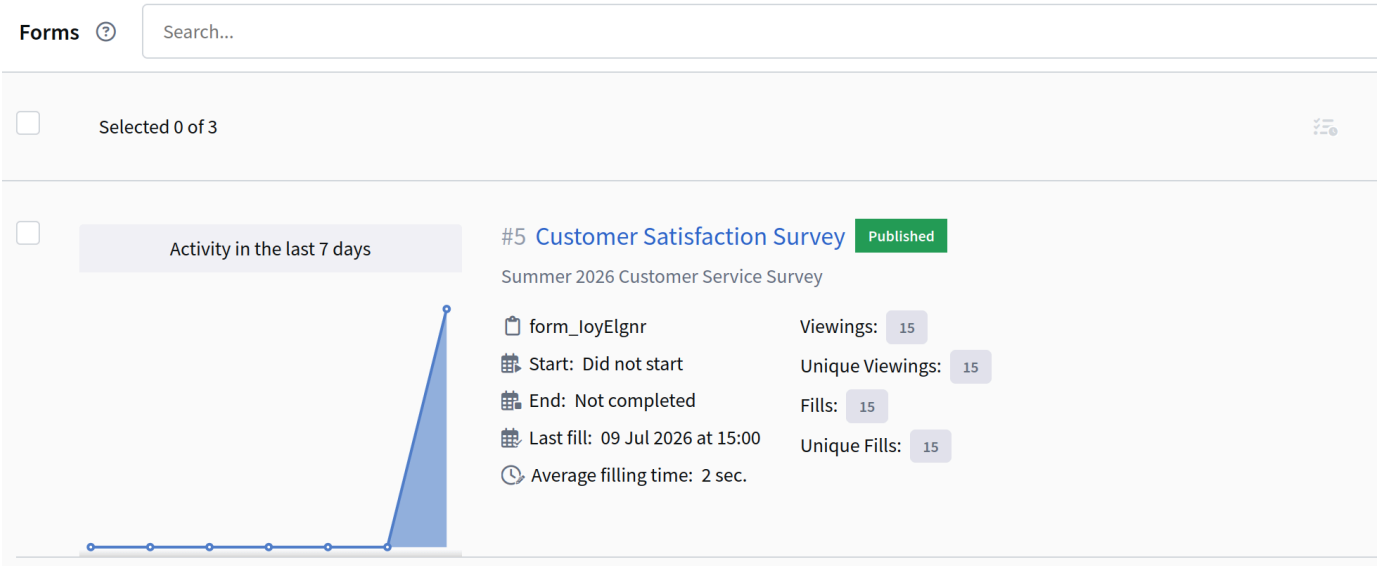
You can add other elements to the form if you wish. Once you have made your changes, save the form an NPS survey is ready.

TIP

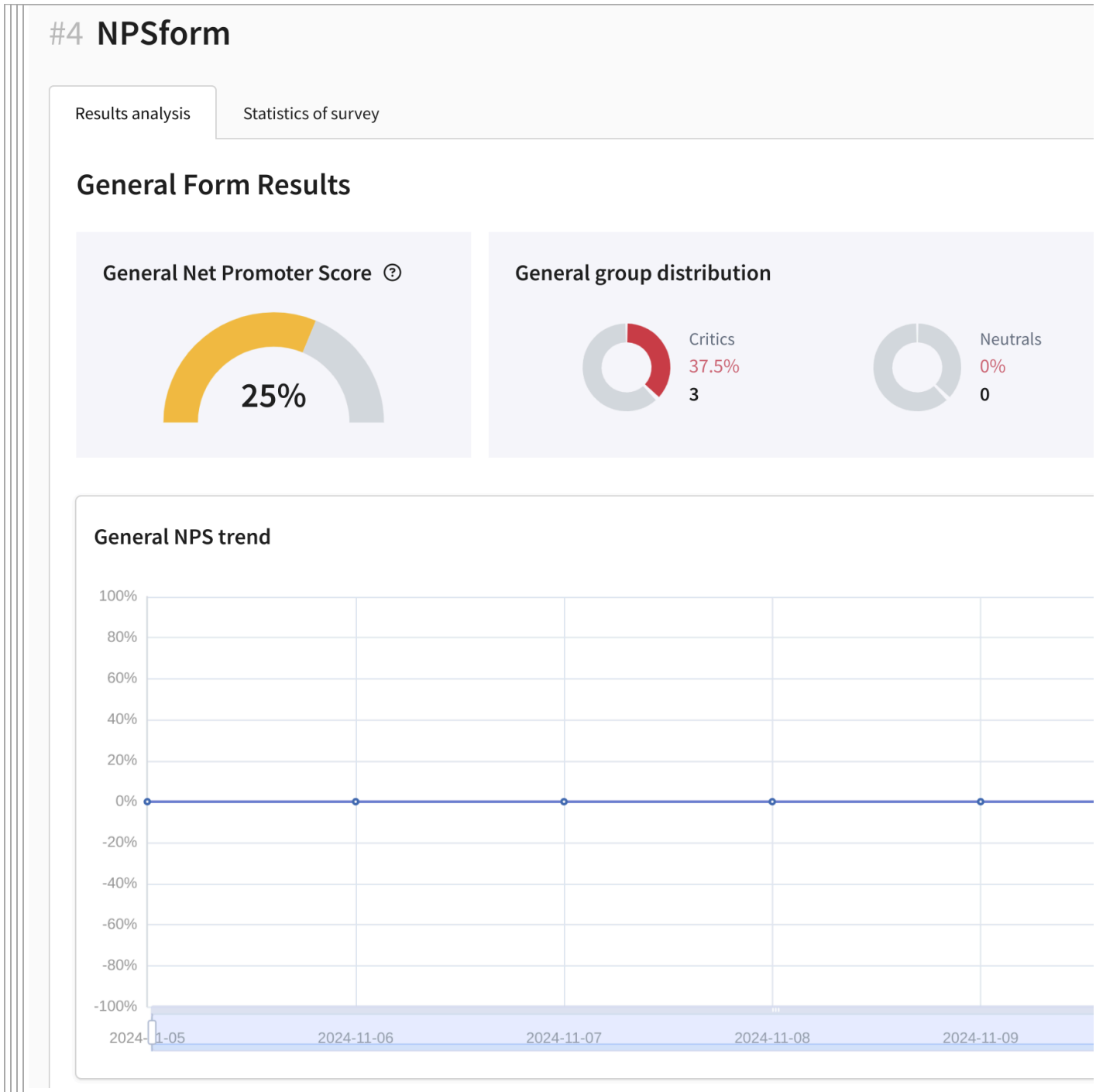
You can read more about creating and customizing a form on this page.

Testing results

You don’t need to manually collect information from forms and calculate the NPS index, you can view the in statistics of form. To do this, open the Weblayer - Forms section, find the form you need and click on t actions menu:



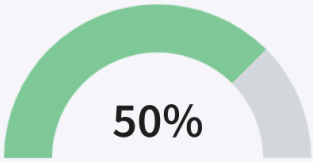
The screen will display the overall form scores - Net Promoter Score, group distribution and NPS dynami



You can also see information on every element on each page:

Form Results

Net Promoter Score



Group Distribution

Critics
25%
100

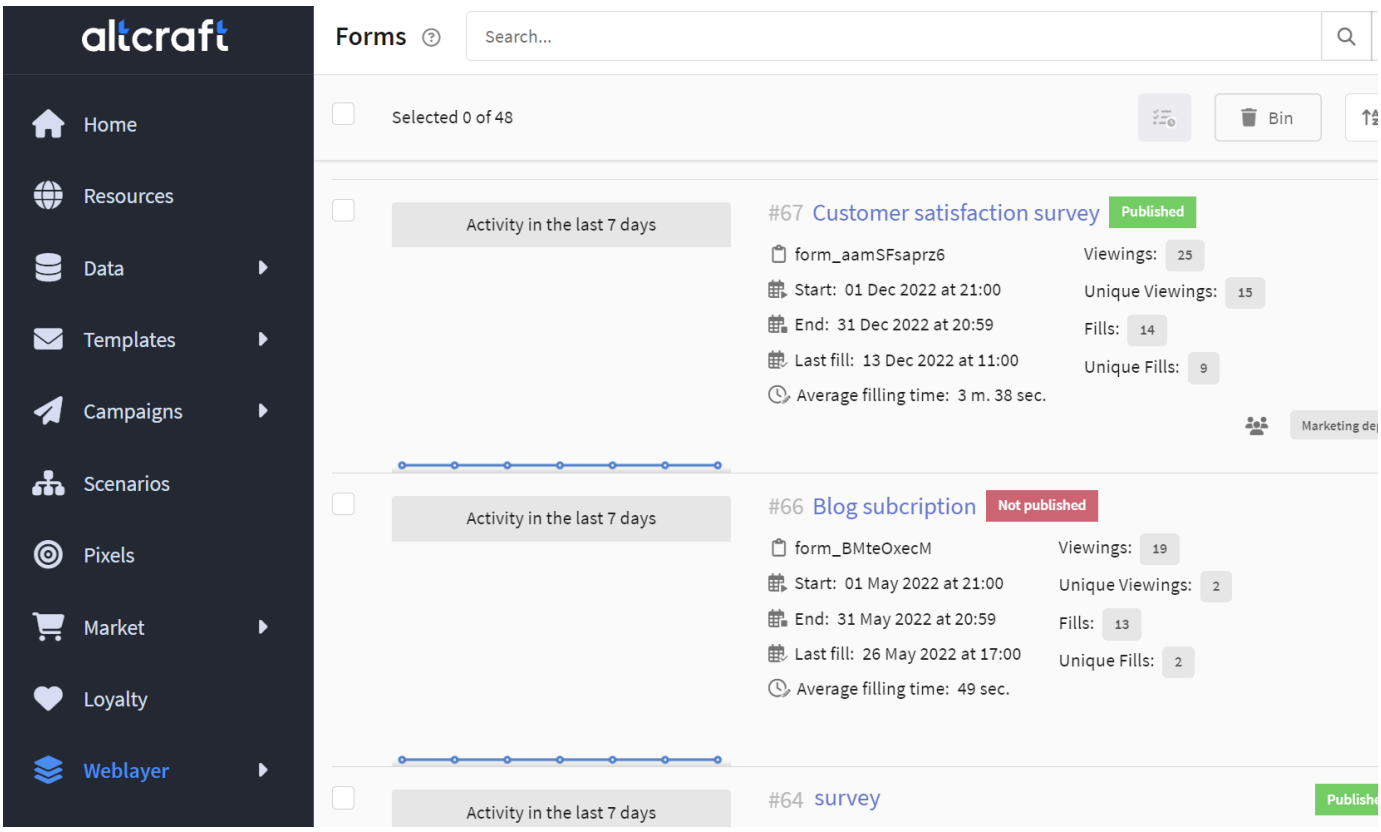
Neutrals
0%
0

Score distribution

😡	😢	😕	😕	😐	😊
0%	25%	0%	0%	0%	75%
0	100	0	0	0	300

Create a form

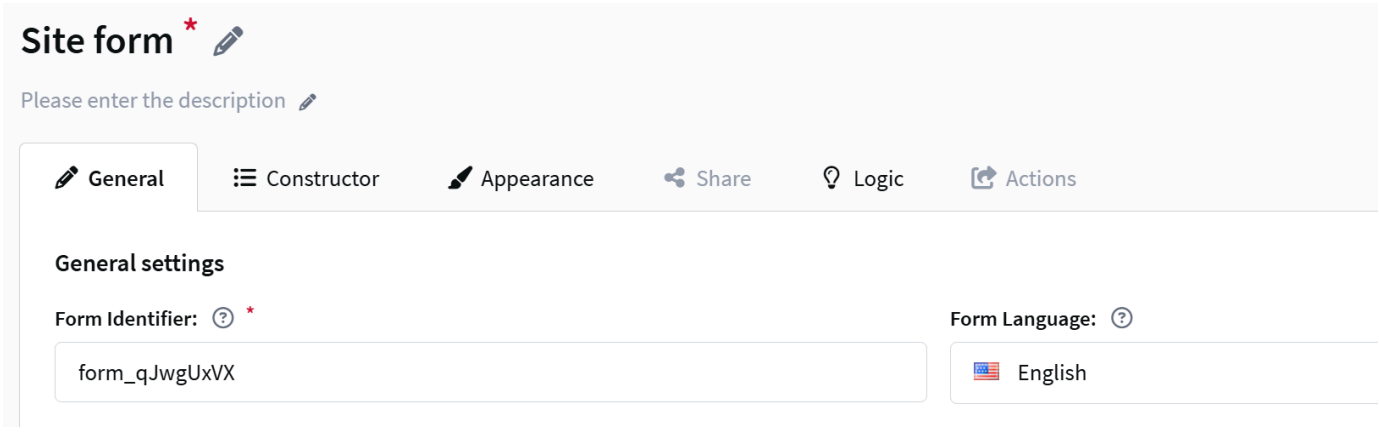
Forms allow you to collect information from users: contact details, feedback, requests, event registratio
form can be embedded on a website, sent via email, or used in newsletters.
To create a form, go to Forms in the main menu and click the Create button:



General settings

The General tab contains the main form settings: name, description, publication options, and additional f
parameters of the form before moving on to its content and design.
Key features:

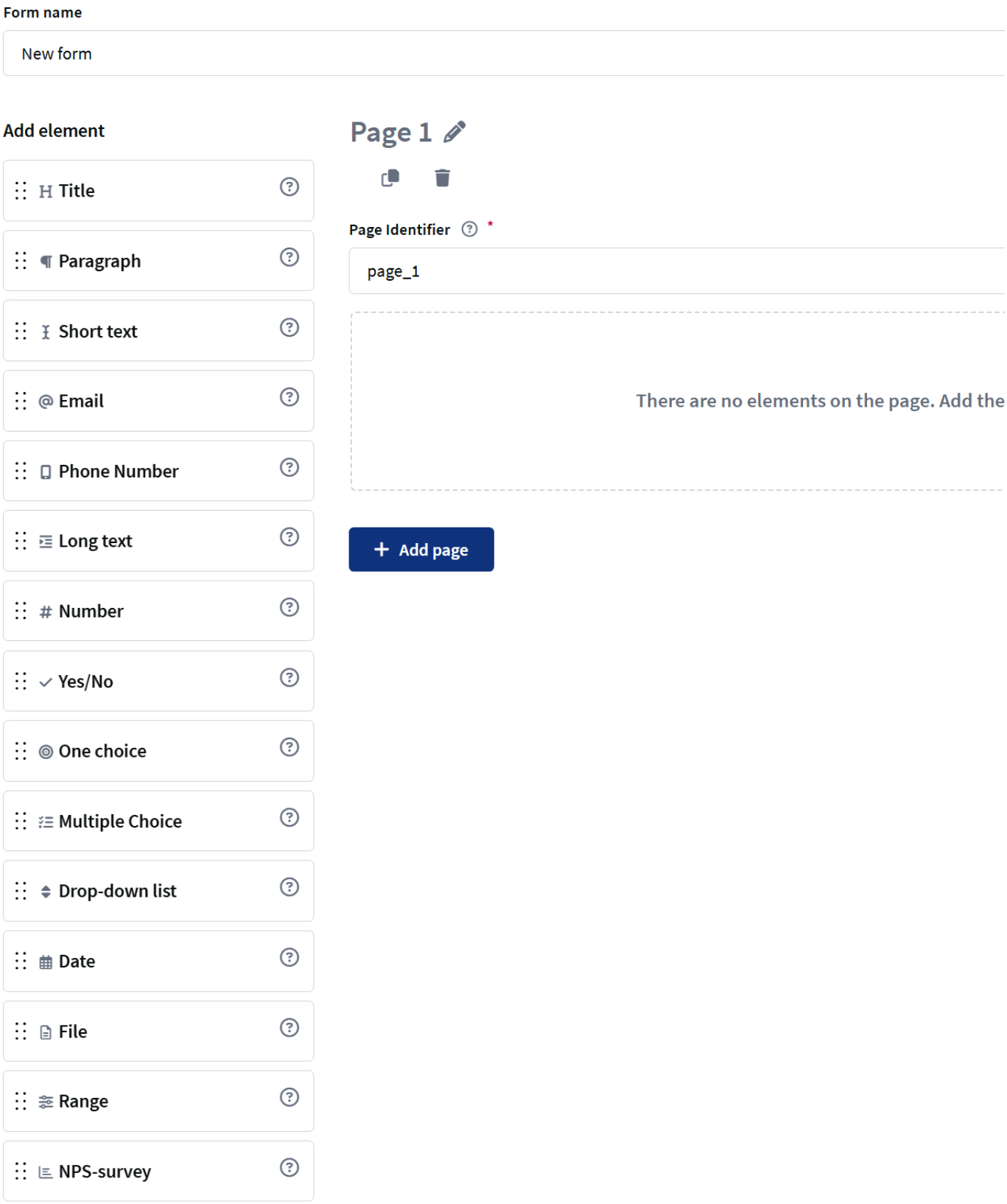
- **Configure the name, description, and form identifier**
- **Manage publication and scheduling**
- **Configure the final page after form submission**
- **Restrict repeat submissions**
- **Add custom code for customization**



TIP
You can learn more about general settings in [this article](#).

Form constructor

The Constructor tab is used to build the form structure — here you add fields, configure questions and a is the main tool for creating the form content.



To add a selected field, simply drag it from the left panel. The following field types are available:

Field Type	Description
Title	**A text header row**
Paragraph	**A text block for information**
Short text	**A field for a short answer**
Email	**An email input field with validation**
Phone number	**A phone number input field**
Long text	**A multi-line text field**
Number	**A field for numeric values**
Yes/No	**A toggle for consent**

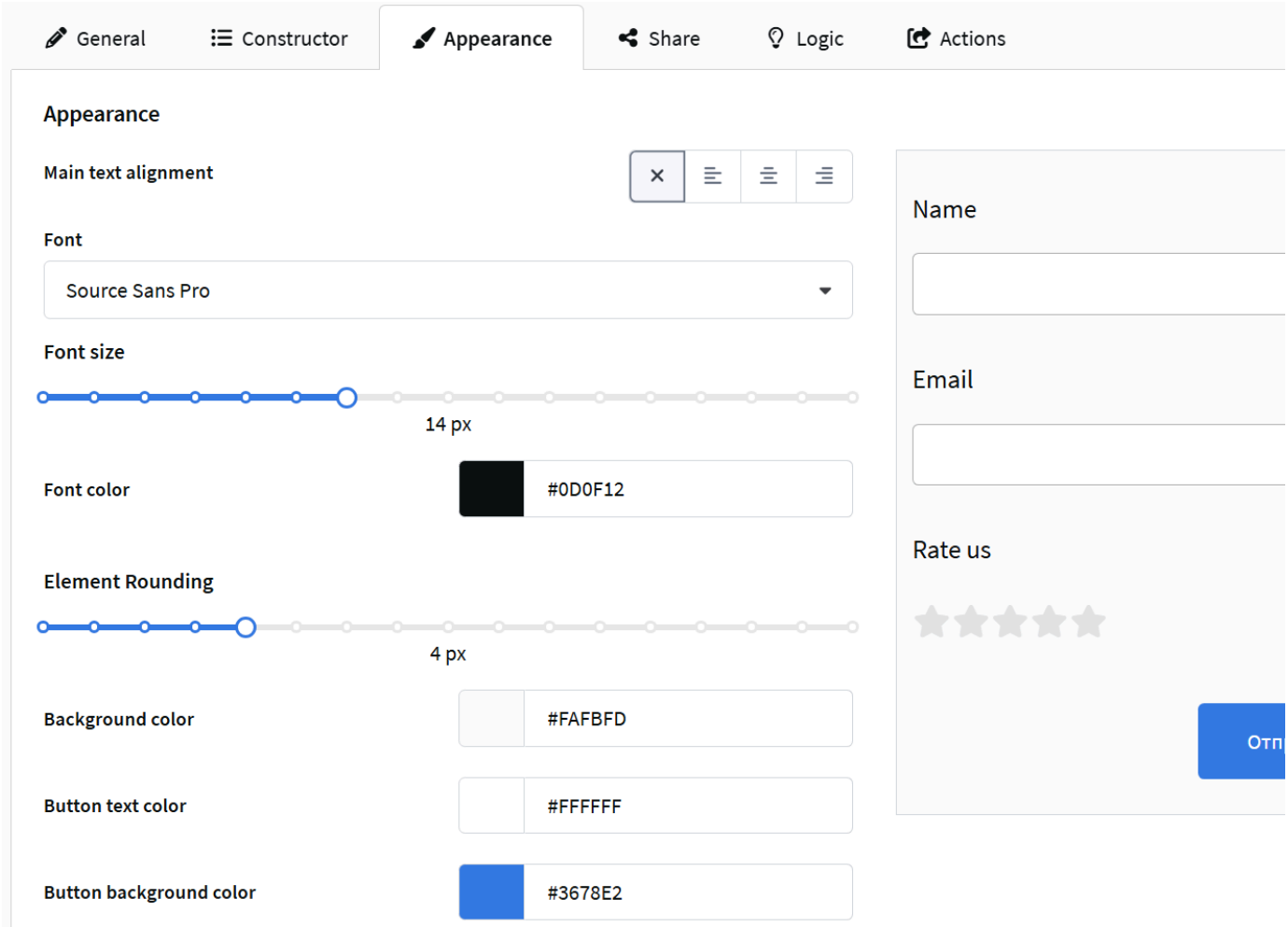
Field Type	Description
One choice	**Radio buttons for selecting a single option**
Multiple choice	**Checkboxes for selecting multiple options**
Drop-down list	**A dropdown list of options**
Date	**A date picker field**
File	**File upload (up to 20 MB)**
Range	**A rating scale (stars, numbers)**
NPS Survey	**A field for NPS testing**

TIP

You can learn more about the form constructor in this article.

Appearance

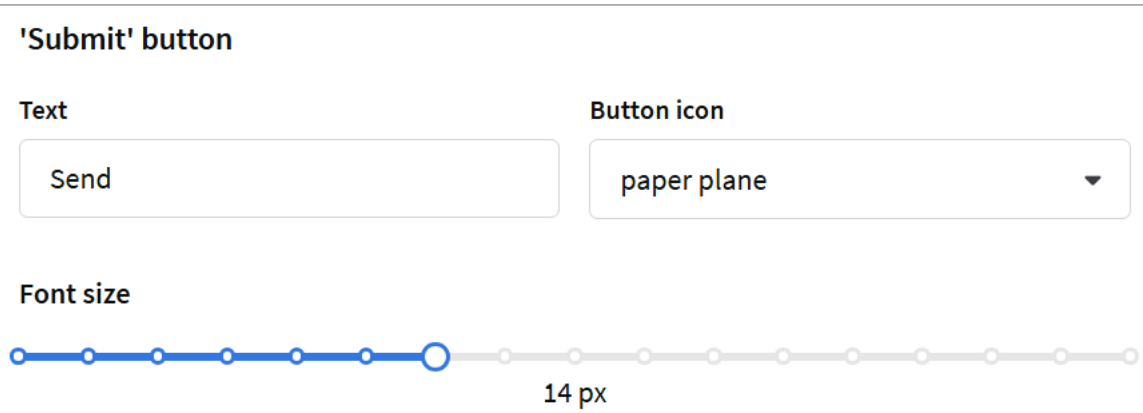
The Appearance tab allows you to customize the form's visual design — colors, fonts, element layout, an the form design to match your website or brand style.



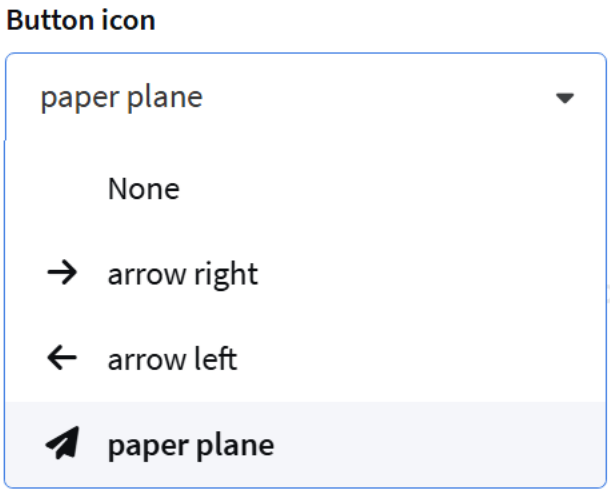
Main settings:

- **Fonts — choose the font style and size**
- **Color scheme — configure text, background, and element colors**
- **Alignment — set the form element layout**

Button styling:



For each button, configure the text, font size, and icon:



TIP

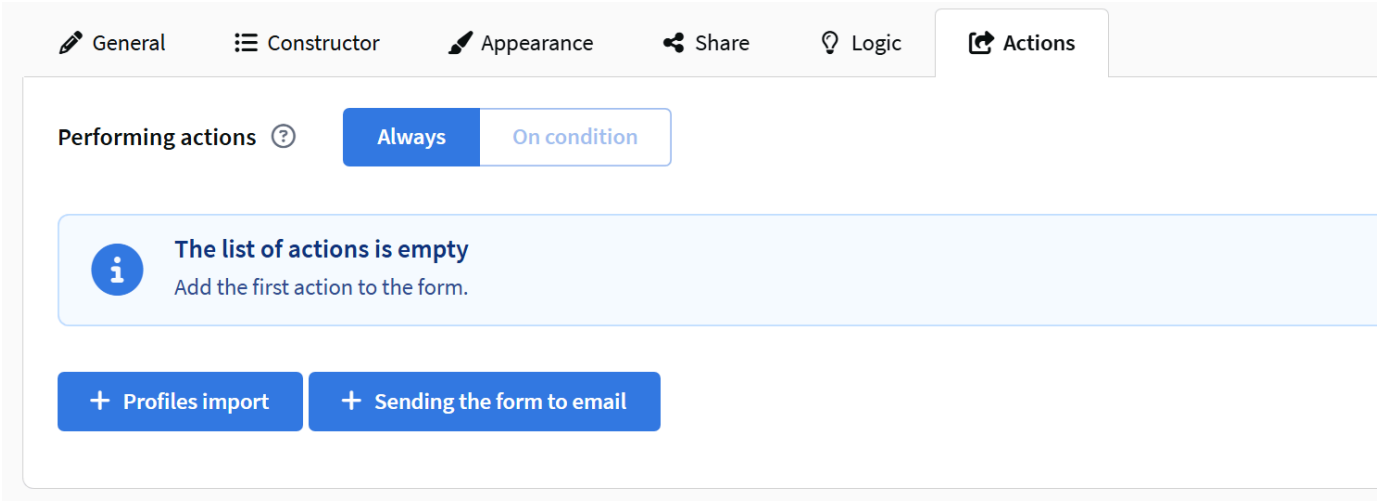
You can learn more about form appearance in this article.

After saving the form

After configuring all the parameters, you need to save the form. Only then will additional tabs for configu available.

Actions

The Actions tab configures automated processes after form submission — importing data into the databa you define what happens to the information after the form is submitted:



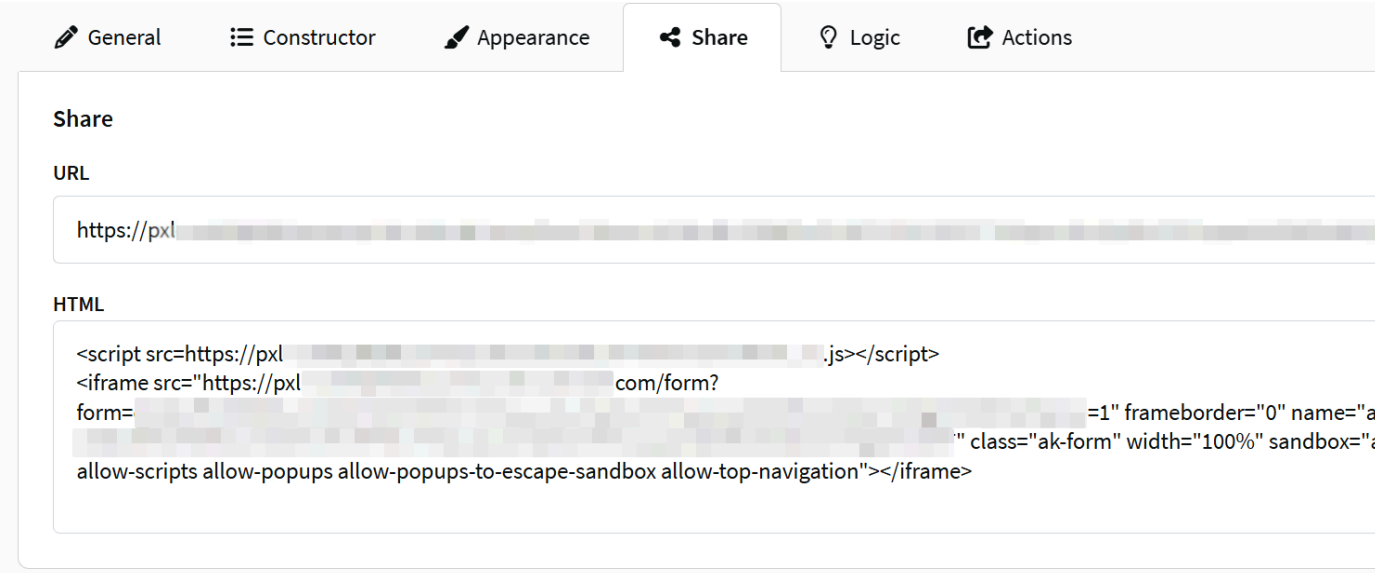
- **Profiles import — upload data to the database**
- **Send email notifications — email alerts about form submissions**

TIP

Learn more about the Actions tab.

Share

The Share tab provides access to codes and links for publishing the form on your website. Here you get r the form:



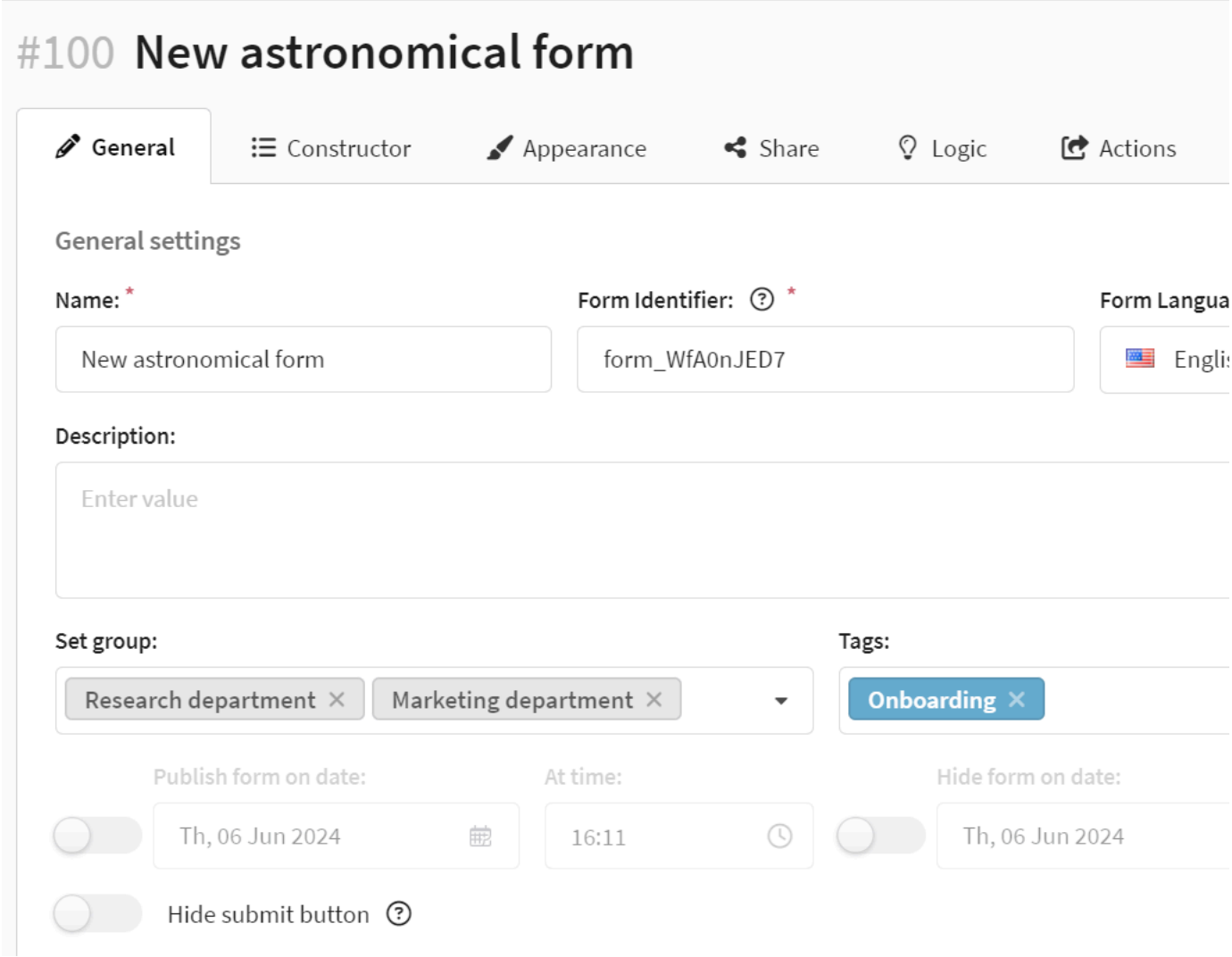
- **URL — a direct link to the form**
- **HTML code — code for embedding on the website**
- **Embedding parameters — as an iframe or popup window**

TIP

Detailed information can be found in this article.

General settings

On the General tab, you can configure the basic form parameters: name, description, publication setting



Basic form settings

- **Name — form name for identification in the system**
- **Description — additional information about the form**
- **Identifier — unique form ID**
- **Language — form interface language**
- **Access group — access rights restrictions for the form**
- **Tags — labels for convenient form organization**

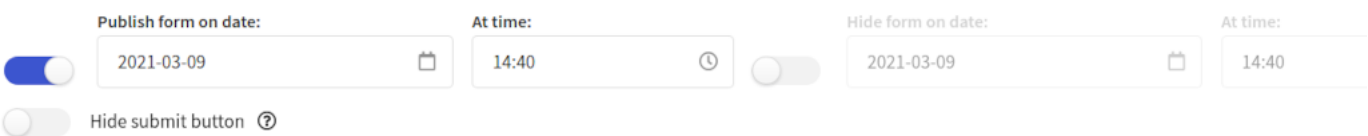
Form publication

To make the form available via URL or as an iframe element, activate the Published switch in the upper r

> CAUTION
> **While the form is published, it cannot be deleted.**

Publication scheduling

Set the start and end date and time for publishing the form for temporary access:



The form will be available for filling out on the specified date and time and will be automatically unpublis
Published switch should remain inactive — the system will update its status automatically.

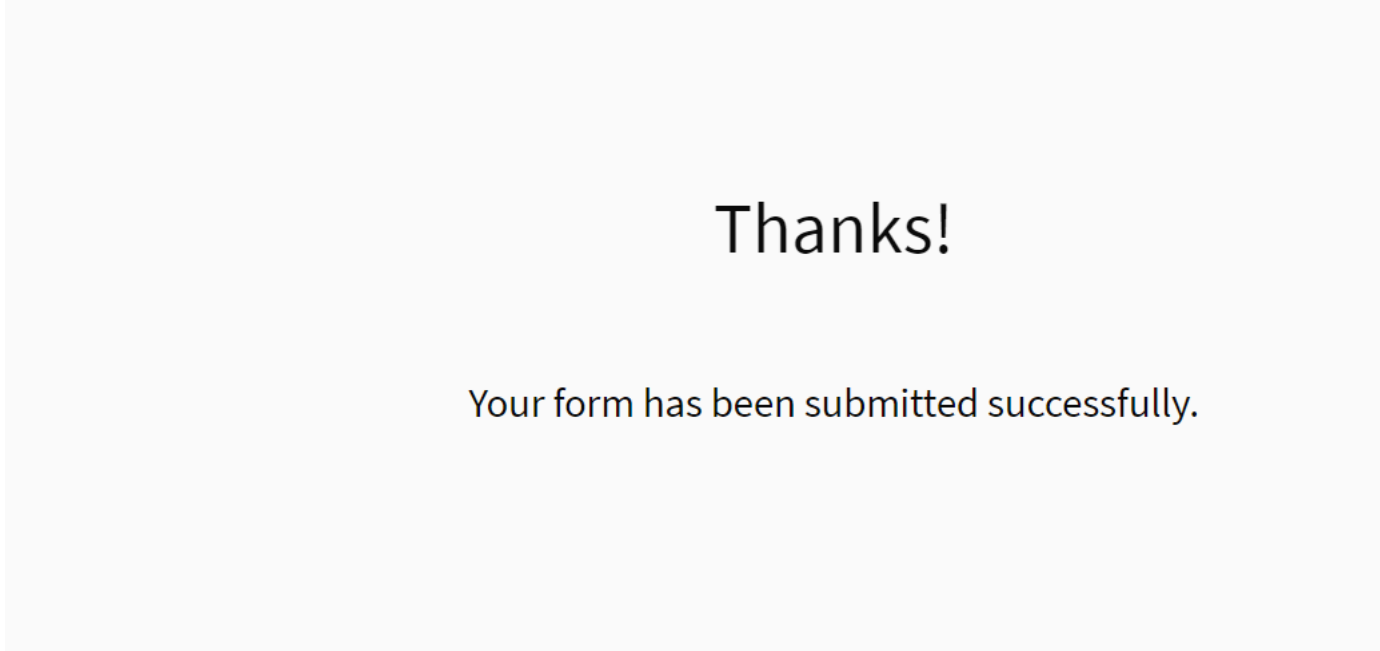
INFO

The system uses a cache to store forms. Changes in form publication may take up to 3 minutes to take

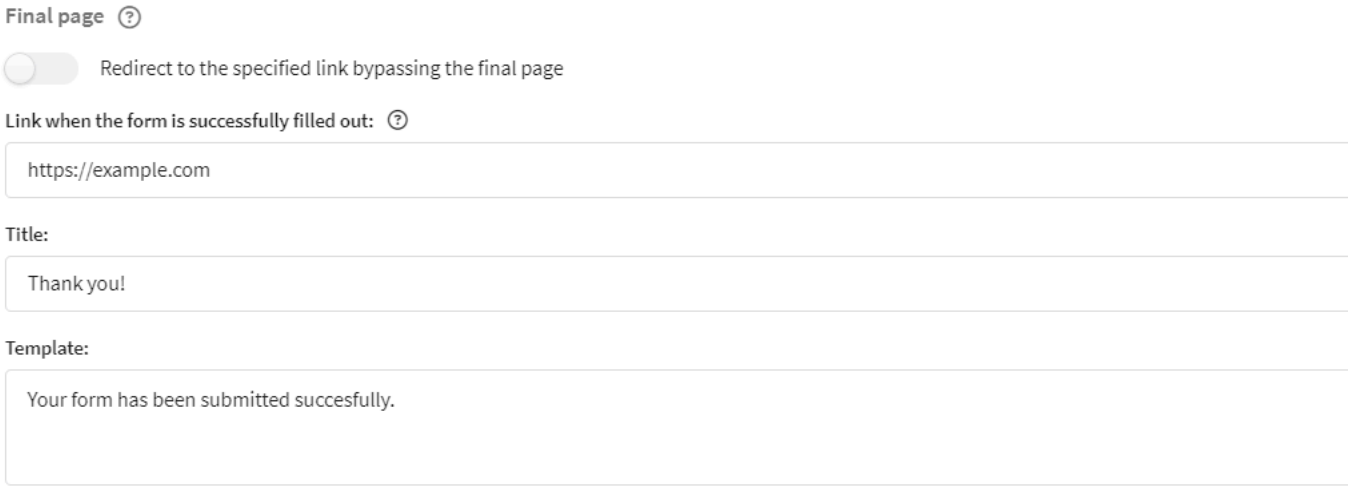
When the form is not published, users trying to access it will see the message Form is unavailable.

Final page

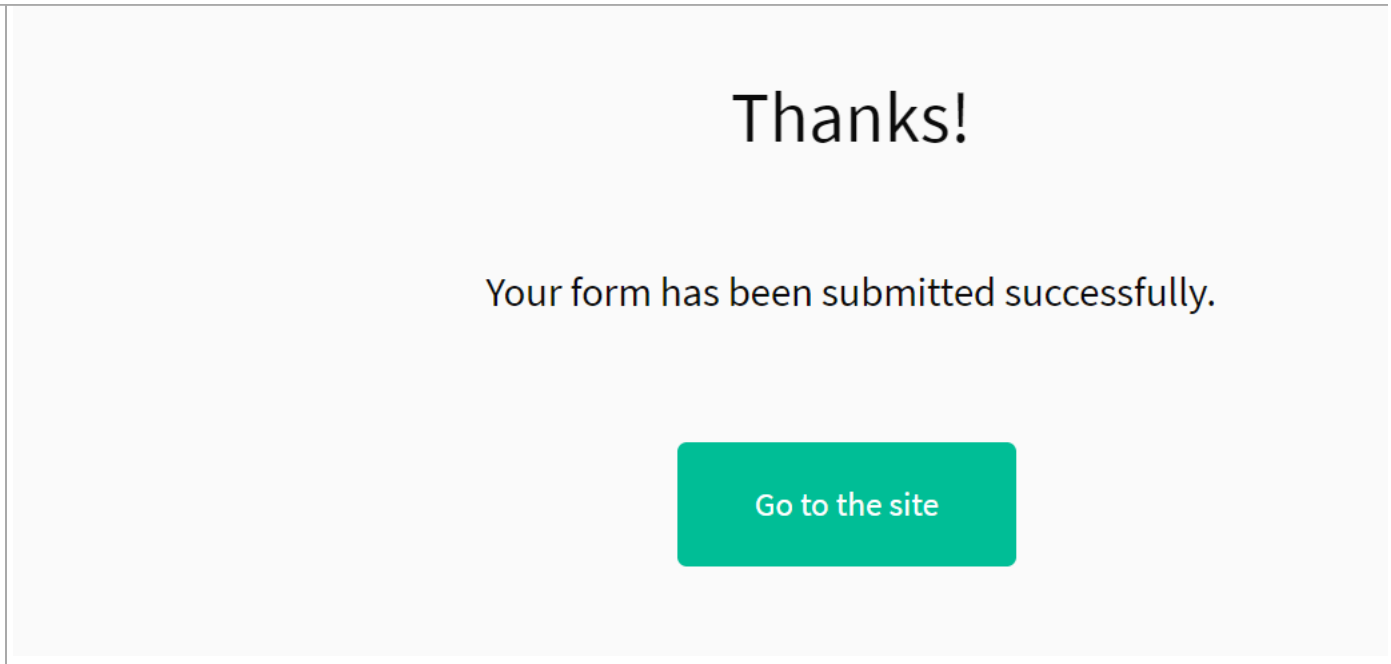
Configure the page that users will see after submitting the form:



Change the Title and Message on the final page:



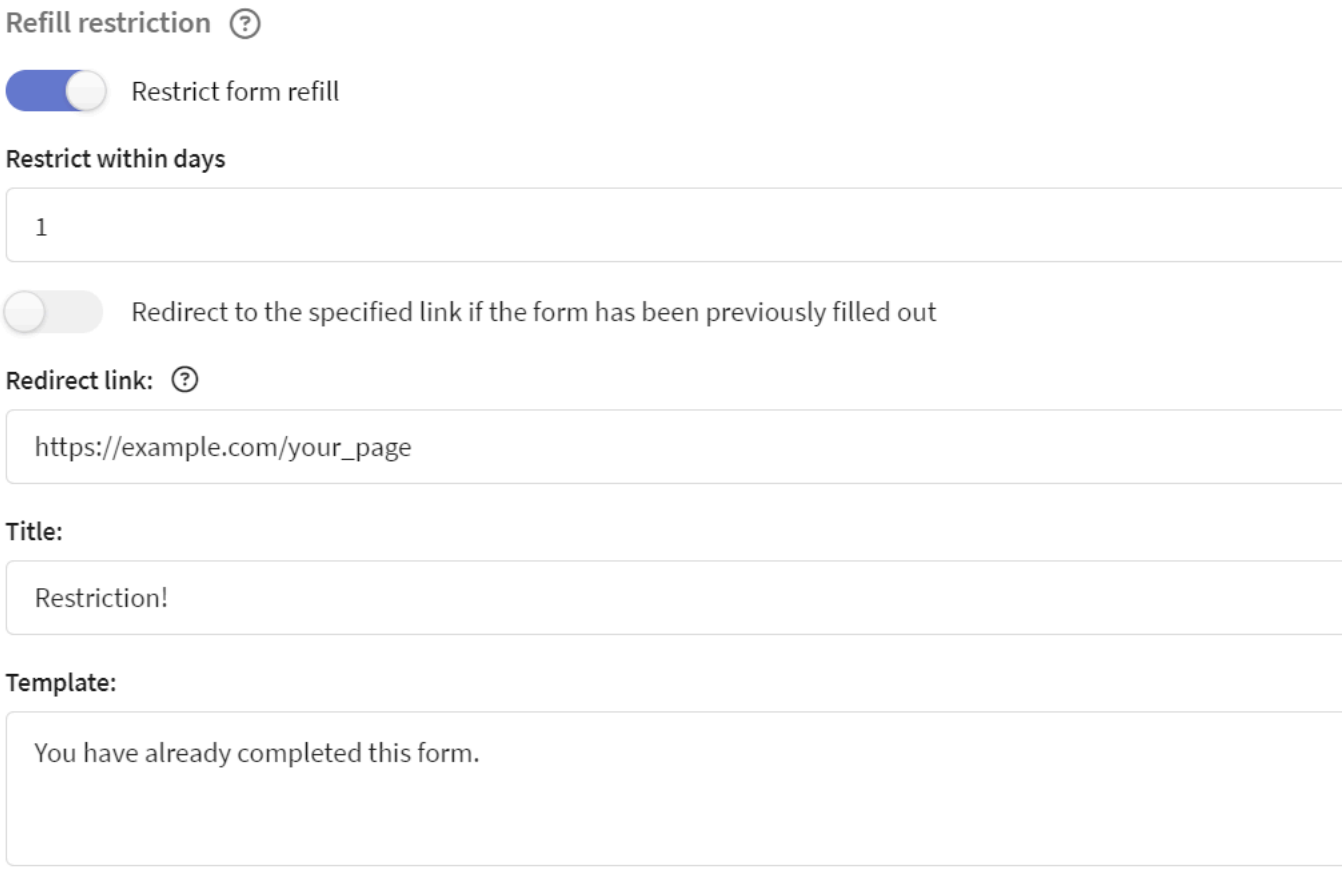
The link when the form is successfully filled out is displayed as a Go to the site button. The button text c tab:



Activate the Redirect to the specified link bypassing the final message switch to redirect the user imme

Refill restriction

Protect the form from multiple submissions:

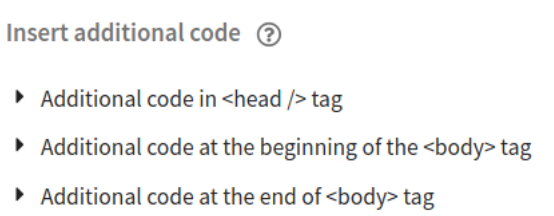


Activate the switch and set the number of days to restrict refilling.
The system records form submission data in cookie files. When attempting to reopen the form, the user
You have already completed this form. The message text can be changed in the Title and Message fields
Additional options:

- **Specify a redirect link when attempting to resubmit**
- **Activate Redirect to the specified link if the form has been previously filled out for automatic redirec**

Form customization

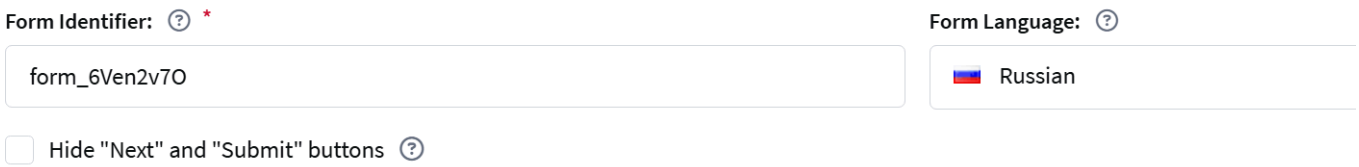
Add custom code for advanced customization:



- **In the `<head>` tag — for connecting styles and scripts**
- **After the opening `<body>` tag — for additional elements**
- **Before the closing `</body>` tag — for analytics scripts**

Additional options

Hide submit button — activate this option to prevent showing the “Submit” and “Next” buttons to the use fields on the page:

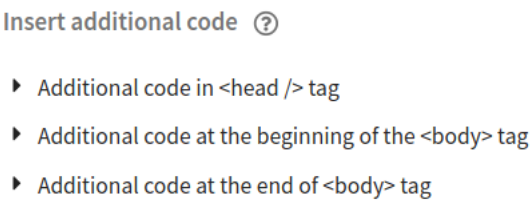


TIP

In the On-premises version of the platform, administrators can customize the appearance and behavior page, etc.) by modifying template files. More details in admin documentation.

Form customization with custom code

The standard appearance settings allow you to change fonts, colors, element rounding, and buttons. But need to add a brand logo, set a background image or gradient, add a shadow or border around the form. For this, the General tab has a Insert additional code block:



Three areas are available for inserting code:

- **In the `<head>` tag — for styles and scripts**
- **After the opening `<body>` tag — for additional elements**
- **Before the closing `</body>` tag — for analytics scripts**

All examples below are inserted into the In the `<head>` tag field.

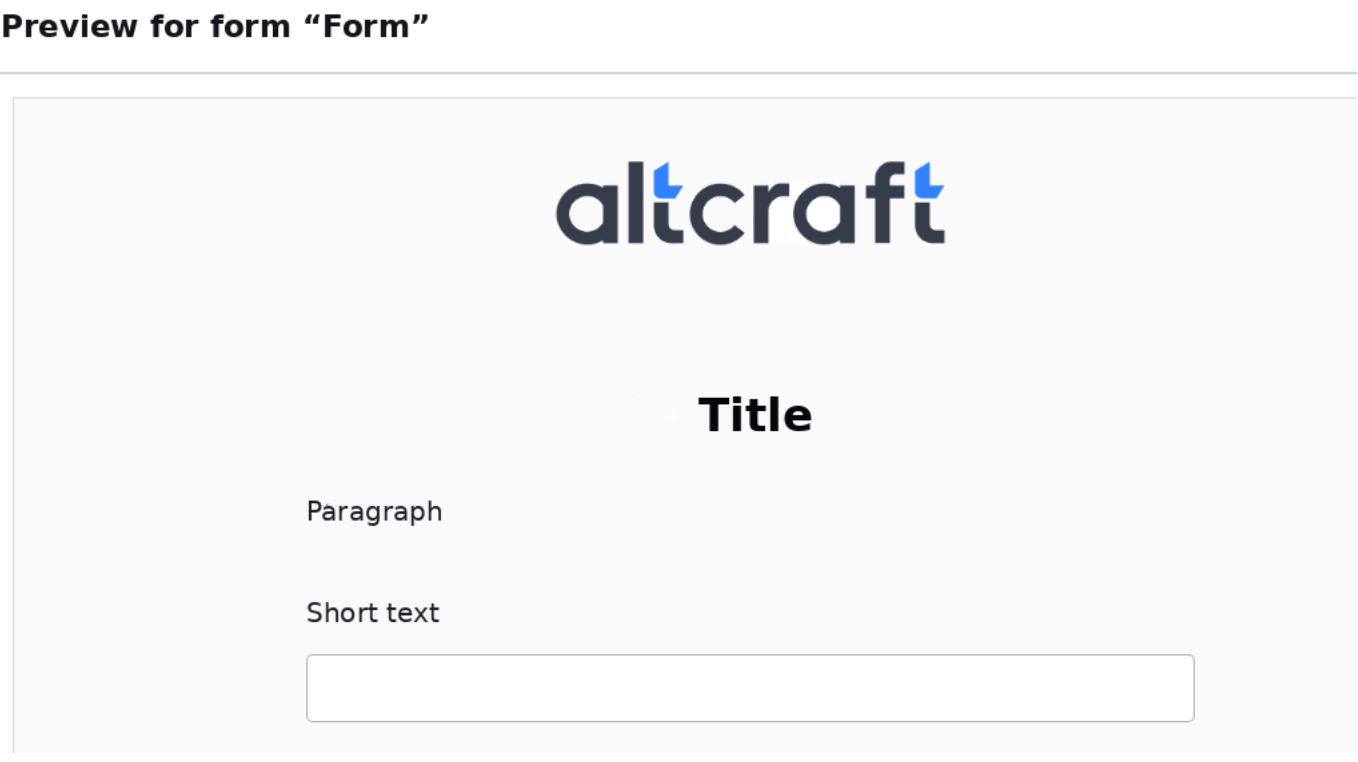
Adding a logo

Insert the code into the In the `<head>` tag field:

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    var firstElement = document.querySelector('.form .element');
    if (firstElement) {
      var logo = document.createElement('div');
      logo.style.textAlign = 'center';
      logo.style.marginBottom = '20px';
      logo.innerHTML = '<img src="https://example.com/logo.png" width="150" />';
      firstElement.parentNode.insertBefore(logo, firstElement);
    }
  });
</script>
```

What to replace:

- **`https://example.com/logo.png` — direct HTTPS link to your logo**
- **`width="150"` — width in pixels (can use percentages, e.g. `width="20%"`)**
- **`'center'` — alignment (`'left'`, `'center'`, `'right'`)**
- **`'20px'` — spacing between the logo and the first form element**



TIP

Use a PNG with a transparent background for the best result.

Background image or gradient

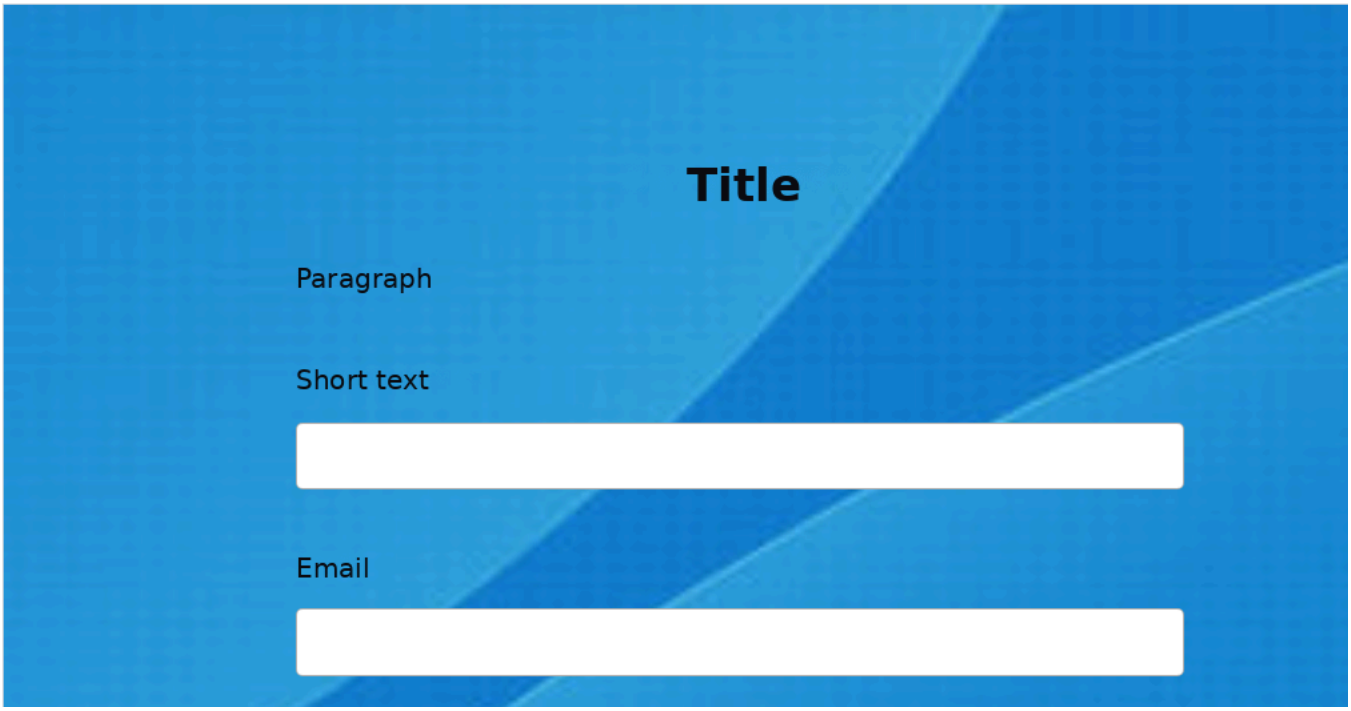
The Appearance interface only supports a solid background color. With custom code, you can set an ima

Background image

```
<style>
  body {
    background-image: url("https://example.com/background.jpg");
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    min-height: 100vh;
    margin: 0;
  }
</style>
```

Replace the link with your image. `background-size: cover` **stretches the image to fill the screen,** `contain` **sho areas on the edges.**

Preview for form “Form”



Gradient

```
<style>
  body {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    margin: 0;
  }
</style>
```

Replace the colors with your brand colors. `135deg` **is a diagonal gradient. You can also use** `180deg` **(top to (bottom to top).**

Radial gradient:

```
<style>
  body {
    background: radial-gradient(circle, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    margin: 0;
  }
</style>
```

Preview for form “Form”

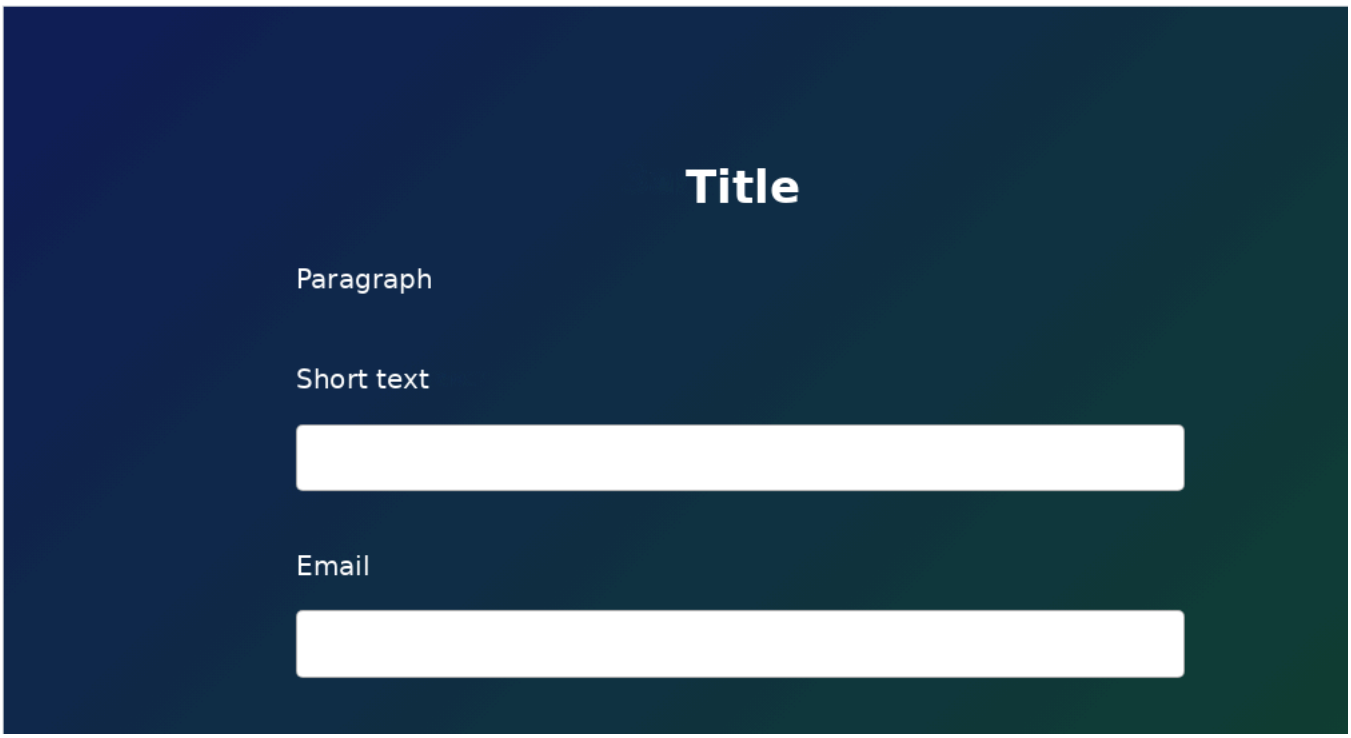


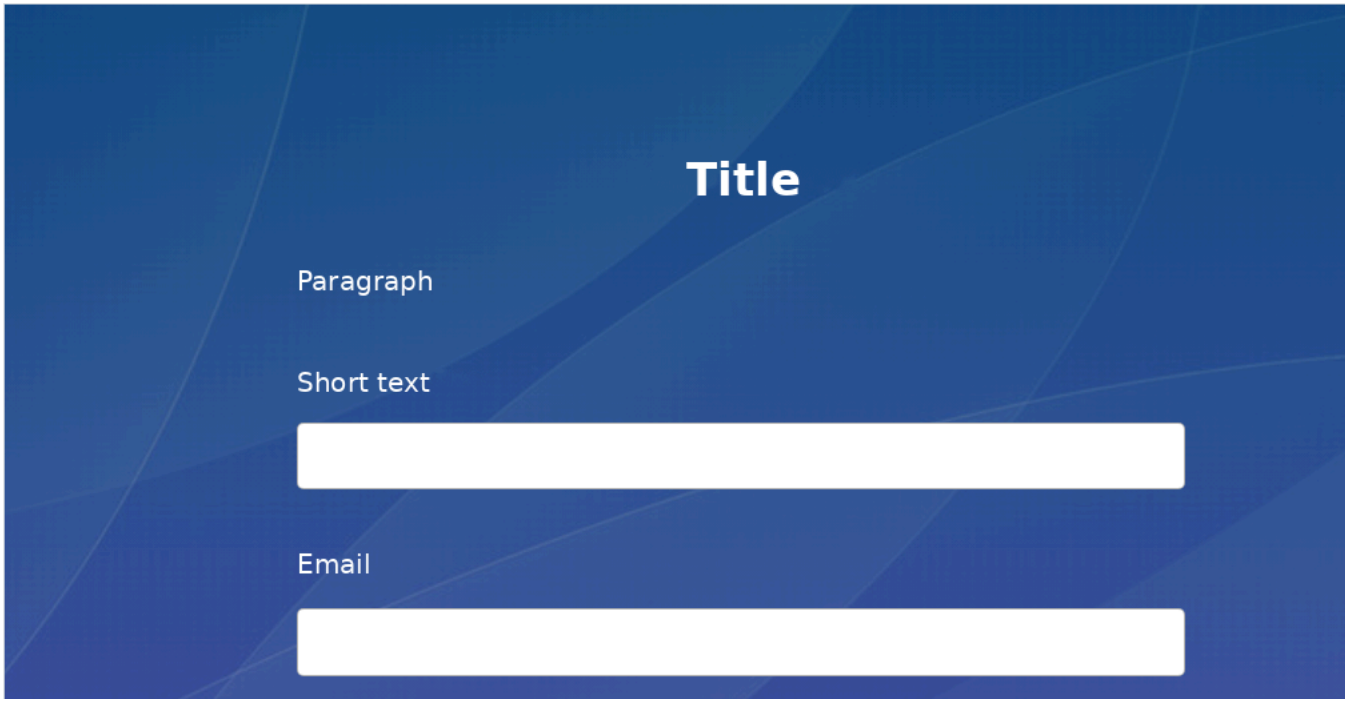
Image with gradient overlay

A semi-transparent gradient over an image improves readability:

```
<style>
  body {
    background:
      linear-gradient(180deg, rgba(102,126,234,0.7) 0%, rgba(118,75,162,0.7) 100%),
      url("https://example.com/background.jpg");
    background-size: cover;
    background-position: center;
    min-height: 100vh;
    margin: 0;
  }
</style>
```

The `0.7` value in `rgba` is the gradient opacity (from 0 to 1). Decrease it to make the image brighter; incre dominant.

Preview for form “Form”



Form card styling

The form card is the white block containing the fields. With custom code, you can add a shadow, border,

Shadow and border

```
<style>
  .form {
    box-shadow: 0 10px 40px rgba(0, 0, 0, 0.3);
    border: 2px solid rgba(255, 255, 255, 0.2);
  }
</style>
```

- `box-shadow` — **shadow (X offset, Y offset, blur, color)**
- `border` — **border (thickness, style, color)**

Width and centering

```
<style>
  .form {
    max-width: 600px;
    margin: 40px auto;
    padding: 30px;
  }
</style>
```

- `max-width` — **maximum form width**
- `margin: 40px auto` — **top/bottom spacing and horizontal centering**
- `padding` — **inner padding**

Rounded corners

```
<style>
  .form {
    border-radius: 16px;
  }
</style>
```

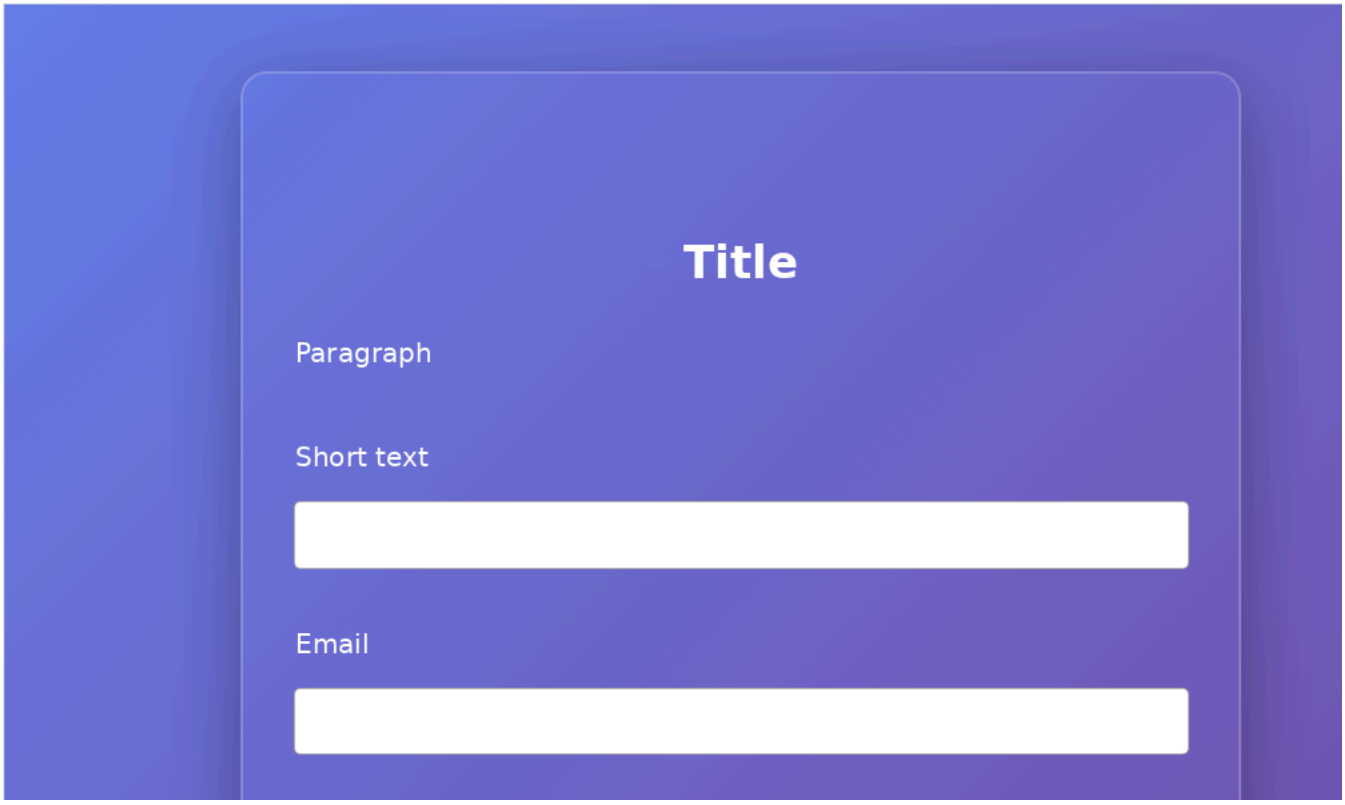
> **NOTE**
>
> **The Appearance tab has a “Element rounding” setting — it rounds fields and buttons inside the form.** `bo`
> **itself.**

Full card example

```
<style>
  body {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    margin: 0;
  }

  .form {
    box-shadow: 0 10px 40px rgba(0, 0, 0, 0.3);
    border: 2px solid rgba(255, 255, 255, 0.2);
    max-width: 600px;
    margin: 40px auto;
    padding: 30px;
    border-radius: 16px;
  }
</style>
```

Preview for form “Form”



How to preview the result

After adding the code, click the preview button

in the upper right corner:

Changes are applied instantly. If something doesn’t look right, go back to the General tab, edit the code,

> TIP
>
> **After publishing the form, changes may take up to 3 minutes to appear due to caching.**

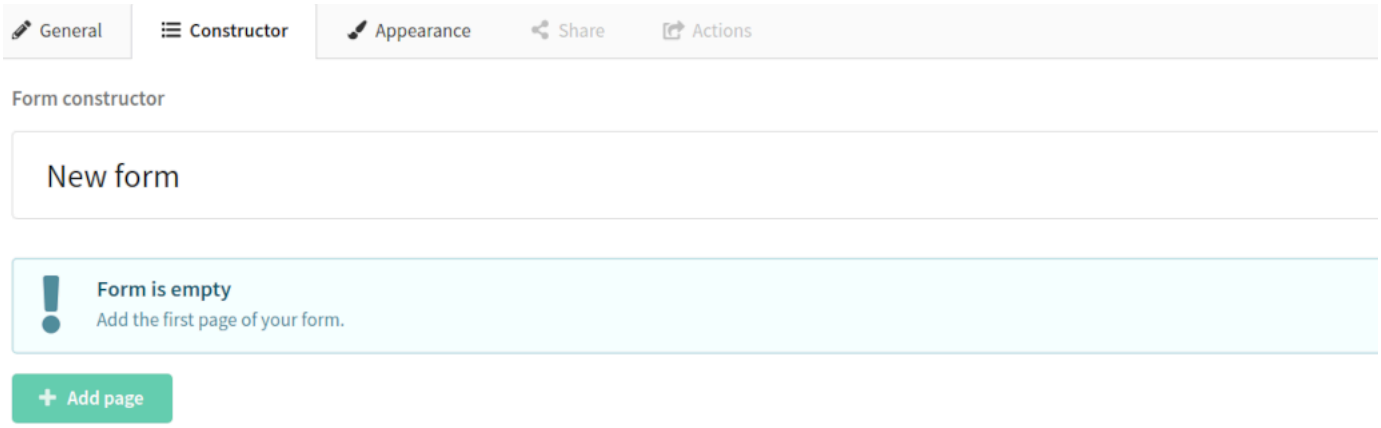
Recommendations

- **Image links must be absolute and start with `https://`. Relative paths do not work.**
- **Always wrap CSS code in `<style>...</style>` tags when inserting into the In the `<head>` tag area.**
- **CSS rules take precedence over Appearance tab settings. If you set values in both places, the code**
- **Test on mobile — switch to mobile view in the preview to make sure the logo and background displa**

- **If the form is embedded on a website via iframe and you want the form background to be transparen add `body { background: transparent !important; }`.**

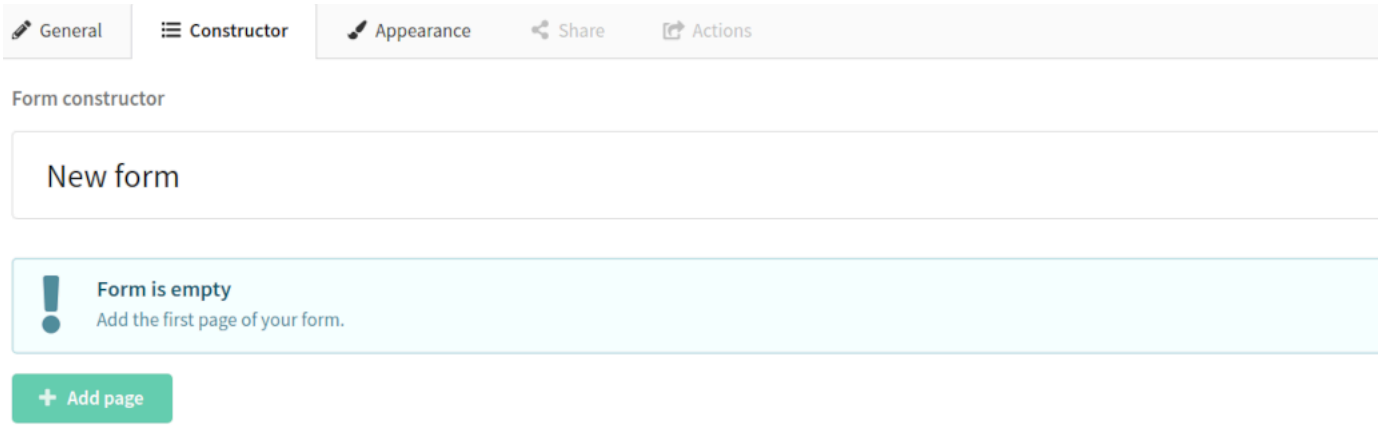
Form constructor

The second tab Constructor is used to configure the form structure and its fields. Here you can set up fo define their type, required status, visibility, and display logic:



- **The right side of the screen displays the form pages and their elements.**
- **The left column shows the element types that can be added to the form.**

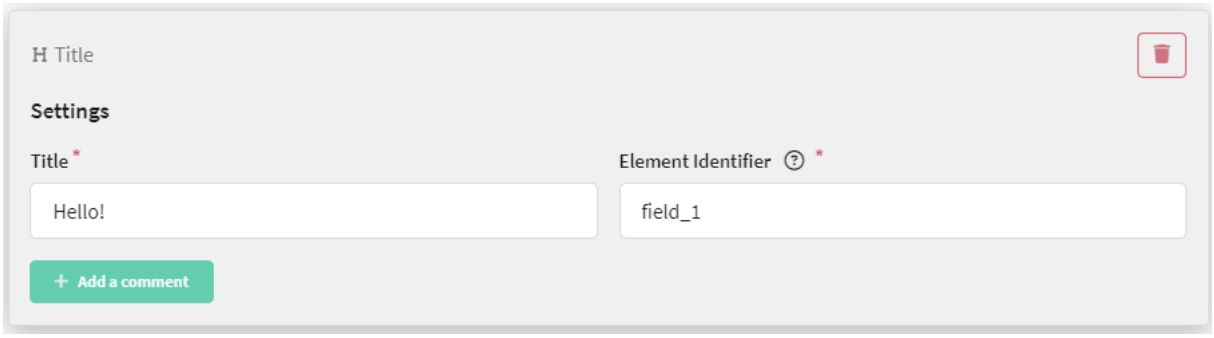
Working in the constructor starts with creating a page — click Add page:



Elements and their description

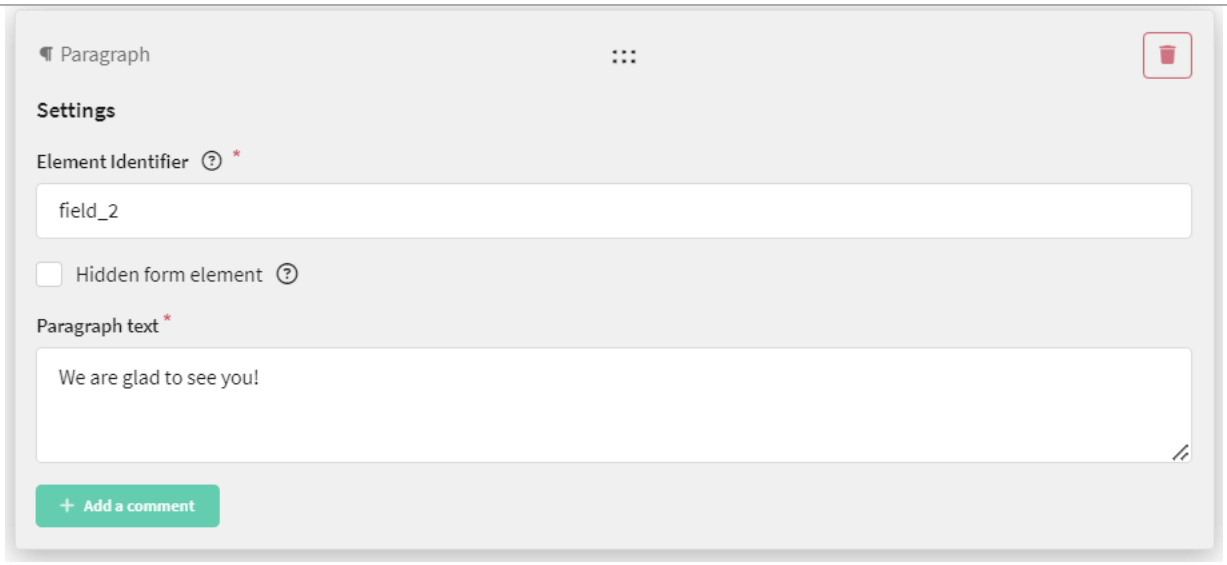
Title

Title is a text element where you can specify the main topic of your form.



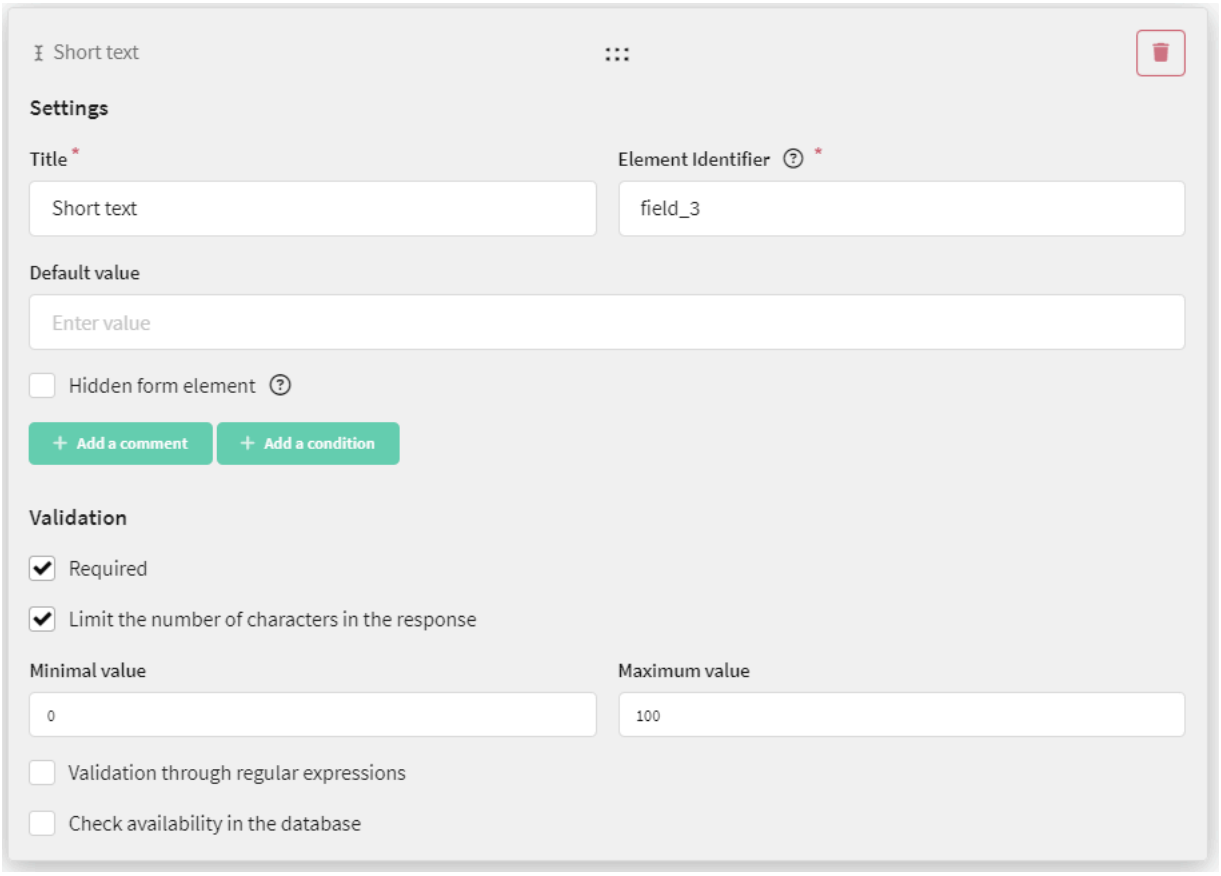
Paragraph

Paragraph is a text element. It can contain information that should be displayed in the form.



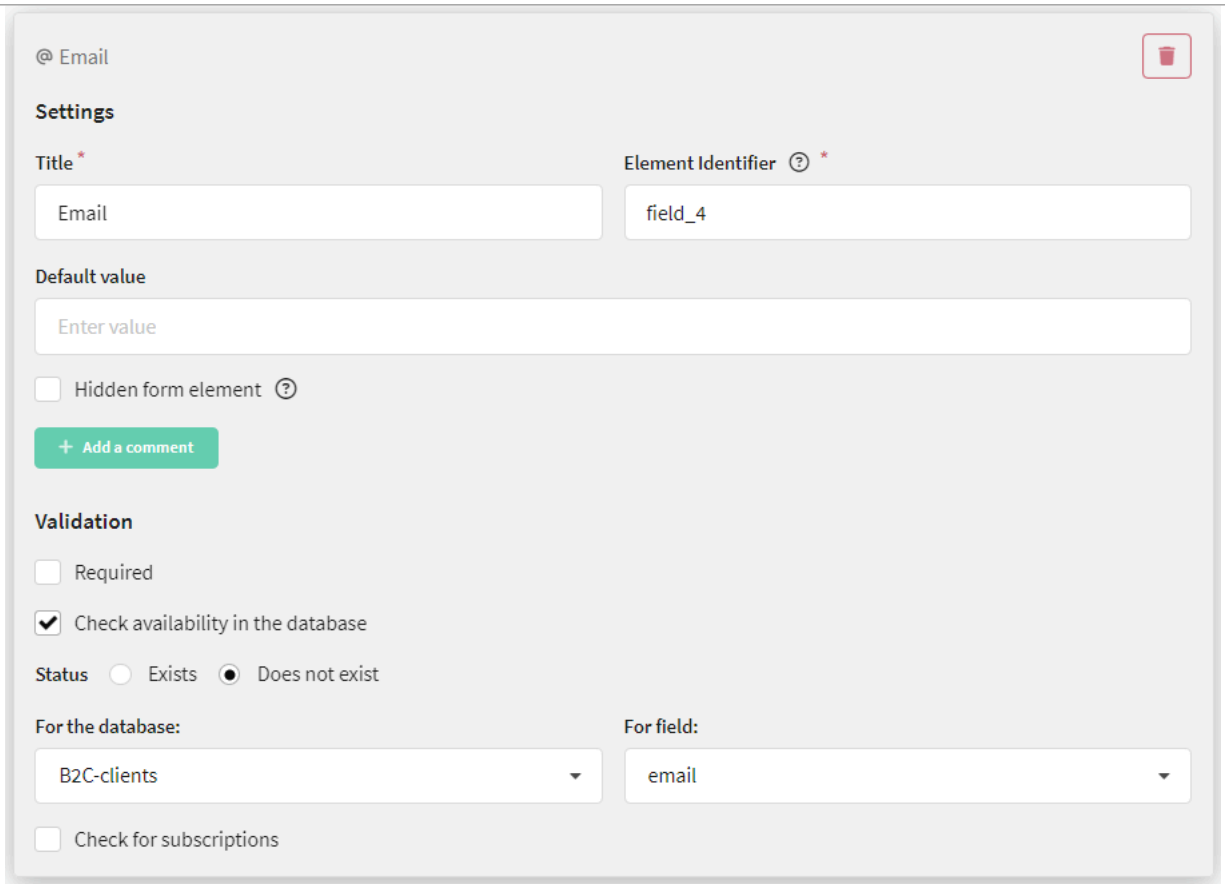
Short text

Short text is a text field where you can ask a question with a short answer. For example, the client's firs
For this field, you can set a title, a default value, and configure element visibility and data validation.



Email

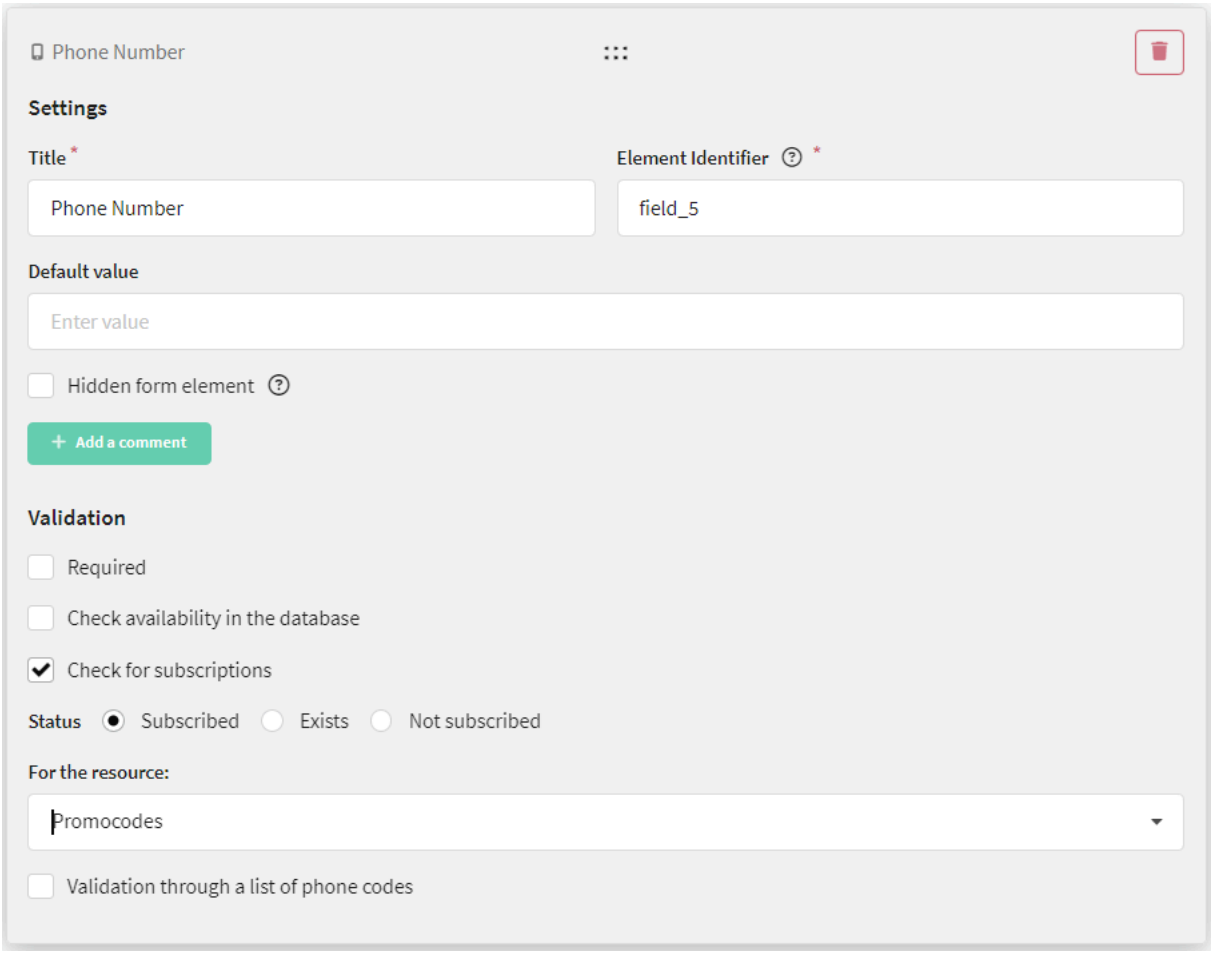
Email is a field where you need to specify an email address.
For this field, you can set a title, a default value, and configure element visibility and data validation.



Phone number

Phone number is a field where you can specify a phone number.

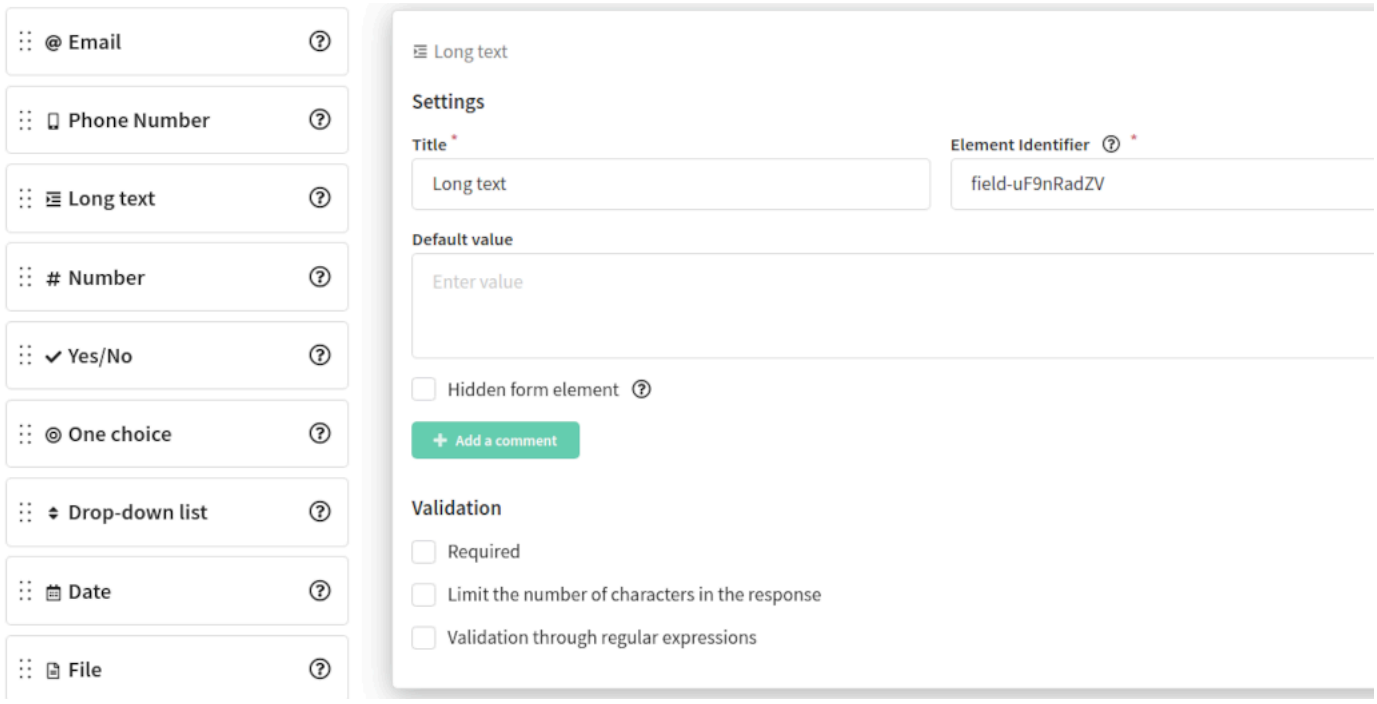
For this field, you can set a title, a default value, and configure element visibility and [data validation](#).



Long text

Long text is a text field where you can ask a question with a long answer, for example, feedback about a

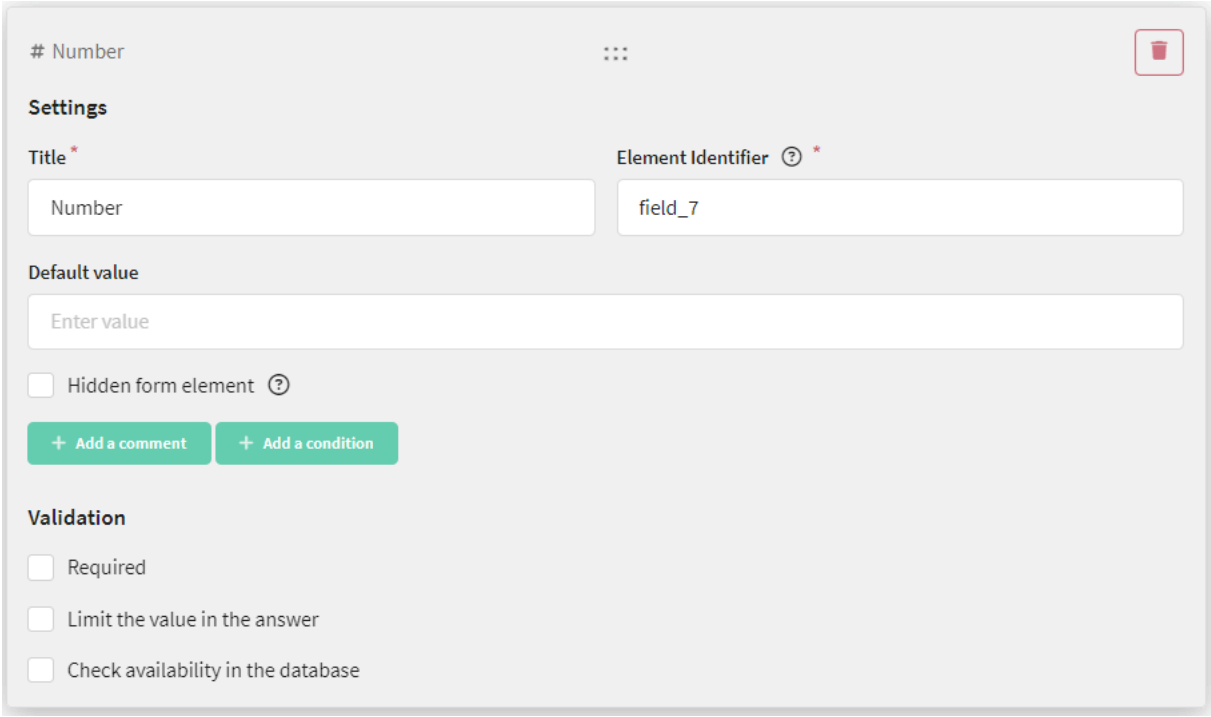
For this field, you can set a title, a default value, and configure element visibility and data validation.



Number

Number is a field where you can ask a question with an answer in the form of an integer or fractional nu

For this field, you can set a title, a default value, and configure element visibility and data validation.

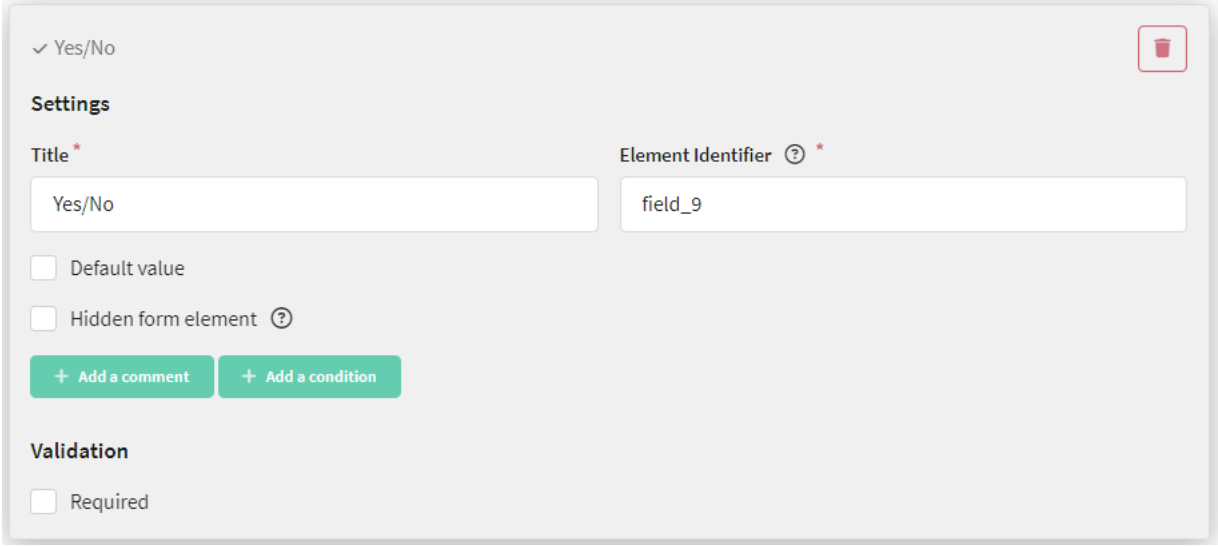


Yes/No

Yes/No is a field where you can ask a question to obtain consent, for example, to receive email newslett

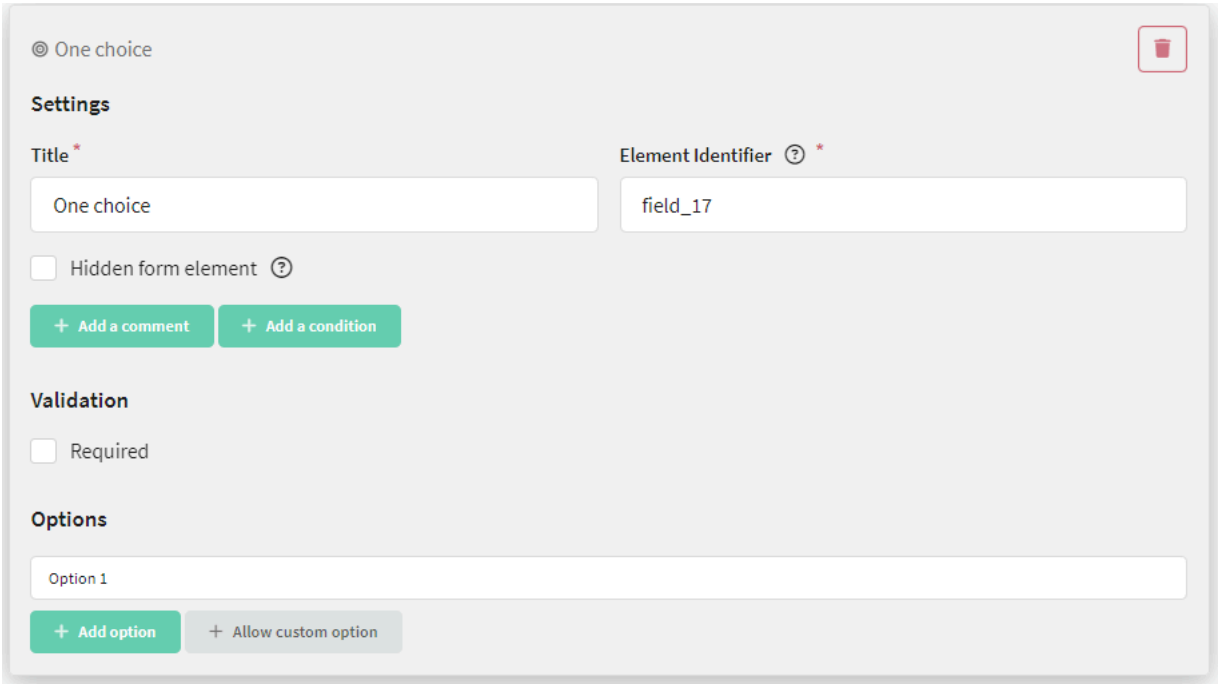
For this field, you can set a title, a default value, and configure element visibility.

If you check the Default value field, the answer “yes” will be selected by default in the form.



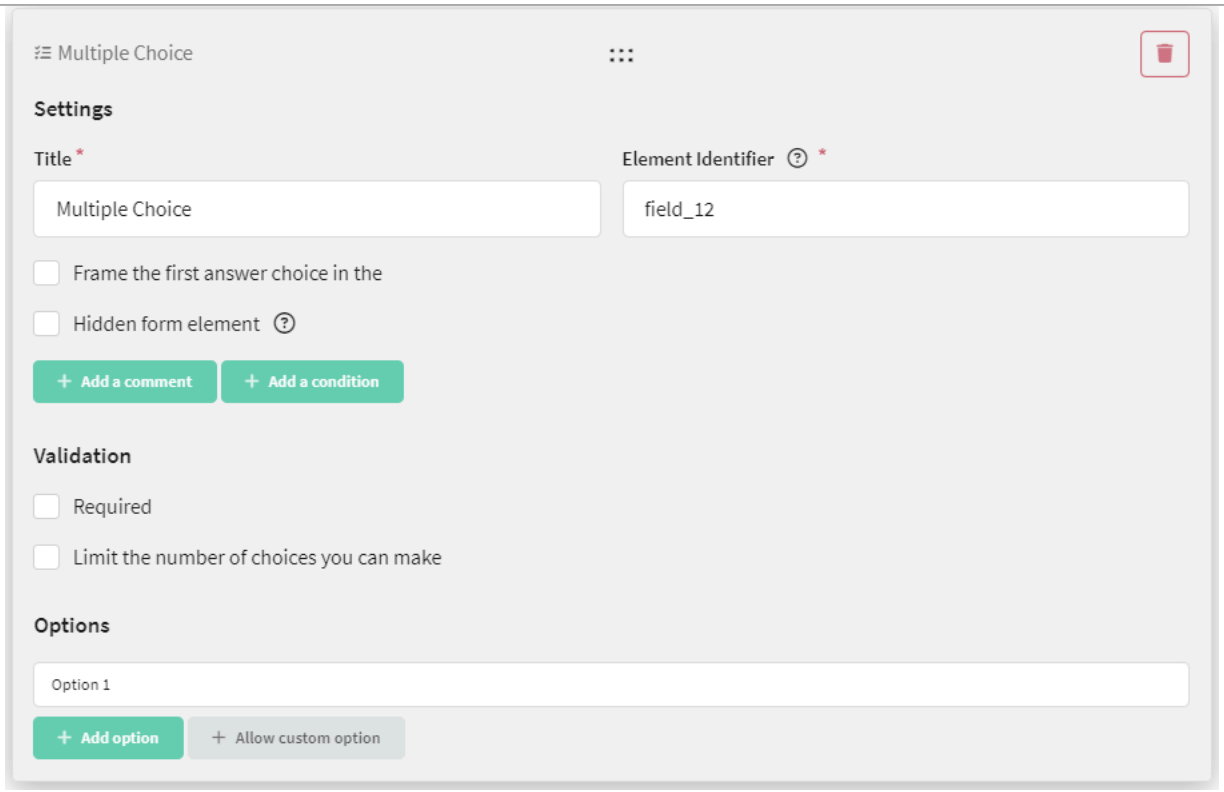
One choice

One choice is a field where you can offer to choose one of several answer options. For example, product
For this field, you can set a title, answer options, and configure element visibility and [data validation](#).
Click the Add option button and enter a value. The number of options is unlimited.
You can allow the user to enter their own answer by clicking the Allow custom option button.



Multiple choice

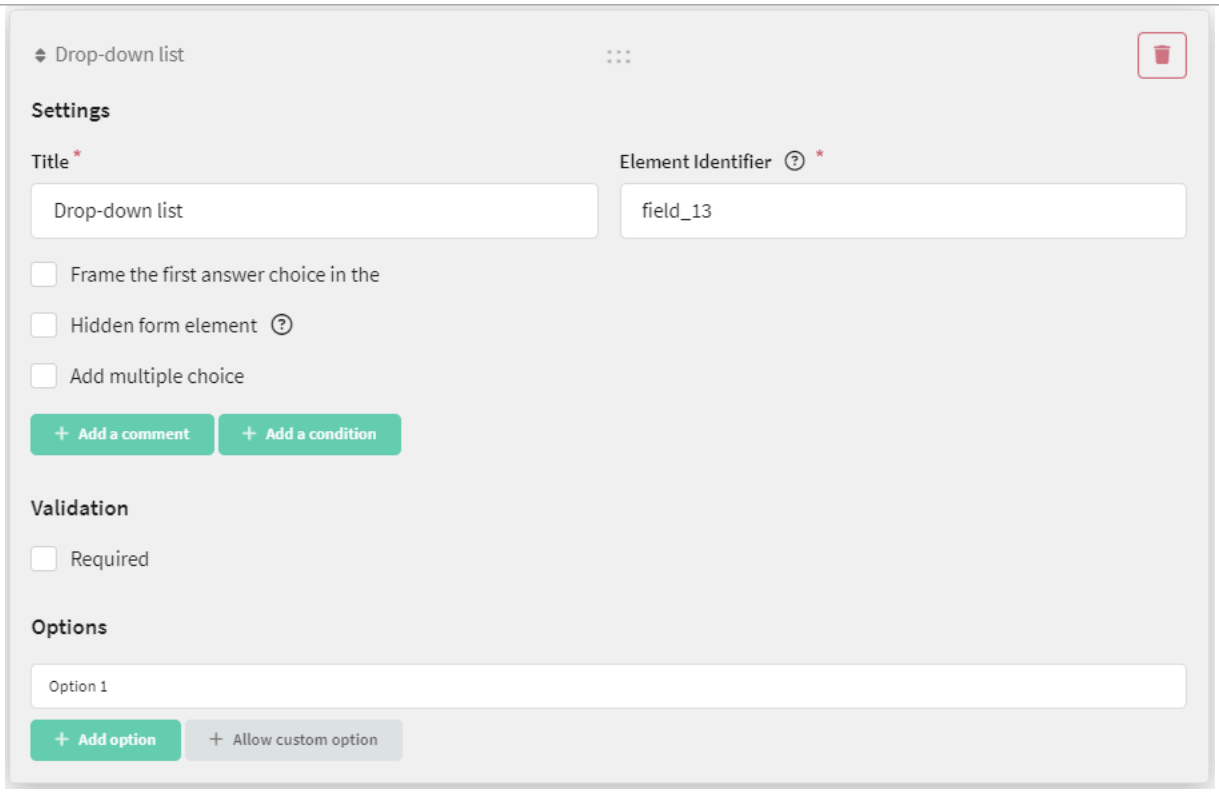
This element allows you to select several answer options from the proposed ones. Use the Add option bu



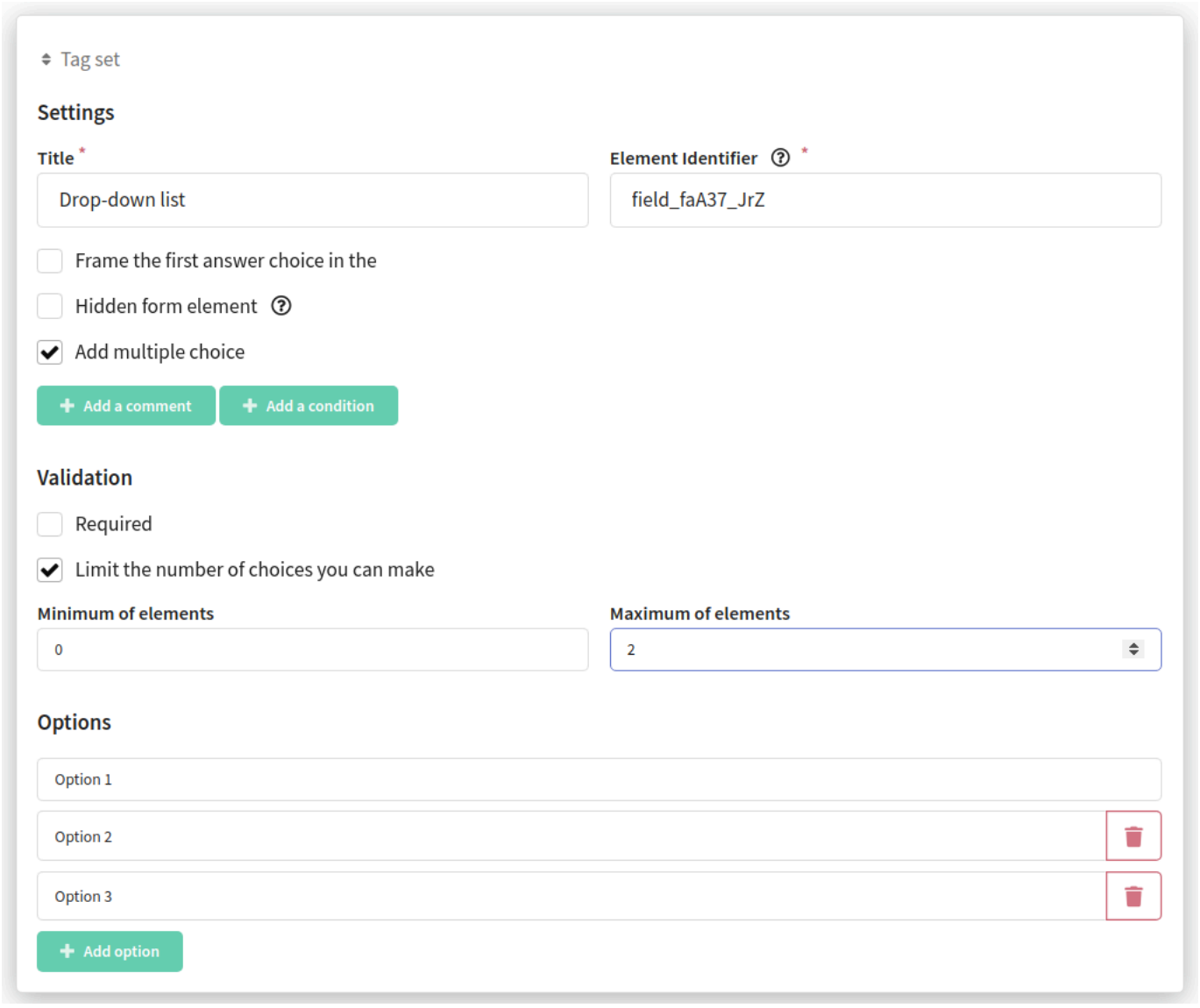
The element can be made required, the number of options to select can be limited, and you can also fran

Drop-down list

Drop-down list is a field where the user needs to select one option from a dropdown list.
For this field, you need to set a title and answer options. Click the Add option button and enter a value. T
You can allow the user to enter their own answer by clicking the Allow custom option button. When this will open for text input.
In addition, you can frame the first answer choice in the field by clicking the corresponding button.
You can also configure element visibility and data validation.



If you want to allow the user to select multiple answer options, click Add multiple choice.
Note that when multiple choices are enabled, a new validation type appears — Limit the number of choic

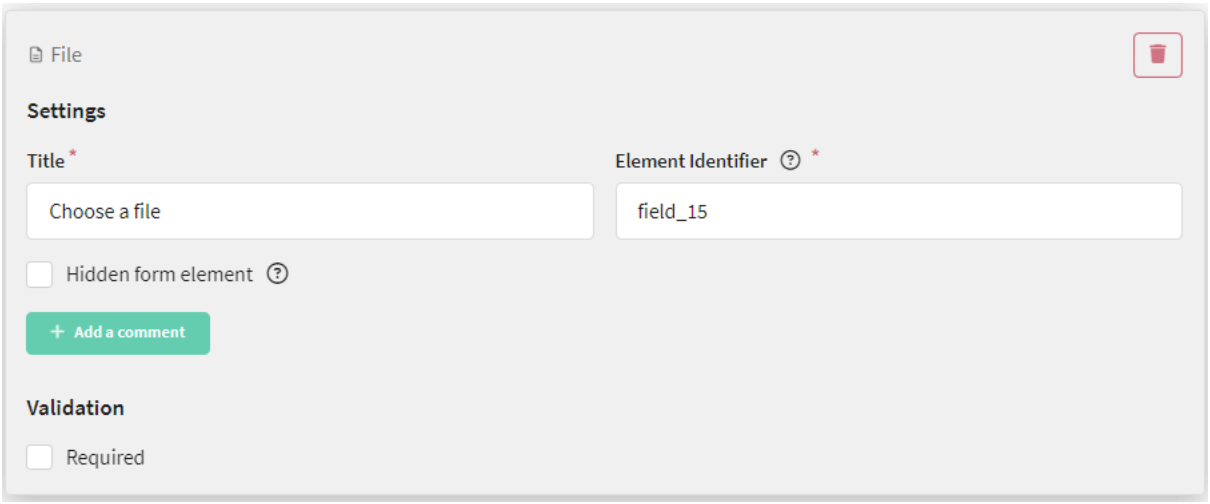


Date

Date is a field where you are asked to specify a date, for example, the client's birthday.
For this field, you can set a title, and configure element visibility and data validation.

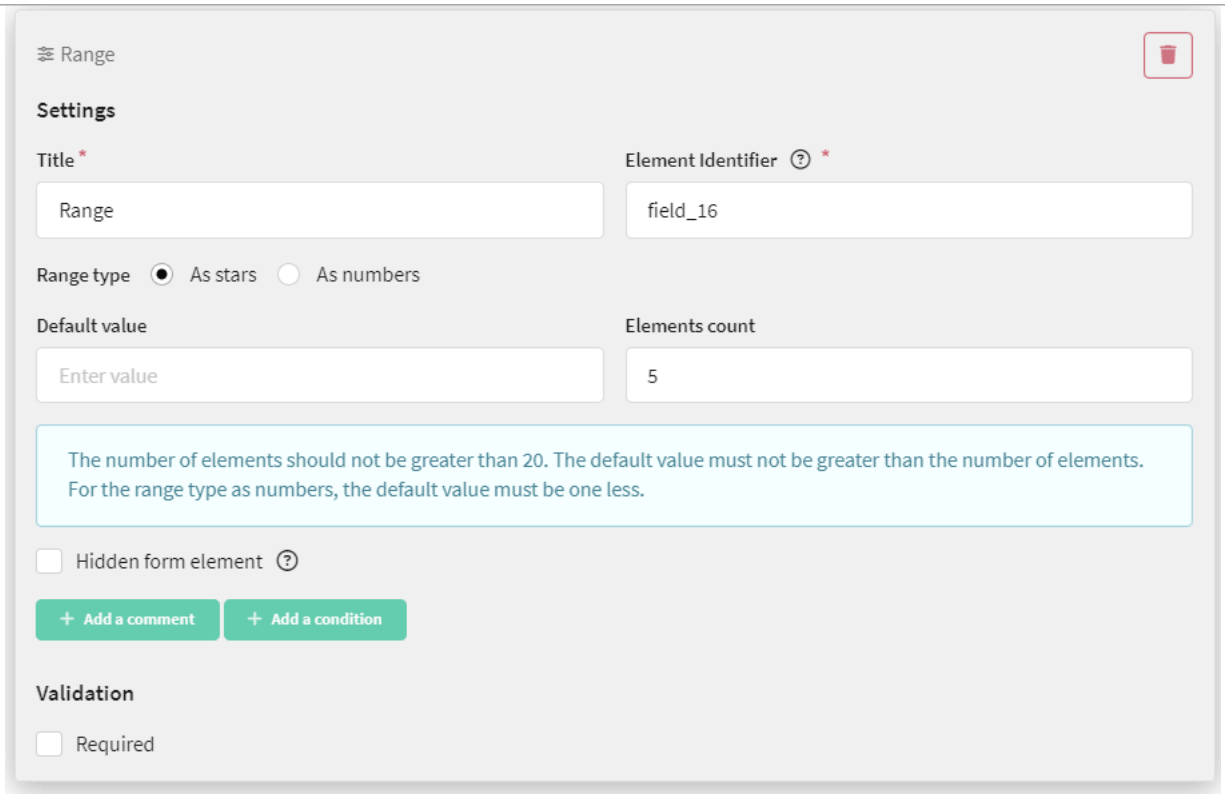
File

File is a field where files can be attached to the form, for example, a photo. The maximum file size is 20
For this field, you can set a title, and configure element visibility and data validation.



Range

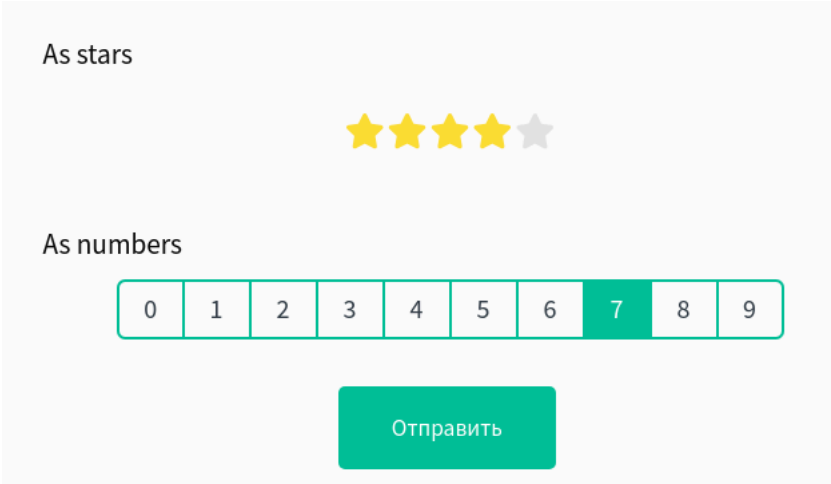
Range is a rating selection scale. For this field, you can set a title, and configure element visibility and d



Additionally, you can choose the range type: as stars or as numbers.
You can also specify the number of elements (for example, 5 stars or 10 numbers).

TIP

If you choose numbers as the range type, note that the count starts from 0.

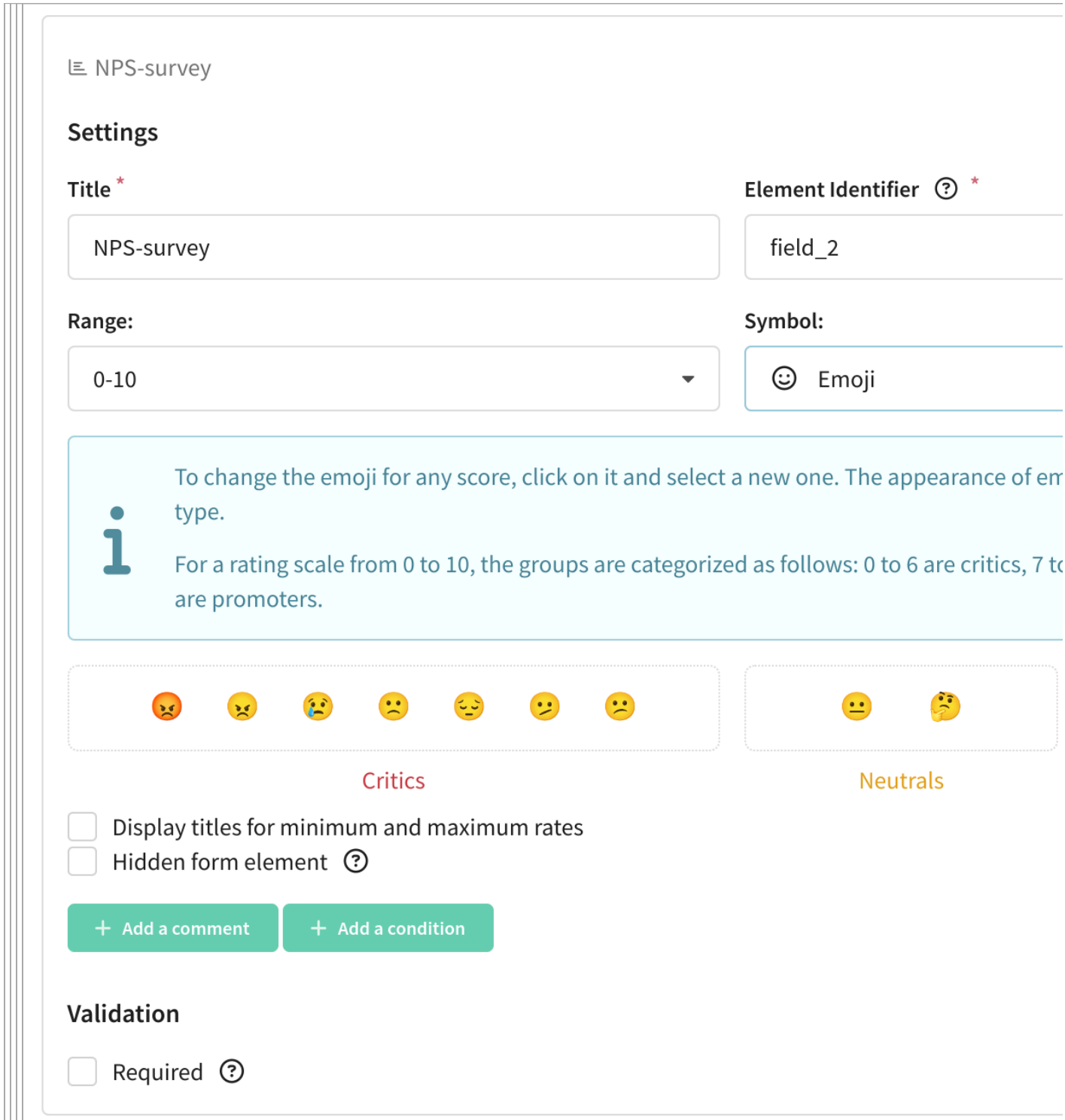


NPS Survey

NPS Survey is a test to obtain the Net Promoter Score. For this field, you can set a title, configure eleme well as set labels for the maximum and minimum scores.

TIP

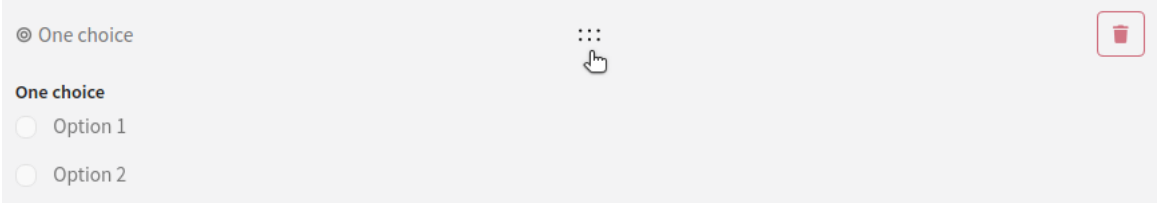
You can learn more about what NPS testing is and how to create it [in this article](#).



You can select the symbol type: as stars, hearts, emoji, or as numbers.
For the scale, you can also specify the size — 0 to 5, 0 to 10, 1 to 5, and 1 to 10.

TIP

You can drag and drop form elements, easily arranging them in the order you want. When you hover ove appears at the top. Hover over them, hold down the left mouse button, and drag the element wherever y



Form field validation

Most fields have one or more validation types:

Validation	Form fiel
Required	**Short text, email, phone number, long text, number, one choice, drop**
Limit the number of characters in the response	**Short text, long text**
Validation through regular expressions	**Short text, long text**
Check the availability in the database	**Short text, email, phone number, number**
Check for subscriptions	**Email, phone number**
Validation through a list of telephone codes **(list the telephone codes allowed in the form using the + symbol)**	**Phone number**
Limit the value in the answer	**Number**
Limit the number of choices you can make **(specify the minimum and maximum number of options the user can select)**	**Drop-down list**

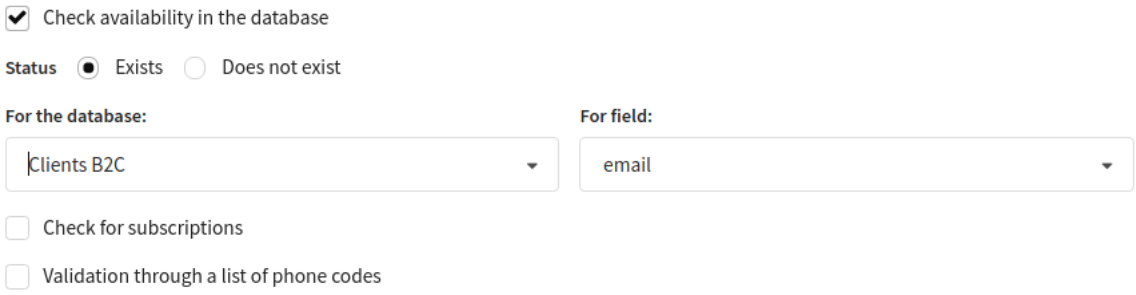
The validation types Check the availability in the database and Check for subscriptions allow you to rest users. For example, you can allow only registered users to submit, or prevent the same user from resubn

Check the availability in the database

At the moment of form submission, the platform will check if the profile of the person who filled out the profile with the specified data does not exist, the user will not be able to submit the form.

When configuring the form, you need to select:

- **the status you want to check (exists or does not exist),**
- **the database in which to look for the profile,**
- **the field in the database to search for an existing profile.**

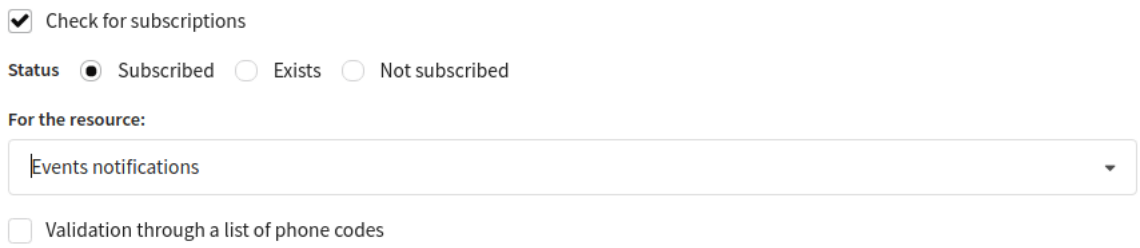


Check for subscriptions

At the moment of form submission, the platform will check if the profile of the person who filled out the resource. If a profile with the specified subscription does not exist, the user will not be able to submit th

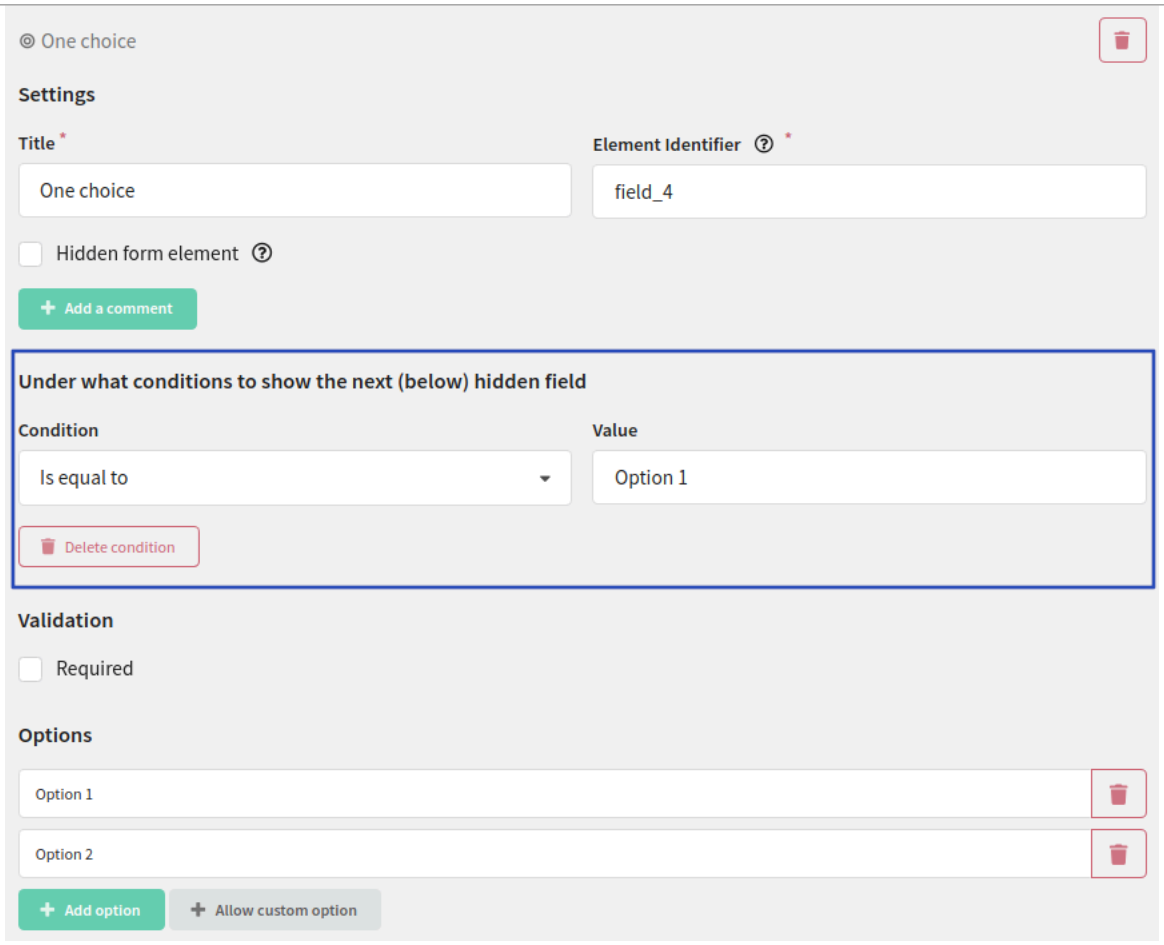
When configuring the form, you need to select:

- **the status you want to check (subscribed, exists, or not subscribed),**
- **the resource to check subscription for.**

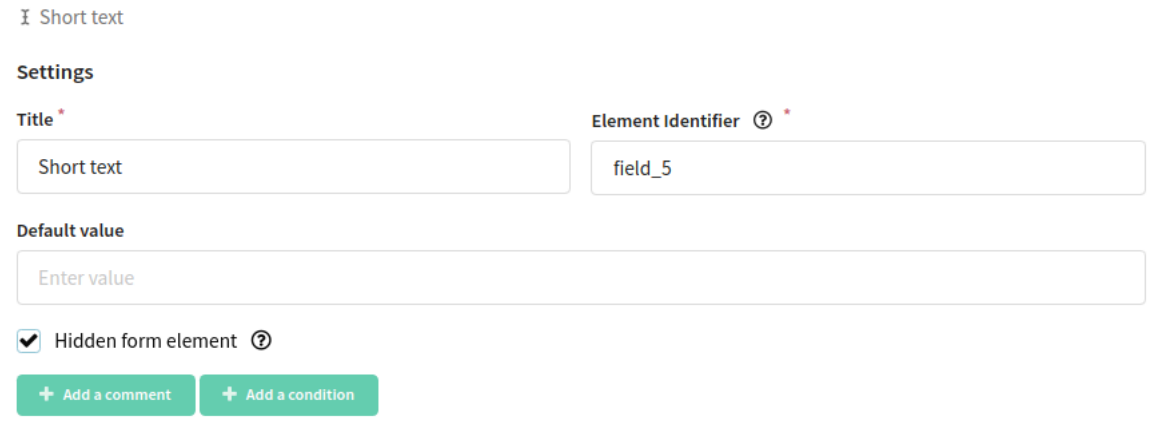


Form element display condition

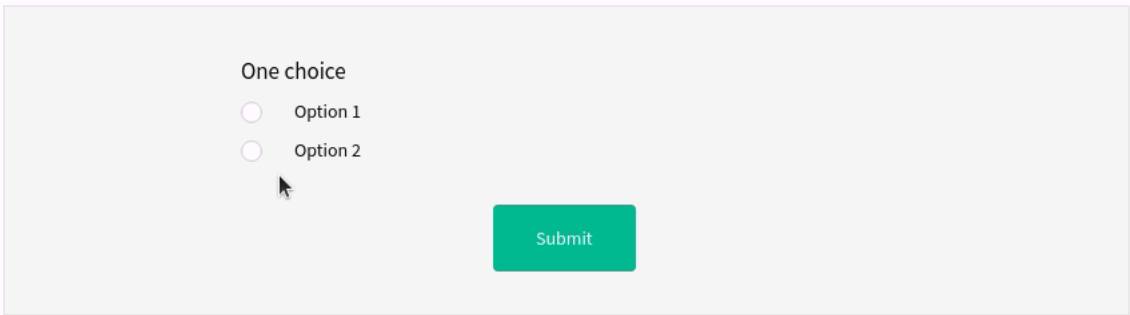
You can show a form field to the user only if they gave a specific answer to the previous question. To do click the Add condition button — additional fields will open (Under what conditions show the next hidden Specify the condition and the value the user should select.



Next, add the element you want to show if the user selects a certain value in the question above. Activa



Now if the user selected a specific answer option, the hidden element will be shown to them. Otherwise,

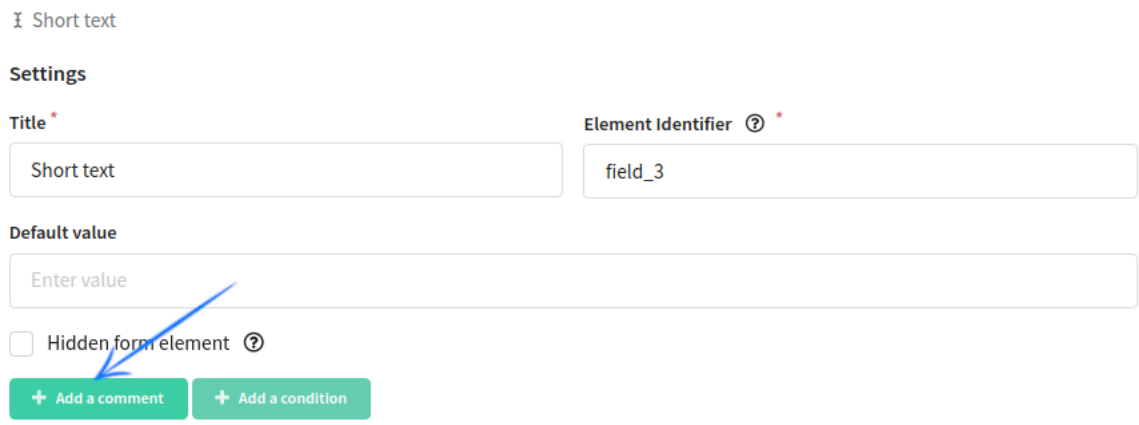


TIP

To add more complex conditions, such as transitioning between form pages depending on answers, con

Comment

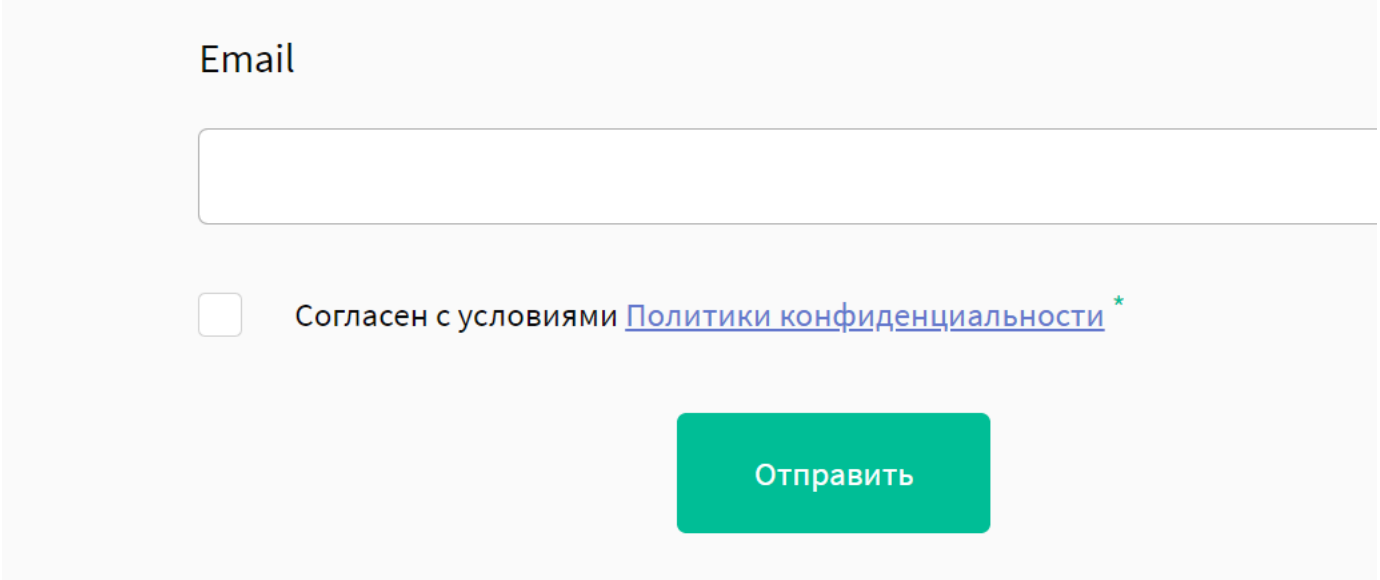
For each element, you can add a comment that will be displayed below the form field.



Links in form fields

You can add a link to any form element using HTML:

```
I agree to the <a target="_blank" href="#">Privacy Policy</a> terms
```



Form example

****Page 1 ****

Around the World in Eighty Days

Best routes and travel ideas

What is your name?

Email *

Your phone number

+7(900)000-00-00

Would you like to receive our newsletter?

You'd like to receive newsletters:

- Once a week
- Once in two weeks
- Once a month
- Custom option

Next

You are interested in routes to:

South America Drop-down list

How many times have you used the services of our company?

Number

Are you satisfied with the service of our company?

Range

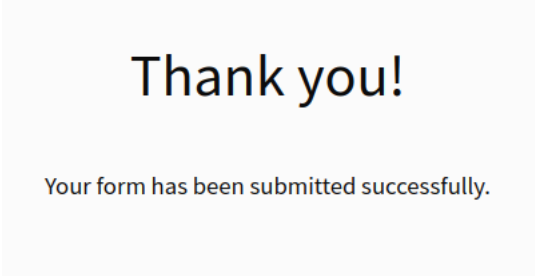
Submit

Page 2

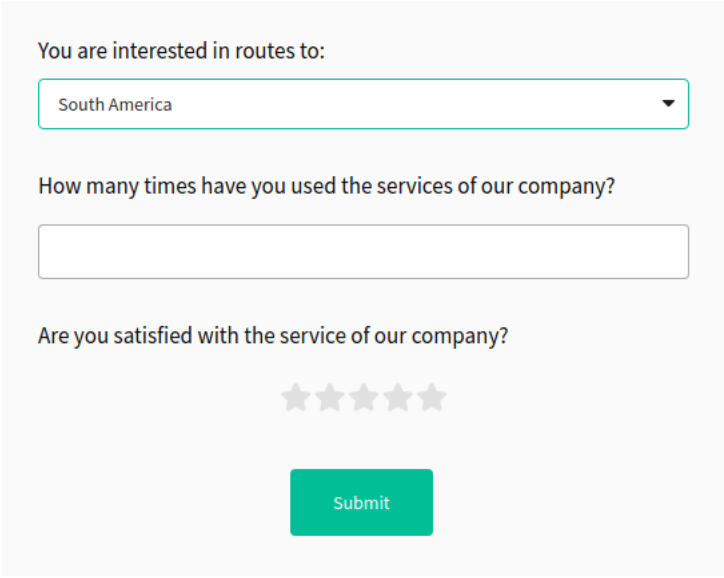
Расскажите о вашем прошлом путешествии, которое мы помогли вам организовать

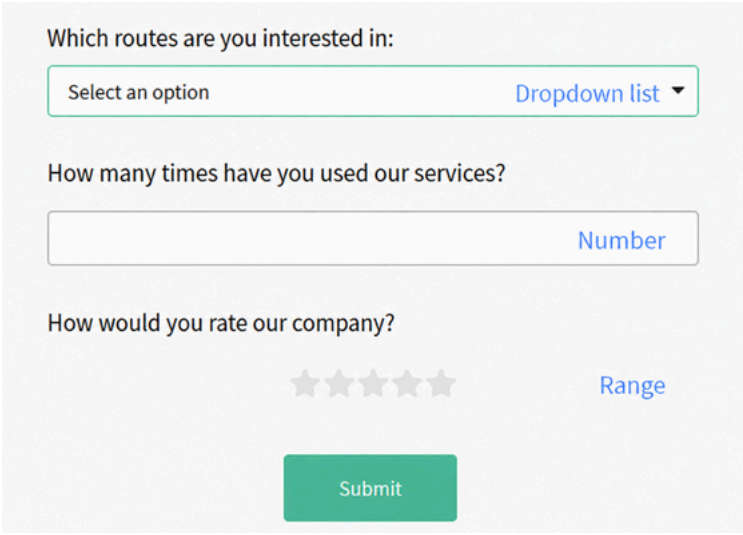
Прикрепите фото вашего прошлого путешествия

Следующая страница

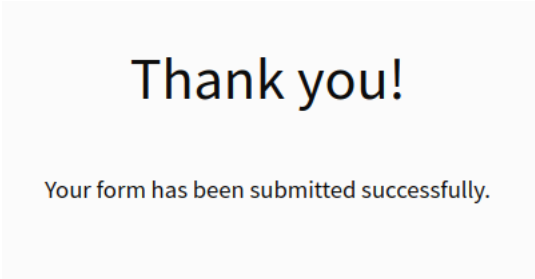


Page 3





Final page



Appearance

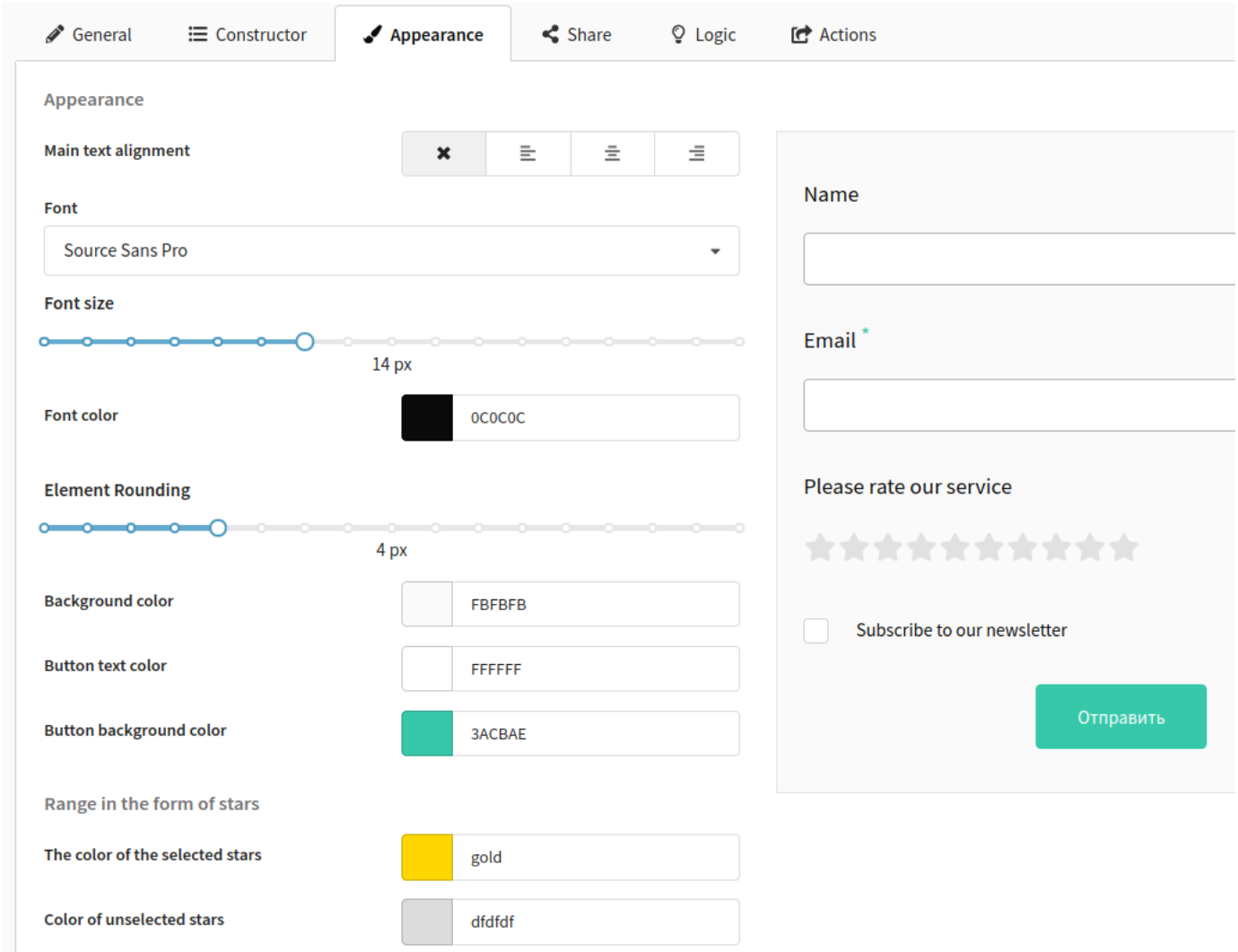
On the Appearance tab, you can customize the form's look and feel: fonts, colors, element alignment, an the form design to match your website or brand style.

Main appearance settings

Configure the visual appearance of the form:

- **Fonts — font family and size**
- **Color scheme — text, background, and button colors**
- **Alignment — element layout (left/right alignment, center)**

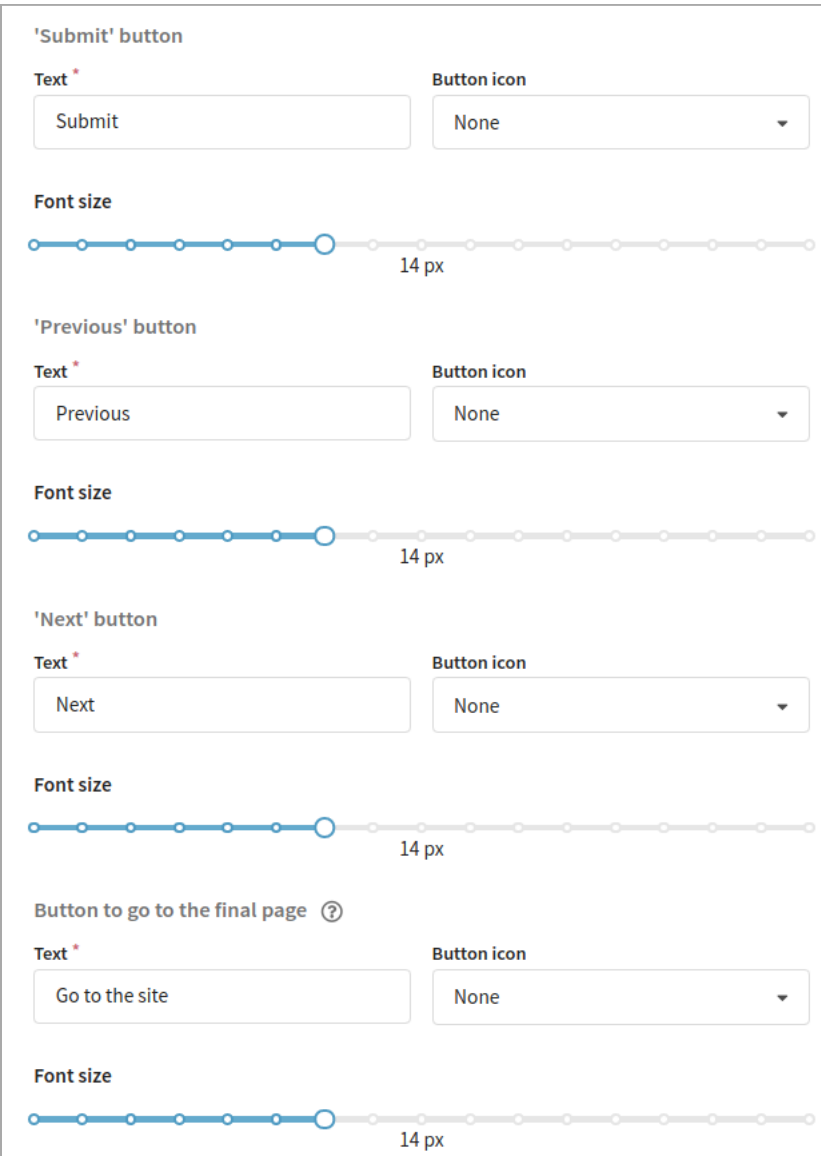
The left side displays a real-time preview of the form.



Button styling

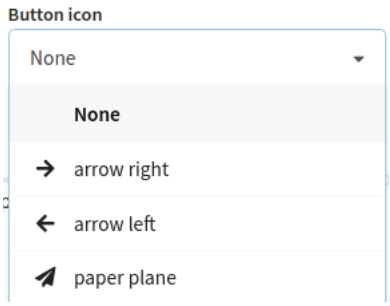
The following buttons are available for styling:

- **Submit — to submit the form**
- **Previous — to go to the previous form page**
- **Forward — to go to the next form page**
- **Button to go to the site — on the final page**



For each button, you can specify:

- **the text displayed on the button**
- **font size (in pixels)**
- **an icon**



TIP: Button to go to the final page

If you leave the button text field empty or do not specify a link to go to after filling out the form on the C settings, the button will not be displayed.

Saving the form

After configuring the field structure and appearance, you need to save the form. Only after this will the a and the ability to publish it become available.

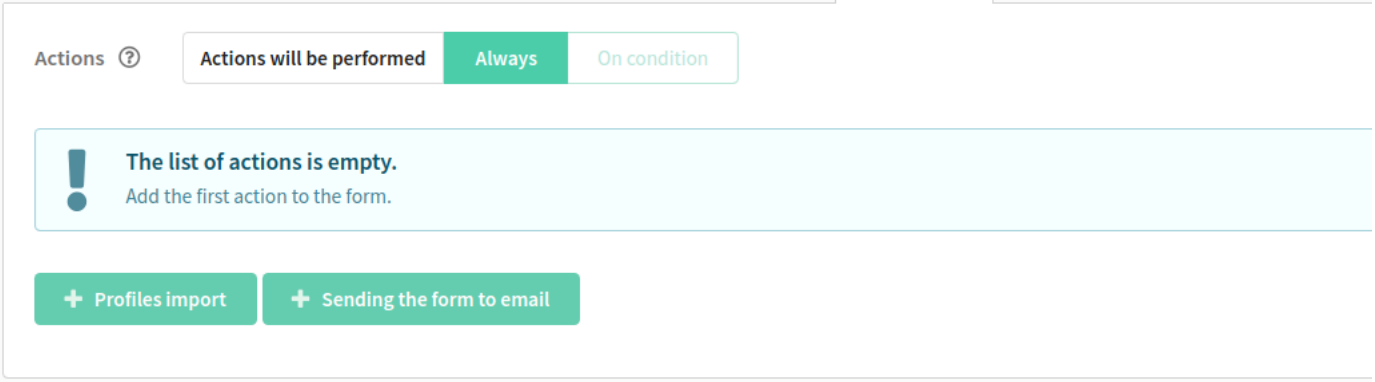
Actions and form publication

On the Actions tab, you can configure automatic processes that run after successful form submission. H into the database and send notifications about form completions.

> **TIP**
>
> **Before you start working with actions, save the form.**

On the Actions tab, specify the actions to be performed upon successful form submission:

- **Profiles import allows you to upload the data of the user who filled out the form into the database;**
- **Sending the form to email enables sending notifications about form completions to specified email a**



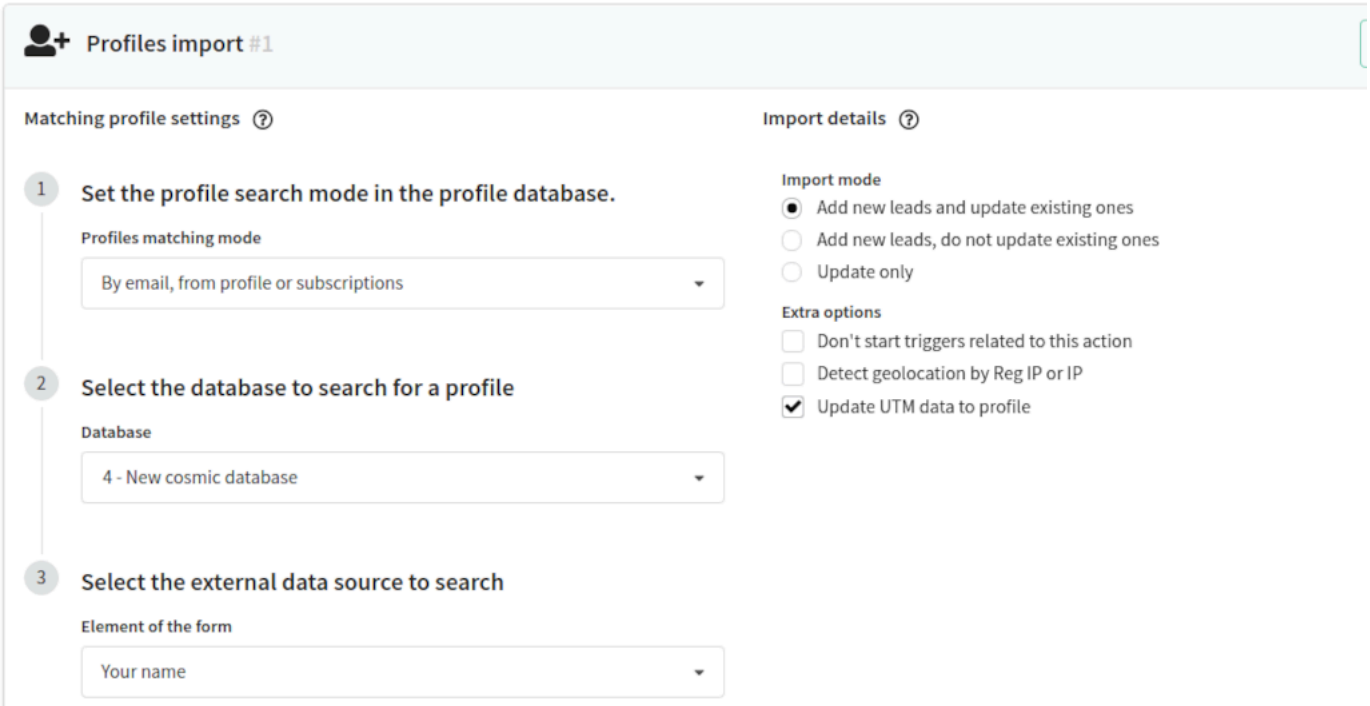
Profiles import

Setting up profile search

Specify the profile search mode in the database:

- **by email address from profile or subscriptions — by the profile field `Email` or by email addresses fro**
- **by email address from profile — only by the profile field `Email`**
- **by Message ID — by the message template identifier**
- **by additional field — by a custom database field**

Next, select the database where the profile will be imported, and the form element from which to take d



In the right part of the window, you can specify the import method:

- **Add new leads and update existing ones;**
- **Add new leads only, do not update existing ones;**
- **Update only, do not add new ones.**

You can also add additional rules:

- **Don't start triggers related to this action: disable trigger notifications for the new profile after impor**

- **Detect geolocation by Reg IP or IP: enable automatic detection of the client's geolocation based on t address) or IP (last known IP address) fields;**
- **Update UTM data to profile: if the form contains UTM tags, the information about these tags is auton structure after import.**

Column mapping

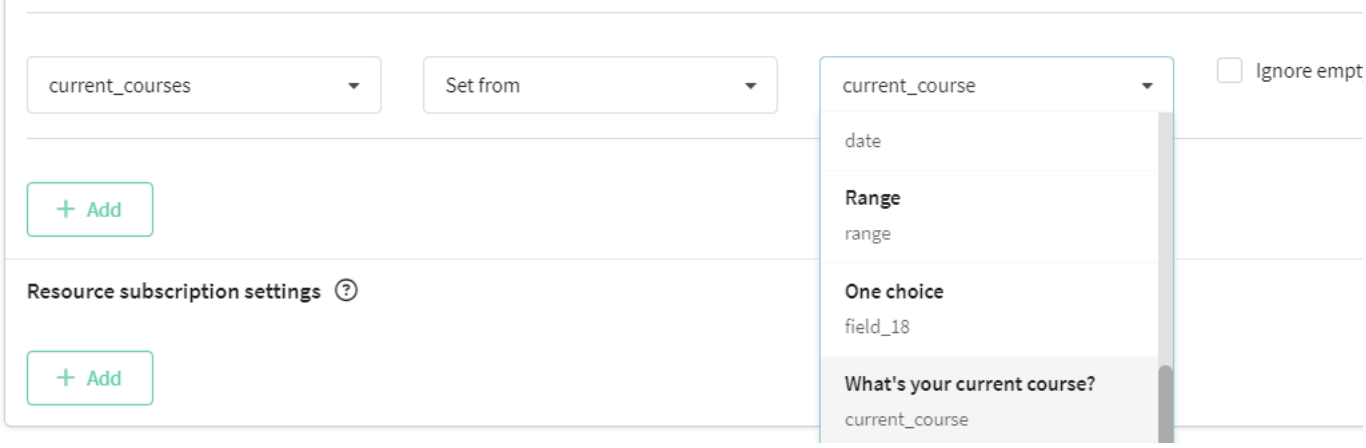
You can set the matching between the form elements and the fields in the profile card.

In the Field in the database drop-down menu, fields from the database are available: system and custom. with the main fields of the form. For example, the Email field in the database corresponds to the field of the Email.

In the Action menu, you can choose three options: set from, set custom, or remove field.

Source column contains fields that exist in the form. You can name them yourself for easier matching. W as the fields, system and custom, are called in the database.

Value handling contains the Ignore empty action. If this field in the form is not filled in, then its value do database that is set by default.

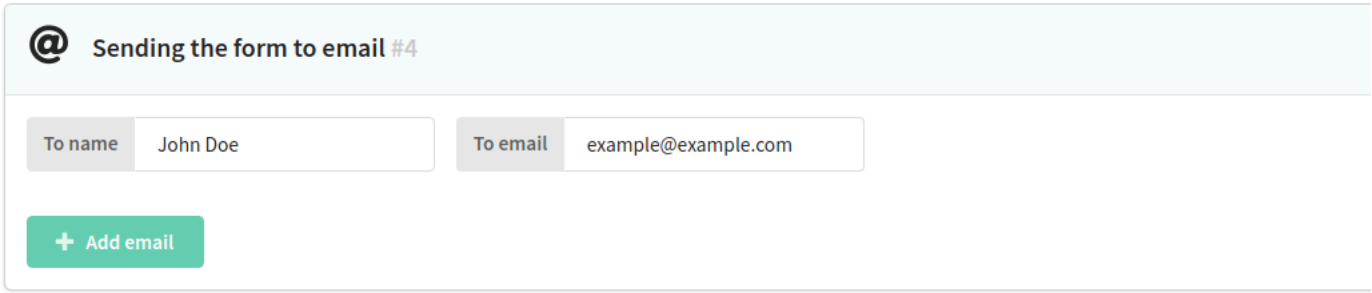


To avoid errors when transferring responses from the form to the profile card, you need to correctly spe database and match them with the form fields.

Form element	Field's type in a database
Short text	**Line**
Email	**Line**
Phone number	**Phones list / Line**
Long text	**Line**
Number	**Integer**
Yes/No	**Boolean**
One choice	**Line**
Multiple choice	**Tags**
Drop-down list	**Tags**
Date	**Date (RFC 3339 (ISO 8601))**
Range	**Integer**

Resource subscription settings

When importing profiles from forms, you can immediately add a subscription to one of your resources fo subscribe to, the channel and subscription status, as well as the field from the form to be used for subsc

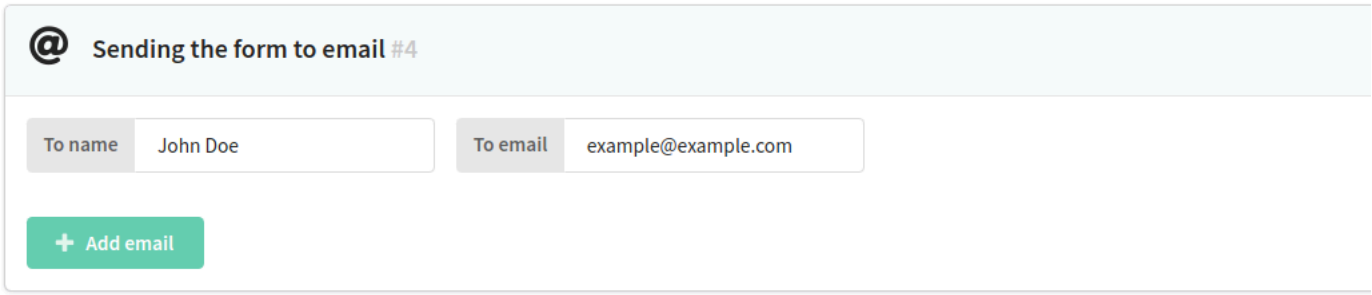


TIP

You can subscribe client profiles to multiple resources at once. To add another subscription, click the +

Sending the form to email

Add this action if you want to receive notifications about form completions. Enter your name and email a recipients.



Each time a user fills out the form, an email will be sent to the specified address with the following data:

- **form completion notification**
- **form name and identifier**
- **link to the form (*Go to view collected data*)**

— test action (send a test message to the specified email address to make sure that the entered data is

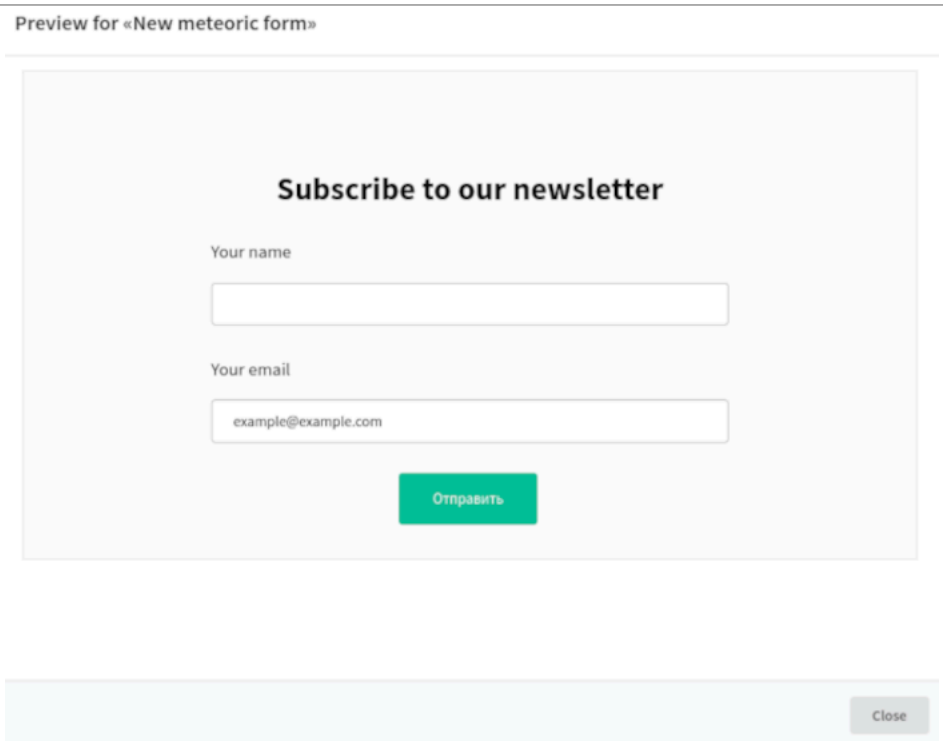
— start action (a message will be sent to the mailbox with information about how many times the form w containing data on fillings)

— remove action

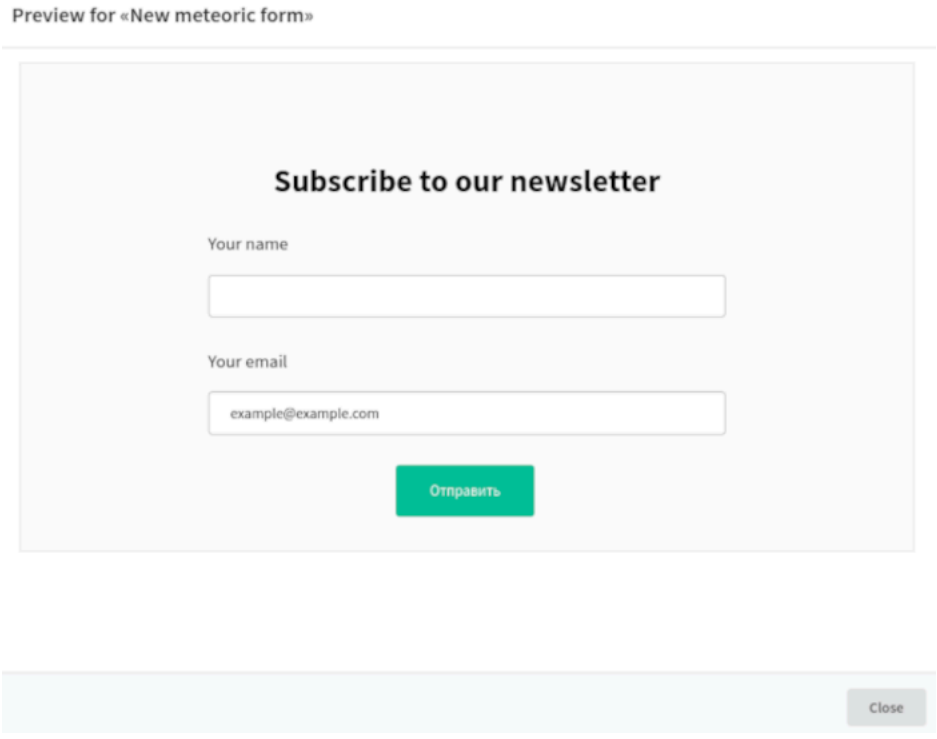
> TIP
> **The Test action and Start action buttons work only if the form is published.**

Form preview

At any time while creating and editing the form, a preview is available. To see how the published form wi the upper right corner of the page



, and the form will open in the way the user will see it.



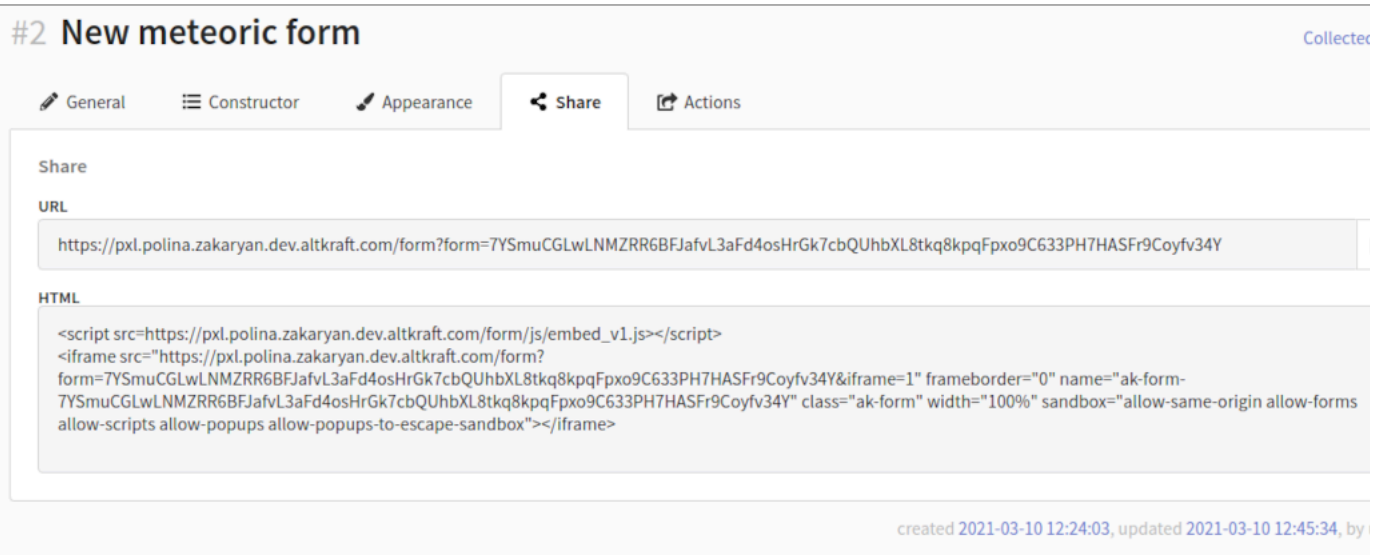
Share

When the form is ready, you can publish it. To do this, activate the Published switch on the General tab.

Once the form is published and saved, it can be used in two ways:

- **URL: The platform generates a URL that leads to a separate page with the form. You can send this li communication channel by adding the form variable to the message template. Learn more.**
- **HTML: The platform generates HTML code that needs to be added to your website page. The form wi**

The URL and HTML code for the form are available on the Share tab:



TIP

After filling out and submitting the form, the user can be redirected to the system's final page or to any settings. Please note that when embedding the form code on the website, this page will not open inside page, but the user will be redirected to a new page.

Conditional logic in forms and surveys

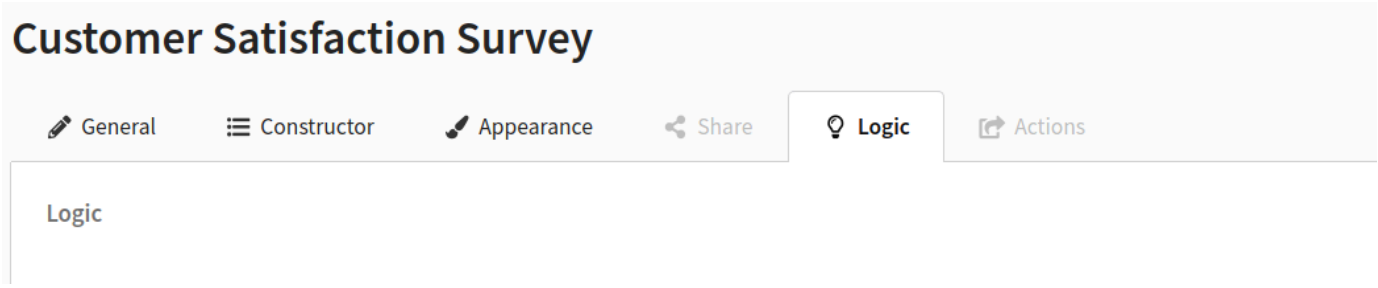
Description

Conditional paging logic allows you to customize the logic of transitions between form pages. Depending you can change the order of displaying pages, as well as set the conditions for displaying a particular pa

How to set up form logic

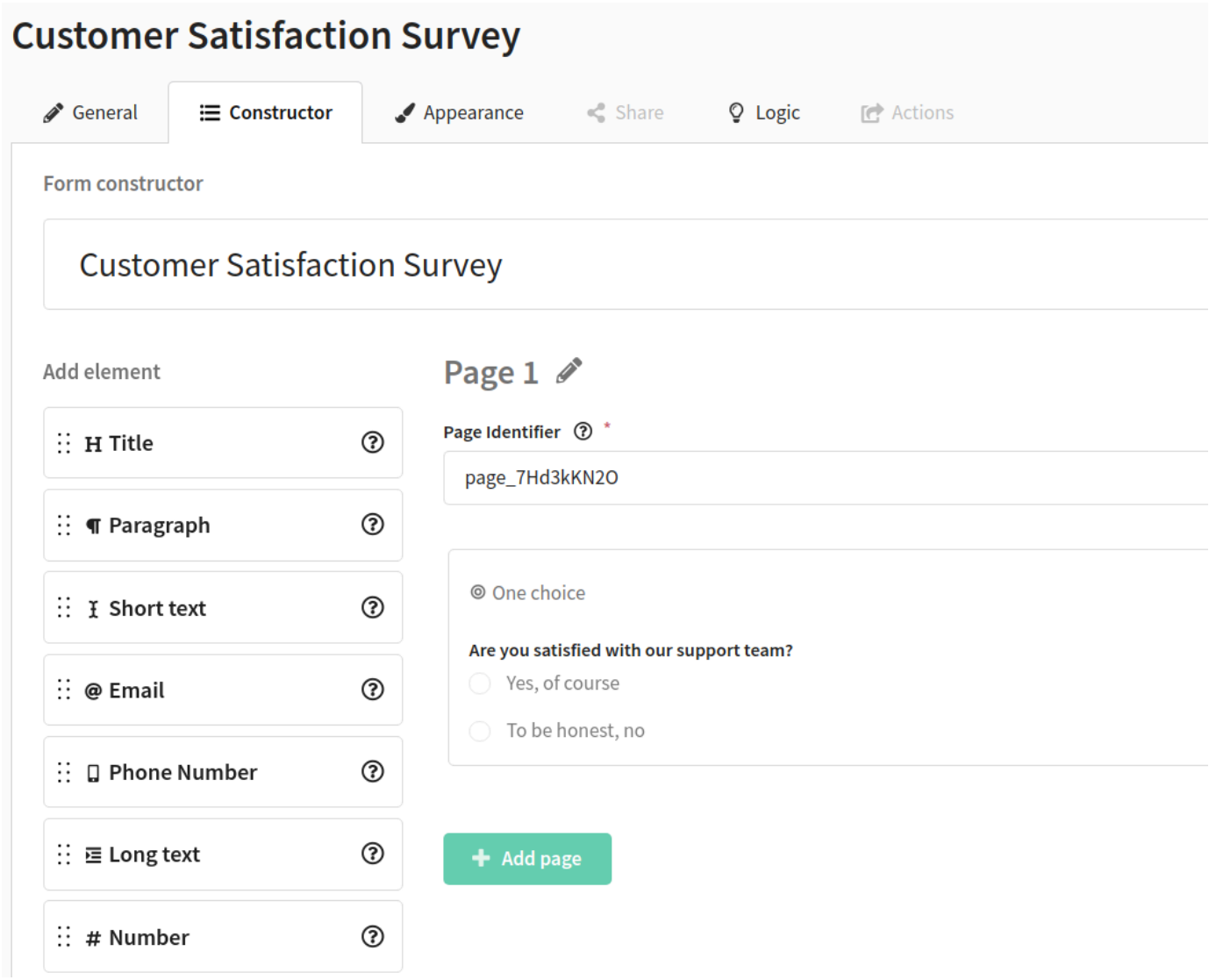
Activate logic

Create a form and fill in the basic settings. After that, go to the Logic tab and activate the option by clicl corner.



Create form structure

Go back to the Constructor tab. Create form pages and add the required elements:

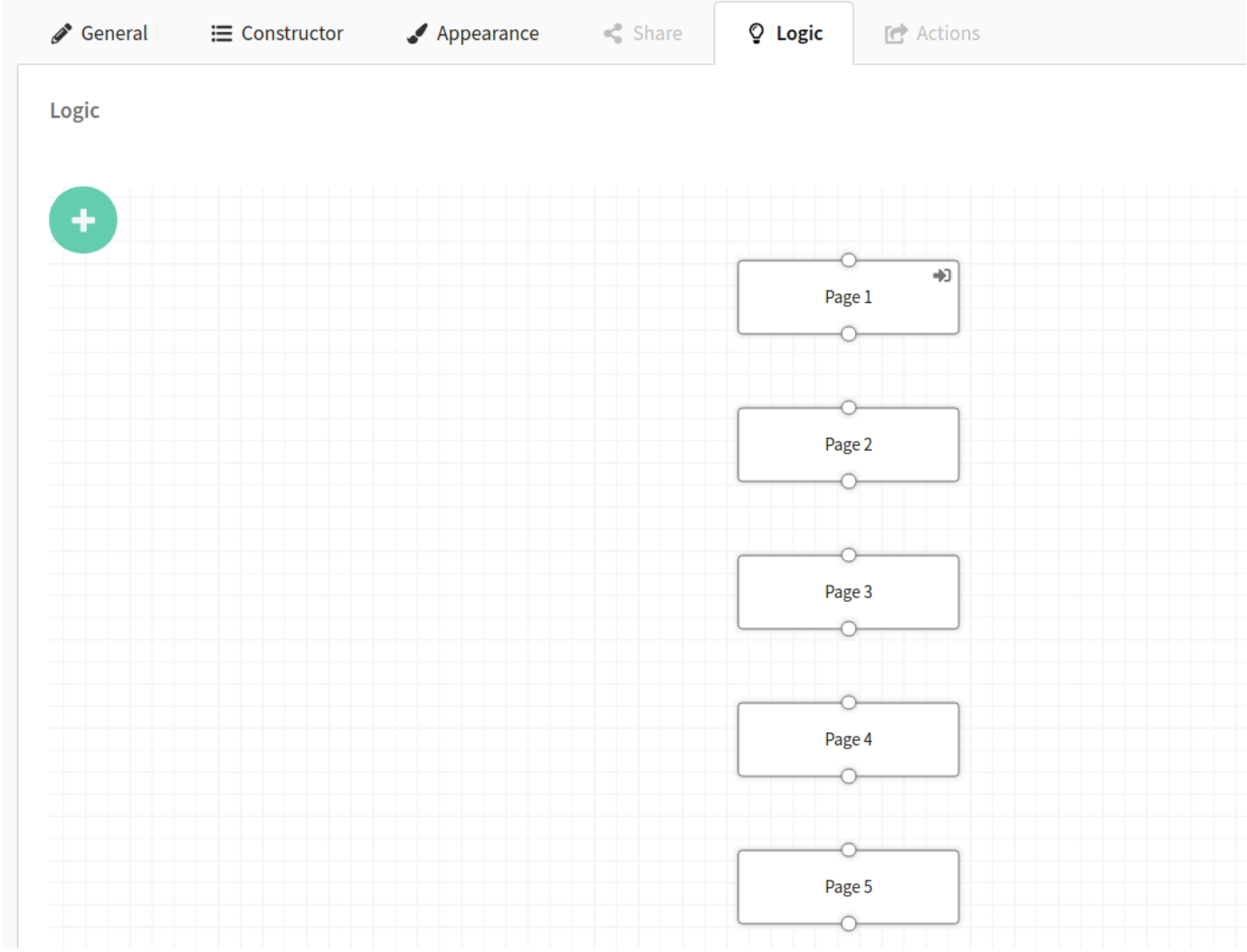


INFO

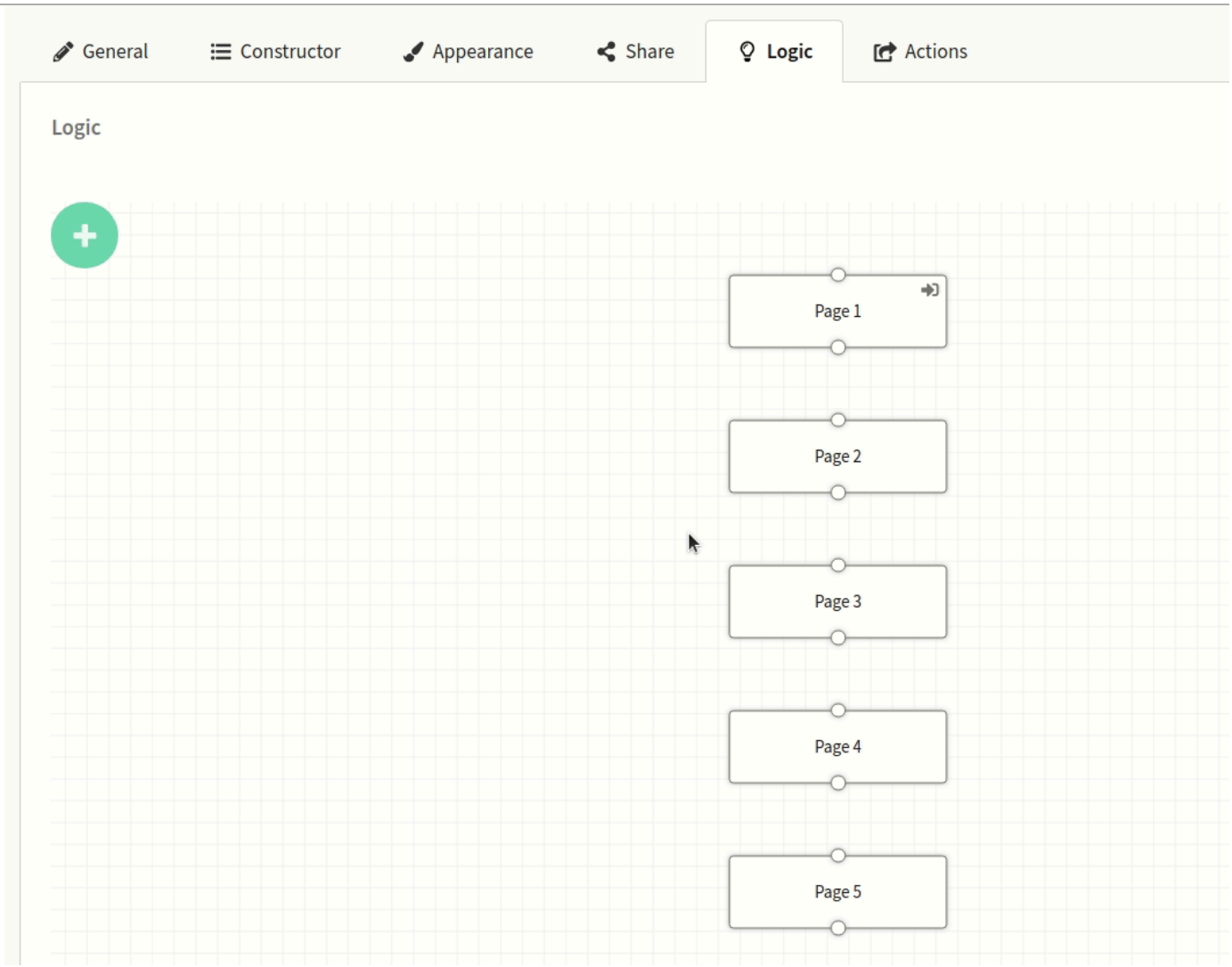
Please note that you are configuring the logic for transitions between pages. In order for the user to go answer to a question, it is necessary to create separate pages for each question, after which the transi

Working with logic

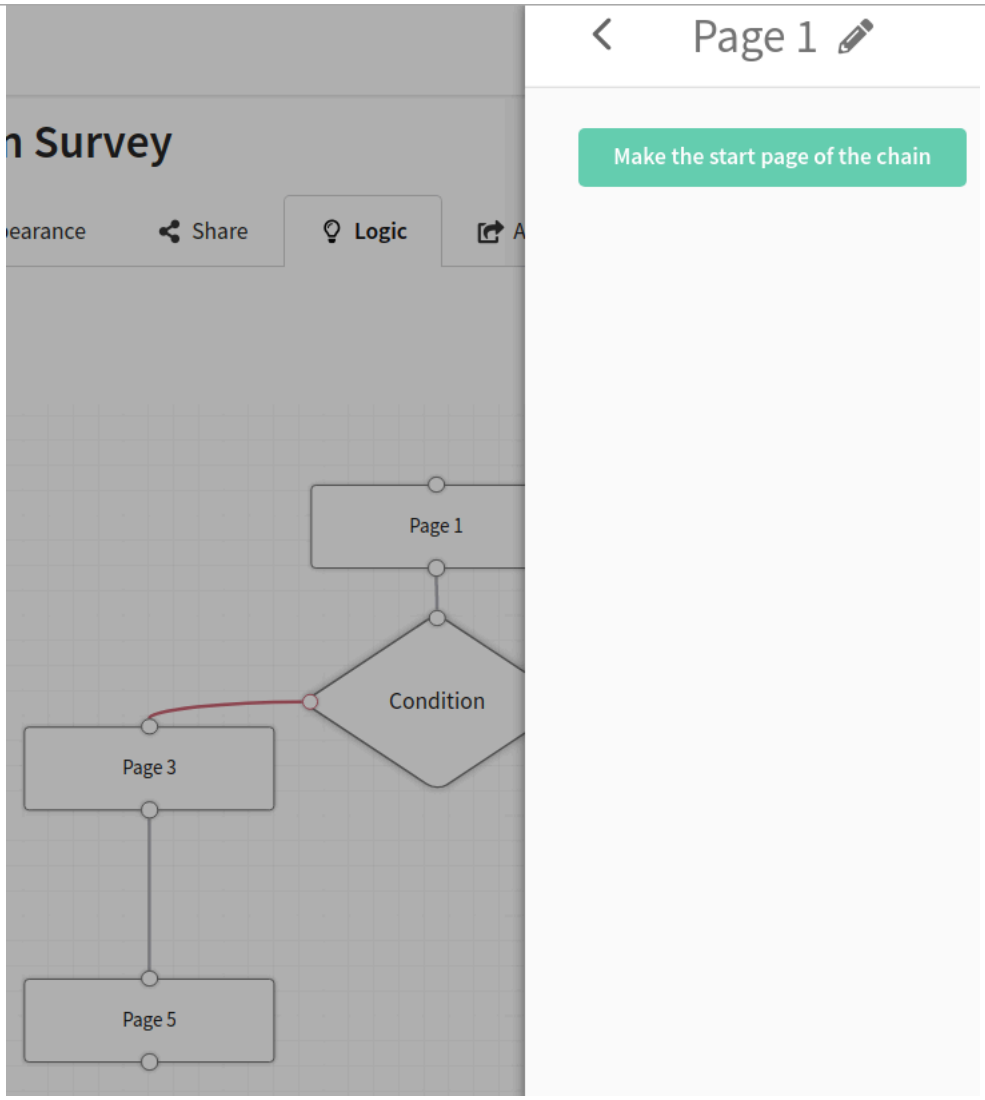
Go to the Logic tab. All created pages will be displayed in a diagram:



To change the order of pages, drag the blocks with your mouse:



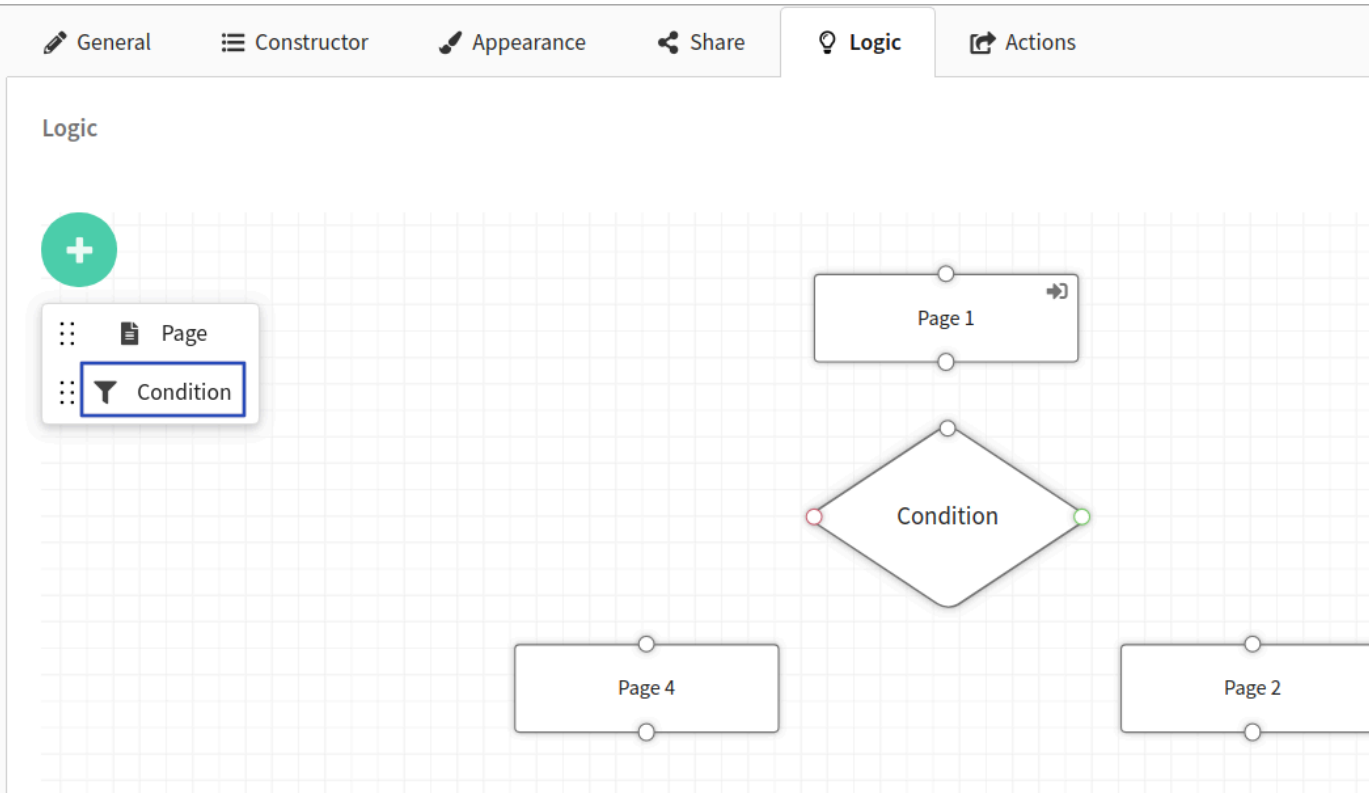
Select the start page by clicking on the page block and then clicking Make the start page of the chain. T form.



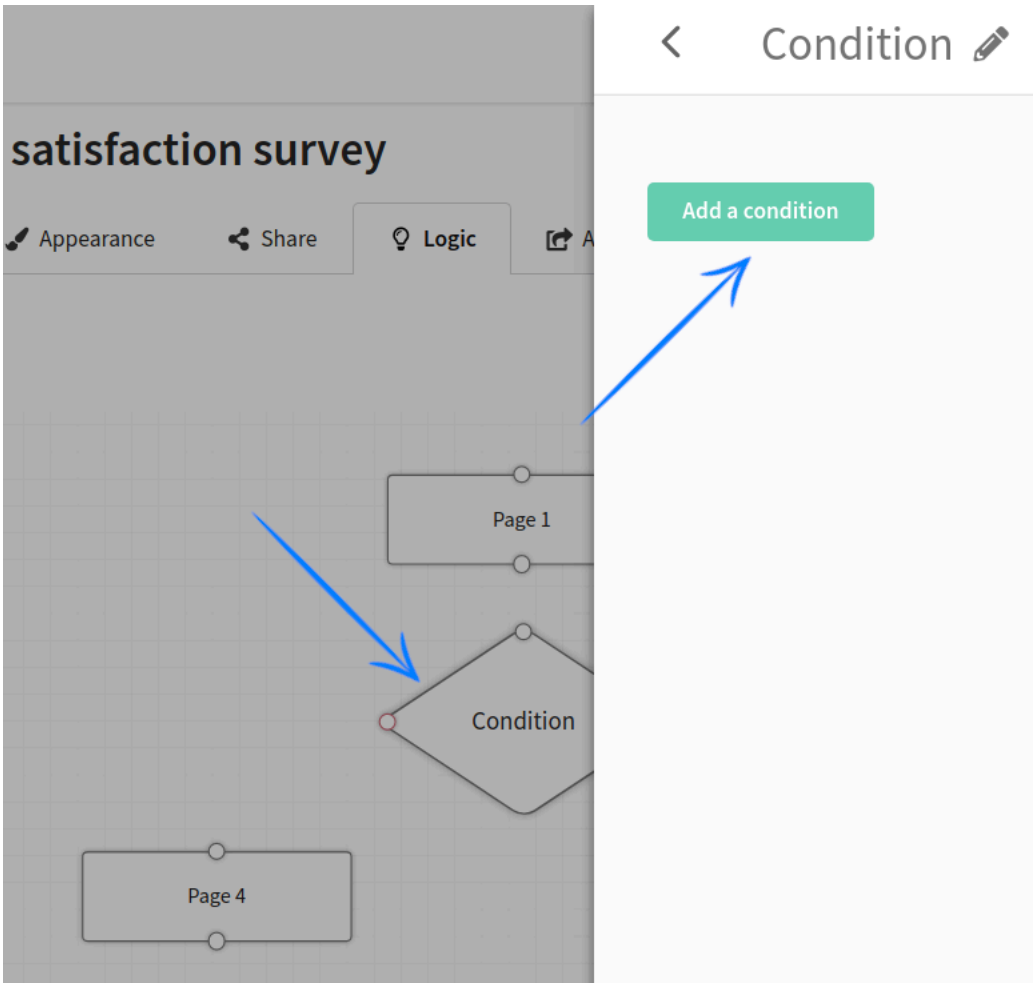
Adding conditions

To add a condition block, click the button

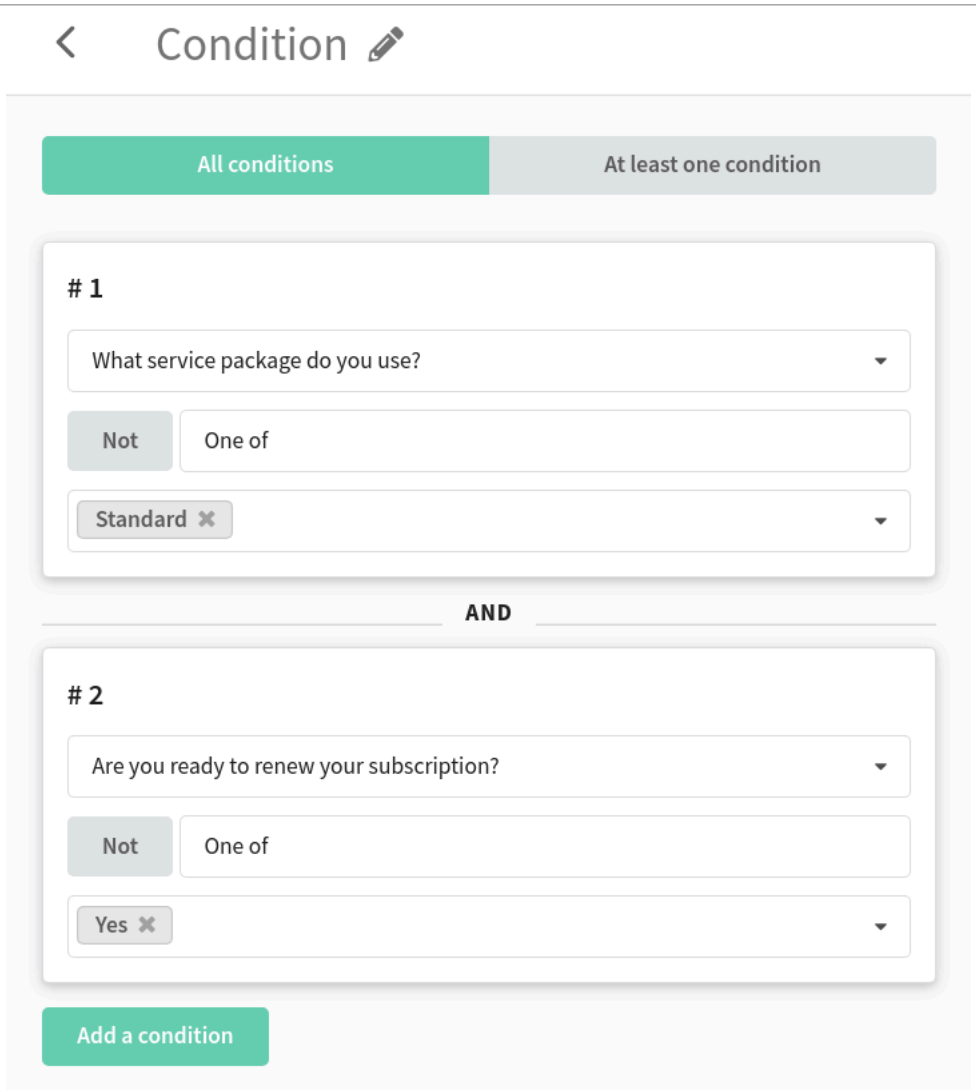
and select Condition:



Click on the condition block (diamond) and click Add a condition:



For one condition block, you can add several conditions at once. It is possible to check the user's respon conditions) or for meeting at least one of them (At least one condition).



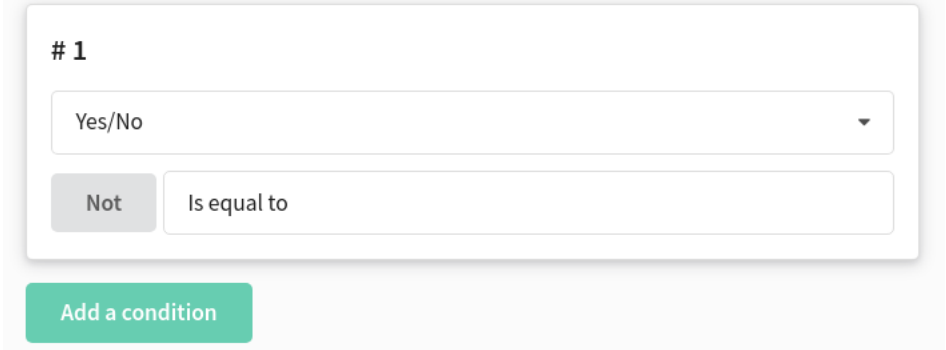
TIP

To remove the entire condition block from the board, use the trash can icon in the upper right corner.

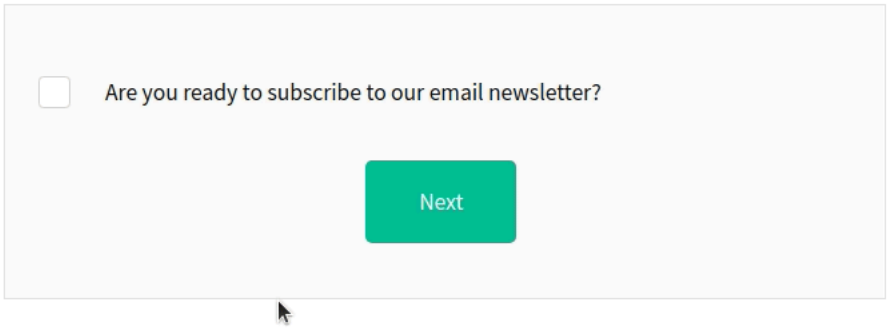
Condition types

Yes/No

Yes/No is the simplest condition. If the user gave a positive answer, they will go to one page; if negative

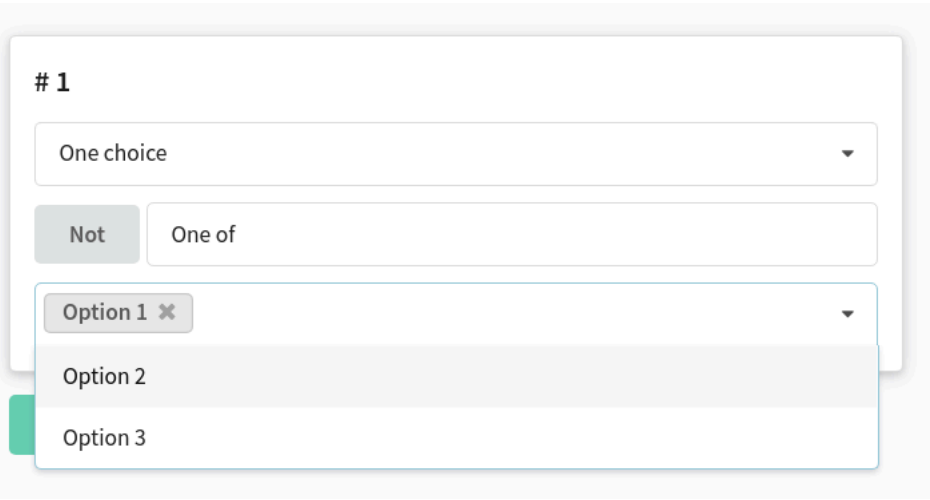


Example

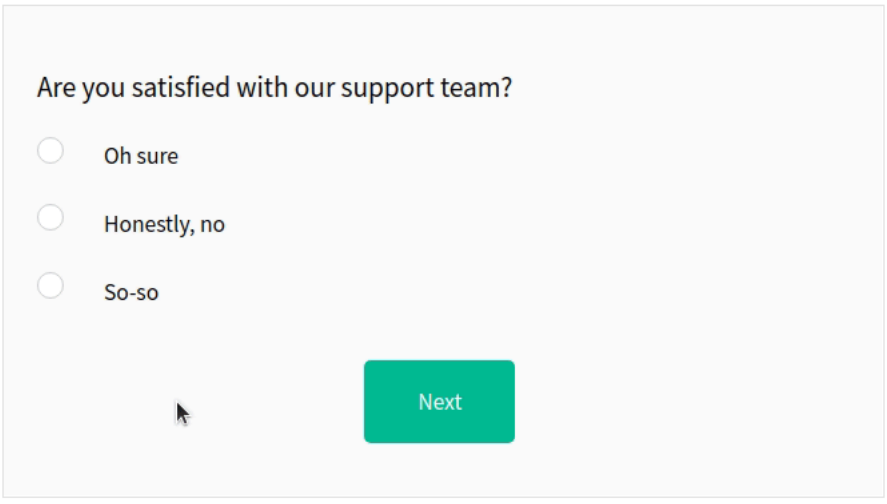


One choice

This element allows you to redirect the user to a specific page depending on the answer option they sele



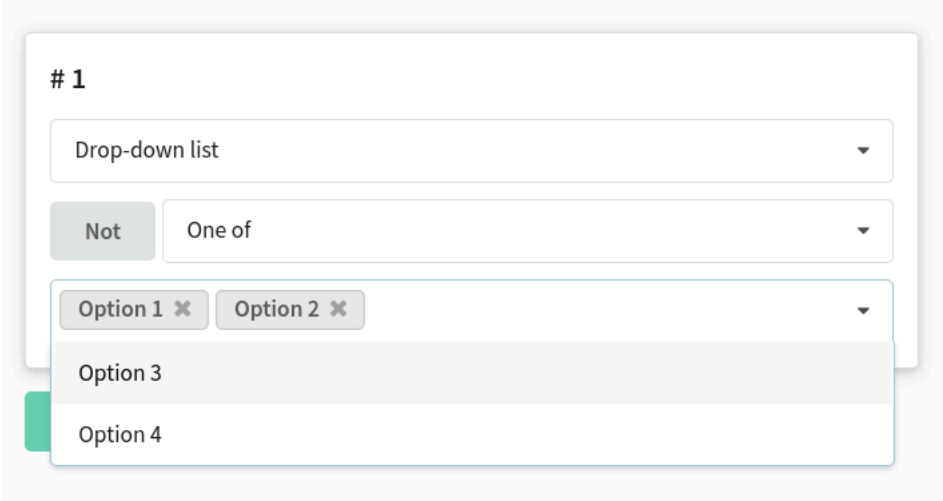
Example



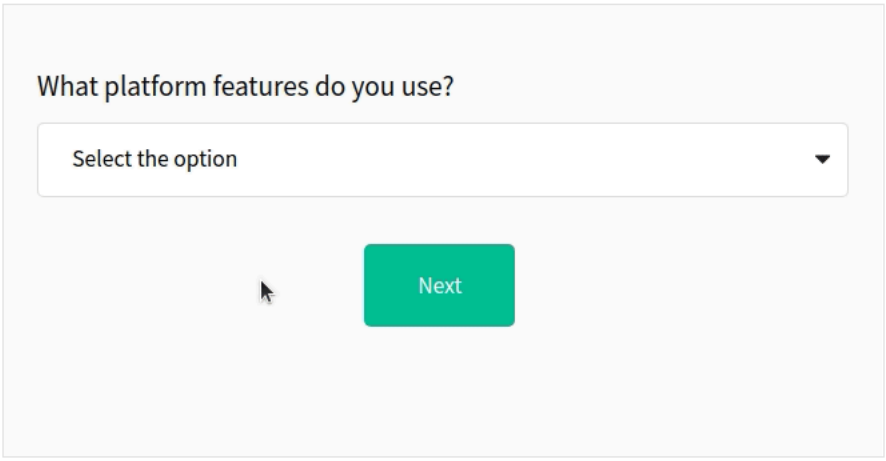
Drop-down list

The drop-down list allows the user to select one or more answer options from the proposed ones. You ca certain options are selected, the user is taken to one page of the form, and when others are selected, th For a drop-down list with multiple choices, the following conditions are available:

- **One of — at least one of the answer options selected by the user matches the specified value**
- **All — all answer options selected by the user match the specified value**



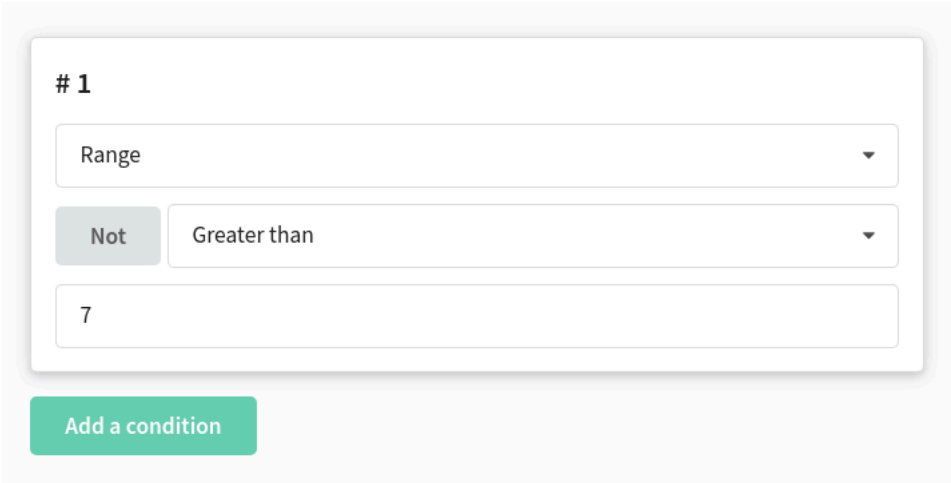
Example



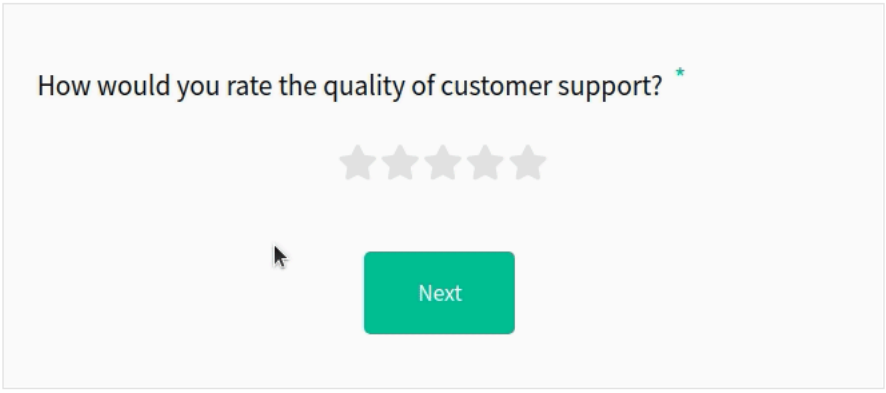
Range

The user will go to a specific page depending on which range value they check. You need to select:

- **the value with which the user's response will be compared**
- **the comparison operator (equal, greater than, less than)**



Example

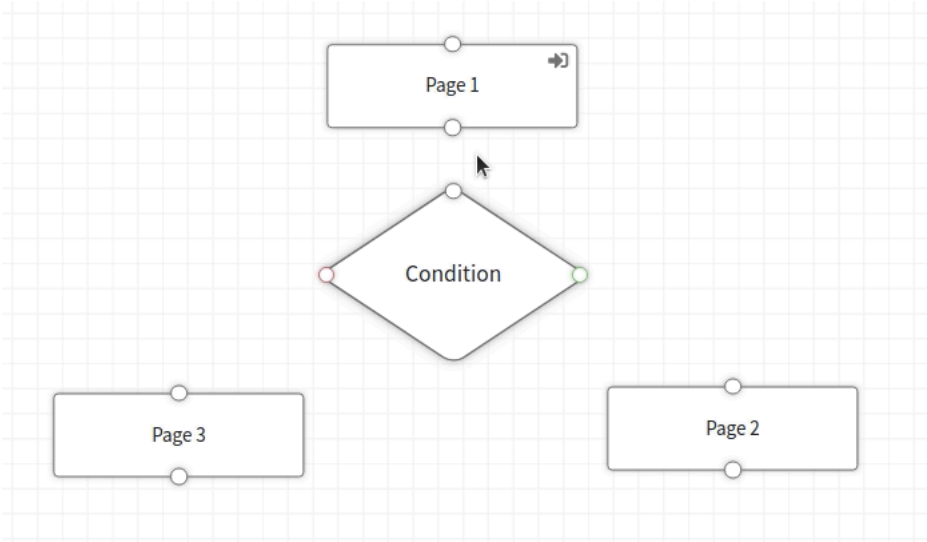


Linking blocks

Link the form pages in the required sequence. Click and hold the icon

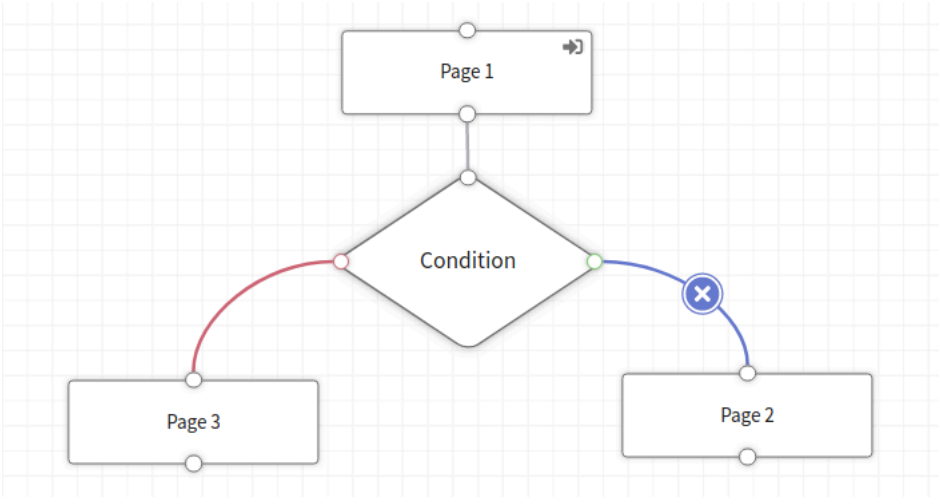
on the block border and drag the line to the page the user should be taken to.

- **Gray line — transition between pages without conditions**
- **Green line — transition if the user's answer matches the condition**
- **Red line — if the condition is not met**



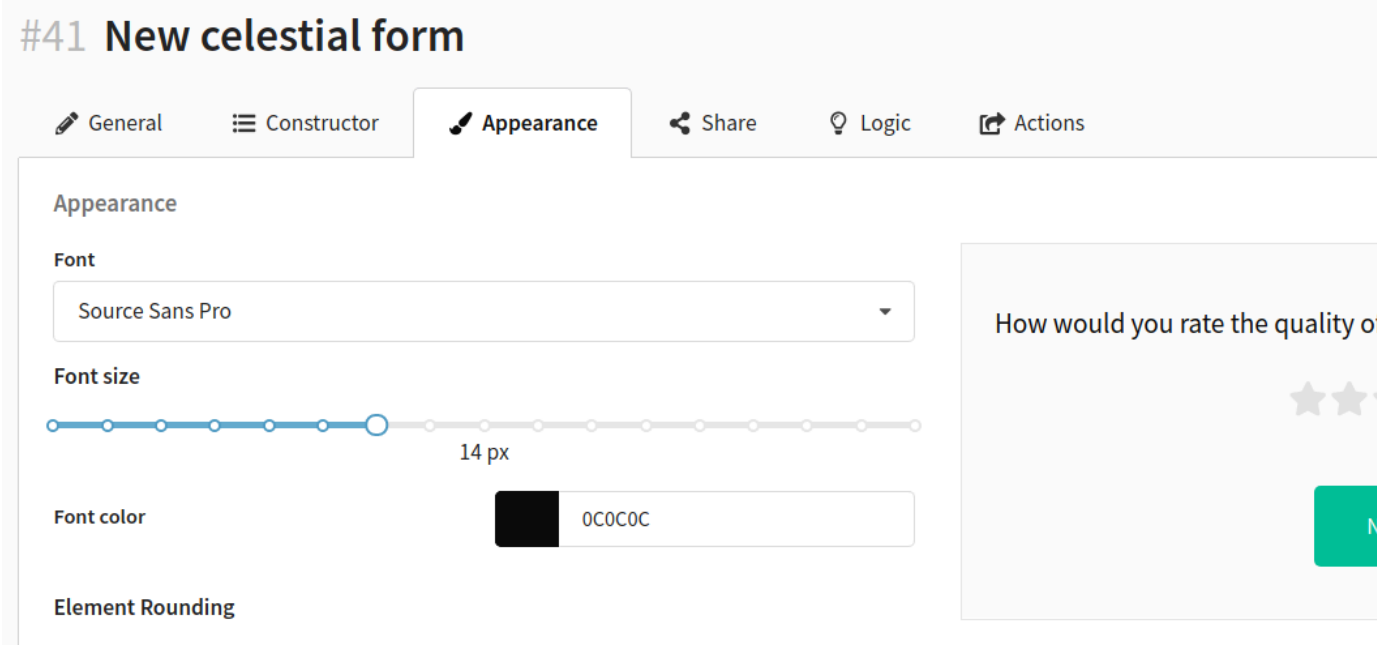
To remove a link, click on the line and then on the icon

:



Preview and publication

You can check the form logic on the "Appearance" tab:



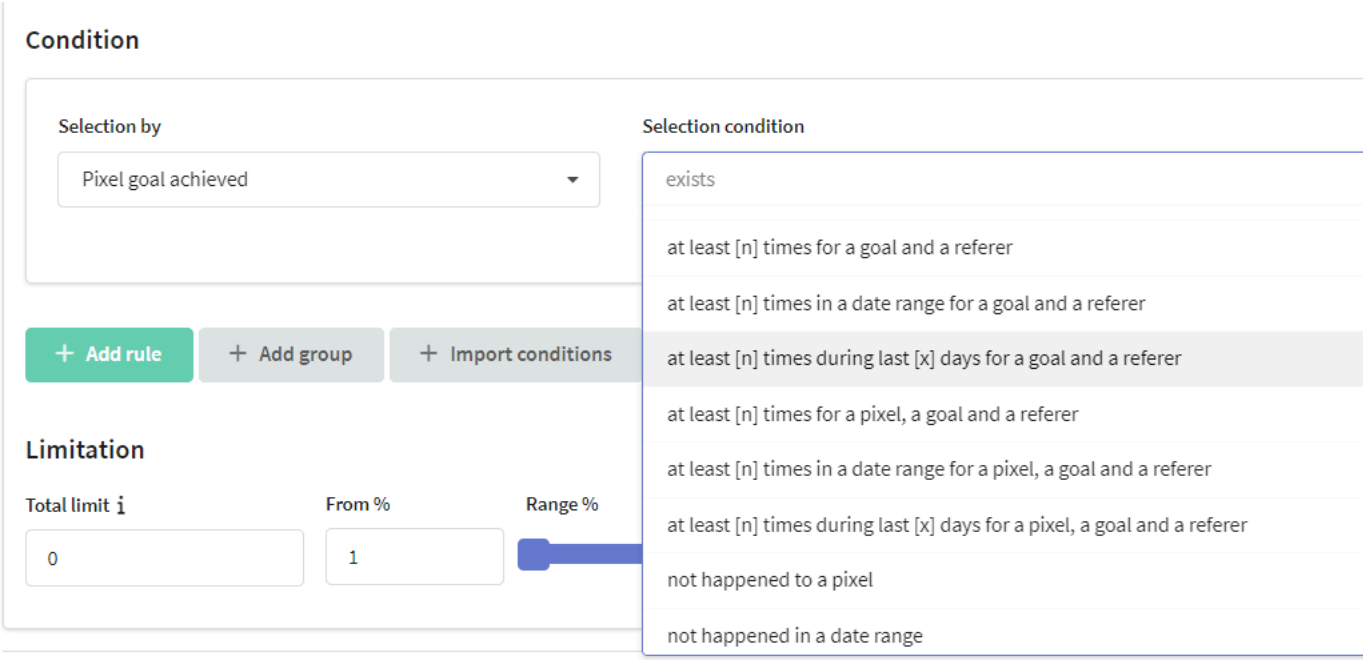
After checking, save the form. After that, you can publish it and use it on the website, or send the client

Pixels

Tracking Pixels are a tool for collecting data about user actions on a website. A pixel is a small piece of transparent image, invisible to the visitor. It can be linked to a page load or any interactive element — a
When a user triggers the pixel through their actions, a request with event information is sent to the platf used to track:

- **the geography of impressions and clicks (country, city, region) determined by IP;**
- **the browser and device platform;**
- **subscriber activity on the site before and after subscription;**
- **the URL of the page from which the user navigated;**
- **subscriber interests and preferences.**

This data is collected from request headers. Additionally, a pixel can transmit business information via g target action on the site (for example, a product purchase), and a value is a numeric quantity associated purchase amount). Goals and values are defined manually according to your needs.
The collected data is stored in the user's profile in the database if the platform can identify the user. Lat segmentation, scenario building, and triggering automated mailings. For example, when creating a profil use pixel goals, website addresses, time intervals, and the number of pixel activations:



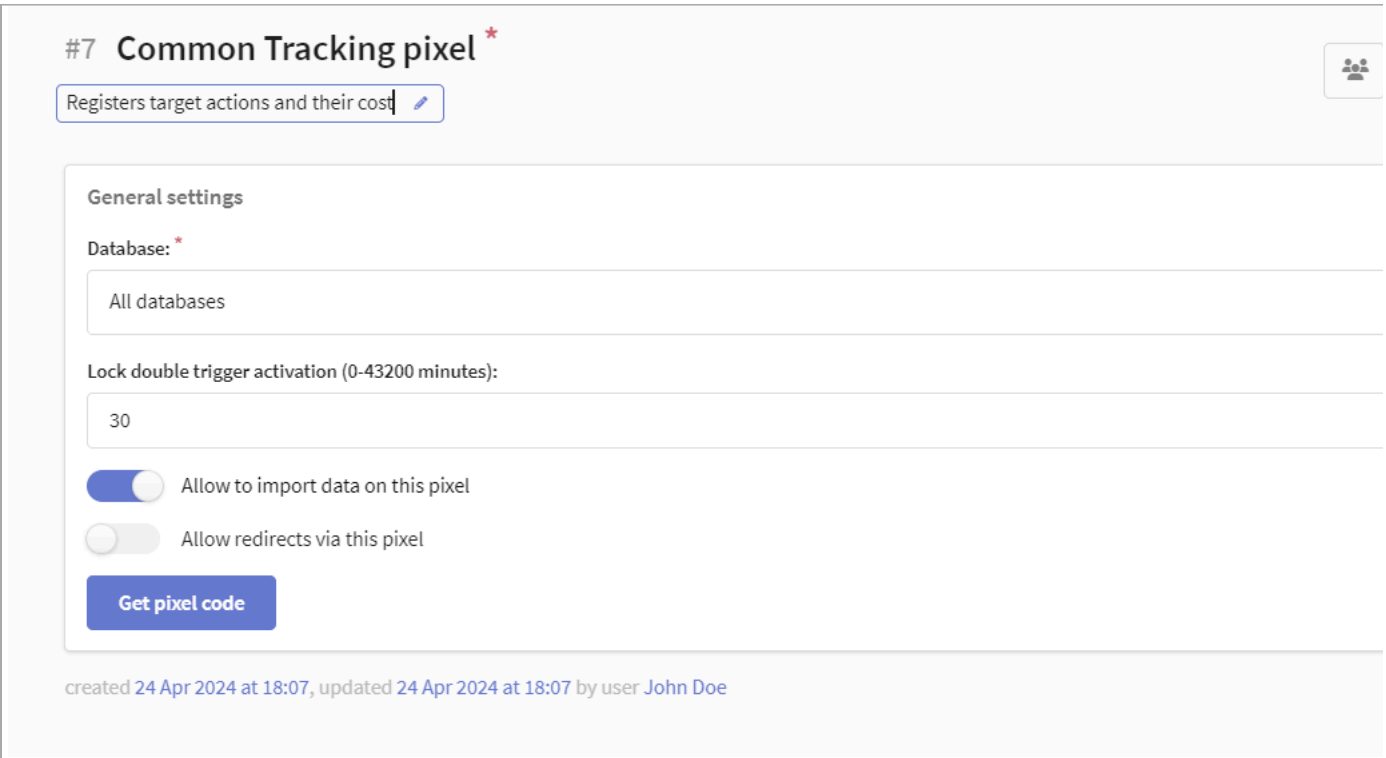
Pixel technologies also take a great part in keeping your databases up to date by refreshing profile locat
Pixel goal achievement can be monitored via Goals report.
More details about using pixels you can see in this article.

Enabling pixels

![img] (/img/attachments/3114601/pixels_nav_mednu_en.png)	**When using Altcraft Platform On-premises on your hardware, Pixels must be first enabled in platform administrative p** NOTE: To use pixels, you need to configure <a href="/en/admin-guide/platform-configuration/3113225" target="_blank">HTTPS protoc **After activation you'll see the new section "Pixels" in the platform interface.**

Creating pixels

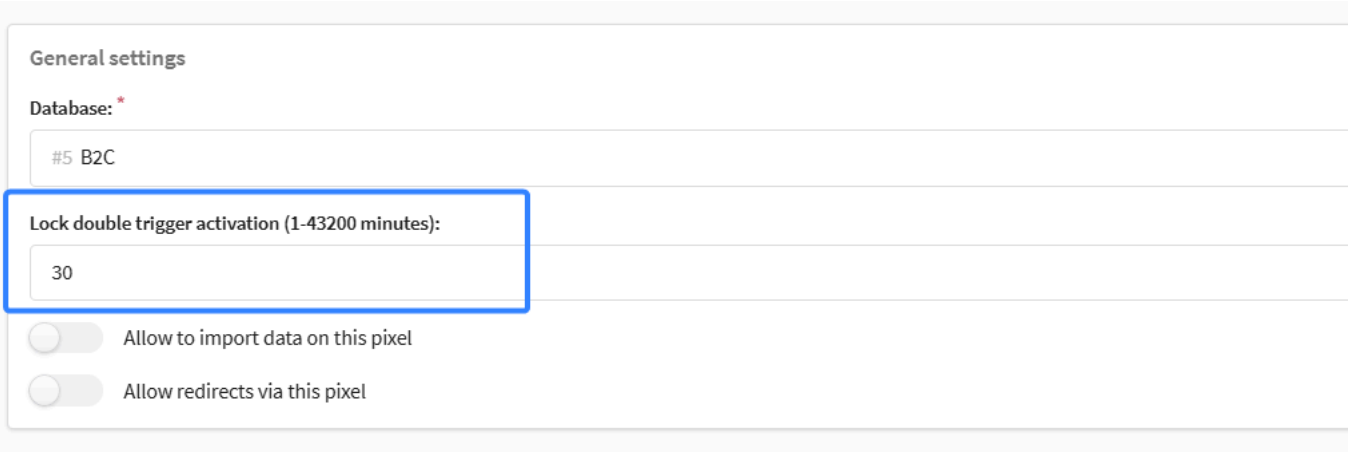
Let's create a new pixel script. First of all, you have to give it a name, write a description, set access gr
If you'd like to prevent unwanted pixel double activation, limit the time before another activation in the L



The trigger will be blocked for each specific goal rather than for the entire pixel. This means you can use different goals for tracking.

TIP

If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is re interval.



INFO

Allow to import data on this pixel — use this field to enable customer profile import at the moment your code.

Pixels can update your existing profiles or create the new ones, including for anonymous website visits:

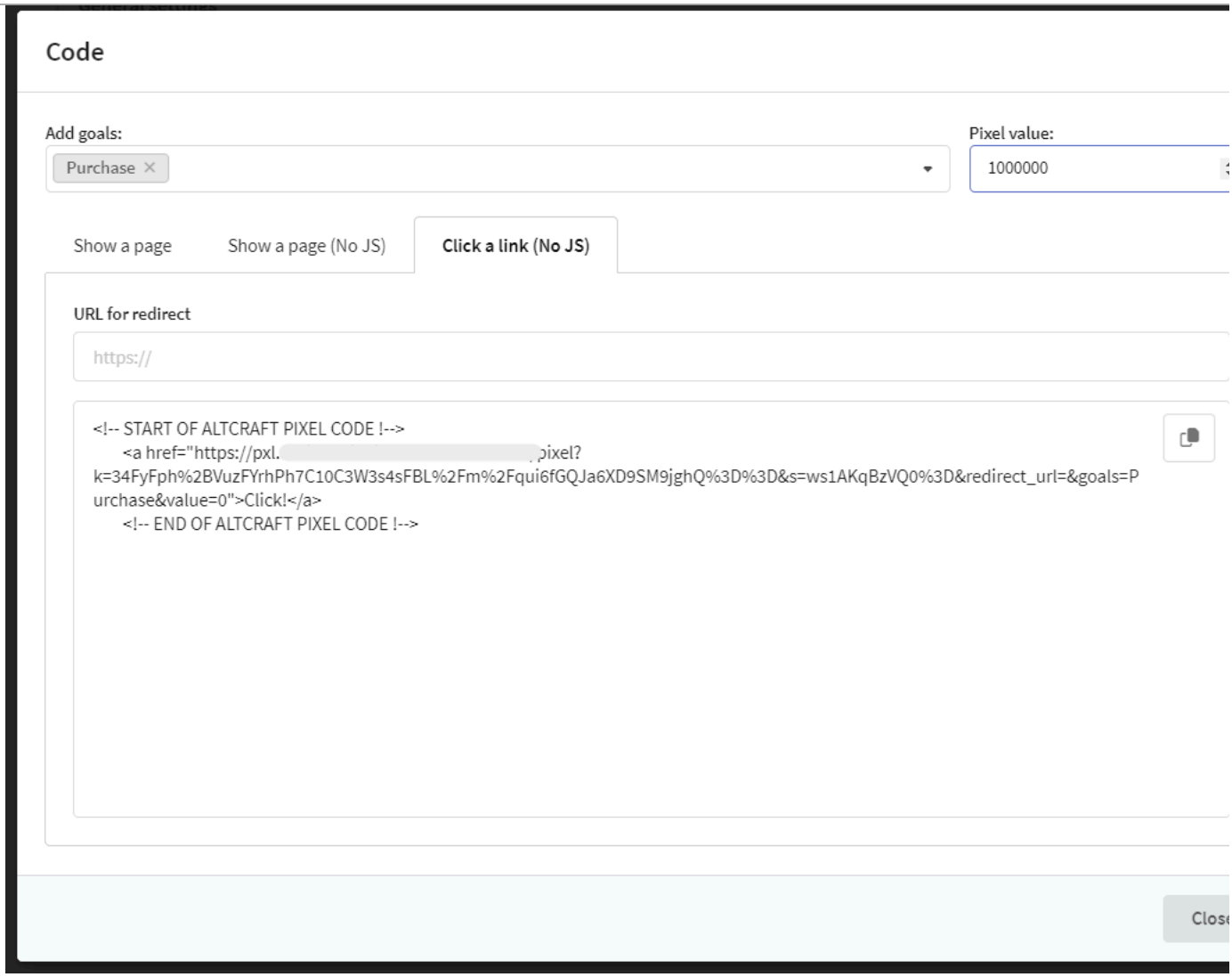
- **HTML pixels (image and URL redirect) use your customers' session data (cookie) for matching profi (e.g. pened mail from the same device) is not found - a new anonymous profile will be imported.**
- **JavaScript tracking codes can use both cookie data and customer profiles information. If a profile i the profile isn't found, a new profile will be created with the data you can specify.**

JS pixel import is described further on this page.

After you save a new pixel, you'll be able to open Get pixel code interface. The first thing is to decide up

- **Show a page — a JavaScript code that can be customized to detect user actions like scrolling a pag**
- **Show a page (No JS)— this is a somewhat classic pixel technology, representing a transparent imag request. It can be used for both your website and messages — no JavaScript required.**
- **Click a link (No JS) — this technology is a hyperlink wrapped in a redirect tracking request. It can be external resources.**

For all pixel types you can set goals and goal values:

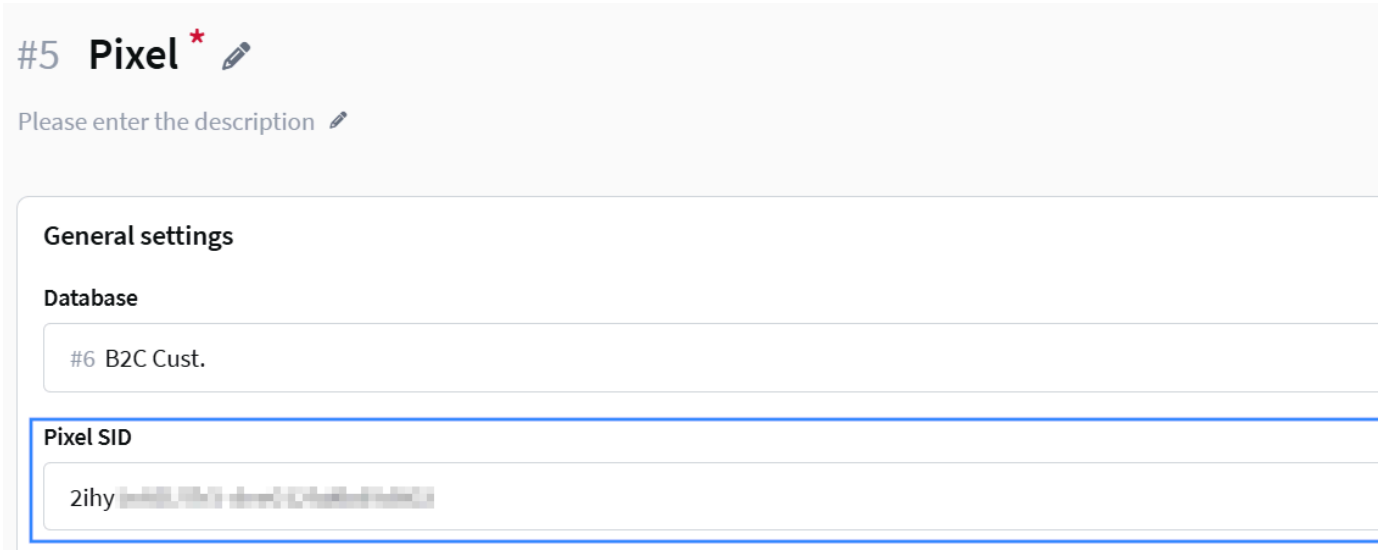


When a pixel code is created, give it to your website developers to implement it into your web site or app tracking your customers' actions.

For each script, you can select several goals and a common value for them (cost).

When placed on the page, the code will register the specified goals, no additional action is required. Eac associate goals with that particular pixel. This logic can be used to compare conversions across differer collection forms.

Pixel SID is a service string identifier of the pixel. The SID is generated automatically after the pixel is s editor. It is used by the platform to register events and is not required for standard website tracking.



The SID may be required when configuring mobile events via mSDK.

Web page tracking

Configuring the code

The script is placed on those pages where you want to track your customers' actions. When a user visits specify in pixel settings (Add goals). You can record one value for these initial goals.

INFO

Symbols available for use in goal names:

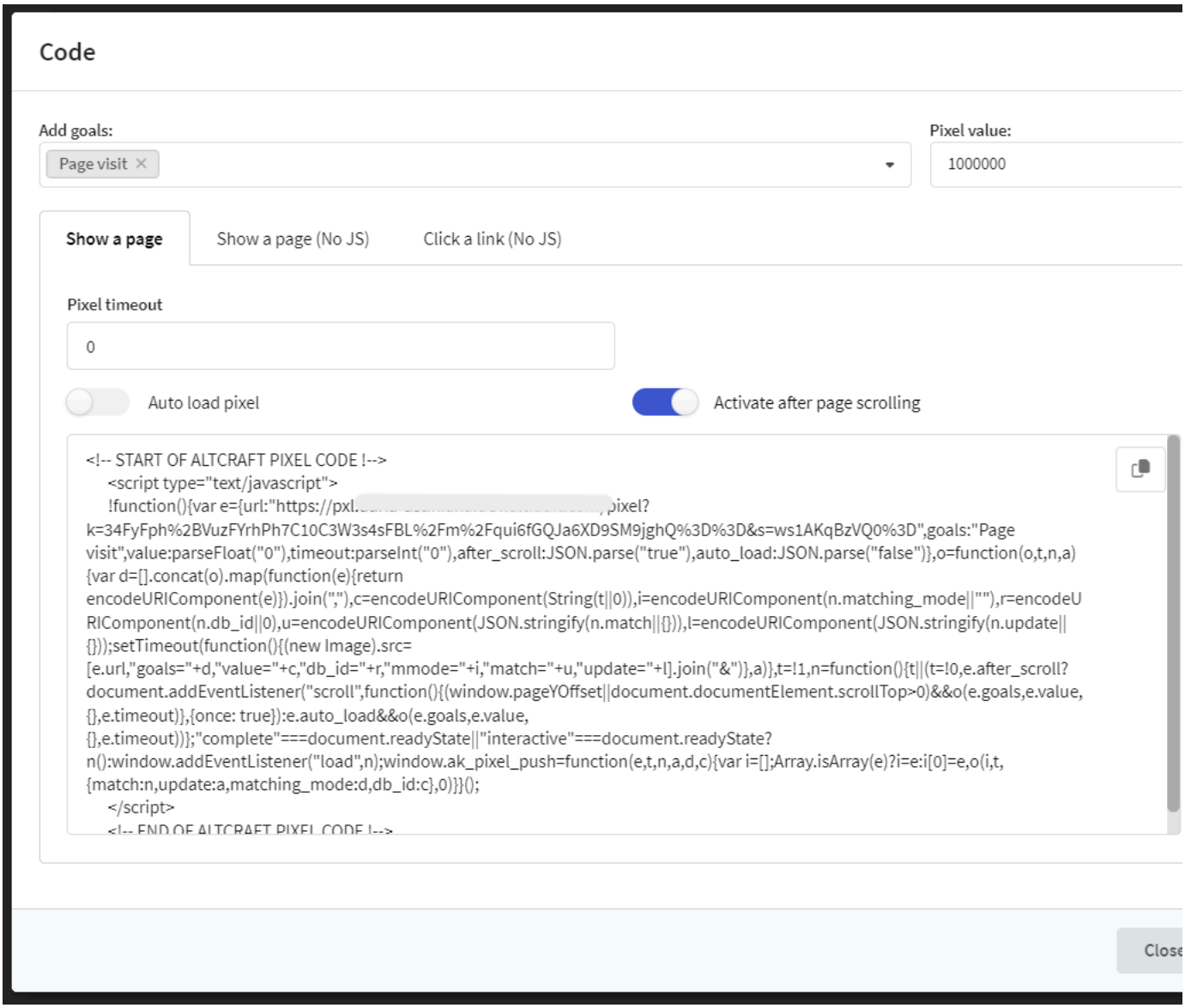
- **letters of the Latin and Russian alphabet (upper and lower case)**
- **spaces and tabs**
- **currency symbols**
- `! # $ % & * + - . , : ; '`
- `= ? @ ^ _ ( ) [ ] {} / \ |`

The maximum length is 128 characters.

Below you can set to activate pixel at once when the page is loaded, or to wait for page scrolling to excl query.

TIP

After you configure a new pixel, give its code from the Show a page tab to your website developers alon page.



Recording goals via API methods

On the web page with the script implemented you can track any of your customers actions based on bro be a mouse focus on a specific banner — or a popup activation — or leaving a page. On these events you and their values. For commercial actions you can register both profit and fees, if any.

For the page elements and browser events use the following method - `ak_pixel_push()`**:**

These arguments are used to record target actions:

- **tags — goal name or names. Can contain a string value or an array of strings.**
- **value — value or price for the goals. Can contain positive and negative floating point values.**

Description	
Simple call	`ak_pixel_push(`
Goal registration	`ak_pixel_push(`
Goal and value registration	`ak_pixel_push(`
Several goals with a negative value. One value will be recorded for all the goals	`ak_pixel_push(`

Customer profiles import and update (JS)

If customer profiles import is enabled in pixel settings, you can add arguments for searching an existing profile data: ak_pixel_push(tags, value, match, update, matching_mode):

- **match — this one is used for profile search. It is an object with one or two properties: "search field": database ID" — in case a database is not specified for the pixel.**

> **TIP**
>
> `db_id` **can be ignored when a profile database is specified in pixel settings.**

- **update — can contain any customer profile data including subscriptions.**
- **matching_mode - can contain any type of matching.**

> **TIP**
>
> `matching_mode` **can be ignored when searching by custom fields.**

Example of goal registration with profile import

Below are examples of goal registration with different type of matching:

email

```
ak_pixel_push(
  "click",
  100,
  {
    db_id: 1,
    email: "user@mail.ru"
  },
    {
        "_fname": "John",
        "_lname": "Doe",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "john.doe@example.com"
            }
        ]
    }
);
```

email_profile

```
ak_pixel_push(
  "click",
  100,
  {
    db_id: 1,
    email: "user@mail.ru"
  },
    {
        "_fname": "John",
        "_lname": "Doe",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "john.doe@example.com"
            }
        ]
    },
  "email_profile"
);
```

custom

```
ak_pixel_push(
    "click",
    100,
    {
      db_id: 1,
      field_name: "client_id",
      field_value: 1,
    },
    {
      _fname: "John",
      _lname: "Doe",
      subscriptions: [
        {
          resource_id: 1,
          channel: "email",
          email: "john.doe@example.com",
        },
      ],
    },
  "custom"
);
```

phone

```
ak_pixel_push(
    "click",
    100,
    {
      db_id: 1,
      phone: "+79000000000"
    },
      {
          "_fname": "John",
          "_lname": "Doe",
          "subscriptions": [
              {
                  "resource_id": 1,
                  "channel": "email",
                  "email": "john.doe@example.com"
              }
          ]
      },
    "phone"
);
```

Redirects via pixel settings

If you allow redirects via the pixel, the user will be redirected to the URL you specified at the moment he

Redirect links can be used for fraud by sending phishing message with redirect_url. To avoid this, pixel ir redirect by default. To engage this option, you need to activate this flag:
When flag is activated, you can enter domains list allowed for redirect, in the field below

Allow redirects via this pixel

Pixel description:

*example.com

After these settings, redirect in messages with pixel will be available on for enlisted domains. Redirect f

Setting up Altcraft pixel goal registration using Google Tag Manager (GTM)

For the convenience of placing the Altcraft pixel on the site pages, you can use the Google Tag Manager site).

INFO: Altcraft Tag Manager

If you want to manage all tags and pixels centrally without using third-party services, check out the bu learn more about working with it in the article Linking a Pixel and Tag Manager

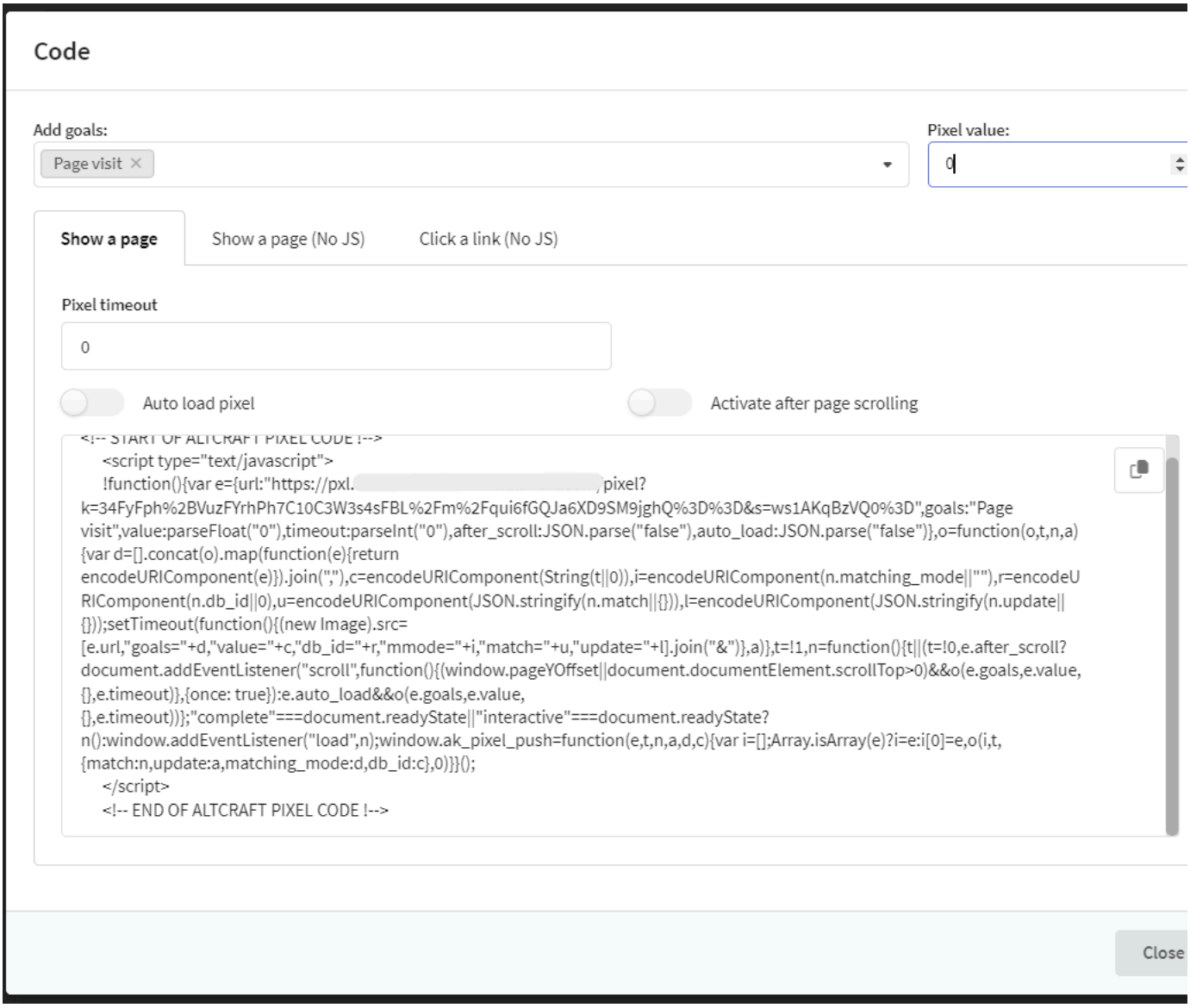
For this:

1. Create a new pixel and set basic settings: pixel name, description, access groups and associated data

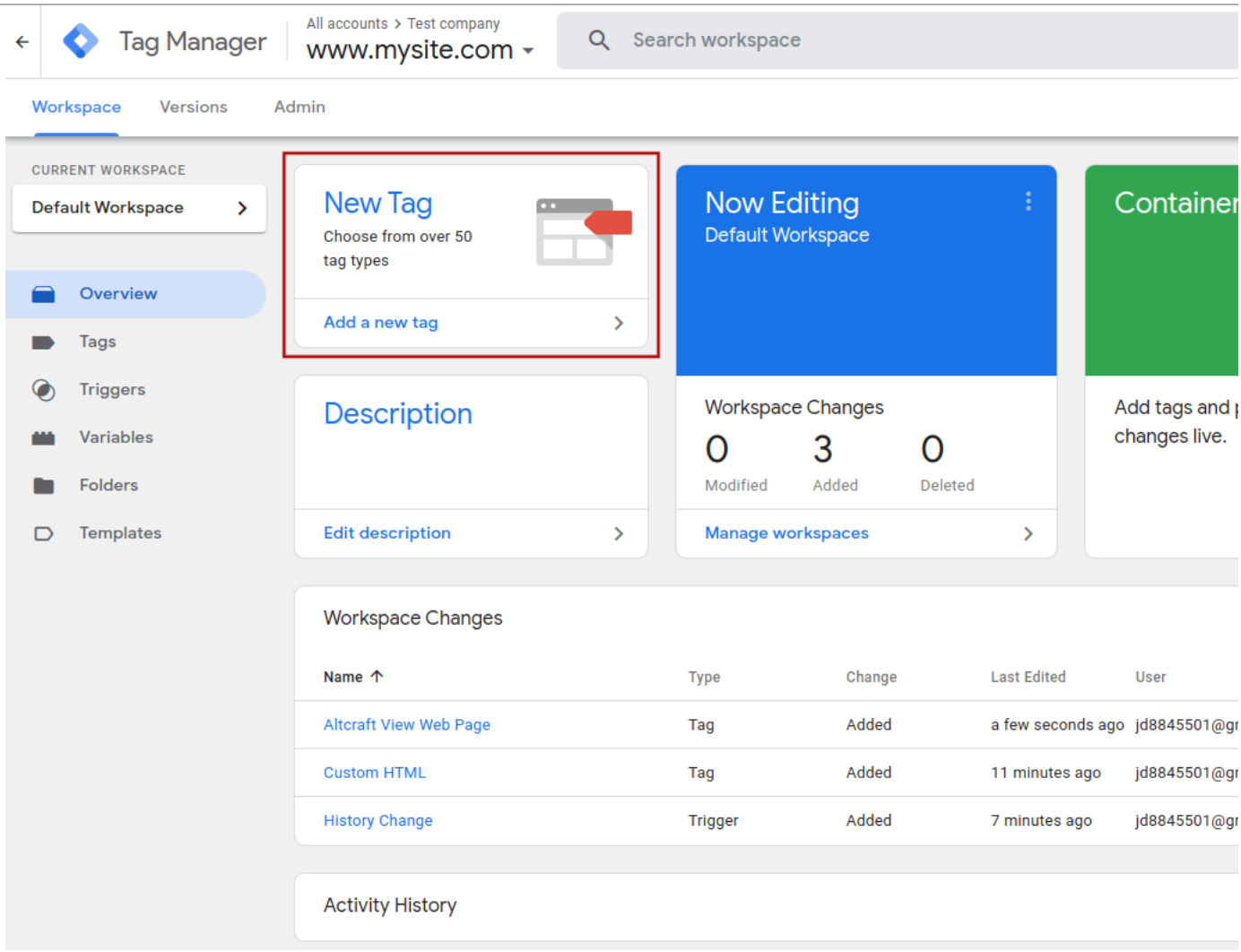
2. After saving the pixel, the tracking code configuration interface will be available. Copy the pixel code

CAUTION

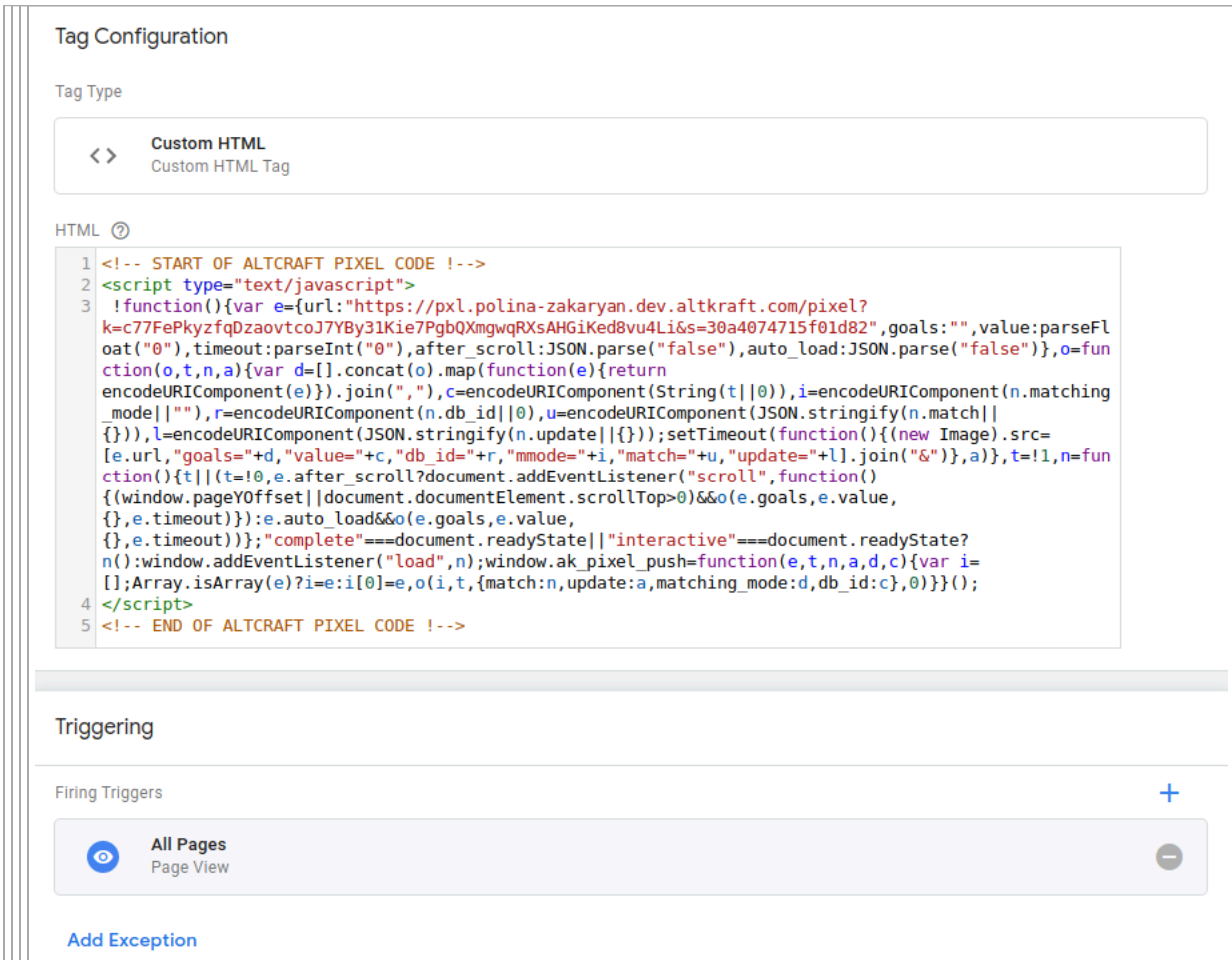
Before copying the pixel code, uncheck the “Auto load pixel” box. Only this way you can use the trackin



3. Your next step is to go to the GTM workspace and create a new tag. Choose Custom HTML as a tag ty



4. Add the Altcraft pixel code to the HTML field in the Tab Configuration window. Choose All pages in the



When you submit the changes, the container with Altcraft pixel code will be loaded on all pages of your s

5. Now, to transfer goals and, if necessary, their values to Altcraft, add the required number of Custom H you need.

```
<script>
        ak_pixel_push("PageView", 0);
</script>
```

CAUTION

Arguments are added by analogy with [Recording goals via API methods](#).

Altcraft View Web Page

Tag Configuration

Tag Type

Custom HTML
Custom HTML Tag

HTML

```
<script>
    ak_pixel_push("PageView", 0);
</script>

```

Triggering

Firing Triggers

History Change
History Change

Goal customer actions and scoring

Description

Altcraft pixels allow you to track customer actions on third-party network resources, such as:

- **Websites and landing pages, including affiliate ones.**
- **Forums, social networks, blog posts and similar services.**
- **Mobile and desktop applications.**
- **CRM and integration services of the company.**

Each registered client action contains two parameters:

- **Goal — the name of the goal action;**
- **Value — the numeric value related to the goal action.**

At the same time, the Pixel itself groups goals by type or traffic source, which can be used in various me

INFO

Symbols available for use in goal names:

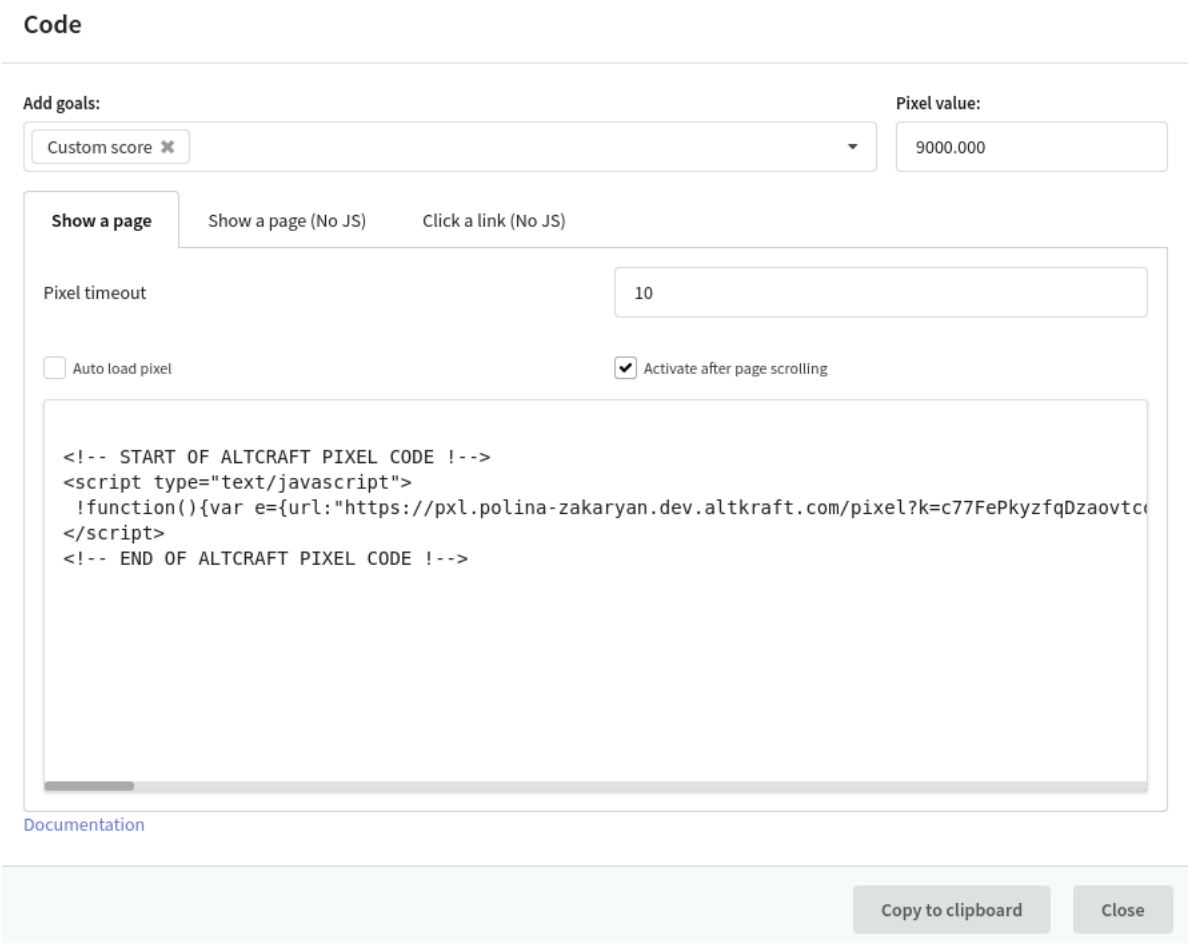
- **letters of the Latin and Russian alphabet (upper and lower case)**
- **spaces and tabs**
- **currency symbols**
- `! # $ % & * + - . , : ; '`
- `= ? @ ^ _ ( ) [ ] { } / \ |`

The maximum length is 128 characters.

Goal actions during viewing a web page

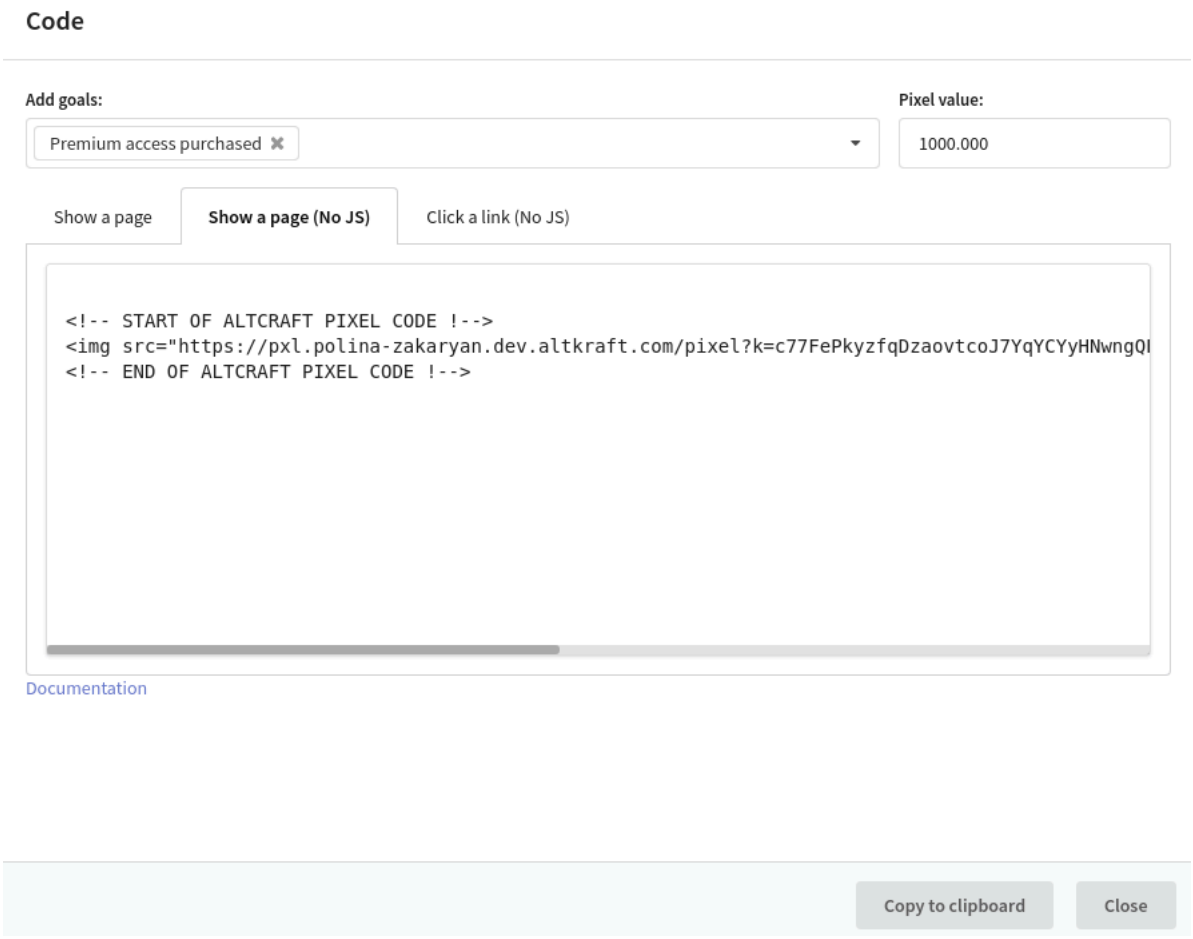
To register goal actions the visitors complete on a specific page when they view it, you can use two type

- **Show a page — JavaScript is used for this kind of tracking. Script activation is configurable, for exar after the user scrolls the page. Select the required pixel, click "Get pixel code" and in the appropriat parameters: Goals, Value, Timeout and scroll options.**



Then copy the code and give it to the developers of your website.

- **Show a page (No JS) — it is the simplest way. Here the pixel looks like a transparent image, that is s server. You add the code to any part of the webpage. Every time the user loaded the page, the pixel value related to it.**



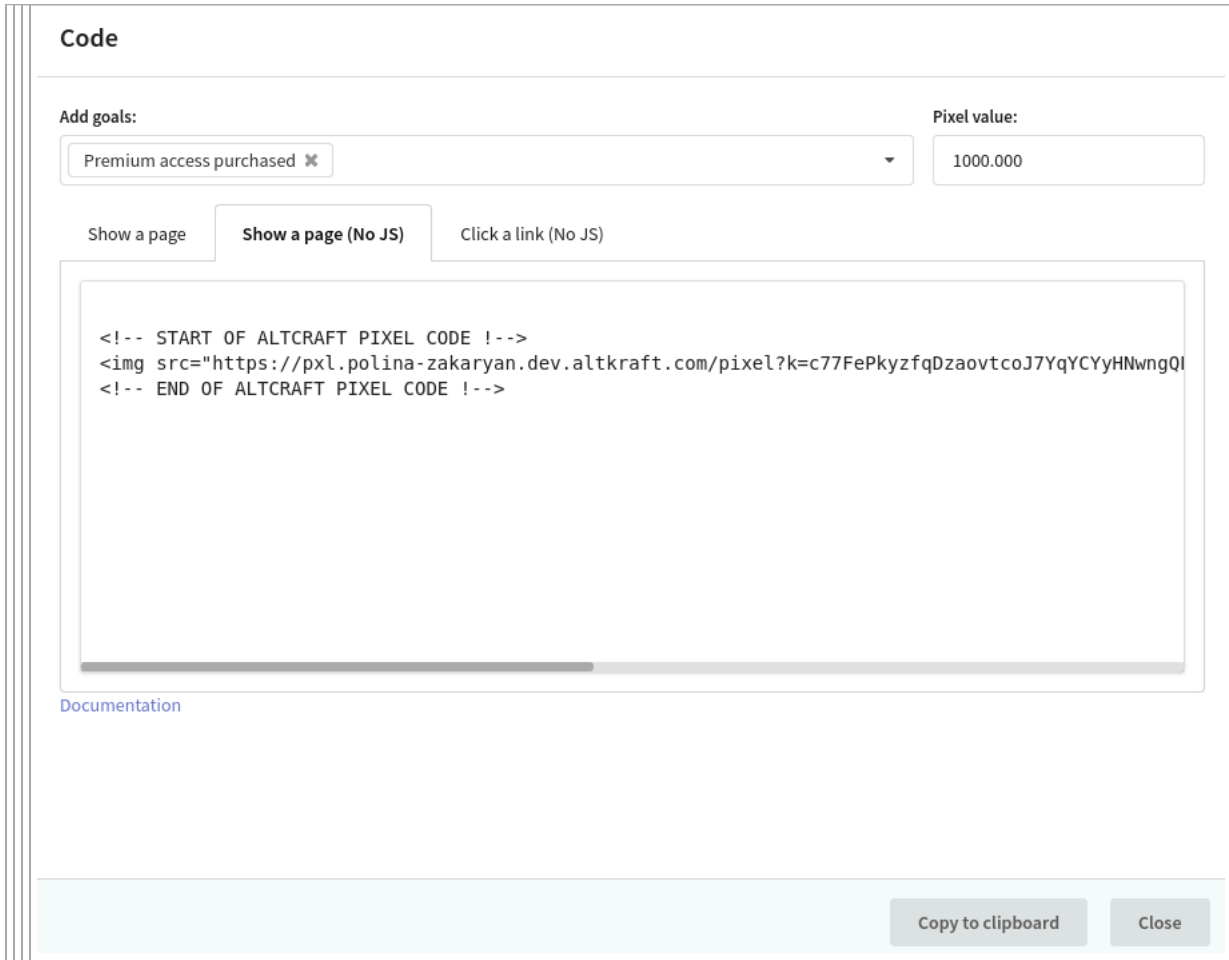
TIP

If you need to register several goals, each with its own value, or if you need a more advanced algorithm [Goal Registration API](#).

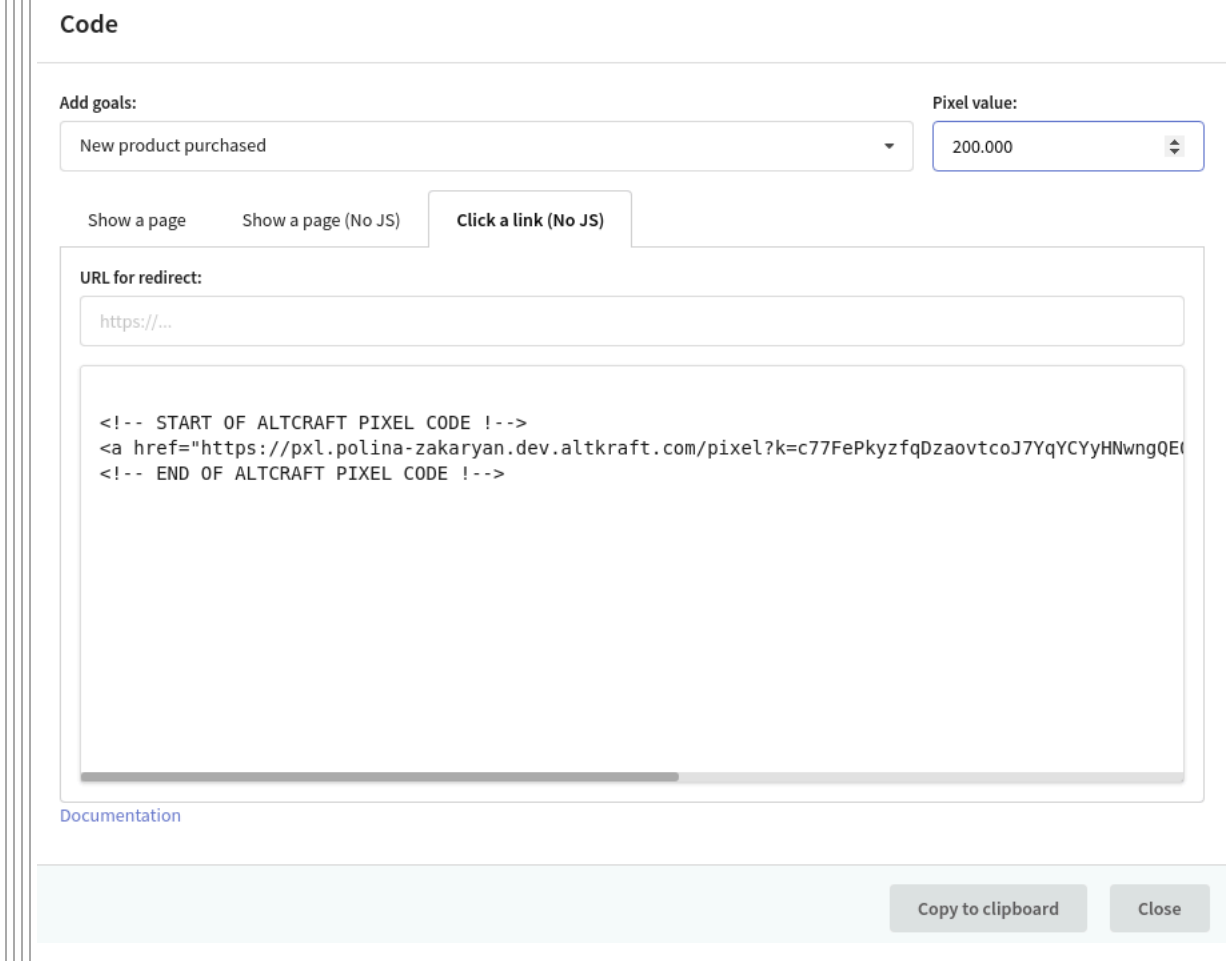
Goal actions in social networks and other public resources

You can use pixels to track the actions of customers on social networks, forums and similar web resour

- **Show a page (No JS) — a transparent image loaded from the platform's tracking server. It's added to external sources. When the client views such a record, the goal action is registered.**



- **Click a link (No JS) — add a redirect to the platform's tracking server for any link you want to place i or on a similar resource. Clicks on such a link will register the specified goal action and its cost.**



Goal actions in desktop, mobile and web applications

In applications tracking of events is carried out by requests to the Goal and Goals Value Registration AP request is the grouping of goals and costs. Unlike ready-made scripts, “goal” + “value” pairs are support several goals, each of which will have its own value.

Pop-ups

![img] (/img/attachments/tag_manager/weblayers.png)	Pop-up is a small window that appears on top of the main content of a web page. It is used to attract user's attention to cert For example, using pop-ups you can offer to subscribe to a newsletter, tell about a discount or promotion, collect applicatior

INFO

Pop-ups appear on the device screen when a certain event occurs — for example, when a page is loade when the page is scrolled to a specified threshold. To set the conditions for a popup to appear, install A

In this section

Creating and publishing a pop-up

This guide contains instructions for creating and configuring pop-up windows for your website using the to style the pop-up, set display rules, and analyze the results.

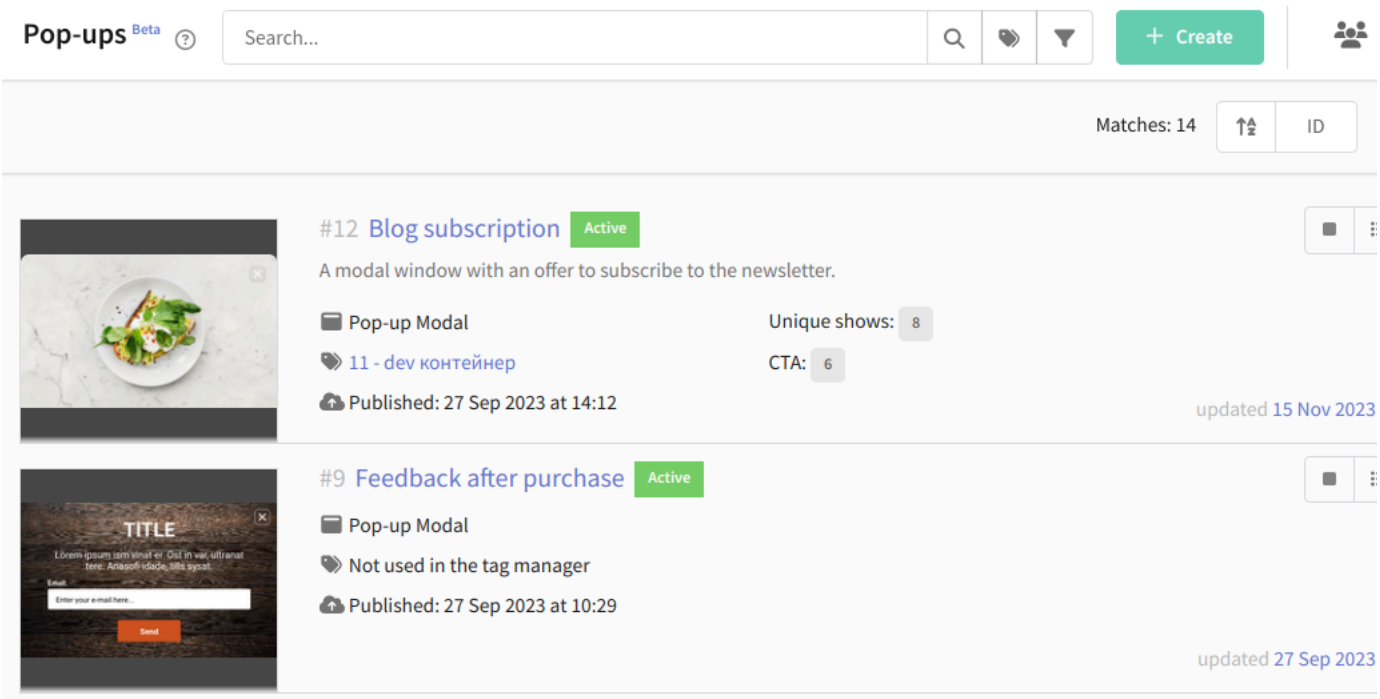
Full workflow for creating and configuring a pop-up

- **Step 1: Creating a pop-up**
  Choose the type of pop-up (modal, floating bar, slide-in, page takeover) and a base template for a qui
- **Step 2: Configuring content and design**
  Work in the visual editor: add text, media, forms, and adjust the appearance to match your brand.
- **Step 3: Configuring actions after form submission**
  Define what the user will see after submitting data (message, redirect), and configure notifications.
- **Step 4: Publishing and configuring display**
  Integrate the code into the site and set rules for display (time, triggers, target audience).
- **Step 5: Analytics and results**
  Monitor key metrics (conversion, impressions, clicks) to evaluate effectiveness and make decisions.

Each of these steps is described in detail below.

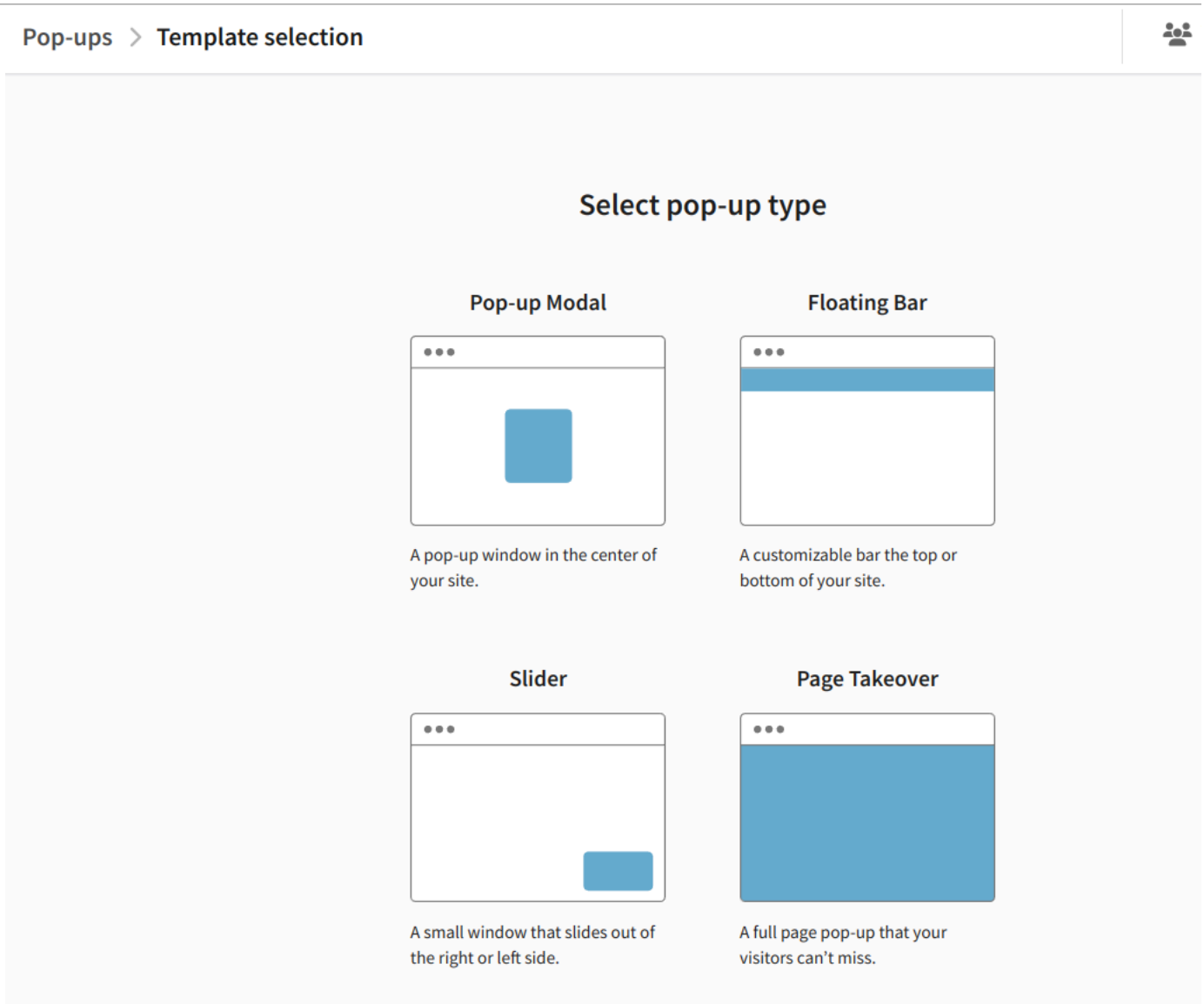
Creating a pop-up

Go to “Weblayers” then “Pop-ups” and click + Create at the top of the page:

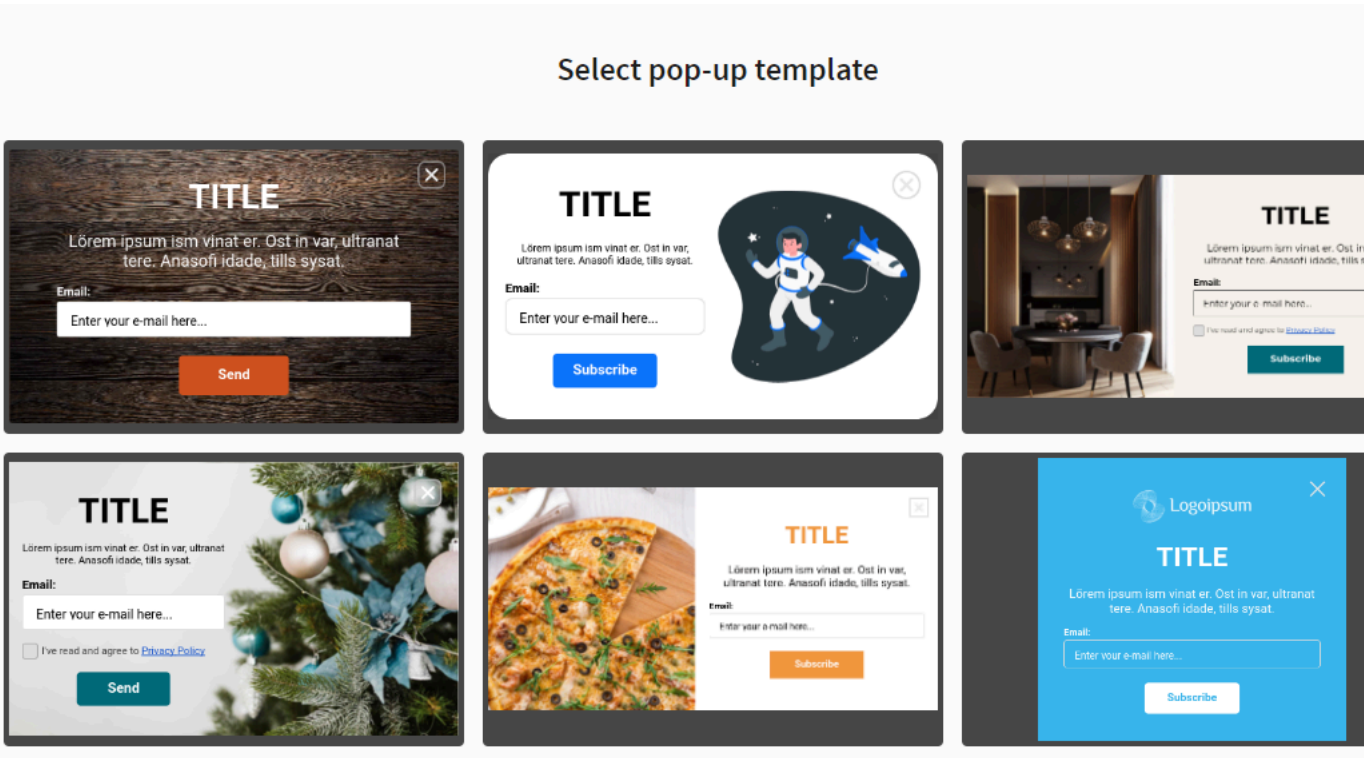


Select the type of pop-up:

- **Pop-up Modal — a pop-up window that appears in the center of the screen. It draws the user’s attent Typically used to collect email addresses.**
- **Floating Bar — a customizable panel placed at the top or bottom of the page. It remains visible even customers about news or limited offers.**
- **Slider — a small pop-up window that appears in the bottom-right corner of the page. It is less intrusi viewing the page content. It can be used to display additional information, offers, or feedback forms.**
- **Page Takeover — a pop-up window that occupies the entire browser screen. This ensures visitors do full-screen pop-ups should be used carefully, as they can be perceived as intrusive advertising.**



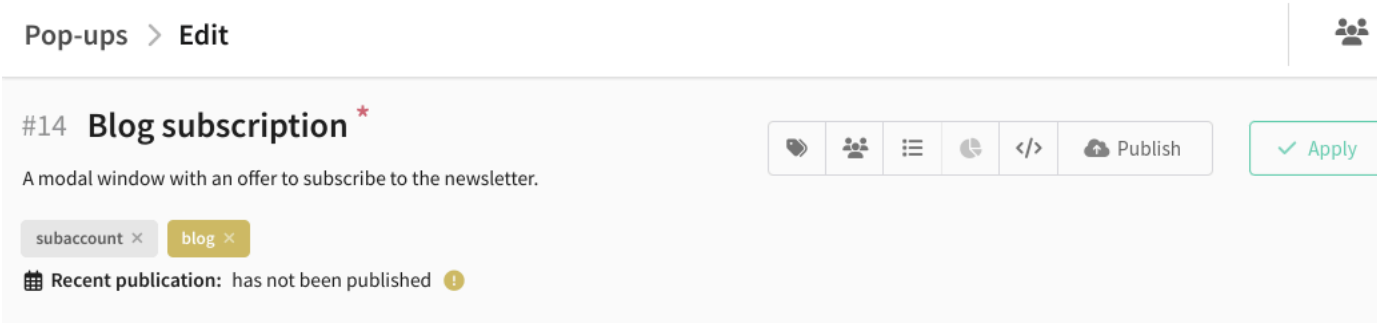
Choose a pop-up template. The platform already includes several ready-made templates. Use any of them fully customize the template (text, styles, fields, etc.) for your tasks.



Set the system name of the pop-up and, if necessary, add a description. You can also set tags for quick s

and assign [access groups]

.



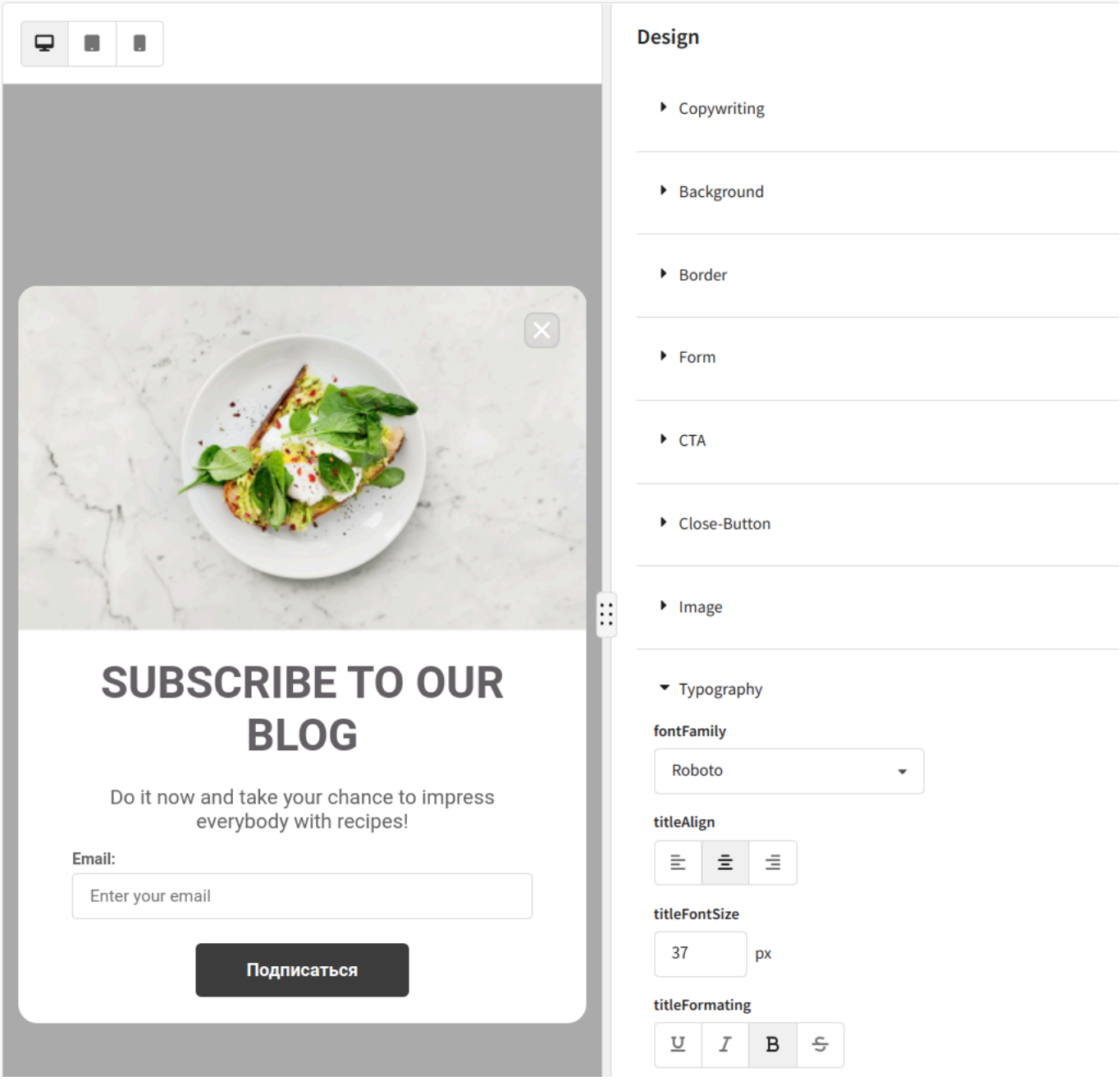
Configuring content and design

The appearance of a pop-up is defined by parameters. Parameters include text, images, and CSS styles. body text, and `backgroundColor` defines the background color.

Two tools are available for editing pop-up content:

- **Visual editor. Suitable for users without coding knowledge. Simply enter text, select images from the**
- **Code editor (HTML, CSS, JS). Suitable for advanced users who want to edit the pop-up code directly.**

All changes are displayed in the preview window in real time. Use the buttons in the upper left corner to



Visual editor

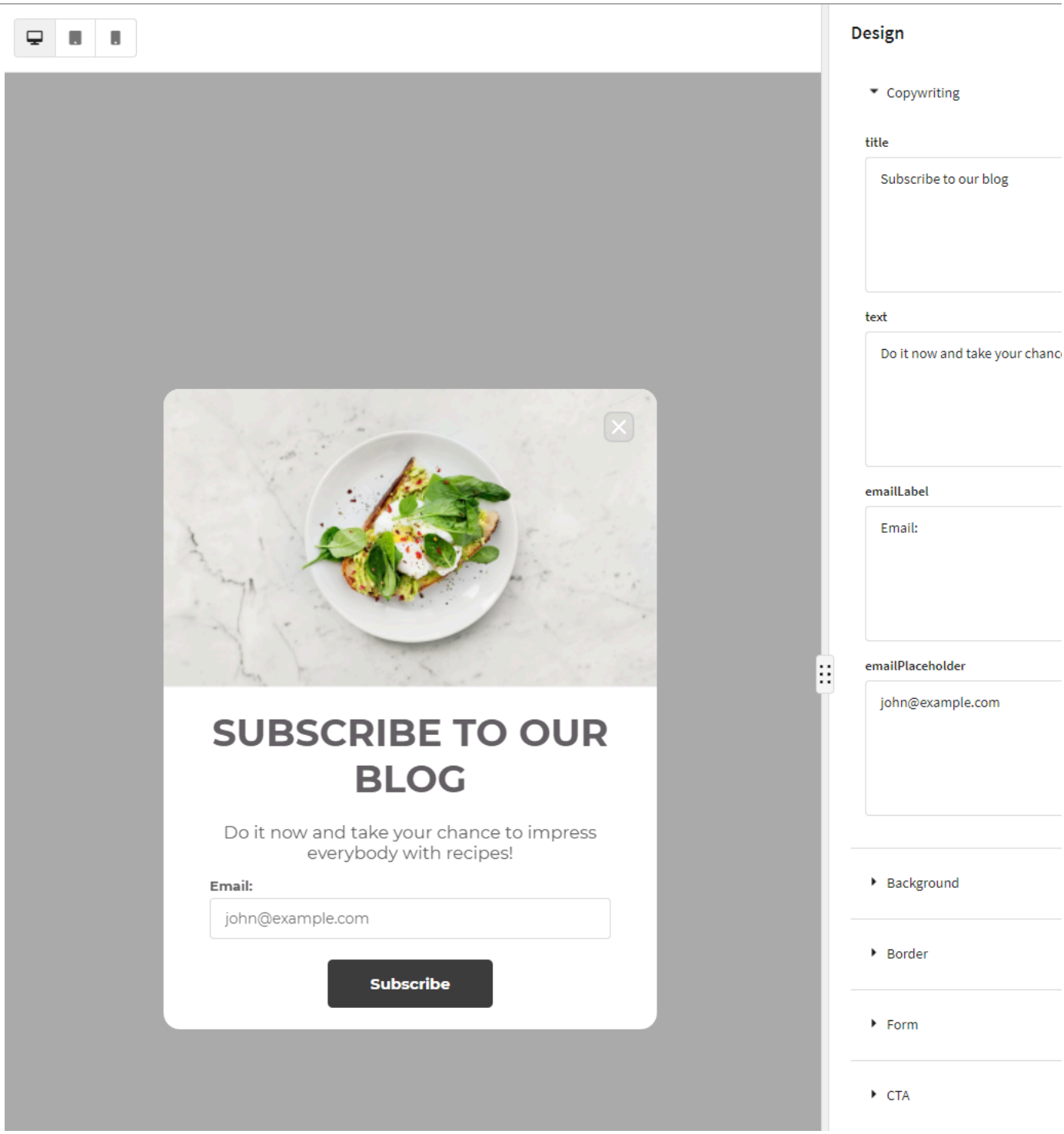
To customize text and design in the visual editor, go to Design. Parameters are grouped into categories:

Copywriting

On the Copywriting tab, set the pop-up text. Available parameters:

- **title — heading;**
- **text — main text;**
- **<field>Label — label for form fields;**
- **<field>Placeholder — placeholder inside form fields.**

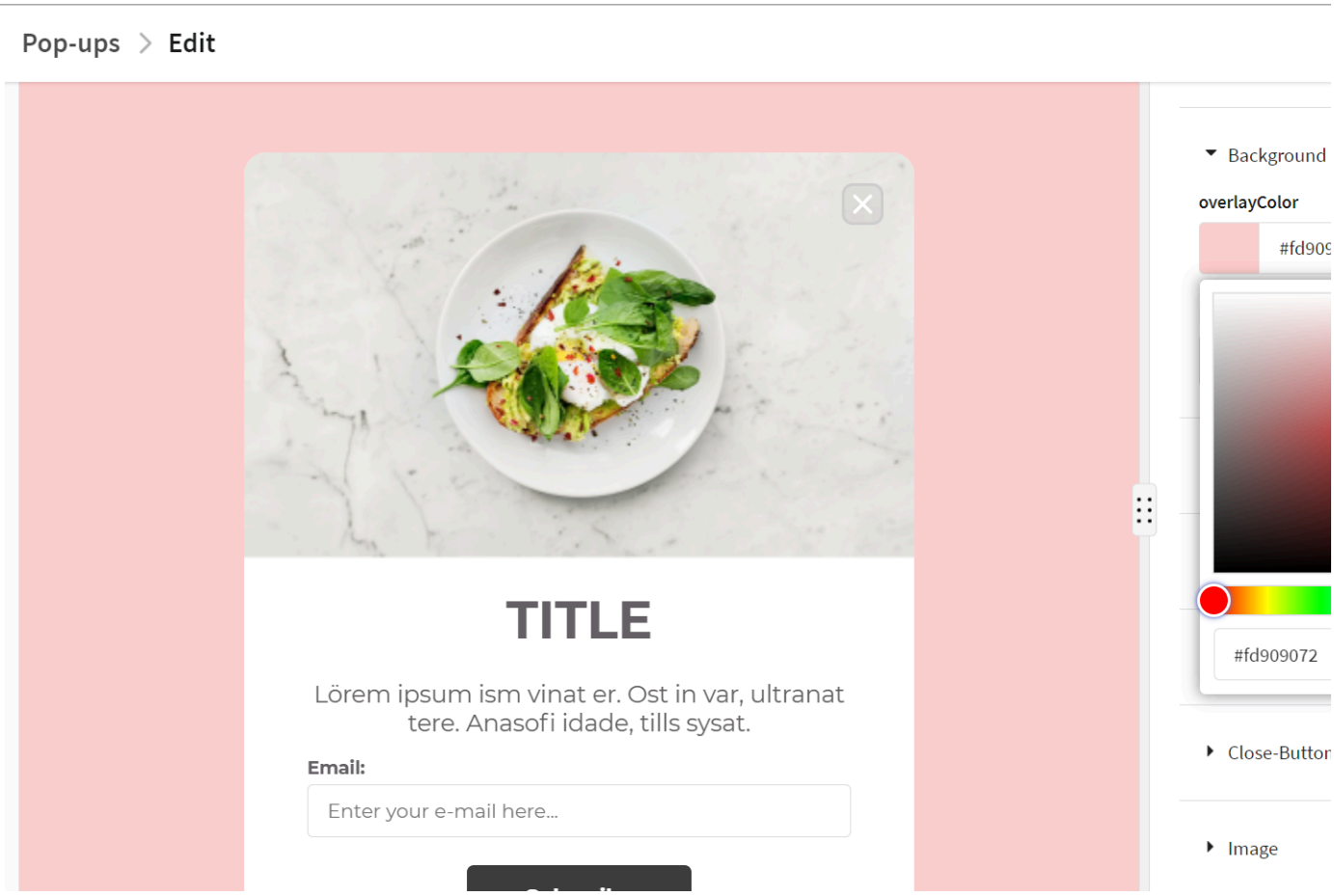
You can insert a `</>` variable into any text element to dynamically pull profile data. See [this article].



Background

- **backgroundColor — pop-up background color;**
- **backgroundImage URL — background image (upload to gallery**

  or specify URL);
- **overlay — background overlay for the page (color and opacity).**

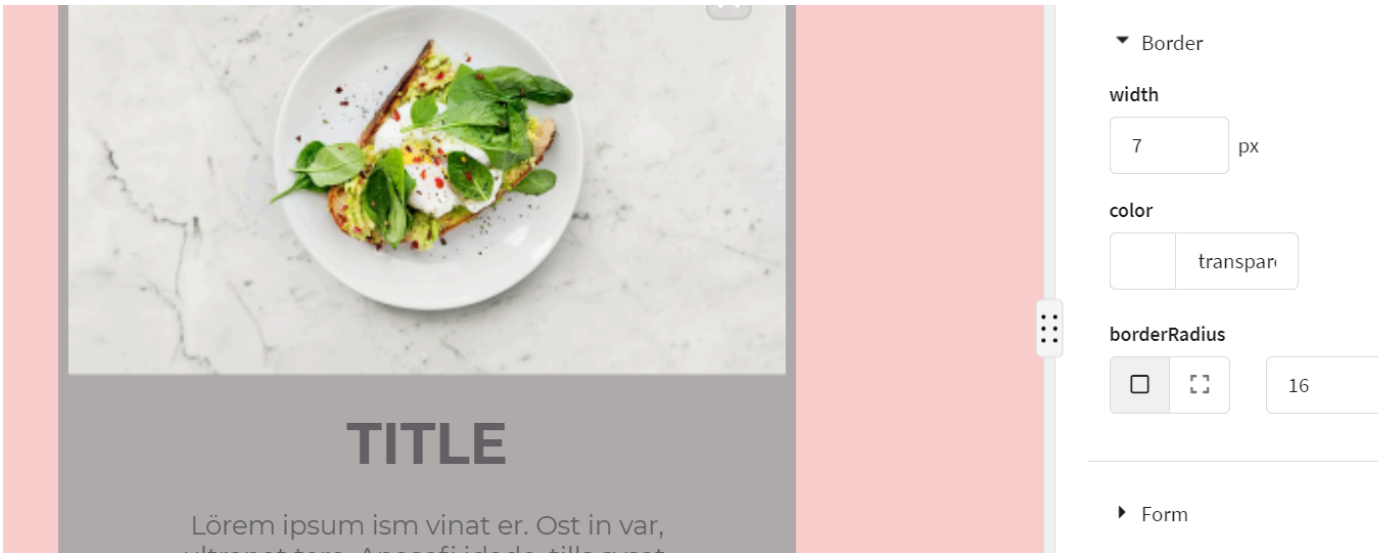


Border

Configure pop-up borders:

- **borderRadius — rounding (px, per corner)**

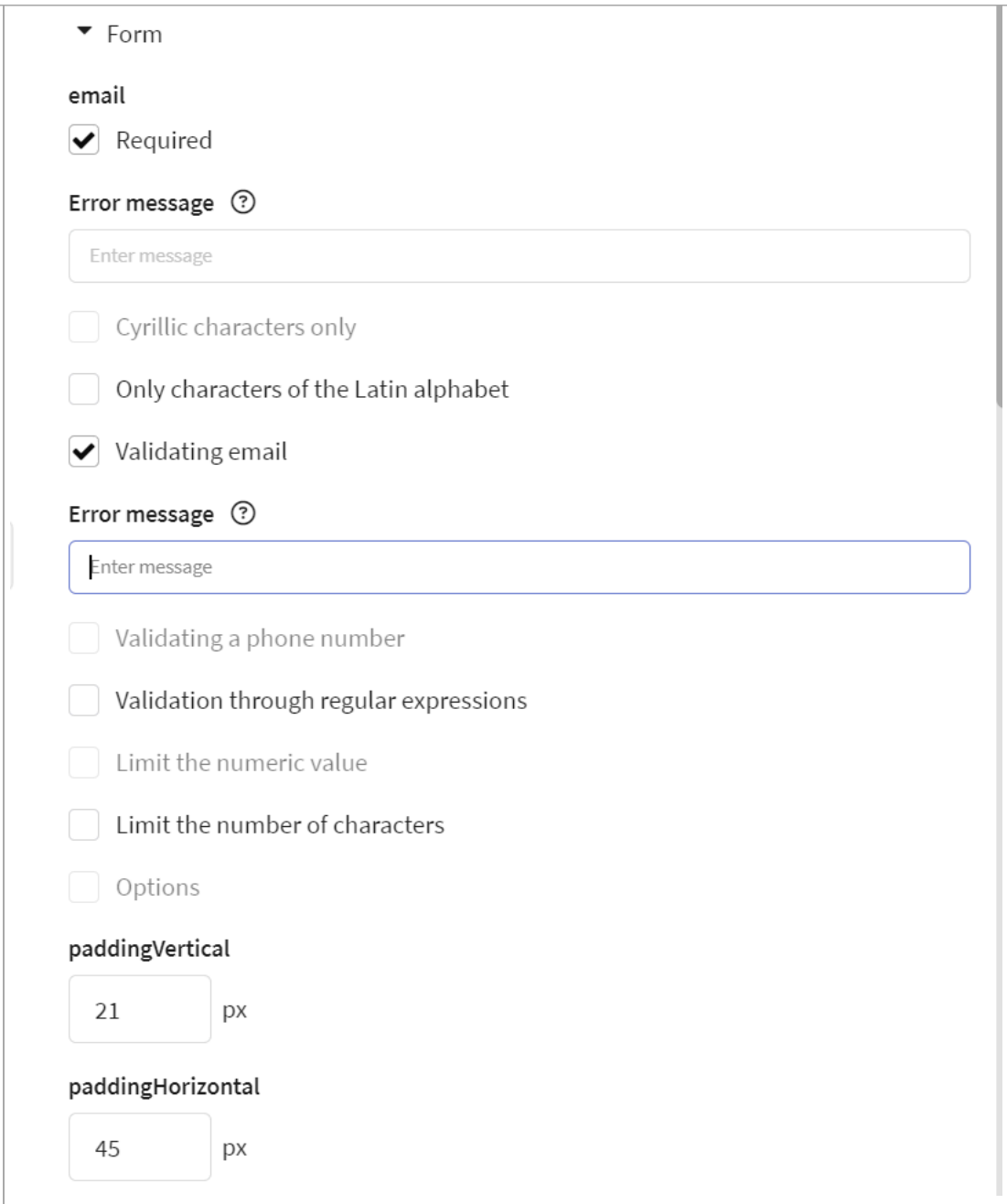
  ;
- **width — border thickness;**
- **color — border color.**



Form

Configure form fields:

- **set up validation rules;**
- **specify error messages shown to the user if input is invalid;**
- **edit CSS styles of form fields.**



Validation options:

- **Required — the user must fill in the field before submitting the form. If left empty, an error message**
- **Cyrillic only — only Cyrillic characters are allowed;**
- **Latin only — only Latin characters are allowed;**
- **Email — the value must match the email address format;**
- **Phone — the value must match the phone number format;**
- **Regex — you can apply your own custom regex validation rules;**
- **Numeric limits — define minimum and maximum values for numeric fields;**
- **Character limits — set the maximum number of characters allowed;**
- **Predefined answers — for fields with a limited set of options. The user must choose one of the allow will appear;**
- **Error message — the text shown to the user when input doesn't meet validation rules. Each field can**

Window styles:

- **paddingVertical — spacing between the button and the bottom border of the popup;**
- **paddingHorizontal — spacing between the fields and the left border of the popup;**
- **width — width of the popup;**
- **fieldsGap — spacing between form fields, including space between the last field and the button;**
- **borderWidth — border width of input fields;**
- **borderColor — border color of input fields;**
- **borderRadius — border radius of input fields. Radius can be set individually for each corner**

.

Input field styles:

- **inputPaddingVertical — vertical padding of the placeholder text inside the input field;**
- **inputPaddingHorizontal — horizontal padding of the placeholder text;**
- **inputFontSize — font size of the placeholder text;**
- **inputBackgroundColor — background color of the input field;**
- **inputColor — color of the user's text input;**
- **inputHoverColor — border color of the input field on hover;**
- **inputActiveColor — border color of the input field on click.**

Checkbox styles:

- **checkboxSize — size of the checkbox;**
- **checkboxLabelFormatting — formatting of the checkbox label text.**

Error styles:

- **errorColor — color of the error message;**
- **errorFontSize — font size of the error message;**
- **inputErrorBorderColor — border color of the input field when showing an error;**
- **inputErrorBackgroundColor — background color of the input field when showing an error.**

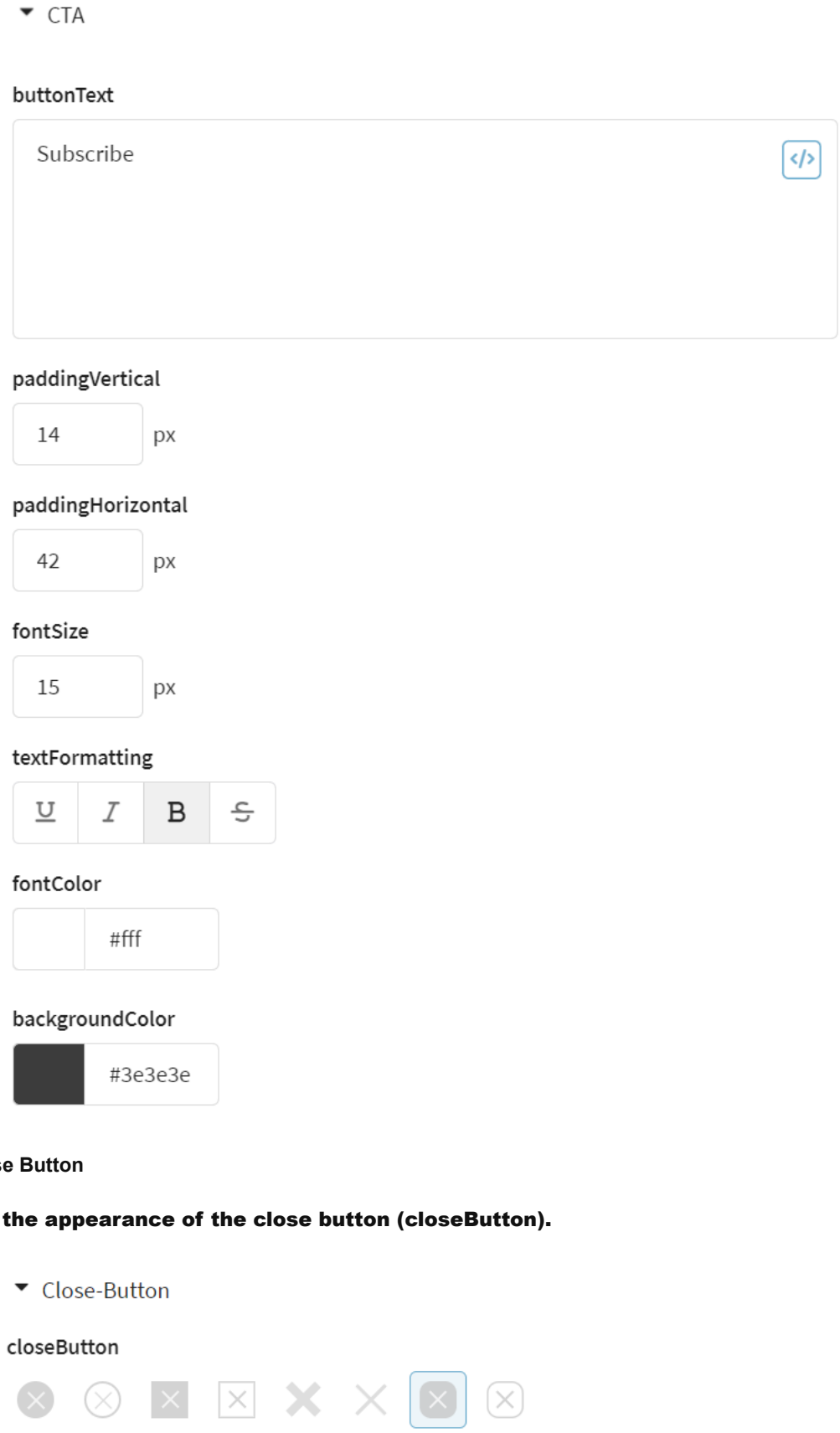
Note: only one form can be created in the visual editor. To add more, use the code editor.

Actions

Edit the submit button:

- **text — button text;**
- **textFormatting — formatting of the button text;**
- **borderRadius — rounding of the button corners. Radius can be set individually for each corner**

  ;
- **backgroundColor — background color of the button;**
- **hoverBackgroundColor — background color on hover;**
- **color — text color;**
- **fontSize — text size;**
- **paddingVertical — vertical padding inside the button;**
- **paddingHorizontal — horizontal padding inside the button.**



Close Button

Set the appearance of the close button (closeButton).

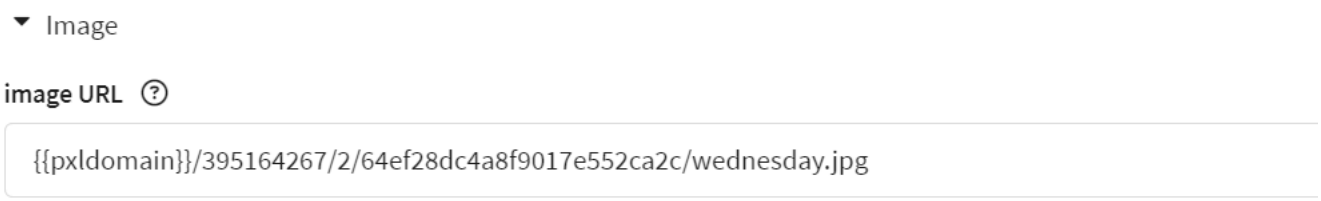
Image

Add an image (not all templates support it):

- **image URL — image URL or upload from the platform gallery**

;

- **width — width in px, height scales proportionally.**



Typography

Change text styles for all text elements:

Category	Parameter	Description
Global font	**fontFamily**	**Font family used for all text elements in the popup.**
Title	**titleAlign**	**Alignment of the title (left, center, right)**
	titleFontSize	**Font size of the title**
	titleFormatting	**Text formatting of the title (underline, italic, bold, strikethrough)**
	titleColor	**Color of the title text**
Main text	**textFontSize**	**Font size of the main text**
	textAlign	**Alignment of the main text (left, center, right)**
	textFormatting	**Text formatting of the main text (underline, italic, bold, strikethrough)**
	textColor	**Color of the main text**
Label	**labelFontSize**	**Font size of the label text**
	labelFormatting	**Text formatting of the label (underline, italic, bold, strikethrough)**
	labelColor	**Color of the label text**

Typography

fontFamily

Montserrat

titleAlign

titleFontSize

37 px

titleFormatting

titleColor

#646364

textAlign

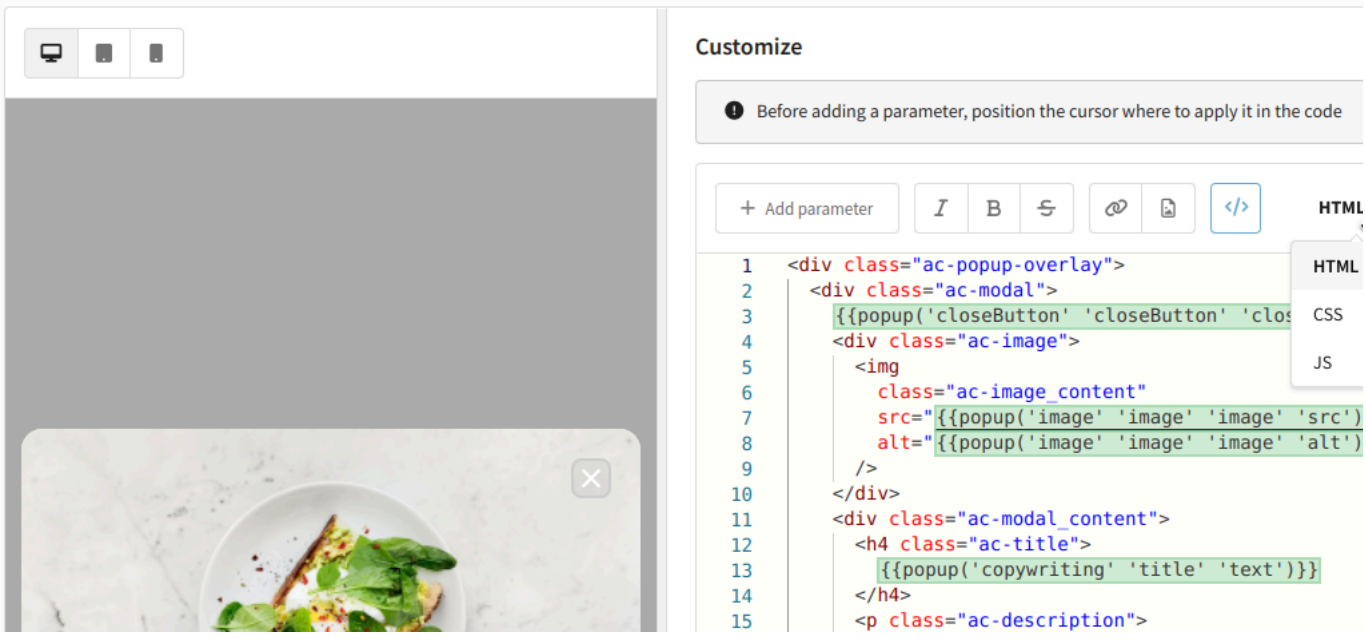
textFontSize

17 px

textFormatting

Code editor

Go to Settings to edit HTML/CSS/JS directly. See [this article](#).



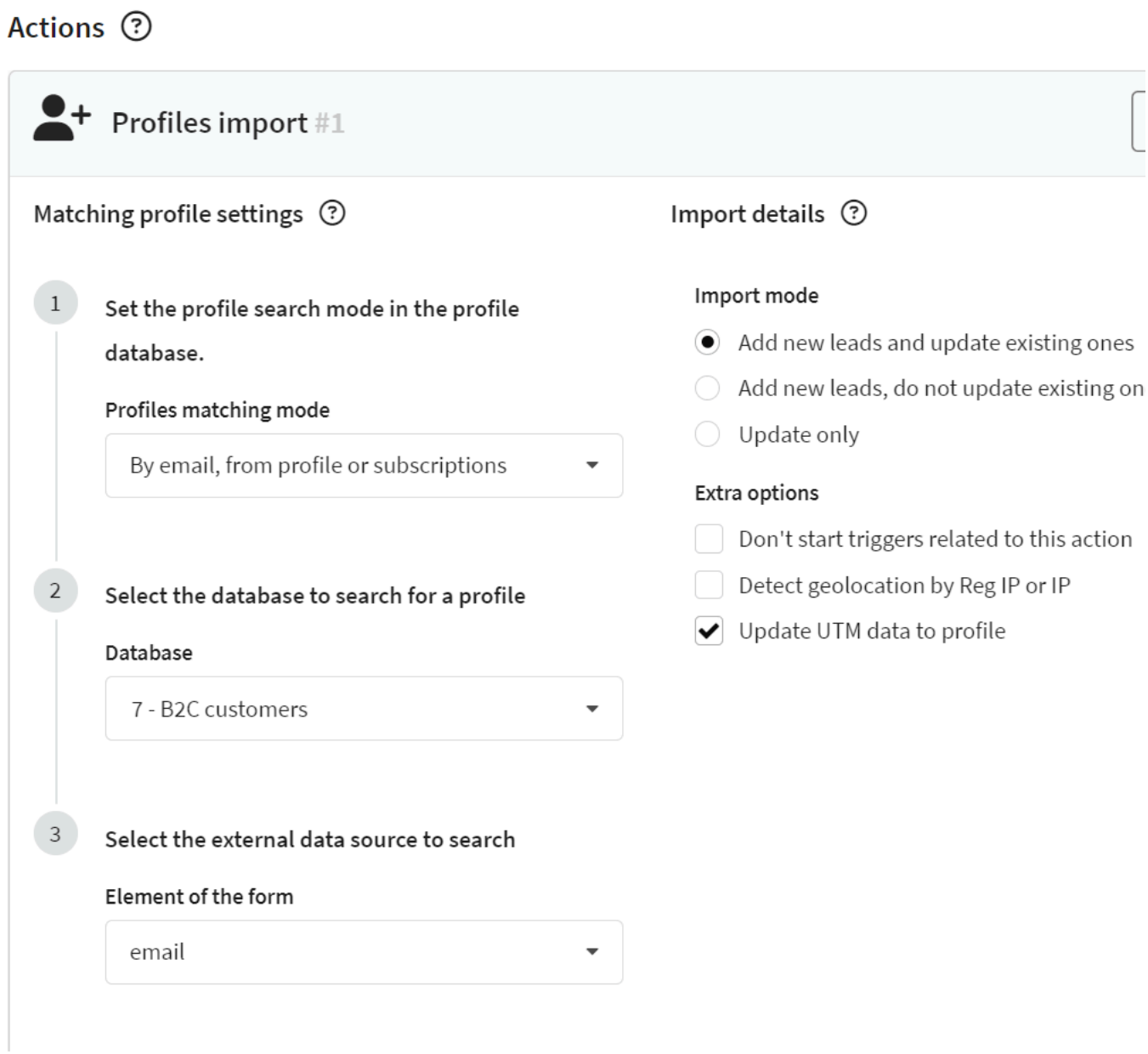
Configuring actions after form submission

Go to Actions. Define actions after the user submits the pop-up form:

- **Import profiles — save data into one or more databases.**
- **Send to email — send submission notifications.**

TIP

Post-submit actions for pop-ups are the same as for standard web forms. See this article.



Configuring publication and display

Two display mechanisms:

- **Manual script insertion — direct control, implement display logic yourself.**
- **Tag Manager — easier: place container code once, configure rules in UI.**

Manual script insertion

Save settings, set status Active, click Publish:

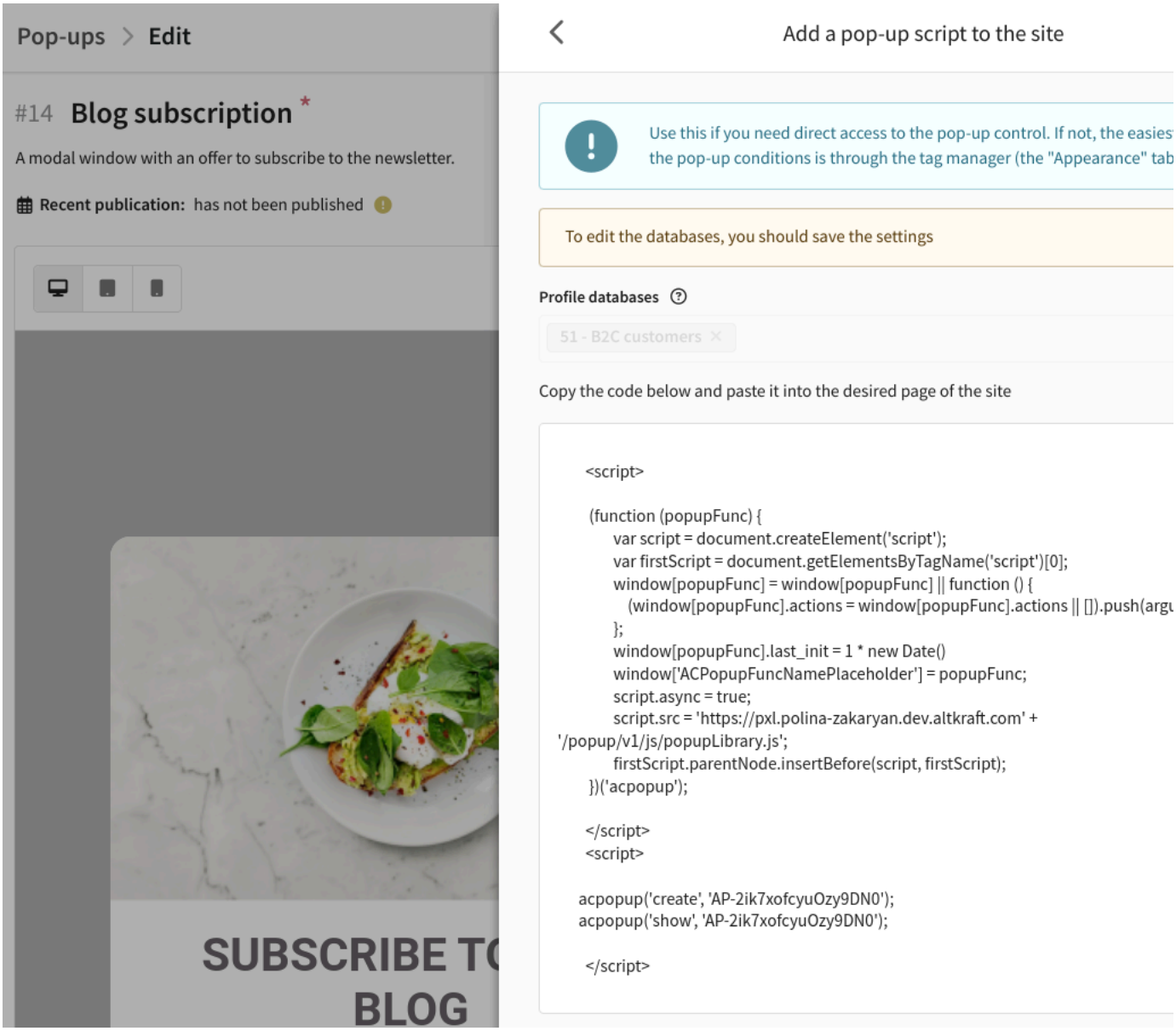
In additional actions, click

</>

. Select databases for profile search. Matching data can be passed with a server data request. If found, c profile is created.

> CAUTION
>
> **Databases here are only for searching existing profiles. Data from form submissions goes to the databa**

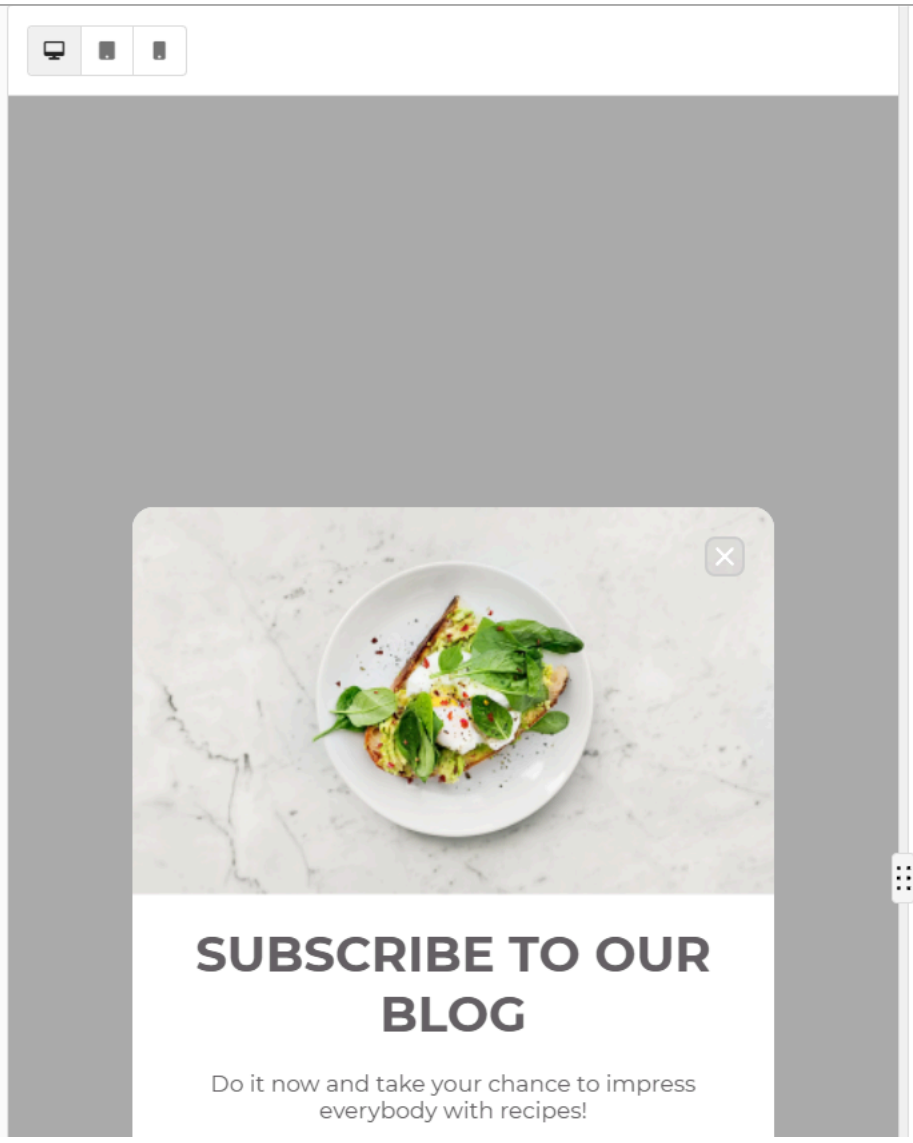
Click Show code then copy and pass to developers:



Tag Manager integration

Binding to a container

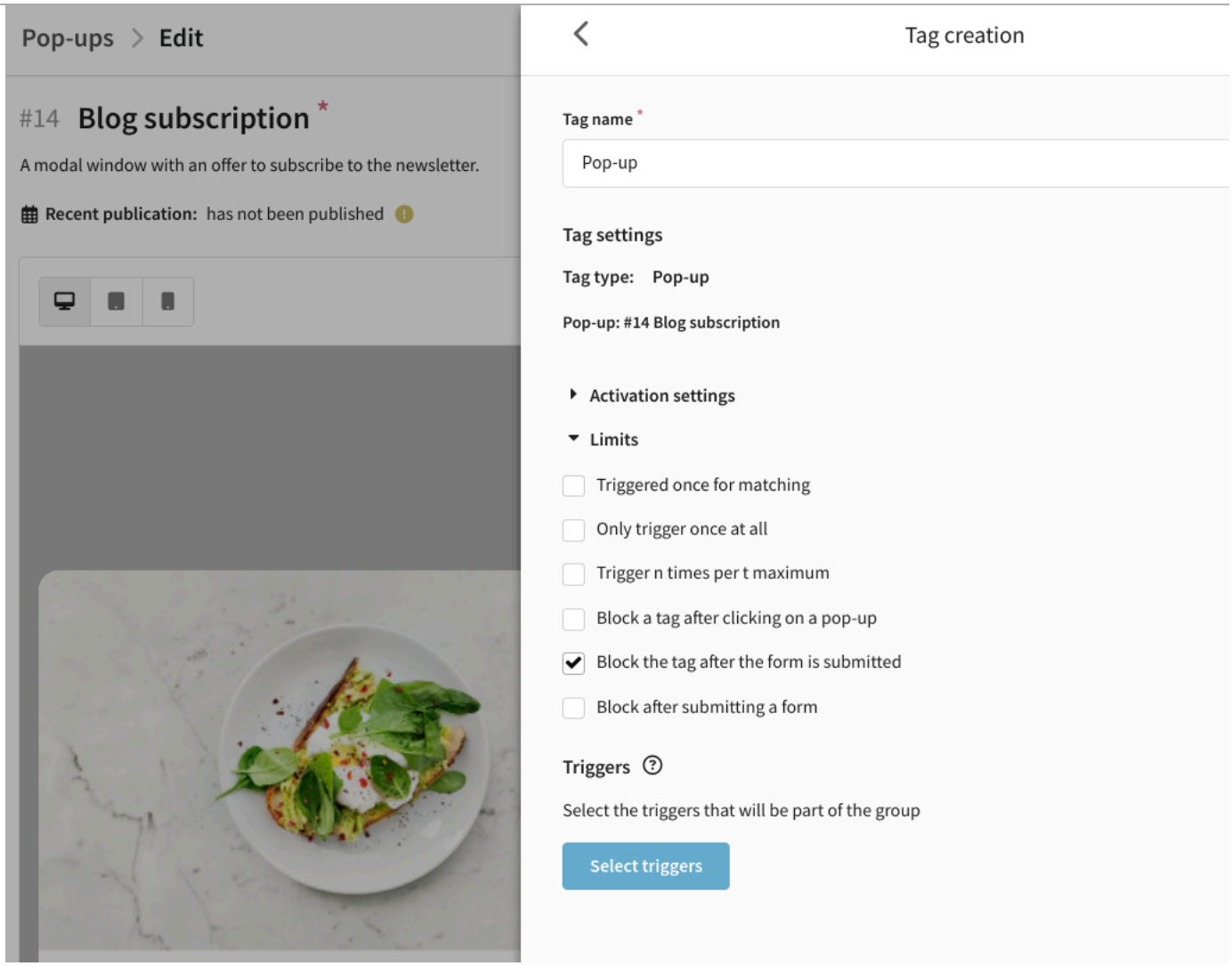
Save the pop-up. Go to Appearance then click Bind:



Select or create a container. Enter a tag name.

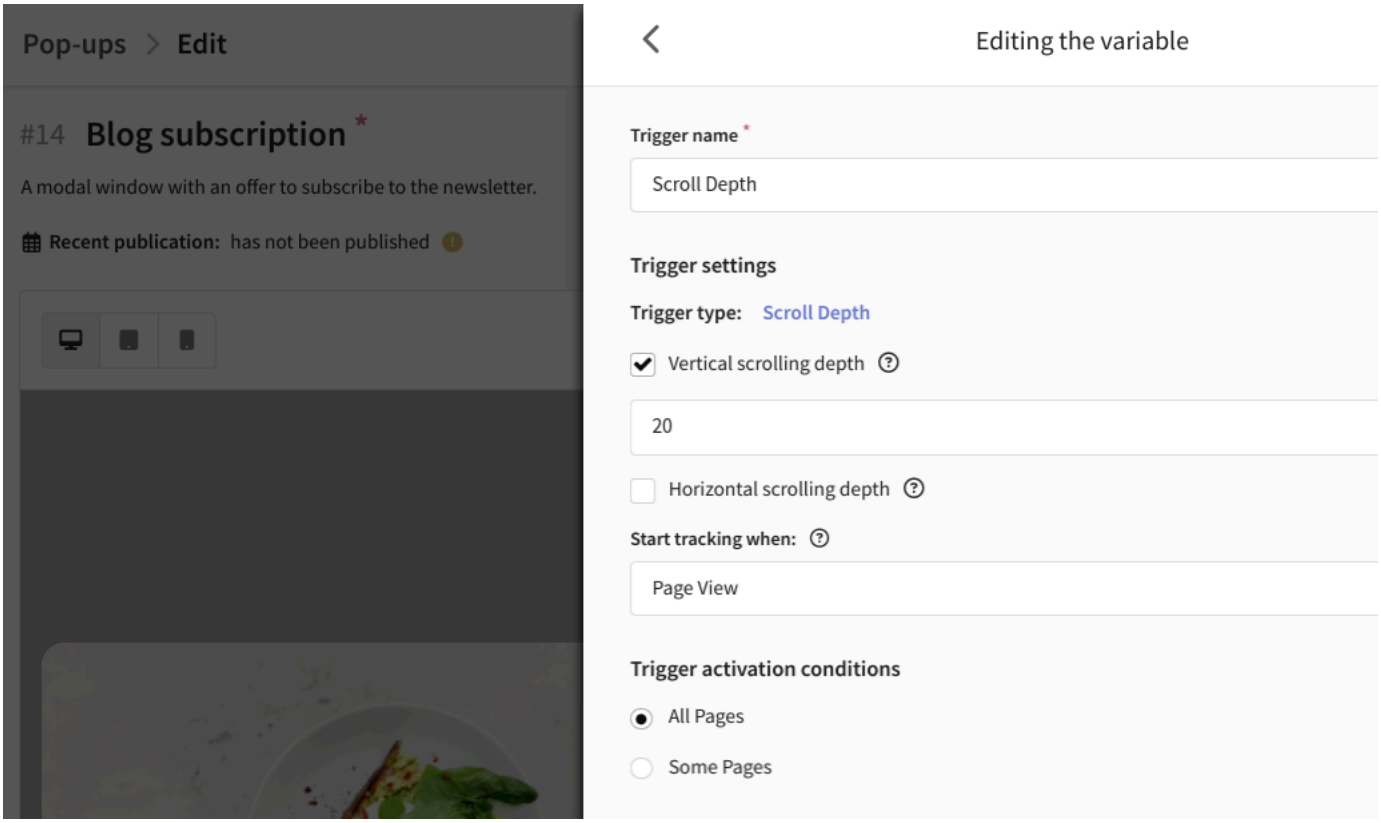
Tag activation settings

Set activation rules and limits:

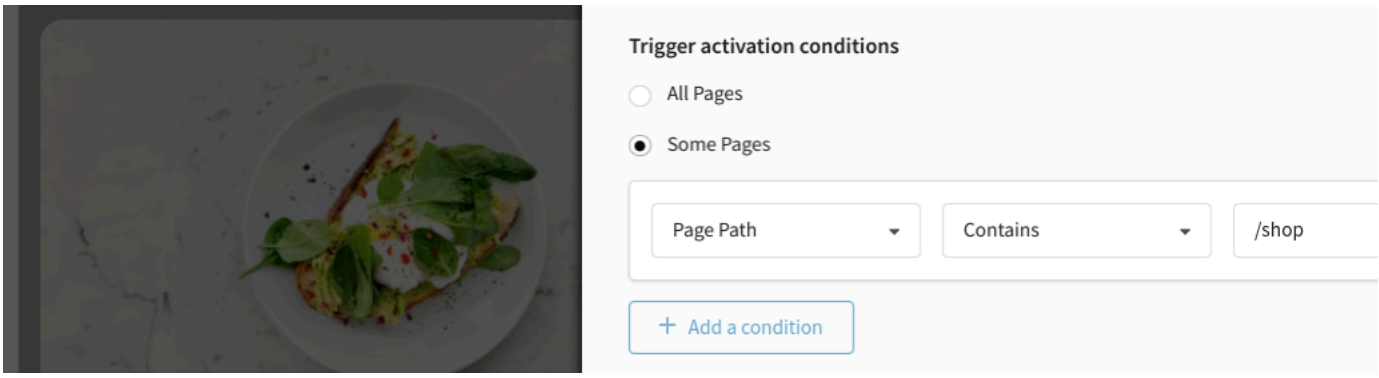


Configuring triggers

Select or create a trigger (page load, click, etc.):

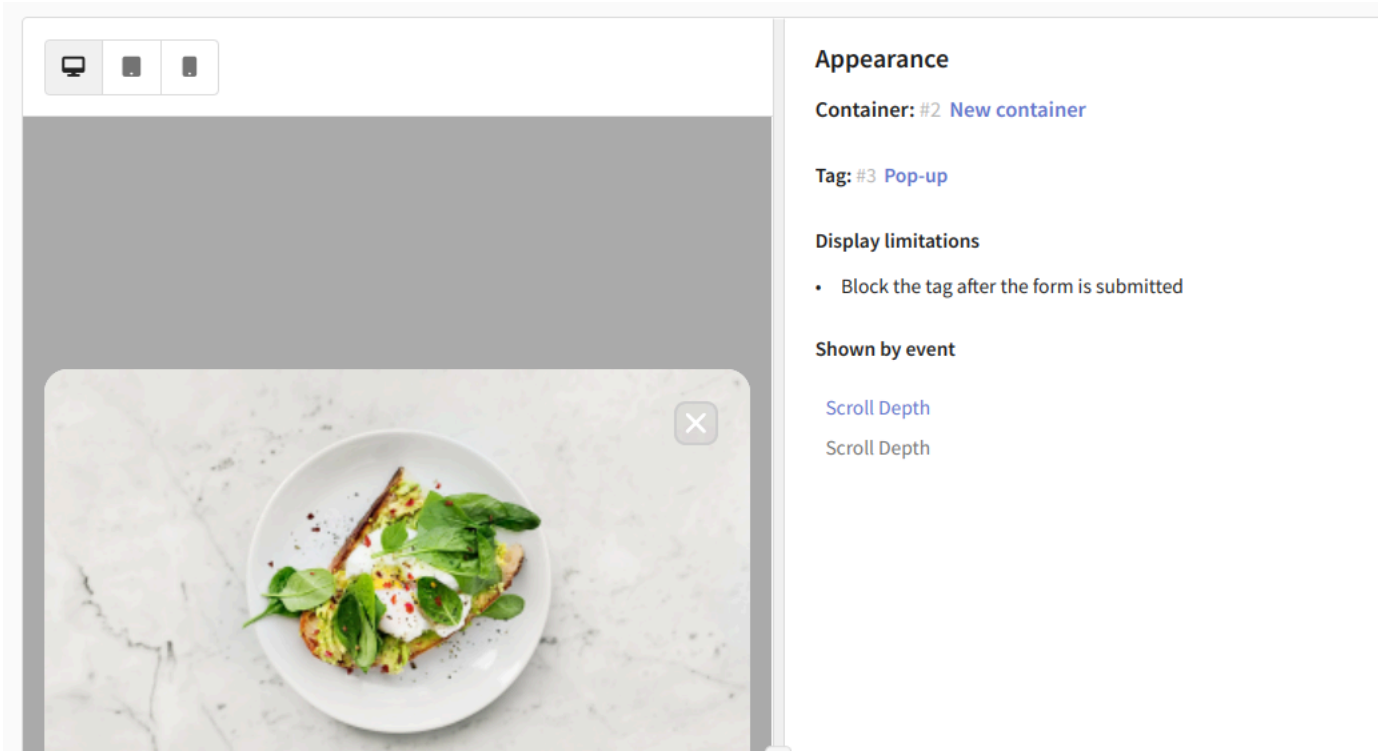


For detailed conditions, set a filter (Some events):

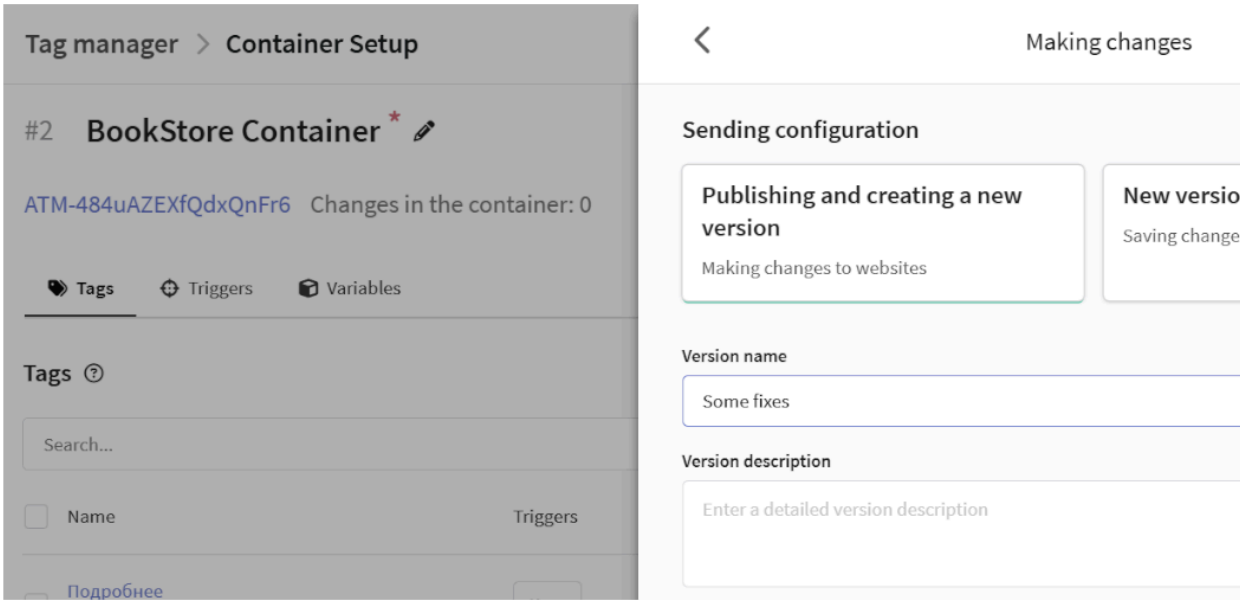


Publishing changes

Save the tag. It will appear in Appearance:



Save and publish the pop-up. Publish the new container version:



TIP

Learn more about containers, tags, triggers, and variables in this article.

Profile matching

You can identify a visitor manually with `acDataLayer.push()`. If a profile exists, its data will be updated.

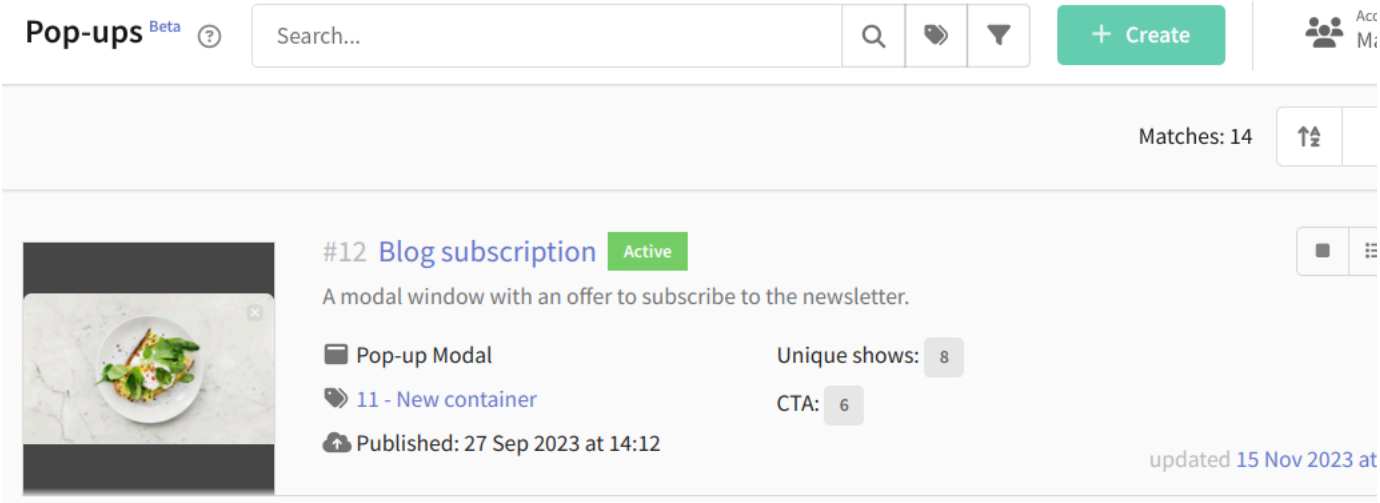
```
acDataLayer.push({
  matching: {
    matching_mode: "email",
    matching: { email: "test@example.com", db_ids: [1, 2] },
  },
});
```

- **matching_mode — search mode (“email”, “email_profile”, “phone”, “profile_id”, “custom”).**
- **matching — object with parameters for search and database IDs.**

Analytics and results

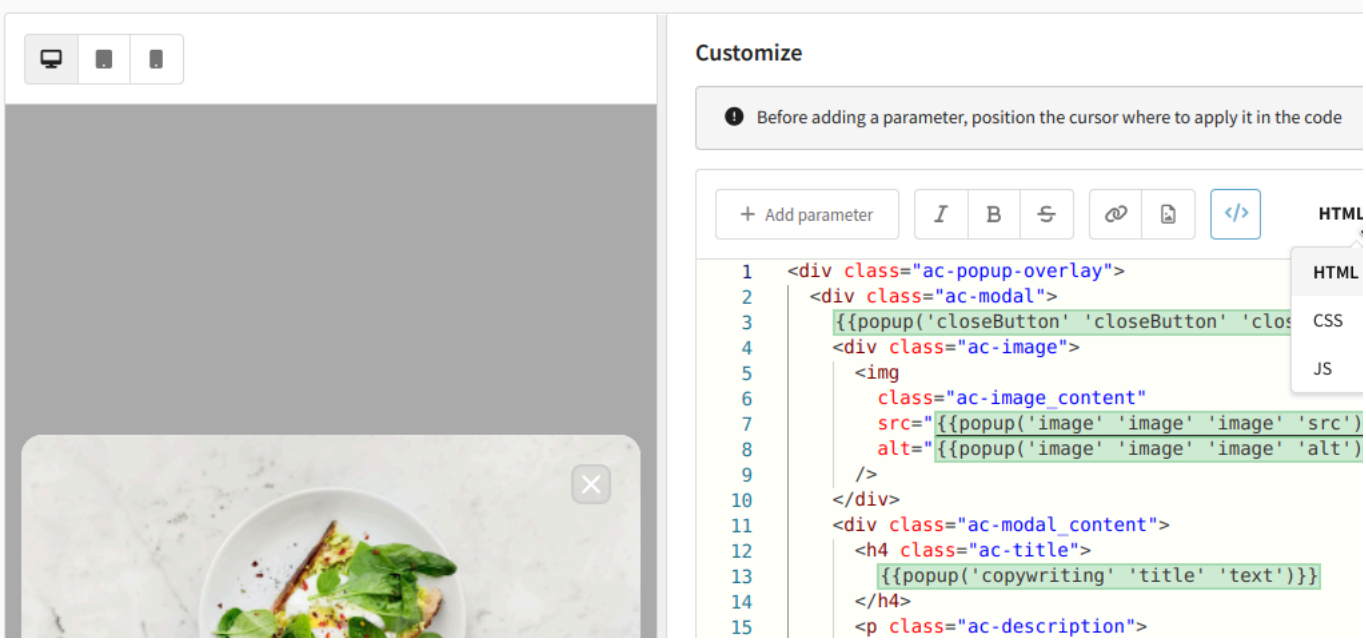
Analyze statistics in Analytics. Click the

icon in the editor or pop-up list:



Setting up a popup in the code editor

To work directly with the pop-up code, go to the Settings section. In the upper right corner you can swit JavaScript code:



To make it easier to work in the editor, use the following functions in the toolbar:

B

— bold text;

I

— *italic*;

~~S~~

— ~~strikethrough text~~;

— adding or editing links;

— adding images from the platform gallery;

</>

— the list of variables and functions of the editor. Using this button you can add dynamic content to the

INFO

You cannot add other HTML elements before the top-level `div` element, otherwise the template preview

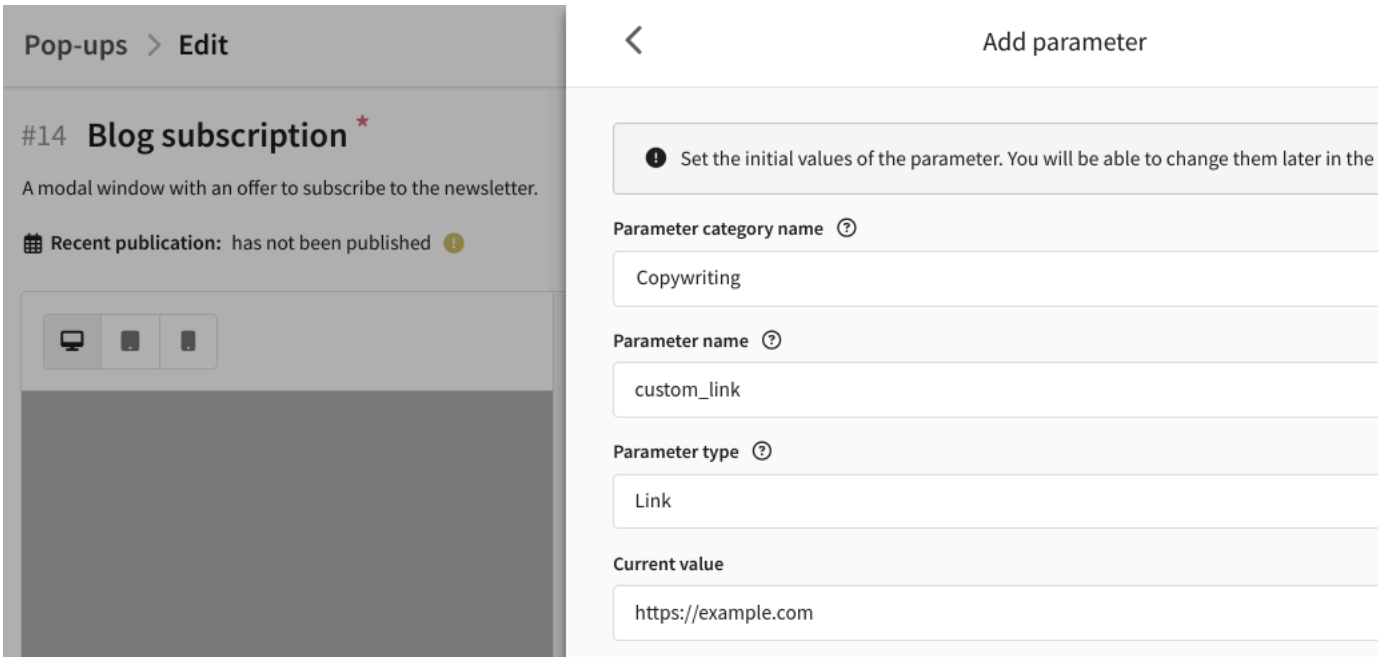
```
<em></em><div class="ac-popup-overlay">
    <div class="ac-modal">
        <div class="ac-image">
            <img
                class="ac-image_content"
                src="{{popup('image' 'image' 'image' '
                alt="{{popup('image' 'image' 'image' '
            />
        </div>
```

Creating a parameter

You can also create new settings in this section. When saved, they will appear on the Design tab. Thus, code will be able to change the values of custom parameters in the visual editor.

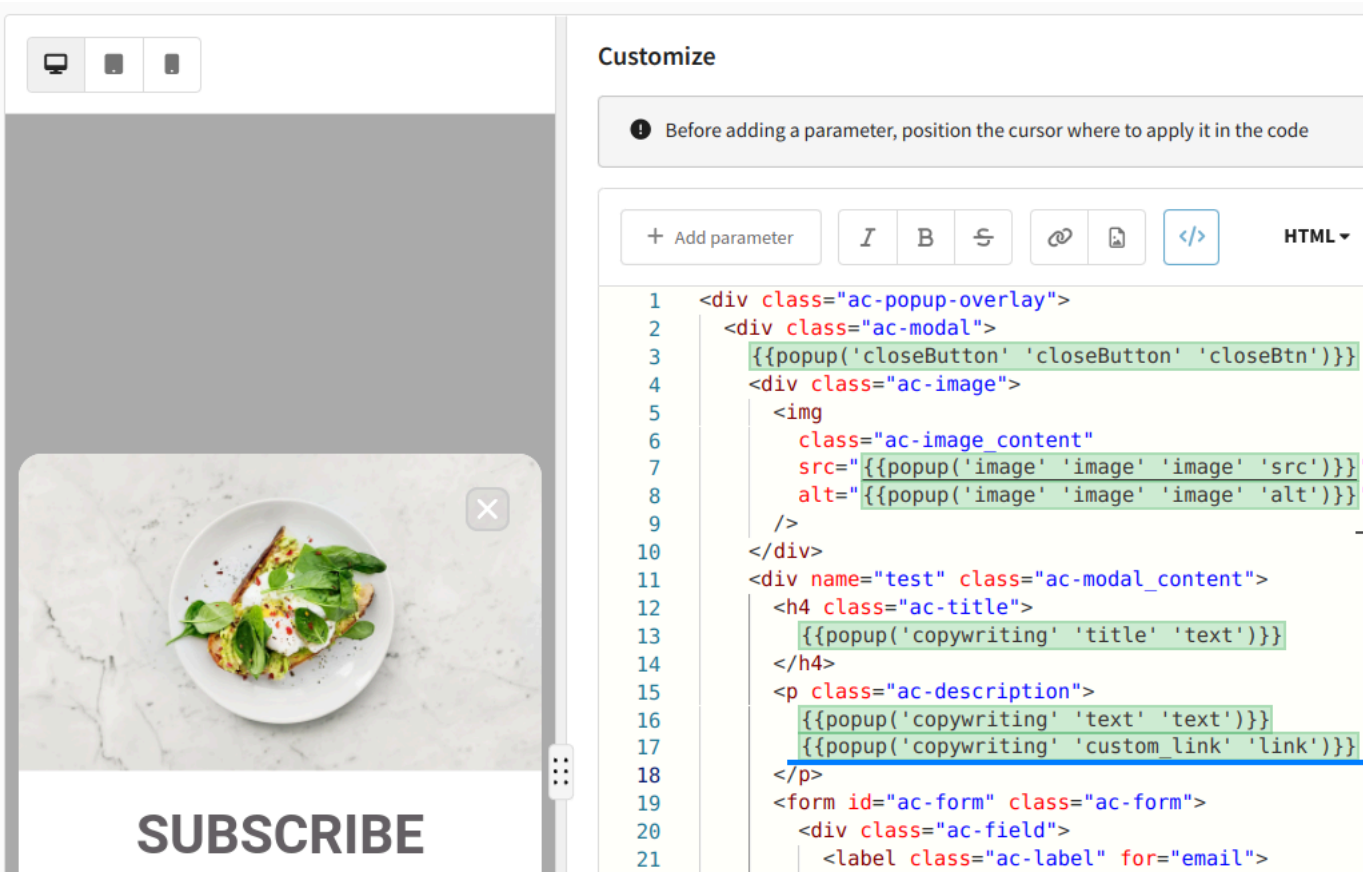
To create a new parameter, place the cursor in the editor line where you want to add the parameter. Clic upper left corner of the editor. In the new window select:

- **Parameter category name. Built-in parameters in the visual editor are grouped by category. You nee the custom parameter. You can also create your own category. To do this, enter the category name a**
- **Parameter name. Enter the name of the parameter to be used in the editor. The name must be uniqu**
- **Parameter type. Select the type of value that the parameter will contain (for example, text, numeric etc.).**
- **Current value. Specify the default value for the parameter. This field is optional.**

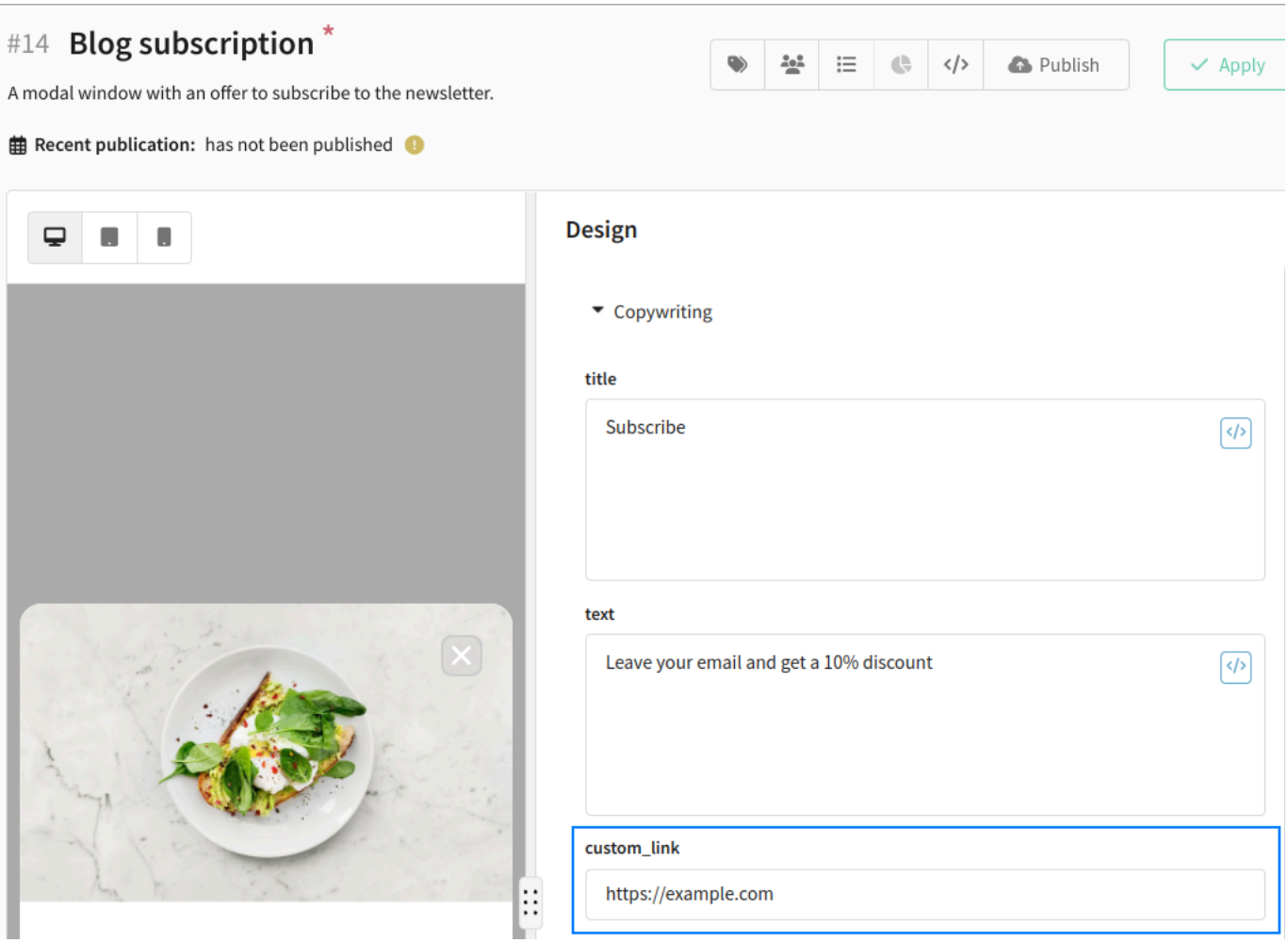


Once configured, click the Insert button at the bottom of the page. The parameter will be added in the fo

```
{{popup('<category_name>' '<parameter_name>' '<parameter_type>')}}
```



The created parameter will appear in the visual editor in the selected category. After saving the popup, t be edited by other platform users:



Popup elements settings

**When creating a pop-up, the platform offers many ready-made templates, but sometimes you need to mal
Below are recommendations on how to do this.**

INFO

The field name (`name` attribute) before and after the expression should not contain extra characters:
Correct usage: `name="{{popup('form' 'week-field' 'fieldData')}}"` **Incorrect usage:** `name="{{popup('form' 'week-fie`

Overlay layer

The overlay layer is designed to visually highlight the pop-up. Defined by an element with class `ac-popup-o`
translucent background and occupies the entire available screen area.

Usage example:

```
<div class="ac-popup-overlay">
    <!--Popup contents-->
</div>
```

Text content

By default, two types of text are available: heading and paragraph.

Usage example:

```
<!-- Title -->
<h4 class="ac-title">
    {{popup('copywriting' 'title' 'text')}}
</h4>

<!-- Paragraph -->
<div class="ac-description">
    {{popup('copywriting' 'text' 'text')}}
</div>
```

Images

There are two ways to add an image to a popup:

- **as an element with the** `<img>` **tag**

- **using the `background-image` CSS property**

An example of adding an image using the `<img>` tag:

```
<div class="ac-image">
     <img class="ac-image_content img"
         src="{{popup('image' 'image' 'image' 'src')}}"
         alt="{{popup('image' 'image' 'image' 'alt')}}"
     />
</div>
```

An example of adding an image using the `background-image` CSS property:

```
background-image: url("{{popup('background' 'backgroundImage' 'image' 'src')}}");
```

Form

For the popup to work correctly, all input fields and the form submit button must be placed in an element

Usage example:

```
<form id="ac-form" class="ac-form">
     <!-- Form content -->
</form>
```

Input fields

For correct display, each input field must be inside an element with the `ac-field` class. To add a title, it is with the `ac-label` class. Be sure to specify the `name` attribute, if it is not specified, the value of the input and will not be saved in the platform.

If you intend to use validation rules, you must add an element with the class `ac-validation-msg`.

Example of an email input field:

```
<div class="ac-field">
     <label class="ac-label" for="email">
         {{popup('copywriting' 'emailLabel' 'text')}}
     </label>
     <input
         id="email"
         type="email"
         class="ac-input"
         name="{{popup('form' 'email' 'fieldData')}}"
         placeholder="{{popup('copywriting' 'emailPlaceholder' 'text')}}"
     />
     <div class="ac-validation-msg" for="email"></div>
</div>
```

Example of a phone number input field:

```
<div class="ac-field">
    <label class="ac-label" for="phone">
        {{popup('copywriting' 'phoneLabel' 'text')}}
    </label>
    <input
        id="phone"
        type="tel"
        class="ac-input"
        name="{{popup('form' 'phone' 'fieldData')}}"
        placeholder="{{popup('copywriting' 'phonePlaceholder' 'text')}}"
    />
    <div class="ac-validation-msg" for="phone"></div>
</div>
```

> CAUTION
>
> **To correctly collect statistics for input fields, you should set the `type` attribute, the value of which will data.**

The following field types are available:

- **`<input type="text">` - basic text field as a string.**
- **`<input type="email">` - field for email address. Before sending, it is checked that the input value contai e-mail address.**
- **`<input type="tel">` - input field for a phone number; line breaks are automatically removed from the inp**
- **`<input type="number">` - input field for a float format number.**
- **`<input type="search">` - one-line text field for entering search strings.**
- **`<input type="url">` - field for URL editing. The entered value must contain either an empty string or a v**
- **`<input type="color">` - a color control element.**
- **`<input type="date">` - control element for entering date (year, month and day, no time).**
- **`<input type="datetime">` - control element to enter the date and time (hour, minute, second and fraction time zone.**
- **`<input type="datetime-local">` - control element for entering date and time without time zone.**

- `<input type="week">` - control element for entering the date containing the number week-year and week
- `<input type="month">` - control element for entering month and year without time zone.

Form submit button

To create a form submit button, it is recommended to use an element with class `ac-form_submit`, which co submit and class `ac-button`.

Usage example:

```
<div class="ac-form_submit">
     <button type="submit" class="ac-button">
         {{popup('cta' 'text' 'text')}}
     </button>
</div>
```

Close button

To create a button to close a popup, it is recommended to use the popup function, passing the appropria

Usage example:

```
{{popup('closeButton' 'closeButton' 'closeBtn')}}
```

The element has several different options, which can be viewed on the Design —> “Close” button tab.

Additional event tracking

To track events, you need to add the `data-tracking-event="[event-name]"` attribute to the html element with w `[event-name]` with the name of the event. The following events are supported:

- `click` - click on an element
- `close` — closing the popup

An example of registering clicks on a popup element:

```
<div data-tracking-event="click"></div>
```

> **TIP**
>
> **If the popup contains an HTML form element, then when the form submit button is clicked, the action o automatically registered. That is, in this case, to register an event, you do not need to add an additiona**

Managing pop-ups manually via script

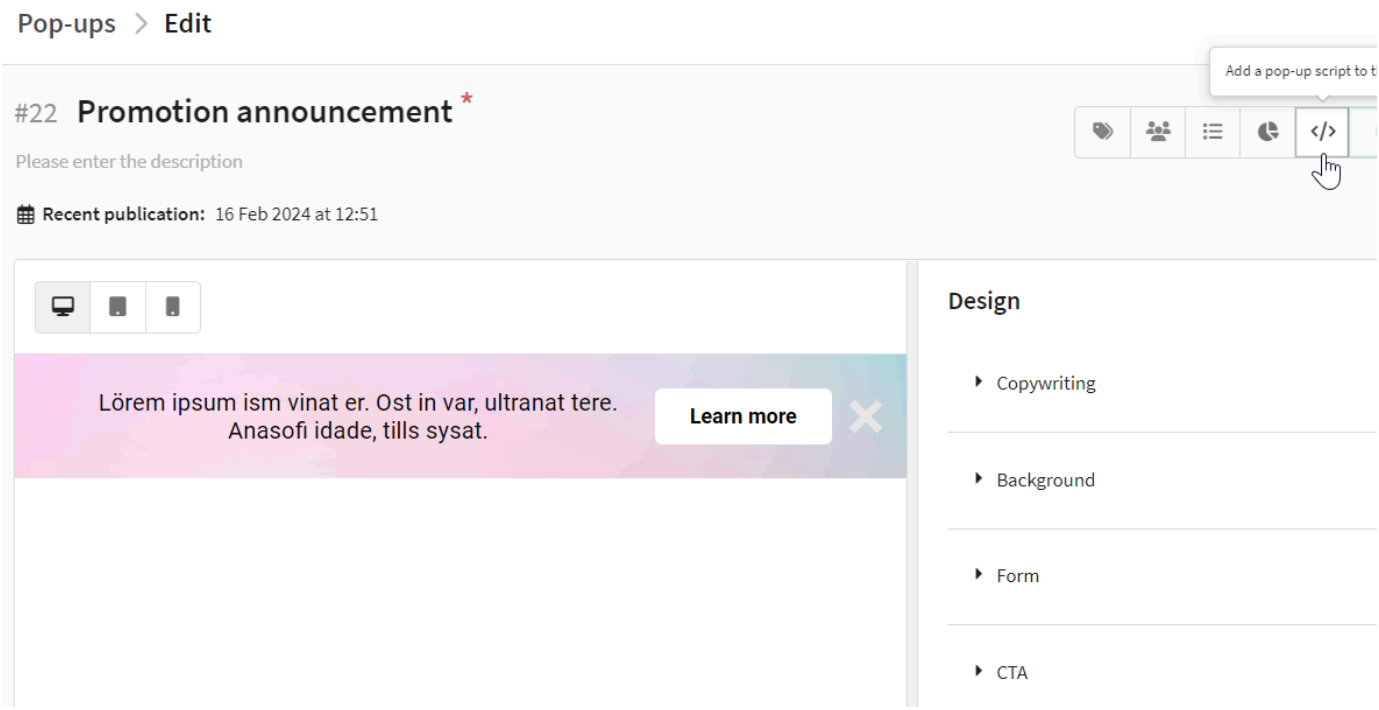
This guide is tailored for platform users who implement pop-ups on their site using scripts rather than th

Obtaining the script

To obtain the script, go to the pop-up editor and in the additional actions menu, click on the icon

</>

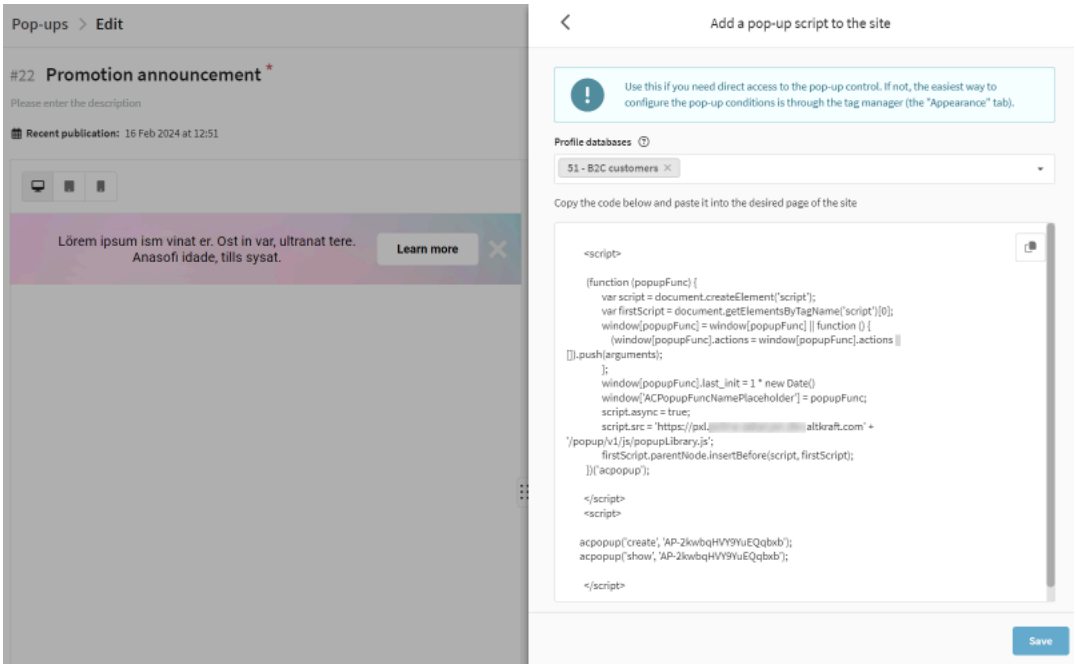
:



In the new window, select the databases for profile search. When a user fills out the pop-up form, their d platform checks if there is a profile with such data in one of the selected databases. If there is, the data not, a new profile is created in the database.

Note that here you are setting the databases to search for existing profiles. User data, filled out in the p database specified in the Actions section of the pop-up editor.

Click the Show code button. This code needs to be placed on the pages of the site where you want to sh configuring the pop-up appearance logic are provided below.



Managing a popup via the script

The following code is used to work with popups on the site:

```
<!-- Loading the script onto the page -->
<script>

  (function (popupFunc) {
        var script = document.createElement('script');
        var firstScript = document.getElementsByTagName('script')[0];
        window[popupFunc] = window[popupFunc] || function () {
            (window[popupFunc].actions = window[popupFunc].actions || []).push(arguments);
        };
        window[popupFunc].last_init = 1 * new Date()
        window['ACPopupFuncNamePlaceholder'] = popupFunc;
        script.async = true;
        script.src = 'https://pxl.altkraft.com' + '/popup/v1/js/popupLibrary.js';
        firstScript.parentNode.insertBefore(script, firstScript);
  })('acpopup');

</script>

<!-- Requesting pop-up data from the server + displaying the pop-up on the page -->
<script>

  acpopup('create', '<popup_id>');
  acpopup('show', '<popup_id>');

</script>
```

If you add this code to your site, the popup will be shown as soon as the page loads because of this scri

```
<script>
  acpopup('create', '<popup_id>');
  acpopup('show', '<popup_id>');
</script>
```

You'll need to set up how and when the pop-up appears.

Example 1. Pop-up appears 10 seconds after the page loads:

```
<script>
    window.onload = function() {
        setTimeout(function() {
            // Запрос данных попапа с сервера
            acpopup('create', '<popup_id>');

            // Показ попапа
            acpopup('show', '<popup_id>');
        }, 10000);
    };
</script>
```

Example 2. Pop-up appears when a button is clicked:

```
<button onclick="showPopupClick()">Show the pop-up</button>

<script>
    function showPopupClick() {
        acpopup('create', '<popup_id>');
        acpopup('show', '<popup_id>');
    }
</script>
```

To manage the pop-up, use the `acpopup` function. It has 3 modes:

- **create**
- **show**
- **sendEvent**

create — request pop-up data from the server

In the `acpopup` function you must pass the pop-up ID as the second argument. As the third argument you c about matching* and *SMID*. The table describes the structure of this object:

Parameter	Type	Required	Description
`matching_mode`	**string**	**No**	**Profile search mode. It can be `"email"`, `"email_profile"`, `"phone"`, `"profile_id"` or `"custom"`.**
`matching`	**JSON object**	**No**	**An object with one or two parameters: `"search field": "field value"` and `"db_id": "database ID"` — the array of the databases in which the pr database is selected in the pop-up settings.**
`smid`	**string**	**No**	**Send message ID**

Example:

```
acpopup('create', 'AP-6eVbo2dP2Nnh0wtBp', {
    matching_mode: 'email',
    matching: {
        email: 'john@example.com',
        db_ids: [1]
    },
    smid: 'send_message_id',
})
```

show — show a pop-up to users

In the `acpopup` function you must pass the pop-up ID as the second argument. The third argument can be will be called after the popup appears on the screen.

Example:

```
acpopup('show', 'AP-6eVbo2dP2Nnh0wtBp', {
  onShow: function () {
      console.log('Look at the screen')
  },
})
```

sendEvent — send a pop-up custom event

In the `acpopup` function you must pass the pop-up ID as the second argument. The third argument must b You can also pass matching data for a specific event and SMID to this object. The table describes the str

Parameter	Type	Required	Description
`web_event_name`	**string**	**Yes**	**Event name**
`web_event_value`	**int**	**No**	**Numeric value of the event**
`web_event_data`	**JSON object**	**No**	**Object with optional data for the event**
`matching_mode`	**string**	**No**	**Profile search mode. It can be `"email"`, `"email_profile"`, `"phone"`, `"profile_id"` or `"custom"`.**
`matching`	**JSON object**	**No**	**An object with one or two parameters: `"поле для поиска": "значение поля"` and `"db_id": "ID базы"` — databases in which to search for a prof in the pop-up settings.**
`smid`	**string**	**No**	**Send message ID**

Example:

```
acpopup('sendEvent', 'AP-6eVbo2dP2Nnh0wtBp', {
    web_event_name: 'new_event',
    web_event_value: 500,
    web_event_data: {
        status: 'active'
    },
    matching_mode: 'email',
    matching: {
        email: 'john@example.com',
        db_ids: [1]
    },
    smid: 'send_message_id',
})
```

TIP

If you’re adding multiple pop-ups to one page, you only need to place the coode for loading script onto t manage them using the `acpopup` function.

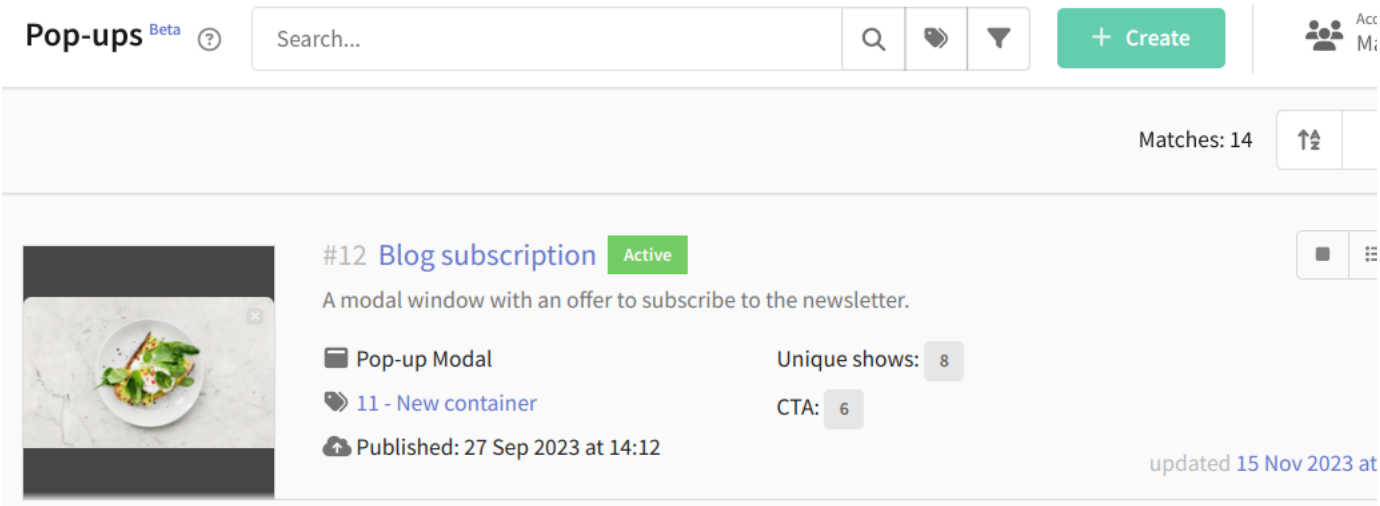
Popup analytics

Description

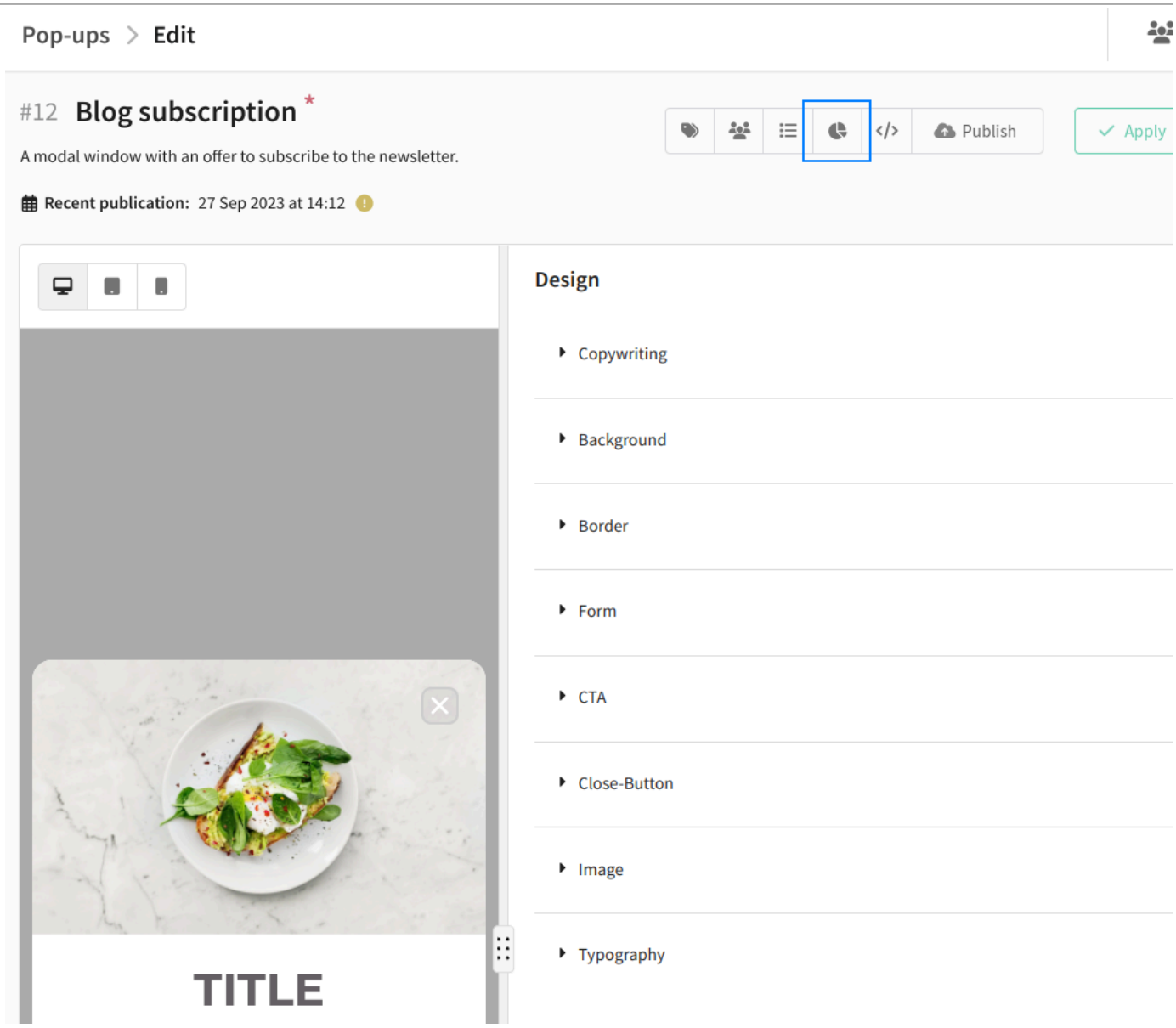
Analytical tools allow you to view key popup metrics (hits to targeting, shows, form submissions), as wel presented in the form of a graph and a funnel. You can group any metric based on various criteria, such a tags.

To go to Analytics, find the popup you need in the general list and click on the icon

:



You can also go to Analytics directly from the popup editing window:



There are two tabs available in analytics:

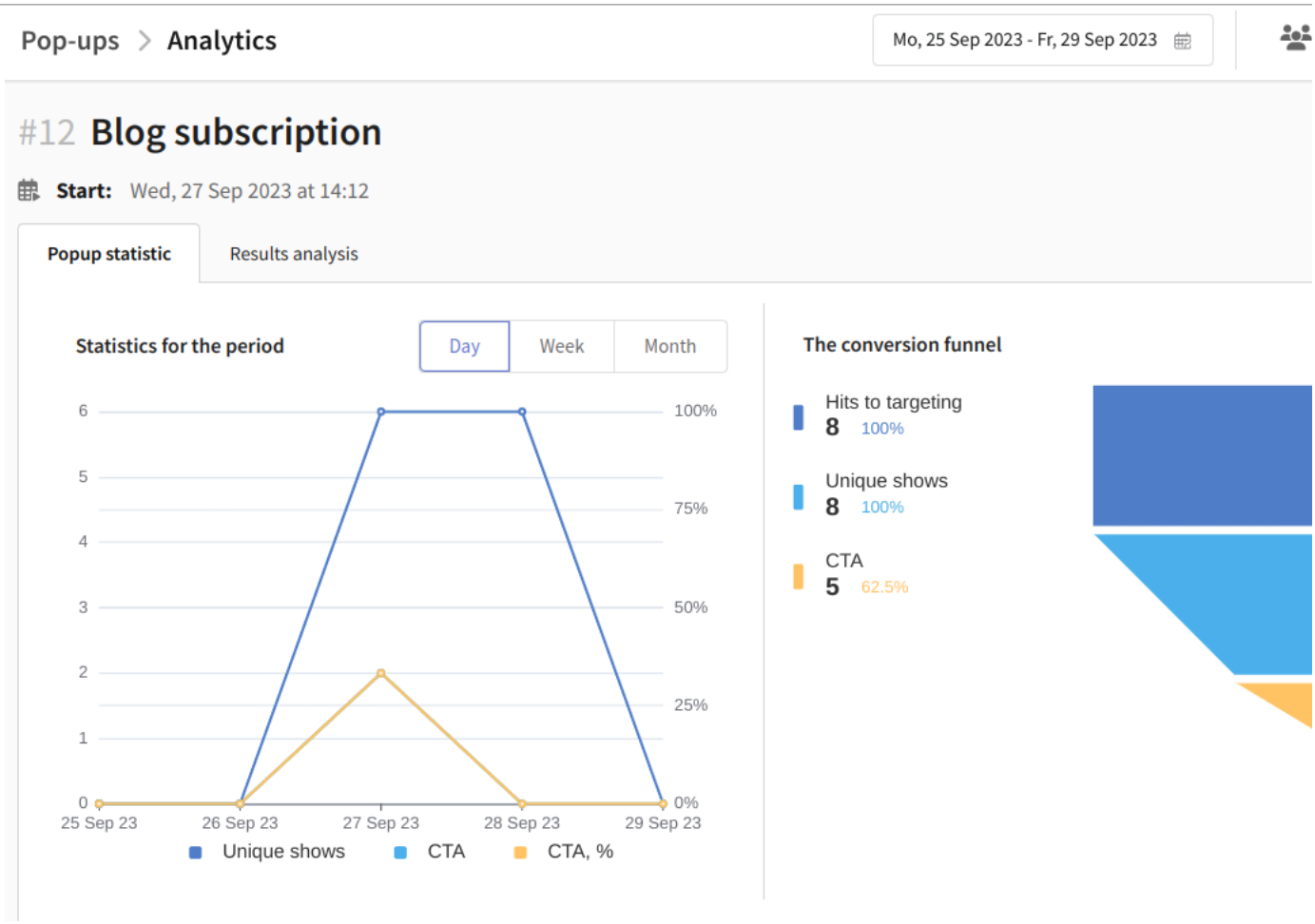
- **Popup statistics — general summary of the popup.**
- **Analysis of results — statistics on filling out fields in the form.**

Popup statistics

This tab contains information about popup impressions and form fillings. On the top panel you can specif receive data.

Popup statistics are divided into three blocks:

1. **Statistics for the period**
2. **Conversion funnel**
3. **Analytical slices**



Statistics for the period

The graph displays the dynamics of performance metrics over the selected period:

- **Unique shows — the number of times the popup is displayed to users. If the popup is shown to a use only one unique show is recorded.**
- **CTA — until a form submission event is registered, this metric reflects clicks on any form element. O submitted at least once, the metric reflects the number of submissions.**
- **CTA, % — (Unique shows / CTA) * 100%.**

Data can be grouped by days, weeks, or months. If you want to hide a metric from the graph, click on its

Conversion funnel

The funnel is a sequence of steps through which a group of website visitors passes. This group is define to targeting. The report then shows how many users moved to the next stage. With each step, the numbe

- **Hits to targeting — the number of times the conditions for the appearance of the popup, set in the tri might not have been shown to the user due to tag limits specified in the “Activation Settings” and “Li**
- **Unique shows — the number of times the popup is displayed to users.**
- **CTA — until a form submission event is registered, this metric reflects clicks on any form element. O submitted at least once, the metric reflects the number of submissions.**

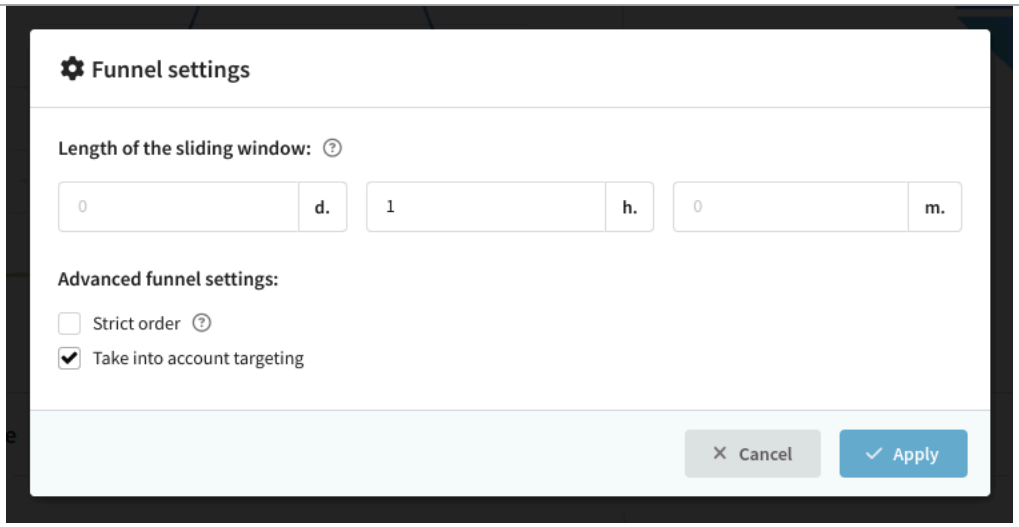
TIP

Events in the funnel are unique. For example, if a popup was shown to a user multiple times within one be counted.

Additionally, you can customize the funnel to suit your tasks. To open the settings window, click

⚙

in the top right corner.



Sliding window width is the time period between the first and last conditions in the funnel. Default is 1 h the moment you hit the target until the CTA event is registered, then the last step of the funnel will not b

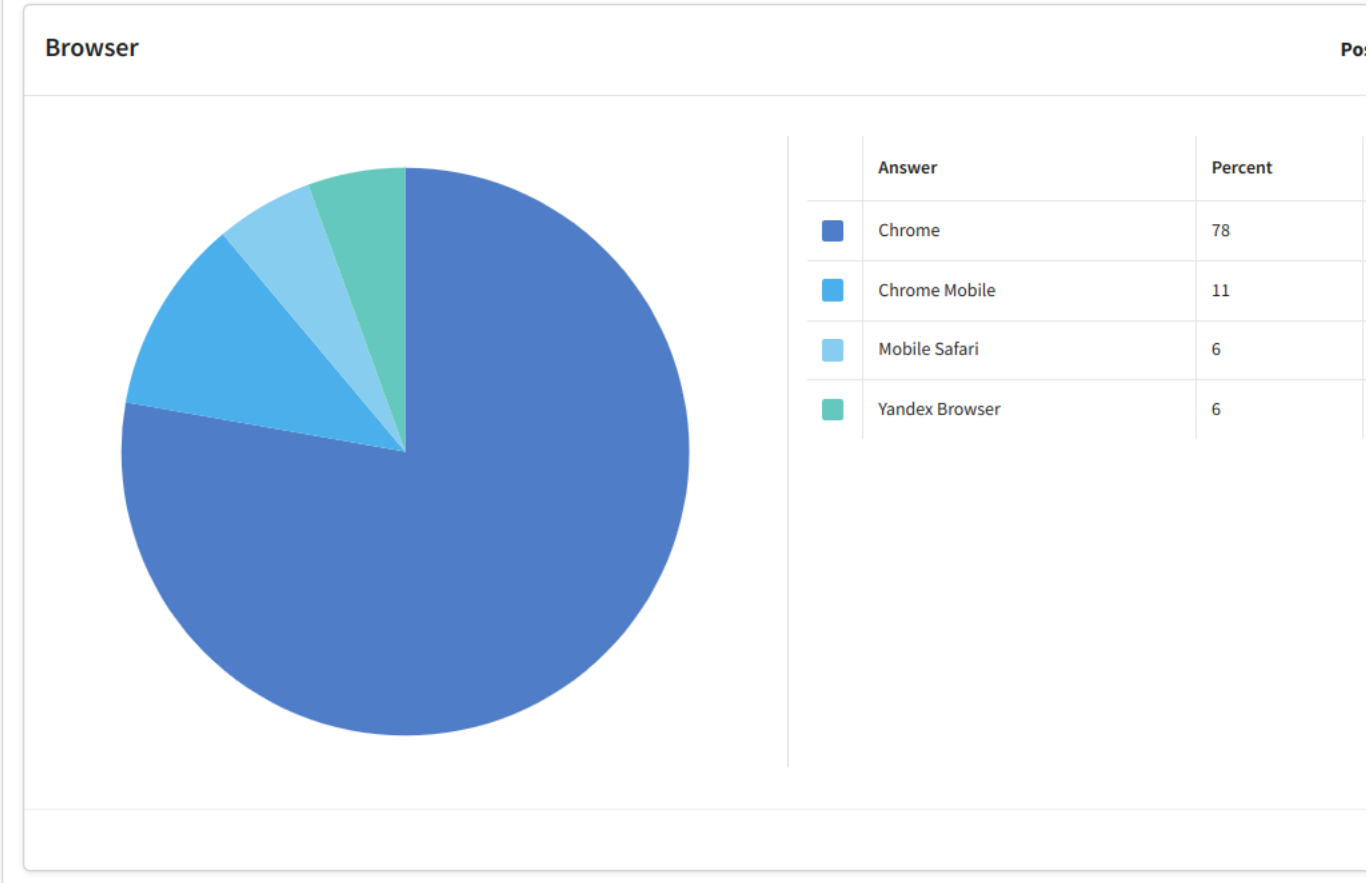
Strict order prohibits extraneous events in the desired sequence. For example, if in a funnel with the seq *display —> CTA* between the first and second step one more event will be recorded, then all subsequent case, the event that violated the order will also not be counted.

Take into account hitting the target — by unchecking the checkbox, you will remove hitting the target as

Analytical slices of the report

Below are pop-up indicators grouped by the following criteria:

- **By time zone**
- **By city**
- **By country**
- **By language (browser language)**
- **By OS (operating system on the user's device)**
- **By browser**
- **By referrer (URL of the page on which the popup was shown)**
- **By UTM tags (content, medium, source, mailing, term)**



For any slice, you can select a specific indicator:

- **CTA — while the form submit event is not registered, the metric displays clicks on any form element. submitted at least once, the indicator will change to “Completions”.**
- **Fillings — the user filled out and submitted the popup form.**
- **Impressions — the popup is shown to the user.**
- **Unique impressions — the popup is shown to the user. If a popup was shown to a user several times impression will be counted.**
- **Close — click on the close button in the popup.**

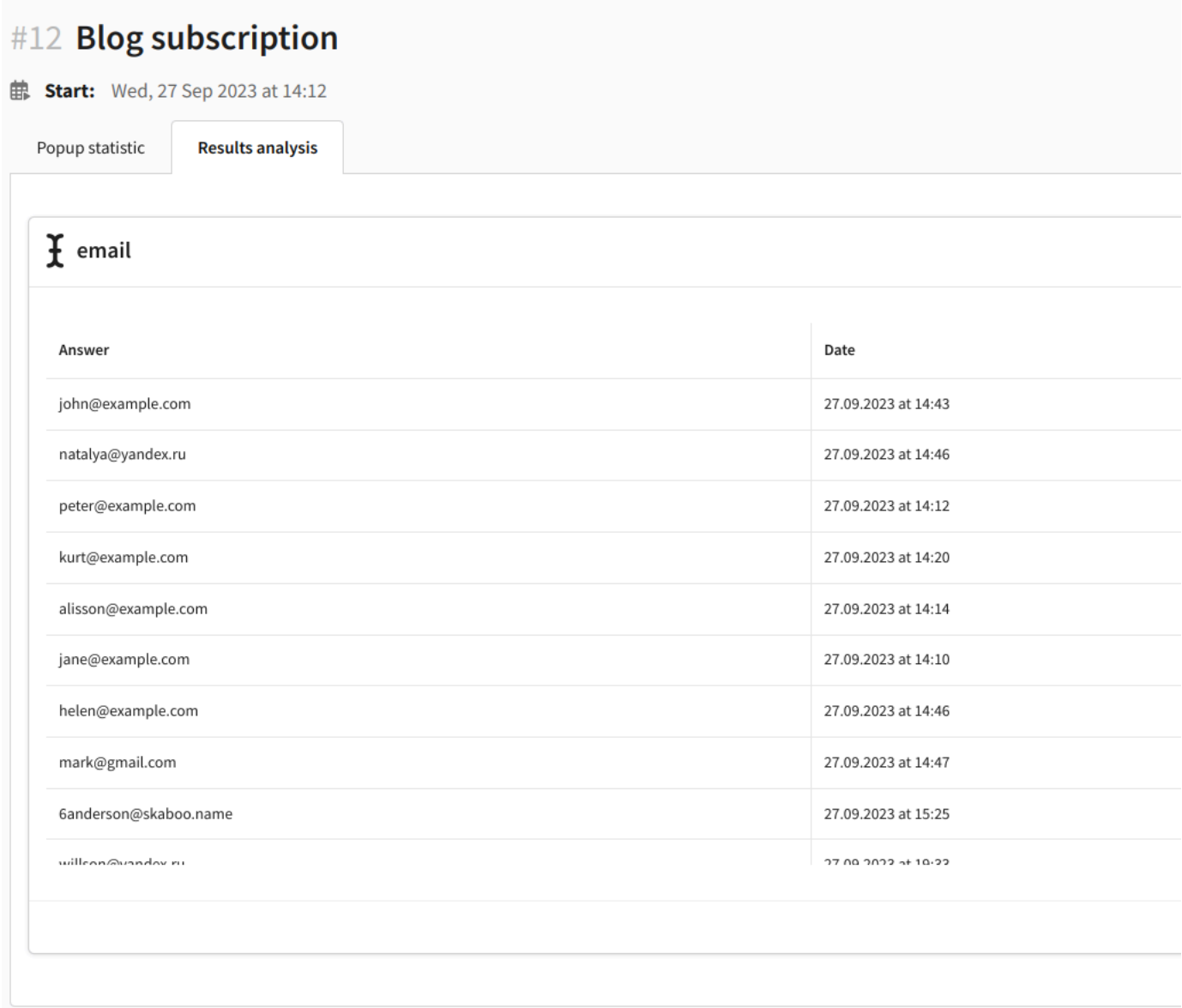
In the table on the right, the data is presented in percentage and quantity. The bottom of the block show selected event was registered. Additionally, you can change the way the data is displayed:

— pie chart

— bar chart

Analysis of results

On this tab you can get information about filling out form fields in the popup:



Statistics are generated depending on the type of input field specified in the popup code: `<input type="<fi` correctly, you should set the `type` attribute, the value of which will correspond to the type of input data. in [this article](#).

Guide: pop-up for push subscriptions

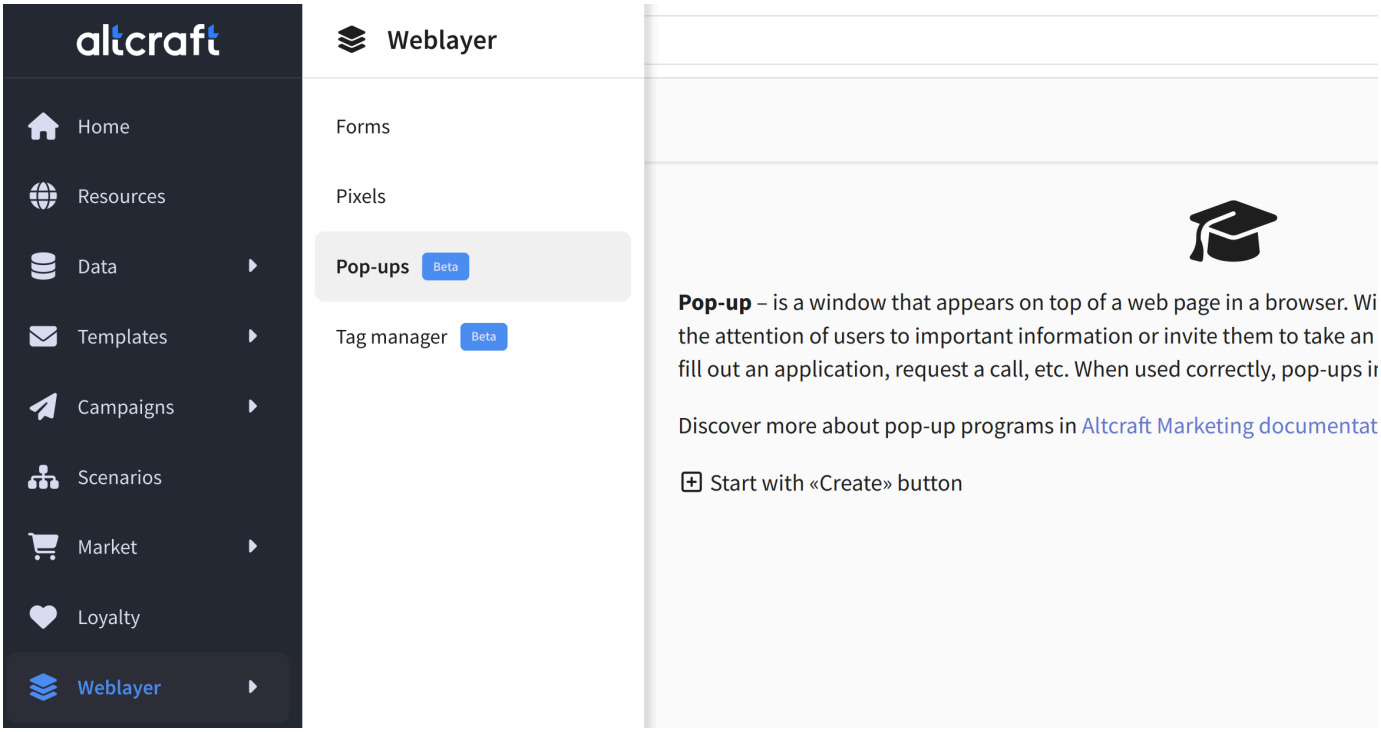
One of the most common types of pop-ups is the push notification subscription pop-up. This kind of pop-u your website visitors to the Altcraft platform resource. In this article, we will show you how to create an on page load for those users who have not yet given permission to display push notifications on your site

INFO

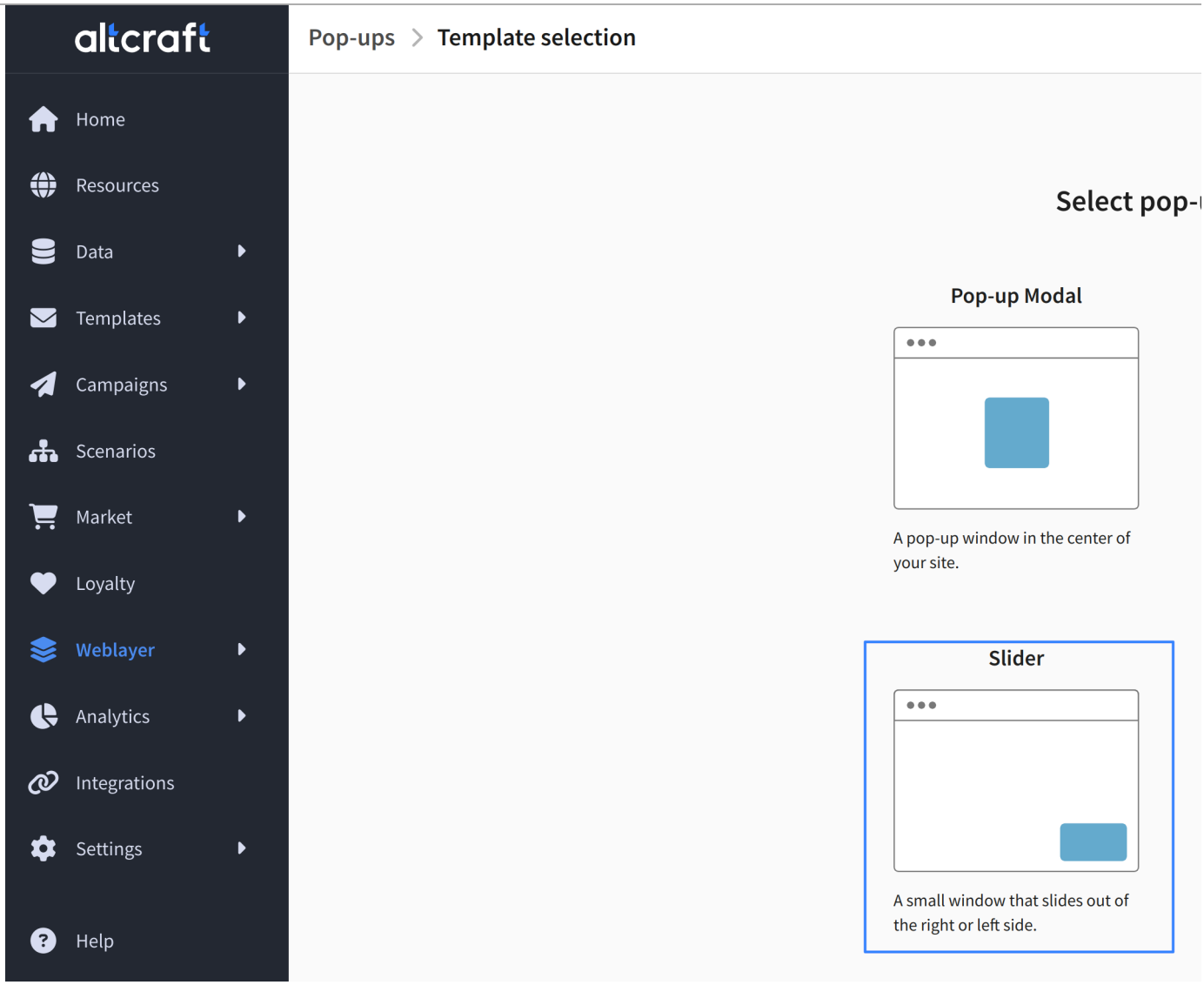
For push notifications subscription to work correctly, you must first set up a resource on the platform, a script to your site. You can learn more about setting up push notifications there.

Creating and publishing a pop-up

Go to Weblayer — Pop-ups. Create a new pop-up:



Select the Slider type:



Select pop-up template. Then go to the “Customize” section of the menu that opens on the right side of t code there:

```
<div class="ac-slider">
  <div class="ac-slider_content">
    <h4 class="ac-title">
      {{popup('copywriting' 'title' 'text')}}
    </h4>
    <p class="ac-description">
      {{popup('copywriting' 'text' 'text')}}
    </p>
    <div class="buttons">
        <button type="button" id="btn_sub" class="ac-button primary">
          {{popup('cta' 'SubscribeText' 'text')}}
        </button>
        <button type="button" id="btn_cancel" class="ac-button secondary" data-tracking-event="close">
            {{popup('cta' 'CancelText' 'text')}}
        </button>
    </div>
  </div>
</div>
```

As a result, the body of the pop-up should look like this:

Customize

Before adding a parameter, position the cursor where to apply it in the code

+ Add parameter

```
<div class="ac-slider">
  <div class="ac-slider_content">
    <h4 class="ac-title">
      {{popup('copywriting' 'title' 'text')}}
    </h4>
    <p class="ac-description">
      {{popup('copywriting' 'text' 'text')}}
    </p>
    <div class="buttons">
      <button type="button" id="btn_sub" class="ac-button pri
        {{popup('cta' 'SubscribeText' 'text')}}
      </button>
      <button type="button" id="btn_cancel" class="ac-button
        {{popup('cta' 'CancelText' 'text')}}
      </button>
    </div>
  </div>
</div>

```

Switch to the CSC styles tab and add the following styles at the end of the code for button placement:

```
.buttons {
    display: flex;
    gap: 8px;
}

.buttons .ac-button {
    flex: 1;
}
```

This will place the buttons side by side and stretch them to the same width.

Change the format to JS and add code that will initialize the push subscription when the subscribe butto

```
const popupContainer = document.querySelector('[data-ac-popup-container]');

if (popupContainer) {
    const popupHost = popupContainer.shadowRoot || document;
    const subscribeBtn = popupHost.getElementById('btn_sub');

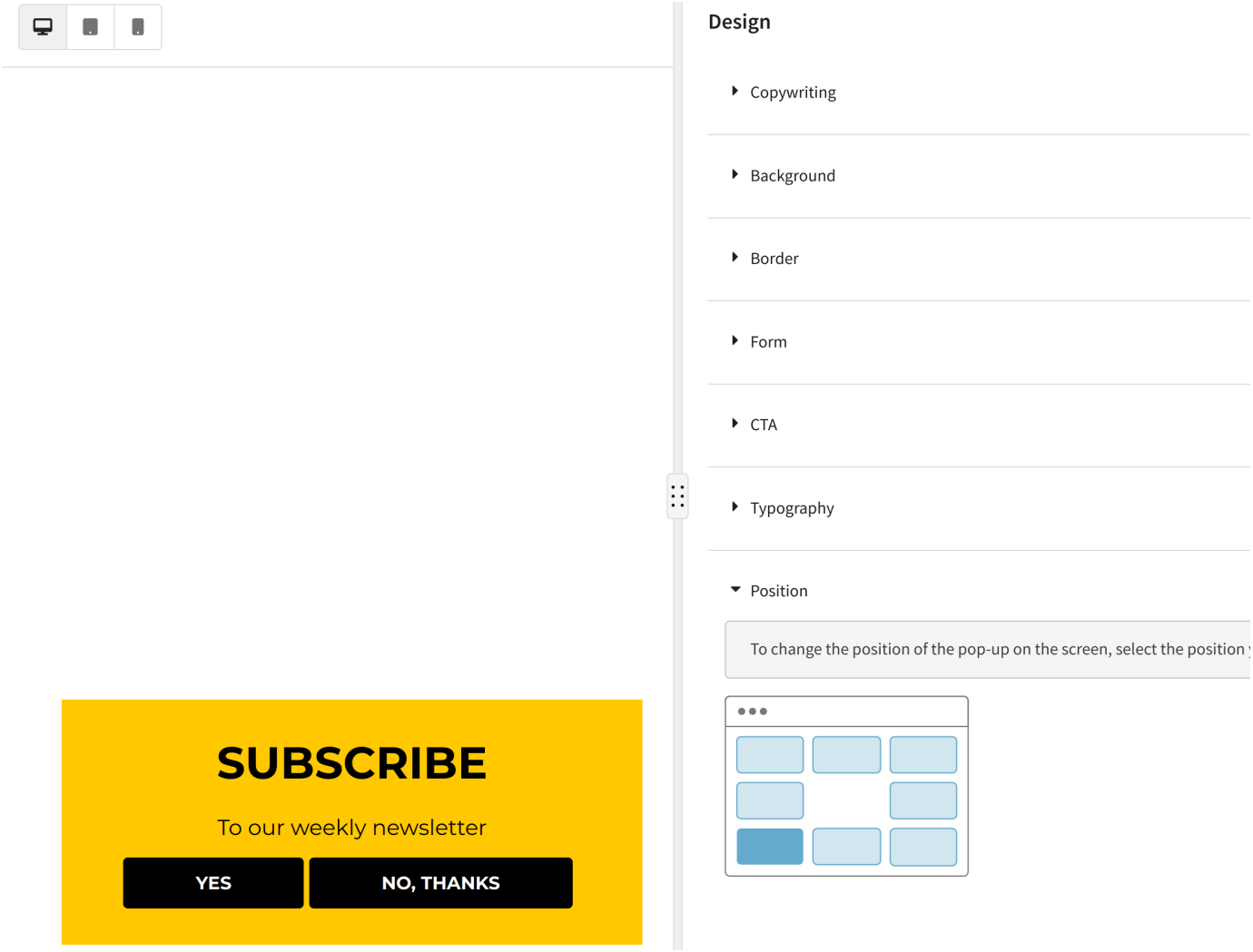
    if (subscribeBtn) {
      subscribeBtn.addEventListener('click', () => {
          // Search for a profile by email. It will also be written to the profile if no matches are found.
          // new AKPush().initSubscription({ email: 'example@email.com' });
          new AKPush().initSubscription();
          popupContainer.style.display = 'none';
      });
    }
}
```

Now you can customize:

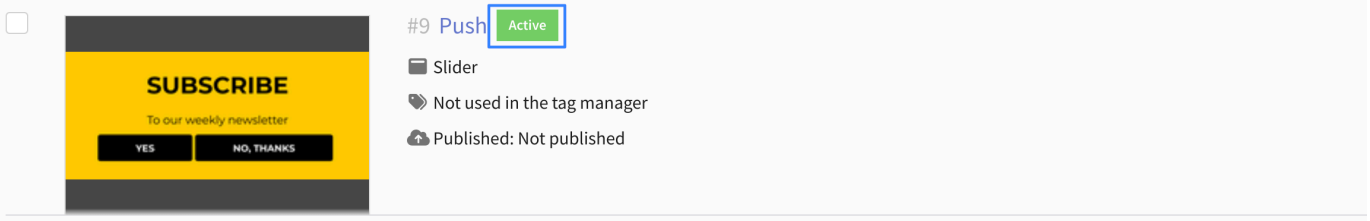
- **The title and description — the Copywriting section**
- **The button text (“Subscribe” / “Cancel”) — the Design — Action section**

> **TIP**
>
> **You can learn more about the layout and customization of pop-ups in the code editor in this article.**

In the Design section, you can adjust the display of elements, their color, the text of inscriptions, and the screen:



When all changes to the pop-up have been made, specify its name and then click Apply and Publish. In th



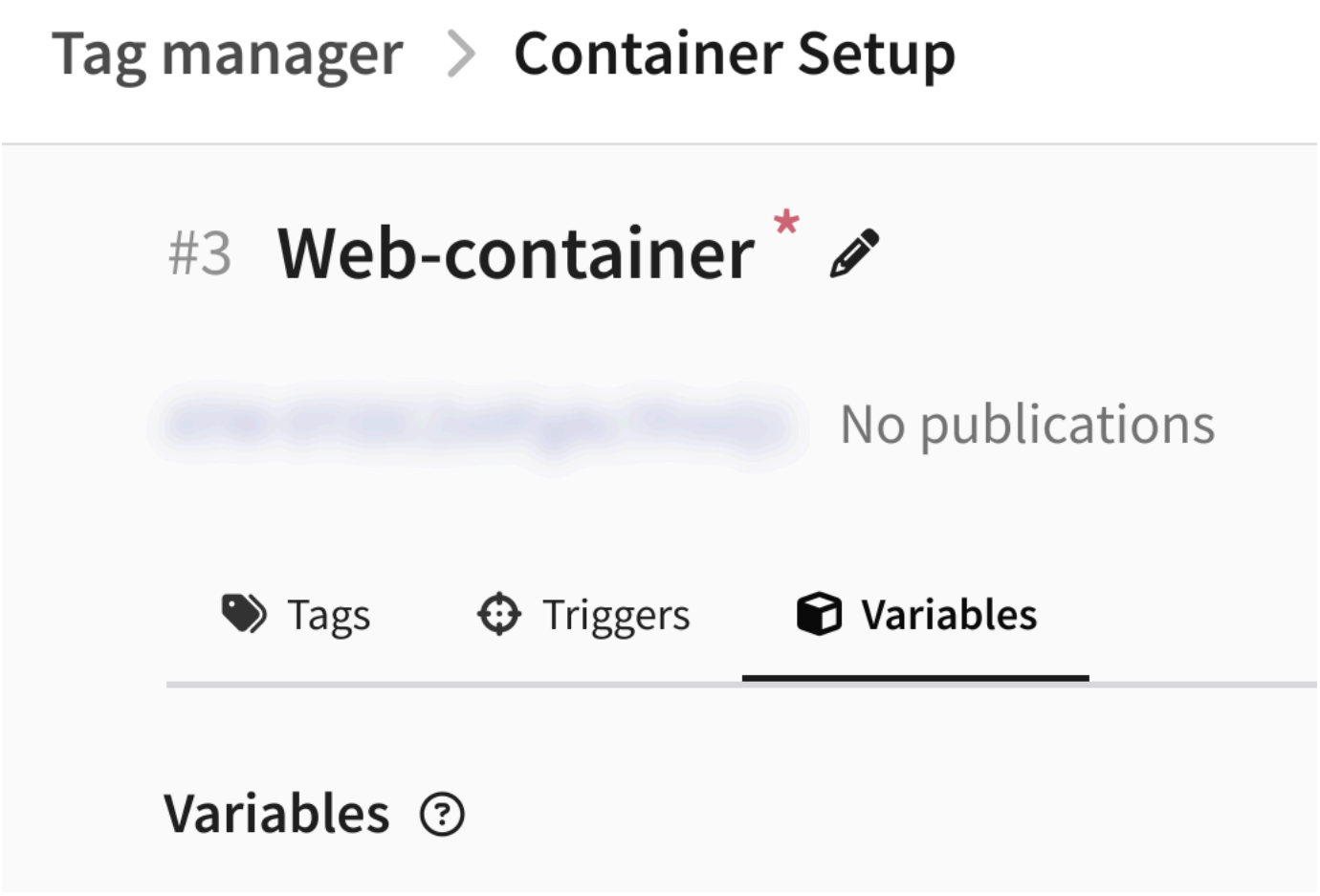
Setting up the Tag Manager

INFO

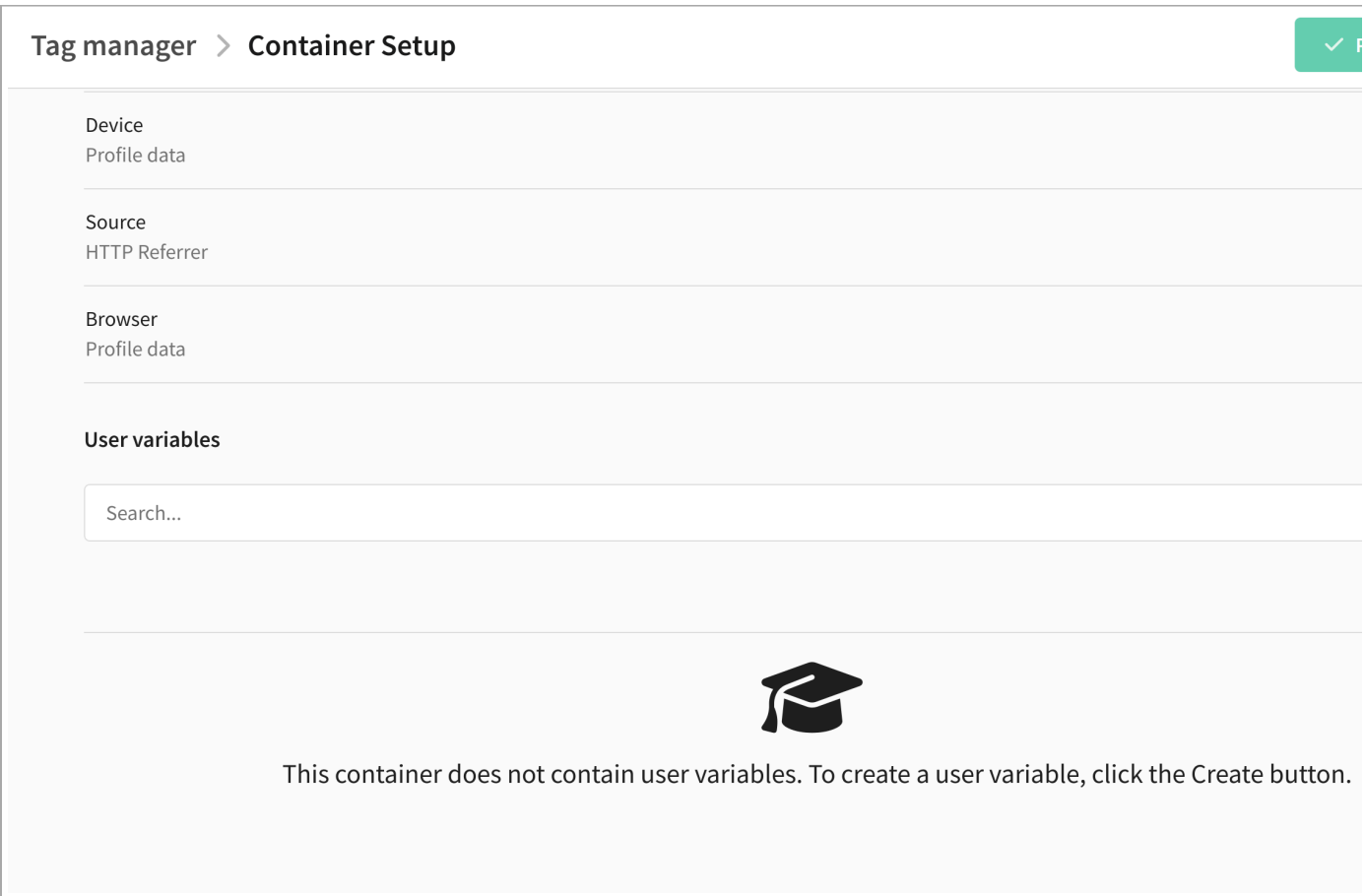
If you haven't worked with Tag Manager or customized the container before, you can read the instructic elements there.

Variable creation

Go to Weblayer — Tag Manager menu. Open the container that you use for the site:



Select Variables and at the very bottom of the page, create a custom variable:



Specify the variable name, for example “Get Notification permission status”, then select the type Custon the following code:

```
function getWebNotificationPermission() {
    return Notification.permission;
}
```

Save the created variable:

Creating the variable

Variable name *

Get Notification permission status

Variable settings

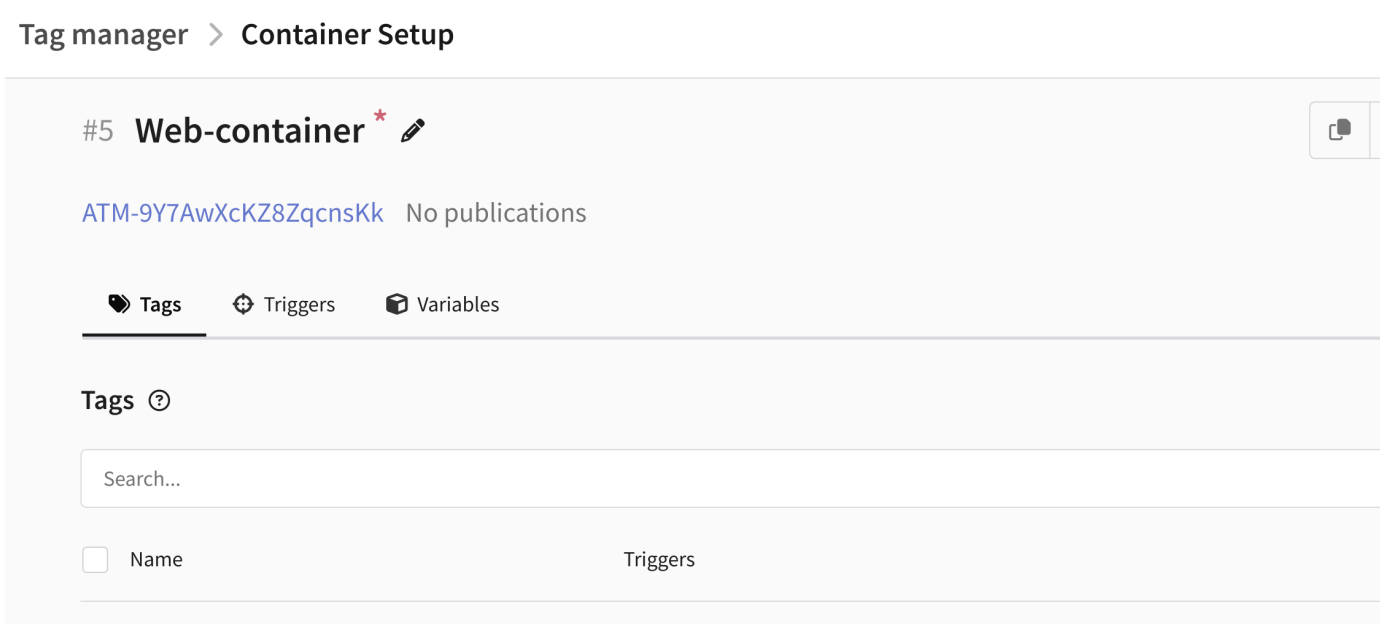
Variable type: Custom JavaScript

Own JavaScript code:

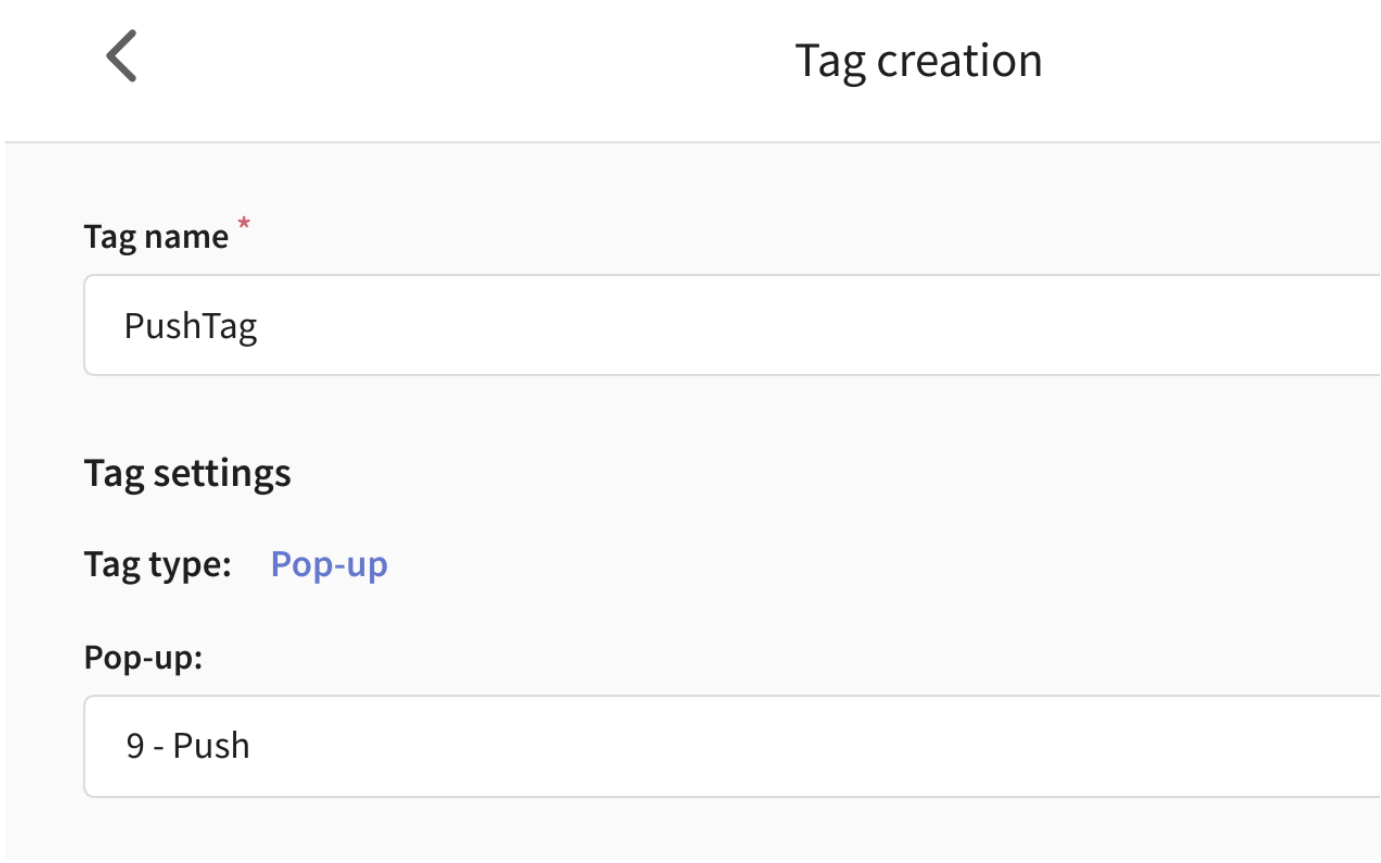
```
function getWebNotificationPermission() {
    return Notification.permission;
}
```

Tag creation

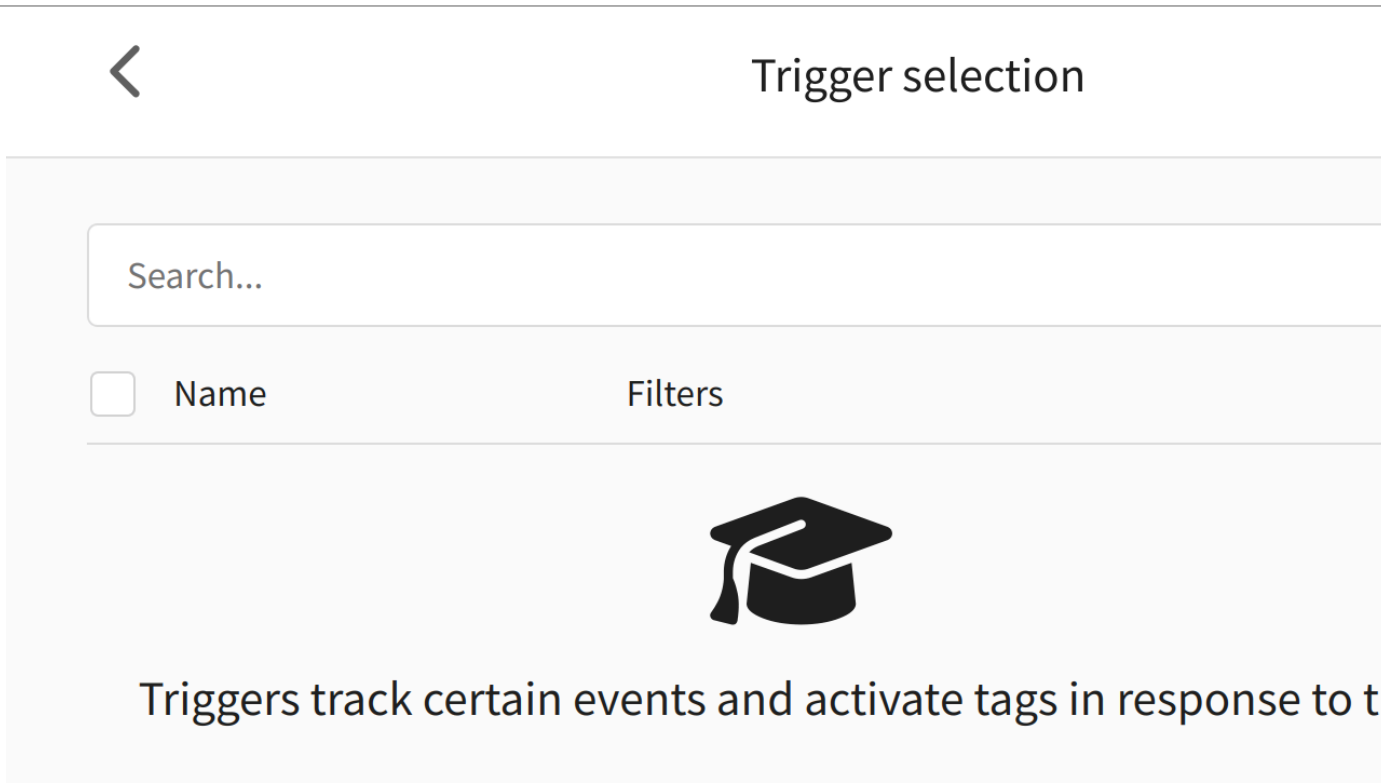
Go to Tags and create a new tag:



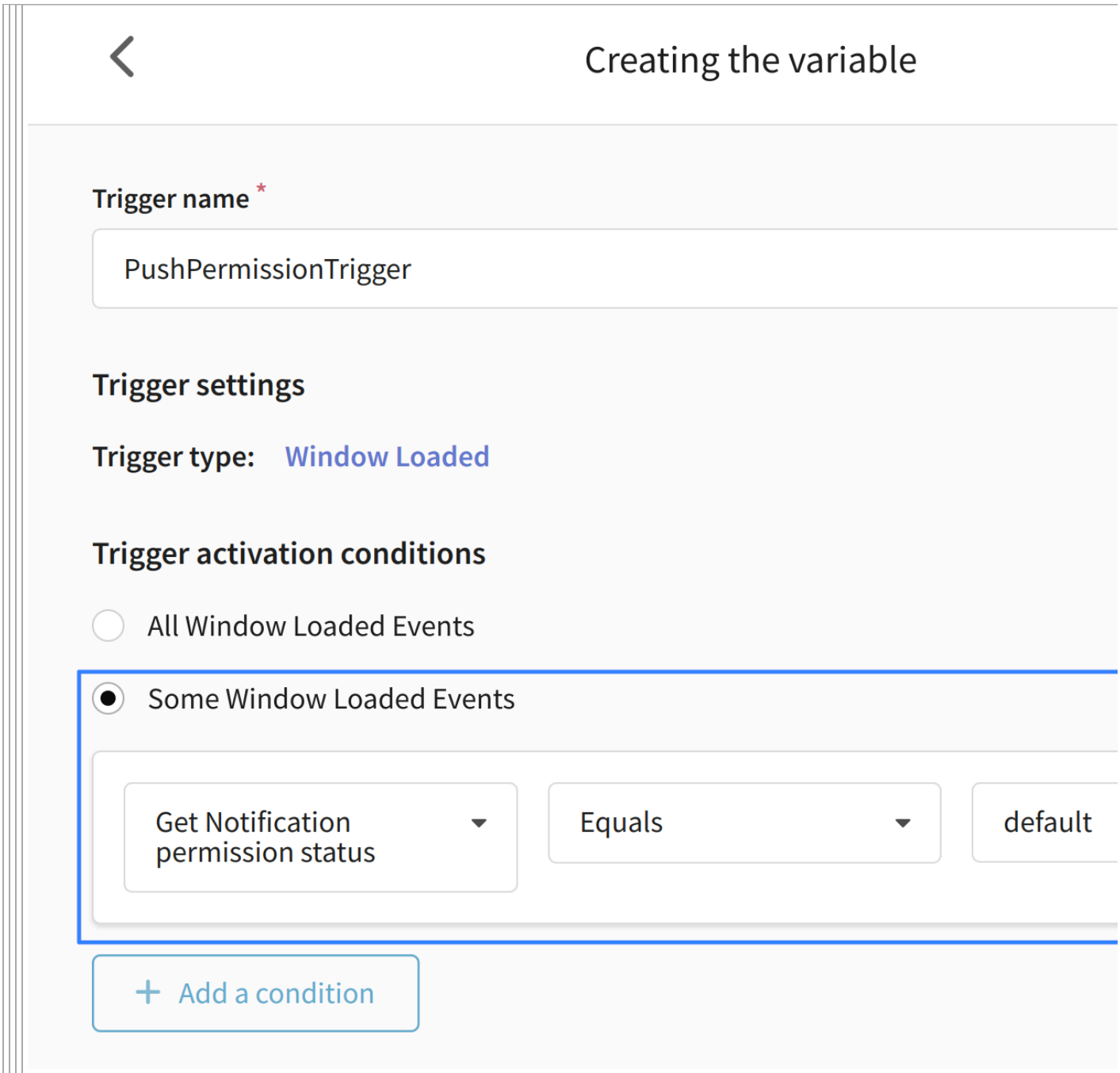
Name it, specify the type Pop-up and select the pop-up you created in the previous step:



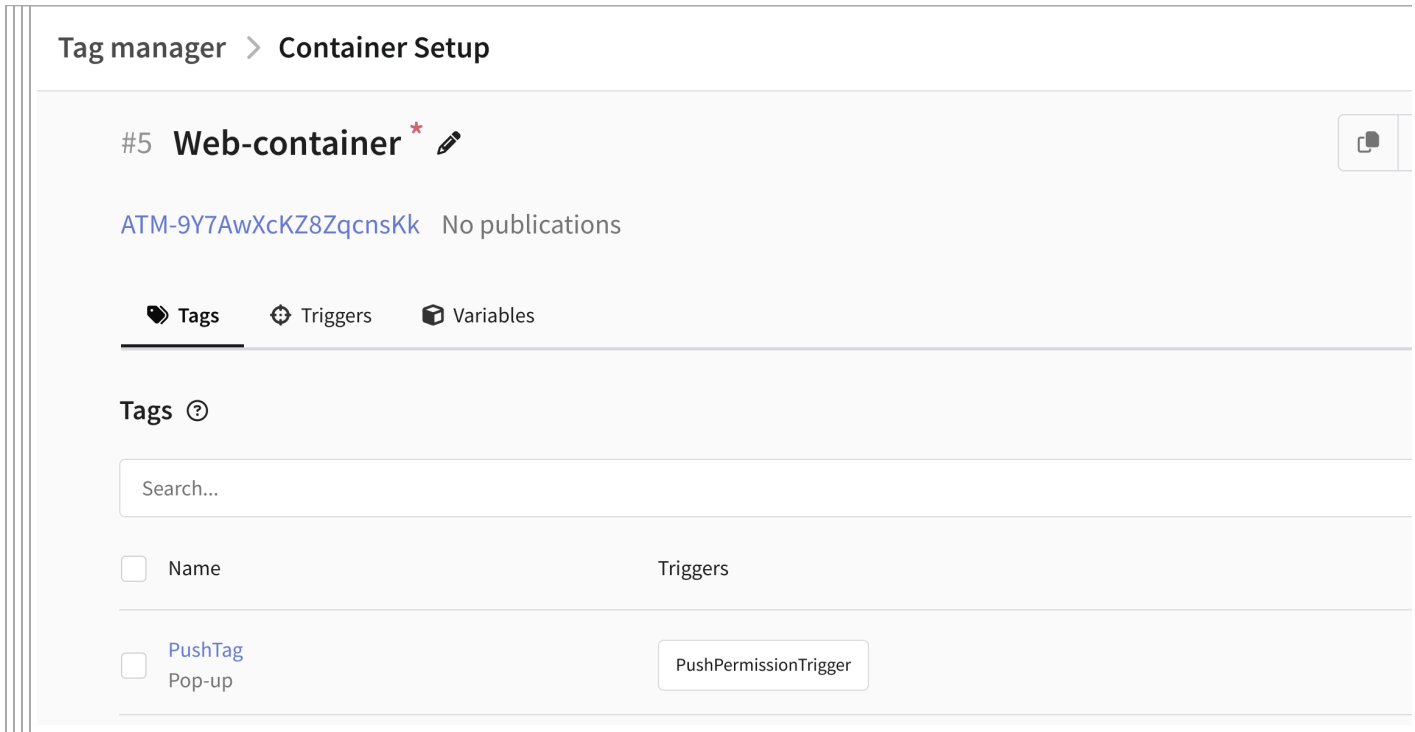
Customize the activation settings and limits if necessary. Then select triggers. If you have not already cr one directly from this window:



**You can customize the trigger yourself for any action you need. In our example we use the “Window load
In the activation conditions specify Some events and the variable created earlier. It is responsible for the
notifications. The variable will be set to `default` in cases where the browser has not yet requested permi
the user. In such situations the pop-up will be triggered:**



Name the trigger and save it. Select it as the trigger to show your pop-up and save the tag. When the tag container have been made, click Publish:



Select the “Publishing and creating a new version” option, provide a title and description for the version

Making changes

Sending configuration

Publishing and creating a new version

Making changes to websites

New version

Saving changes and creatin

Version name

Pushv1

Version description

Push notifications

**The work of the pop-up window has been configured. Now when a user gets to the page, the window wit
notifications will be shown.**

Case: Creating a pop-up with the “Wheel of Fortune” widget

The special “Wheel of Fortune” widget lets you create an interactive pop-up that engages website visito simple form and spins the wheel to receive a random gift: a discount, a promo code, or another bonus. T such a pop-up in the Altcraft builder:

![img](/img/attachments/wheel-of-fortune/en/spining-gif.png)
Before configuring the wheel, it’s recommended to create a field in the profiles database where the won how to create custom fields in [this article](#).

Creating a pop-up and choosing a template

Start by creating a new pop-up in Web Layer — Pop-ups, and click Create. Choose the pop-up type. for a v best fit—it appears in the center of the screen:

![img](/img/attachments/wheel-of-fortune/en/popup-type.png)

Choose a template. Use any ready-made template as a base. For easier customization, choose templates example, one with an astronaut):

![img](/img/attachments/wheel-of-fortune/en/popup-template.png)

Set a name and description: specify the system name and, if needed, a description:

![img](/img/attachments/wheel-of-fortune/en/popup-name.png)

Integrating the widget code

The “Wheel of Fortune” widget is implemented with HTML, CSS, and JavaScript. To add it, replace the st provided ready-made code.

CSS

Go to the CSS tab. Do not remove existing styles. Scroll to the very end of the editor and add the provide container.

```
.formWrapper {
    display: flex;
    align-items: center;
    justify-content: space-around;
    flex-wrap: wrap;
}

.ac-button:disabled {
    opacity: 0.6;
    cursor: not-allowed;
    background-color: #cccccc;
    color: #666666;
    transform: none;
    box-shadow: none;
}

.ac-button:disabled:hover {
    background-color: #cccccc;
    color: #666666;
    transform: none;
    box-shadow: none;
}
```

HTML

In Settings, open the HTML tab. Completely delete the current editor contents and paste the provided HT

```
<div class="ac-popup-overlay">
    <div class="ac-modal">
        <div class="ac-modal_content">
            <div style="display: none;" data-popupId="{{popup('copywriting' 'popupId' 'text')}}"></div>
            <div id="firstPage">
                <h4 class="ac-title">
                    {{popup('copywriting' 'title' 'text')}}
                </h4>
                <div class="ac-description">
                    {{popup('copywriting' 'text' 'text')}}
                </div>
                <div class="formWrapper">
                    <div class="canvas-container">
                        <canvas id="canvas" width="420" height="420"></canvas>
                    </div>
                    <form id="ac-form" class="ac-form" action>
                            <div class="ac-field">
                            <label class="ac-label" for="email">
                                {{popup('copywriting' 'emailLabel' 'text')}}
                            </label>
                            <input
                                id="email"
                                type="email"
                                class="ac-input"
                                name="{{popup('form' 'email' 'fieldData')}}"
                                placeholder="{{popup('copywriting' 'emailPlaceholder' 'text')}}"
                            />
                            </div>
                            <div class="ac-field">
                            <label class="ac-label" for="name">
                                {{popup('copywriting' 'nameLabel' 'text')}}
                            </label>
                            <input
                                id="name"
                                class="ac-input"
                                name="{{popup('form' 'name' 'fieldData')}}"
                                placeholder="{{popup('copywriting' 'namePlaceholder' 'text')}}"
                            />
                            </div>
                            <div class="ac-field">
                            <label>
                                <input type="checkbox" name="{{popup('form' 'privacyPolicy' 'fieldData')}}" id="privacyPolicy" required>
                                <span>I have read the <a href="{{popup('copywriting' 'policy' 'link')}}" target="_blank">privacy policy</a></
                            </label>
                            </div>
                            <div class="ac-validation-msg" for="email"></div>
                            <input type="text" style="display: none;" class="ac-input" id="prizeField" name="{{popup('form' 'prize' 'fieldDat
                        <button class="ac-button" type="button" id="winBtn" disabled>
                            {{popup('copywriting' 'winBtn' 'text')}}
                        </button>

                        <div class="ac-form_submit" style="display: none;">
                            <button type="submit" class="ac-button" id="submitBtn">
                                {{popup('cta' 'text' 'text')}}
                            </button>
                        </div>
                    </form>
                </div>
            </div>
             <div id="secondPage" style="display: none;">
                <h4 class="ac-title">
                    {{popup('copywriting' 'title-congrats' 'text')}}
                </h4>
                <div class="ac-description">
                    {{popup('copywriting' 'text-before-prize' 'text')}}
                    <span id="prizeName"></span>
                    <div>
                        {{popup('copywriting' 'text-after-prize' 'text')}}
                    </div>
                </div>
            </div>
        </div>
        {{popup('closeButton' 'closeButton' 'closeBtn')}}
    </div>
</div>
```

JavaScript

Switch to the JavaScript tab. Completely delete the existing contents and paste the provided JS code. It selection, and form behavior.

```
// Prize list. Length may vary
// id - Identifier sent to the server
// text - Label displayed on the wheel sector
const options = [
  {id: 1, text: '$100'},
  {id: 2, text: '$10'},
  {id: 3, text: '$25'},
  {id: 4, text: '$250'},
  {id: 5, text: '$30'},
  {id: 6, text: '$1000'},
  {id: 7, text: '$1'},
  {id: 8, text: '$200'},
  {id: 9, text: '$45'},
  {id: 10, text: '$500'},
  {id: 11, text: '$5'},
  {id: 12, text: '$20'}
]

// Wheel sector colors. If there are fewer colors than sectors,
// colors are repeated cyclically
const colors = ['white', '#3583FF']

// Initial wheel rotation angle
let currentAngle = 0;

// Wheel spin duration is randomized
// Minimum spin duration (seconds)
const minSpinTime = 4;

// Maximum spin duration (seconds)
const maxSpinTime = 7;

// Delay between wheel stop and prize message display (seconds)
const pageSwitchDelay = 2;

// Color of sector labels on the wheel
const textColor = "black";

let isPreview = true;
let myPopup = null;

const allPopups = document.querySelectorAll('[data-ac-popup-container]');

allPopups.forEach(item =>{
    const idContainer = item.shadowRoot.querySelector('[data-popupId={{popup('copywriting' 'popupId' 'text')}}]');
    if(idContainer){
        myPopup = item.shadowRoot;
        isPreview = false;
    }
})

if (!myPopup) {
  myPopup = document;
}

const winBtn = myPopup.getElementById('winBtn');

function initInputs(){
  let emailInput = myPopup.getElementById('email');
  let nameInput = myPopup.getElementById('name');
  let policyInput = myPopup.getElementById('privacyPolicy');

  function updateButtonState() {
      const isEmailEmpty = !emailInput.value.trim();
      const isNameEmpty = !nameInput.value.trim();
      const isPolicyChecked = policyInput.checked;

      console.log(myPopup.getElementById('winBtn'))
      myPopup.getElementById('winBtn').disabled = isEmailEmpty || isNameEmpty || !isPolicyChecked
  }

  emailInput.addEventListener('input', updateButtonState);
  nameInput.addEventListener('input', updateButtonState);
  policyInput.addEventListener('input', updateButtonState);
  myPopup.getElementById("winBtn").addEventListener("click", spin);

  updateButtonState();
}
if(isPreview) {
  setTimeout(()=>{initInputs()}, 1000);
```

```
} else {
  initInputs();
}

let isInitialized = false;

// Initialize wheel rendering once
function initializeWheel() {
    if (isInitialized) return;

    const canvas = myPopup.getElementById("canvas");
    if (!canvas) return;

    const ctx = canvas.getContext("2d");
    if (ctx && !canvas.dataset.initialized) {
        drawRouletteWheel();
        canvas.dataset.initialized = "true";
        isInitialized = true;
    }
}

const observer = new MutationObserver((mutations) => {
    for (let mutation of mutations) {
        if (mutation.type === 'childList') {
            mutation.addedNodes.forEach((node) => {
                if (node.nodeType === 1 && (node.id === 'canvas' || node.querySelector?.('#canvas'))) {
                    setTimeout(initializeWheel, 100);
                }
            });
        }
    }
});

if (document.body) {
    observer.observe(document.body, { childList: true, subtree: true });
}

document.addEventListener('DOMContentLoaded', initializeWheel);

let segmentAngle = Math.PI / (options.length / 2);
let spinTimeout = null;

let spinAngleStart = 0;

let spinTime = 0;
let spinTimeTotal = 0;

let ctx;

// Draw the wheel with all sectors
function drawRouletteWheel() {
  const canvas = myPopup.getElementById("canvas");

  if (canvas.getContext) {
    const centerX = 210;
    const centerY = 210;
    const outsideRadius = 200;
    const textRadius = 130;
    const insideRadius = 0;

    ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, 400, 400);

    ctx.strokeStyle = "black";
    ctx.lineWidth = 2;
    ctx.font = 'bold 12px Helvetica, Arial';

    ctx.strokeStyle = "white";
    ctx.lineWidth = 15;
    ctx.beginPath();
    ctx.arc(centerX, centerY, outsideRadius, 0, 2 * Math.PI);
    ctx.stroke();

    ctx.strokeStyle = "black";
    ctx.lineWidth = 2;

    for(let i = 0; i < options.length; i++) {
      const angle = currentAngle + i * segmentAngle;
      const segmentMiddleAngle = angle + segmentAngle / 2;
      ctx.fillStyle = colors[i % colors.length];
```

```
      ctx.beginPath();
      ctx.arc(centerX, centerY, outsideRadius, angle, angle + segmentAngle, false);
      ctx.arc(centerX, centerY, insideRadius, angle + segmentAngle, angle, true);
      ctx.stroke();
      ctx.fill();

      ctx.save();
      ctx.fillStyle = textColor;

      const textX = centerX + Math.cos(segmentMiddleAngle) * textRadius;
      const textY = centerY + Math.sin(segmentMiddleAngle) * textRadius;

      ctx.translate(textX, textY);
      ctx.rotate(segmentMiddleAngle);

      const text = options[i].text;
      ctx.fillText(text, -ctx.measureText(text).width / 2, 0);
      ctx.restore();
    }

    // Draw pointer
    ctx.fillStyle = "white";
    ctx.strokeStyle = "#666666";
    ctx.lineWidth = 2;

    ctx.beginPath();
    ctx.moveTo(centerX - 10, centerY - 15);
    ctx.lineTo(centerX + 30, centerY);
    ctx.lineTo(centerX - 10, centerY + 15);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
  }
}

// Animate wheel rotation
function rotateWheel() {
  spinTime += 30;
  if(spinTime >= spinTimeTotal) {
    stopRotateWheel();
    return;
  }
  const spinAngle = spinAngleStart - easeOut(spinTime, 0, spinAngleStart, spinTimeTotal);
  currentAngle += (spinAngle * Math.PI / 180);
  drawRouletteWheel();
  spinTimeout = setTimeout(rotateWheel, 30);
}

// Start wheel spin
function spin(e) {
  e.preventDefault();
  const form = myPopup.getElementById('ac-form');
  const isValid = form.checkValidity();

  if(!isValid){
    myPopup.getElementById('submitBtn').click();
    return;
  }
  winBtn.removeEventListener('click', spin);
  spinAngleStart = Math.random() * 10 + 10;
  spinTime = 0;
  spinTimeTotal = Math.random() * (maxSpinTime - minSpinTime) + minSpinTime * 1000;
  rotateWheel();
}

// Handle selected prize
function handlePrize(prize){
  myPopup.getElementById("prizeField").value = prize.id;
  myPopup.getElementById('submitBtn').click();
  myPopup.getElementById('prizeName').innerText = prize.text;
}

// Switch to prize screen
function showSecondPage() {
  myPopup.getElementById('firstPage').style.display = 'none';
  myPopup.getElementById('secondPage').style.display = 'block';
}

// Stop wheel rotation and determine winning sector
function stopRotateWheel() {
```

```
  clearTimeout(spinTimeout);

  const degrees = currentAngle * 180 / Math.PI;
  const segmentDegrees = segmentAngle * 180 / Math.PI;
  const index = Math.floor((360 - degrees % 360) / segmentDegrees);

  handlePrize(options[index]);

  const form = myPopup.getElementById('ac-form');
  const isValid = form.checkValidity();

  setTimeout(()=>{
    showSecondPage();
  }, pageSwitchDelay * 1000);

}

// Easing function for smooth deceleration
function easeOut(currentTime, startValue, changeInValue, duration) {
    const normalizedTime = currentTime / duration;
    const squaredTime = normalizedTime * normalizedTime;
    const cubedTime = squaredTime * normalizedTime;

    return startValue + changeInValue * (cubedTime + -3 * squaredTime + 3 * normalizedTime);
}

drawRouletteWheel();

// Override form submit handler
myPopup.getElementById('ac-form').submit = () => {}
```

Content and design setup in the visual editor

After integrating the code, go to Design. Here you can fill the pop-up with text and tune its appearance u without editing code.

Pop-up width

First, set the pop-up width (Form — width). The wheel canvas size is fixed (420 px). So the pop-up width r the wheel and the input form on the same horizontal row, set the width to at least 810 px.

Copywriting

On the Copywriting tab, fill in the key text parameters:

- `title` **— main headline on the first screen (for example, "Try your luck!").**
- `text` **— explanatory text or campaign description.**
- `emailLabel` **and** `emailPlaceholder` **— label and placeholder for the email field.**
- `nameLabel` **and** `namePlaceholder` **— label and placeholder for the name field.**
- `policy` **— a required link to your privacy policy page. Without filling this field, you won't be able to sa**
- `winBtn` **— text on the button that starts the wheel spin (for example, "Spin the wheel!").**
- `popupId` **— a required parameter. A unique text identifier for the pop-up (for example,** `onlinestore_wheel` **are multiple pop-ups on the page.**

The pop-up has two screens. Parameters for the "win message" screen:

- `title-congrats` **— congratulations headline.**
- `text-before-prize` **— text shown before the prize name.**
- `text-after-prize` **— additional text under the prize message.**

Appearance

Use the other tabs in Design for fine-tuning the look:

- **Background — change the pop-up background color and the page overlay opacity (** `overlay` **).**
- **Border — configure the pop-up corner rounding.**
- **Form — style input fields: spacing, border colors, background, and text.**
- **Actions — style buttons (color, rounding, font).**
- **Typography — choose the common font and styles for all text elements.**

Any changes in the visual editor are immediately shown in the preview pane on the right.

Wheel of fortune customization

The wheel's look and behavior (prizes, sector colors, spin speed) are configured by editing variables in th Go back to Settings — JavaScript. At the beginning of the file, find and edit the following variables.

Prizes (options)

The `options` **variable defines the list of prizes on the wheel. It's an array of objects, where each object c** `text` **.**

```
const options = [
    { id: 1, text: '$100' },
    { id: 2, text: '$10' },
    { id: 3, text: '$25' },
    { id: 4, text: '$250' },
    { id: 5, text: '$30' },
    { id: 6, text: '$1000' },
    { id: 7, text: '$1' },
    { id: 8, text: '$200' },
    { id: 9, text: '$45' },
    { id: 10, text: '$500' },
    { id: 11, text: '$5' },
    { id: 12, text: '$20' }
];
```

How to change:

- **Prize text: replace the** `text` **value with what you need (for example,** `'10% off'` **or** `'Free shipping'` **).**
- **Number of prizes: add new objects to the array or remove existing ones. The number of wheel secto**

Sector colors (colors)

The `colors` **variable is an array of fill colors for the wheel sectors. Colors are applied in order, starting fr**

```
const colors = ['black', 'orange'];
```

How to change: Replace the colors in the array with your own. You can use:

- **English color names:** `'red'`, `'blue'`, `'green'`
- **HEX codes:** `'#4f46e5'`, `'#f59e0b'`, `'#10b981'`
- **RGB/RGBA values:** `'rgb(79, 70, 229)'`

Spin speed

Spin duration is set with two variables:

- `minSpinTime` **— minimum spin time (seconds).**
- `maxSpinTime` **— maximum spin time (seconds).**

The actual duration of each spin will be a random value within this range.

```
const minSpinTime = 4;
const maxSpinTime = 7;
```

How to change: Increase values to make the wheel spin longer, or decrease them for a faster result.

Reveal delay (pageSwitchDelay)

The `pageSwitchDelay` **variable sets the pause (in seconds) between the wheel stopping and the automatic s**

```
const pageSwitchDelay = 2;
```

How to change: Set it to the number of seconds you need.

Text color (textColor)

The `textColor` **variable defines the color of prize labels on the wheel sectors.**

```
const textColor = "white";
```

How to change: Replace the value with any valid CSS color so the text remains readable on the sector ba `"black"`, `"#1f2937"`.

Saving the won prize value

After a user spins the wheel and wins a prize, the result is automatically written into the hidden form fiel user profile in Altcraft, configure the import action in Actions.

In the pop-up editor, open the Actions tab:

![img](/img/attachments/wheel-of-fortune/en/import.png)

Configure profile import:

- **Follow the standard import setup flow described in the pop-up creation article.**
- **In import settings, map the** `prize` **field from the form to the required field in your Altcraft database.**

![img](/img/attachments/wheel-of-fortune/en/matching.png)

By default, the JavaScript code writes the prize ID into the `prize` **field (**`prize.id`**). If you need to save the edit the** `handlePrize()` **function in the JS code and replace** `prize.id` **with** `prize.text`**.**

After configuring import, every form submission will save the prize data (text or ID), along with the user's profile.

Publishing the pop-up

After you finish configuration, save the pop-up and make sure its status is Active.

In Appearance, bind the pop-up to a Tag Manager container:

![img](/img/attachments/wheel-of-fortune/en/choose-container.png)

Set the tag Name and configure the trigger — the condition that shows the pop-up (for example, Timer or

![img](/img/attachments/wheel-of-fortune/en/trigger.png)

More about triggers is in the dedicated article.

After you finish setup, save the pop-up and make sure the status is Active. Once published, the wheel-of- appear to visitors according to your rules.

Automatic delivery of personalized prizes via scenarios

To automatically send promo codes that match the won prizes, create a scenario triggered by changes t promo codes for each prize type and conditional branching in the scenario.

Creating promo codes

Go to Loyalty — Create to create a new loyalty program. In the program settings, create promo codes for

- **10% discount — promo code** `BOOK10`
- **15% discount — promo code** `BOOK15`

- **Free shipping — promo code** `FREESHIP`
- **And so on for each prize**

![img](/img/attachments/wheel-of-fortune/en/promocodes.png)
You can read more about promo codes in [this article].

Creating an email template

Create a universal email template that shows different text depending on the won prize. Use logical expr value. More about logical expressions and variables in templates is in [this article].

The template checks what value the prize field (for example, `Prize`**) contains in the user profile (** `{lead.Pri`
For each prize type, insert a promo code variable into the email body:

```
{if lead.Prize}

    {if lead.Prize equal "1" or lead.Prize equal "$100"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a cash prize of $100!</p>
        <p>Your promo code: {loyalty.prize100.promocode}</p>
        <p>The promo code is valid for all products in our store.</p>
    {end}

    {if lead.Prize equal "2" or lead.Prize equal "$10"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a cash prize of $10!</p>
        <p>Your promo code: {loyalty.prize10.promocode}</p>
        <p>The promo code is valid for all products in our store.</p>
    {end}

    {if lead.Prize equal "3" or lead.Prize equal "10% discount"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a 10% discount on your next purchase!</p>
        <p>Your promo code: {loyalty.discount10.promocode}</p>
        <p>The discount is valid for all products in our catalog.</p>
    {end}
{else}
    <p>Congratulations, {lead._fname}!</p>
    <p>You have won a prize in our promotion!</p>
    <p>Please contact your manager for details.</p>
{end}
```

As a result, you'll get a template where each condition has its own promo code:
![img](/img/attachments/wheel-of-fortune/en/template-promocode.png)

Creating a scenario

Go to Scenarios — Create. Select the database where your campaign participants' profiles are stored. In entry into the scenario for a customer profile. This ensures the scenario starts for each user only once p
After creating the scenario, add an entry trigger. Choose Profile change as the trigger type, select the `Pr`
Changed. This means the scenario will start each time the user's prize field appears or changes.
After setting the trigger, add a Condition element to the scenario. In the condition, check the `Prize` **field** `Prize`**, Operator to** `equal`**, and enter** `1` **(if you store the prize ID) or the text of the first prize, for example**
If the condition is true (the user won prize ID 1 or "$100"), add two nodes:

- **Promo code — select the created loyalty program and the specific promo code that matches this pri**
- **Send email — select the created universal email template.**

If the condition is false, add the next Condition element to check the second prize. Configure it the same contains `2` **or the text of the second prize.**
Repeat this logic for each prize, creating a chain of conditions and matching actions to assign promo co
![img](/img/attachments/wheel-of-fortune/en/scenario.png)
At the end, don't forget to save and activate your scenario.

Basic cases of placing a popup via the Tag Manager

Configuring popup triggers

There are four main popup trigger scenarios available:

- **On page load**
- **On scroll**
- **On click**
- **After a certain time on the page**

> INFO
>
> **To place a popup, you need to add it to the Altcraft Tag Manager and install the container code on your**

Showing a popup on page load

To configure a popup to appear immediately after the page loads, create the corresponding trigger in the depends on your site's architecture:

Site architecture	
For MPA sites (with full page reload)	**Page View**
For SPA sites (without page reload)	**History Ch**

To configure, create a trigger in the Tag Manager: go to the Triggers tab —> "Create".
Select the appropriate trigger type and set the activation conditions:

- **All page views — the popup will appear on any page with the container code.**
- **Some page views — for example:** `Page URL –> equals –> https://site.com/contact`**. The popup will only appe**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup on scroll

The popup will appear when the user scrolls the page to the specified depth. To configure this, create a in percentages (for example, 50%). The trigger can fire on both vertical and horizontal scroll depth.
Set the activation conditions:

- **All page views — trigger on any page.**
- **Some page views — for example:** `Page Path –> contains –> /blog/`**. The popup will only appear when scro**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup on click

The popup will appear when clicking an element on the page. To configure, create a Click on element tri
Set the activation conditions:

- **All clicks — trigger on any click on the page.**
- **Some clicks — for example:** `Click ID –> equals –> subscribe-button`**. The popup will only appear when clic "subscribe-button".**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup after a time delay

With this trigger configuration, the popup will appear after the specified time has passed.
To do this, create a trigger in the Tag Manager. Go to the Triggers tab —> Create.
Enter the trigger name and select its type "Timer".
Trigger settings include:

- **Default event name —** `actm.timer`**. In most cases, you don't need to change the name. You can specify based on a custom event.**
- **Interval — the time between event firings (in milliseconds). For example, if you specify 5000, the eve firing will also occur after the specified time.**
- **Limit — the maximum number of firings. If you leave the field empty, the trigger will keep firing until**

For example, to show a popup 10 seconds after the page loads, set the interval to 10000 and the limit to
After configuration, click "Save" and publish the changes in the Tag Manager.

Tag Manager

![img] (/img/attachments/tag_manager/weblayers.png)	Altcraft Tag Manager is a tool that allows you to manage HTML tags and JavaScript code without any need for developers. It snippets on your site to track user activity and feed data into the platform.

TIP

Altcraft Tag Manager is fully compatible with Google Tag Manager (GTM). You can use both tools on the conflicts.

Basic terms

Container is a wrapper that stores tags, triggers, and variables. It provides the code that needs to be pla

Tag is a piece of code that performs a specific function (for example, showing a popup to site visitors).

Trigger is a condition for executing a tag. Triggers monitor certain events on a site (such as clicks or pag response to them. To fire a tag, at least one trigger must be configured.

Variable is a parameter to which certain values are passed. Variables can be used in both triggers and ta activation filter, they determine when a particular tag should be fired. In tags, variables are used to obta

In this section

Configuring and installing Tag Manager

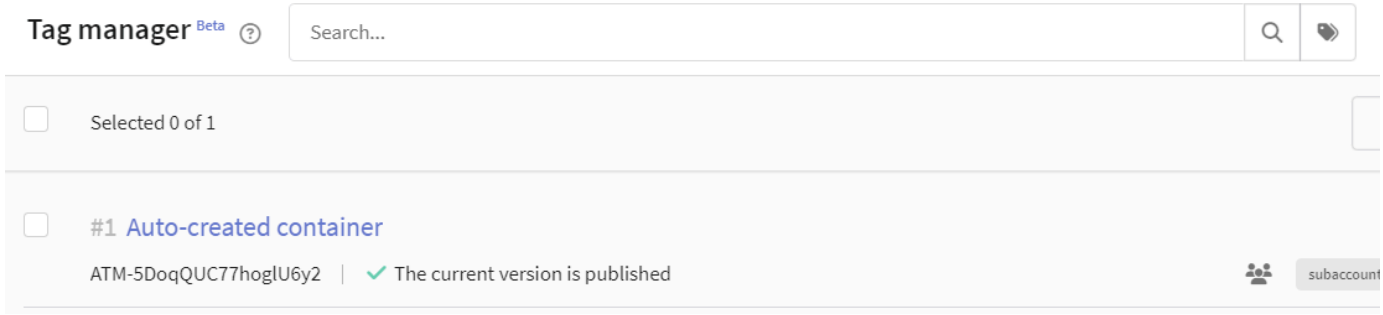
To get started with Tag Manager, go to Weblayer → Tag Manager.

Step 1. Creating a container and installing it on the site

TIP

A container stores tags, triggers, and variables. It is a piece of code that is installed on the site. After i configuration is done in the platform interface without changing the source code.

By default, a container is already created in your account. You can work with it or add a new one. It is re container for each website or mobile app.



The following actions are available in the actions menu on the right side:

— clone the container: a new container is created and the settings, tags, triggers, and variables of the c version history is not copied).

— delete the container along with all created tags, triggers, and variables. Tags from this container will

Creating a new container

To create a new container, follow these steps:

1. Click the + Create button in the upper right corner.

2. Enter a custom name for the container.

3. Select profile databases. Tags track the behavior of site visitors and send data to the Altcraft system. search for existing profiles in the database. Specify at least one database for the search.

4. Install the container on the site. To do this, copy the provided code and pass it to your developers. The every page of the site where you plan to use tracking tags. The container code should be added to the <
tag as possible.

5. Click the Create button to save the container.

Create Container

Container name *

New container

Profiles databases

6 - B2C-clients ×

For the container to function on your site, copy the code below and paste it into the code of the pages on which it should work (you can do this later

Insert this portion of the code into the **<head>** section of the page code as close to the beginning as possible:

```
<script>
  (function (window, document, dataLayerName, containerID) {
    window[dataLayerName] = window[dataLayerName] || [];
    window[dataLayerName].push({ 'start': new Date().getTime(), event: 'actm-js' });
    var firstScript = document.getElementsByTagName('script')[0];
    var script = document.createElement('script');
    var dl = dataLayerName !== 'acDataLayer' ? '&l=' + dataLayerName : '';
    script.async = true;
    script.src = 'https://pxl.daria-ustinkina.dev.altkraft.com/actm/js/actm.js?id=' + containerID + dl;
    script.type = 'module';
    if (firstScript && firstScript.parentNode) {
      firstScript.parentNode.insertBefore(script, firstScript);
    }
  }(window, document, 'acDataLayer', 'ATM-7m52icYWPjjAEHWFM'));
</script>
```

Working in a container

To edit an existing container, click its name. When you click the container identifier (ATM-xxxxxxx), a wi
have not yet placed the container on your site, pass the code to your developers.

To the right of the ID there is a counter that shows the number of unpublished changes. After a version is
zero.

Tag manager > Container Setup

#2 New container *

ATM-7m52icYWPjjAEHWFM Changes in the container: 0

Tags | Triggers | Variables

Tags

Search...

Name	Triggers
Pop-up showing Pop-up	Click on items

For additional settings, a quick actions menu is available on the right:

— view version history;

— select databases for profile search;

— assign tags for quick search;

— configure access groups.

> INFO
>
> **Any changes to the container settings (editing the name or access groups, configuring tags and trigger platform, but they are not applied on the site until you publish the container.**

Step 2. Adding tags and configuring triggers

> TIP
>
> **A tag is a piece of code that runs on the site when a certain condition is met. A trigger is the condition Triggers monitor specific events on a site (such as clicks, page loads, or form submissions) and fire tag To fire a tag, you must configure at least one trigger.**

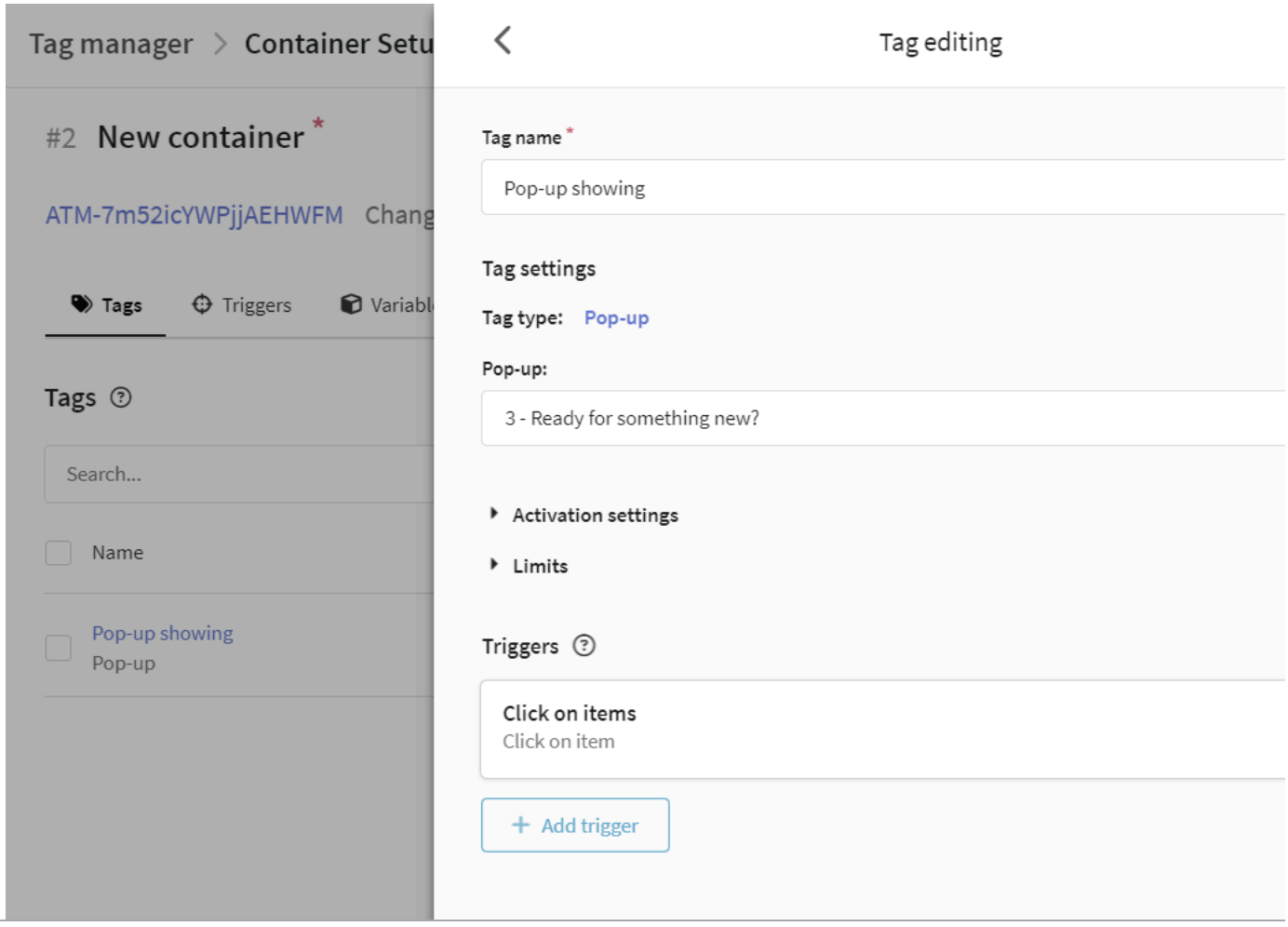
To create a tag and configure the conditions for firing it, follow these steps:

1. On the Tags tab, click + Create.

2. Enter the name of the tag.

3. Select the tag type:

- **Pop-up — used to display a pop-up window to website visitors. You must first create a pop-up and the more about pop-ups in [this article](#).**
- **Custom HTML — used to create a custom tag. Paste your HTML code into the editor window.**

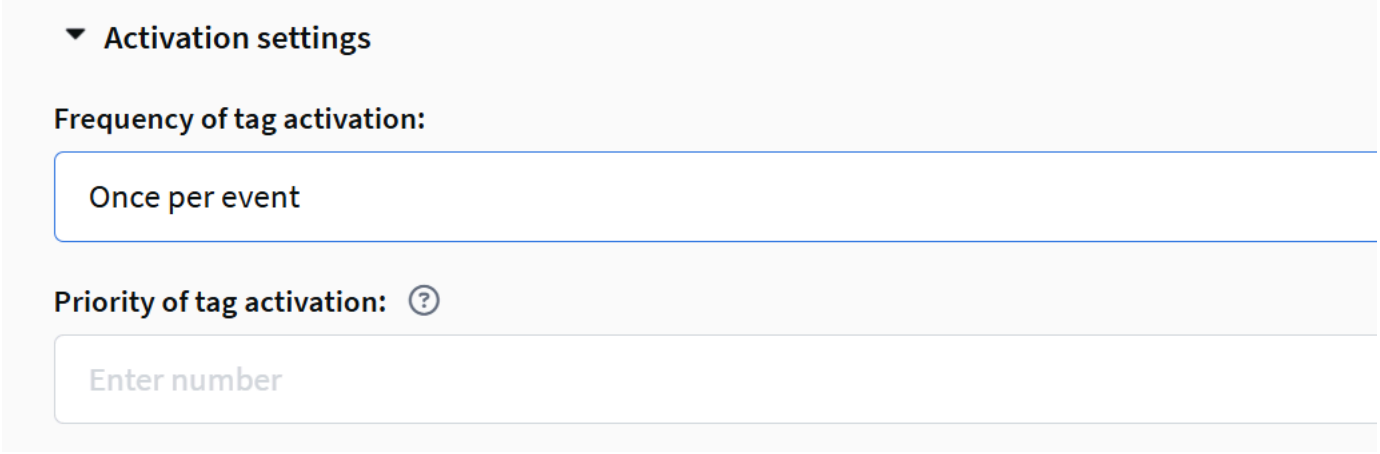


4. Configure activation settings and limits.

Tag activation settings

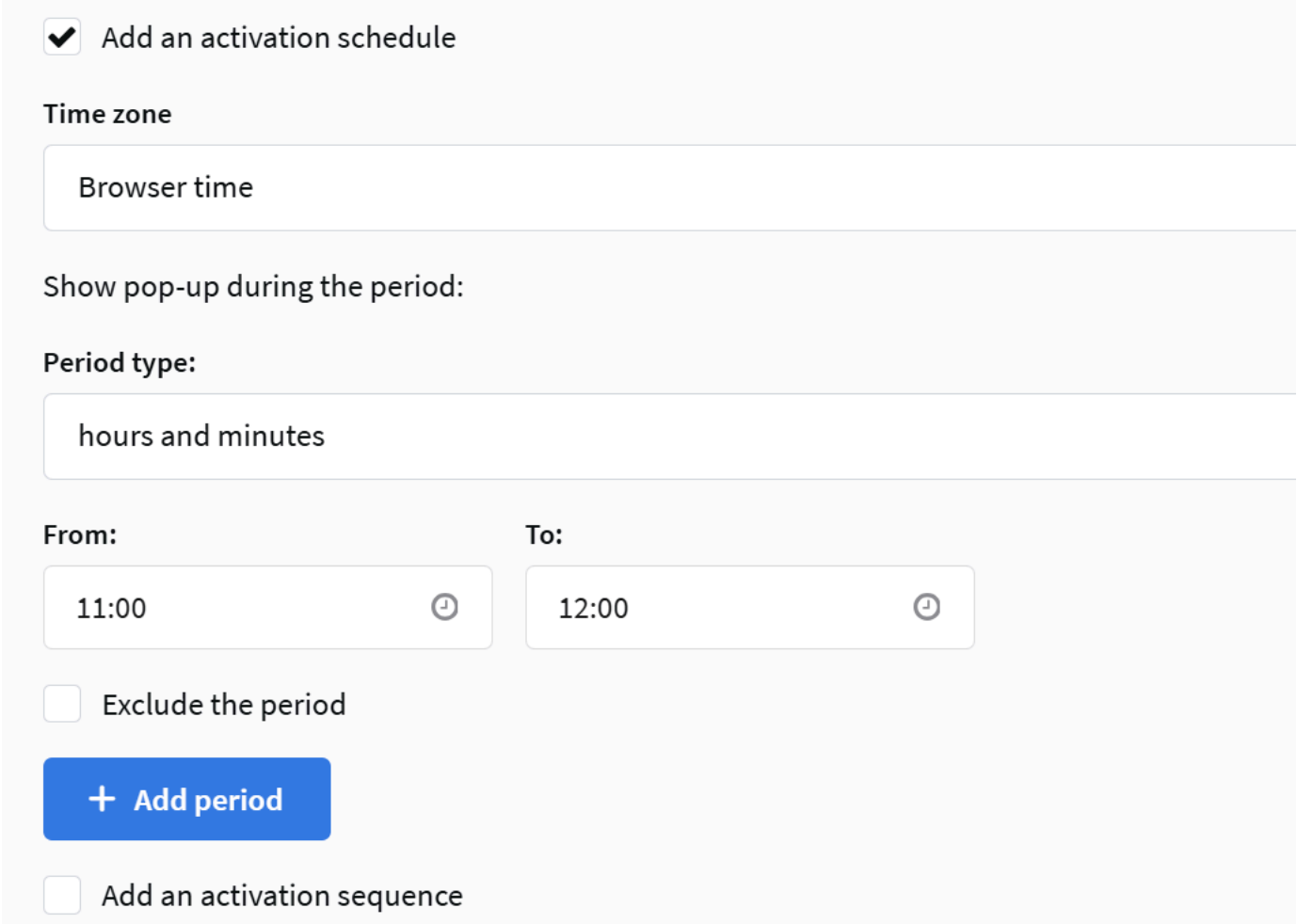
Frequency of tag activation:

- **Once per event — the tag is activated only once when a specific event occurs.**
- **Once per page — the tag will fire only once per page, regardless of the number of events.**
- **No restrictions — the tag is activated each time the event specified in the trigger occurs.**



Priority of tag activation. Sets the order in which tags are fired. By default, the priority of a tag is 0. To c or negative integer. For example, a tag with priority 2 will be fired before a tag with priority 1. Tags run a

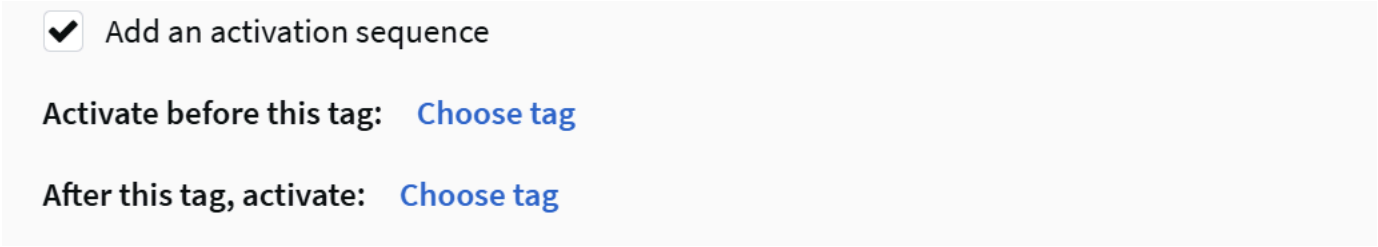
Add an activation schedule. Lets you configure tag activation during a given period. For example, you are one hour. In this case, the banner with information should be shown to site visitors only during this hour. Select the period type, set the start and end date and time. If you want the tag to remain inactive during period option. You can also combine several time periods — click the Add period button.



Add an activation sequence:

- **Activate before this tag. Select a tag that must be fired before the current tag. If the selected tag do creating will not be activated.**

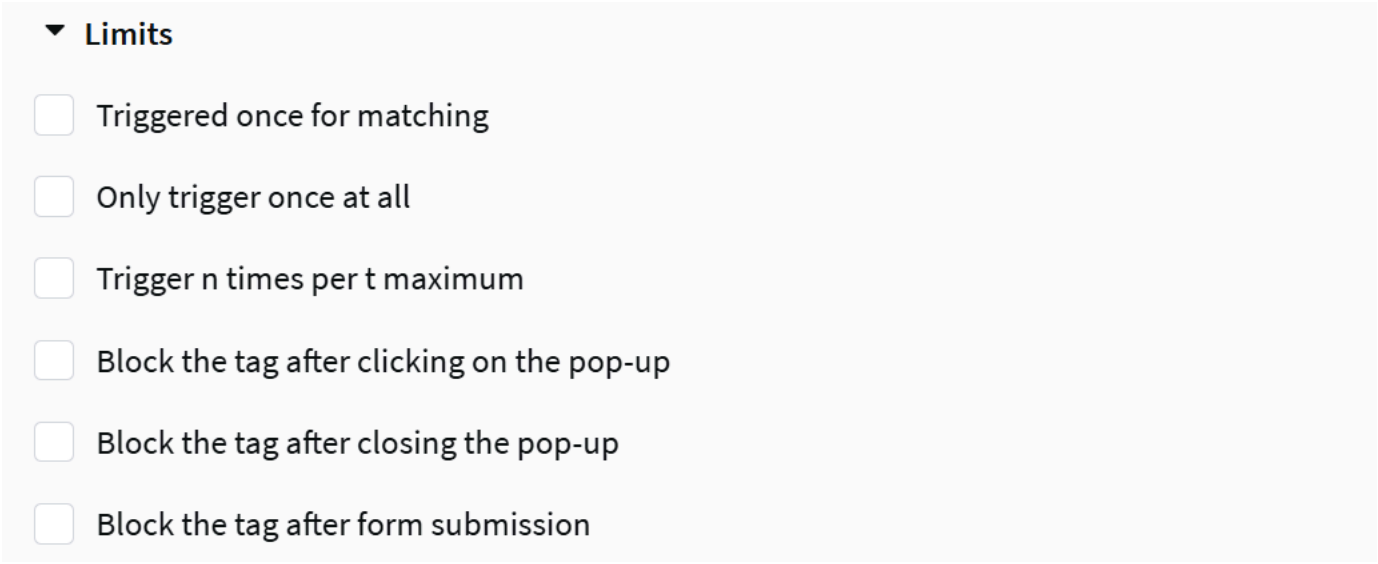
- **Activate after this tag. Select a tag that must be fired after the current tag. If the tag you are curren selected tag will not be activated.**



Tag limits

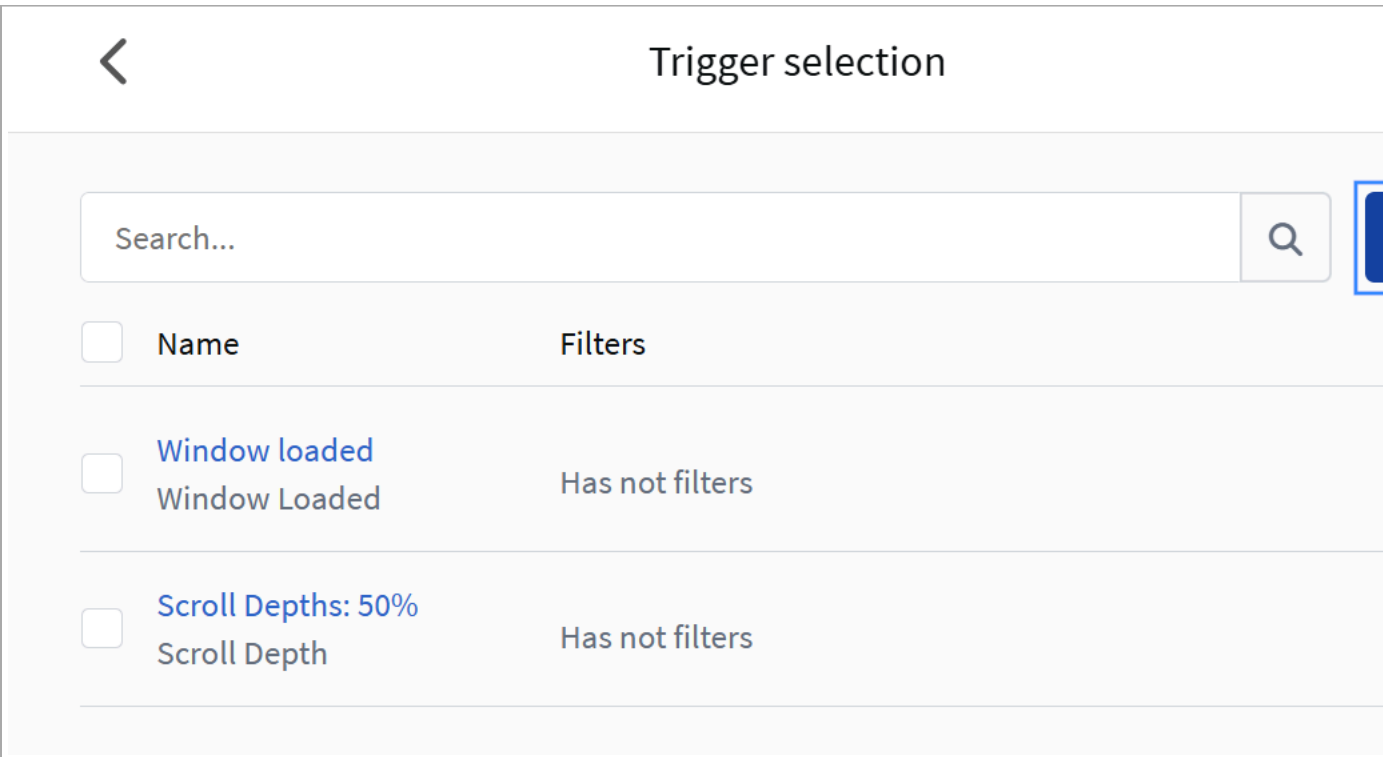
Limits

- **Trigger only once per match — the tag is executed only once each time a condition that matches its condition becomes true again, the tag will not be executed again. This applies to conditions related title, and other parameters that may change while the page is being viewed.**
- **Trigger only once — the tag is executed only once, regardless of which conditions are met. Use this executed only once.**
- **Trigger at most n times per period t — limits how many times the trigger can fire during a certain per**
- **Block the tag after clicking in a pop-up — after clicking any element in the pop-up, the tag will not be listens to occurs again. This prevents multiple tag activations on repeated clicks in the pop-up.**
- **Block the tag after closing a pop-up — after clicking the Close button in the pop-up, the trigger will n again.**
- **Block the tag after submitting a form — after the form associated with the tag is submitted, the trigg conditions for its activation are met again.**



5. Click Select triggers. A new window opens with all available triggers. If you have already created a tri

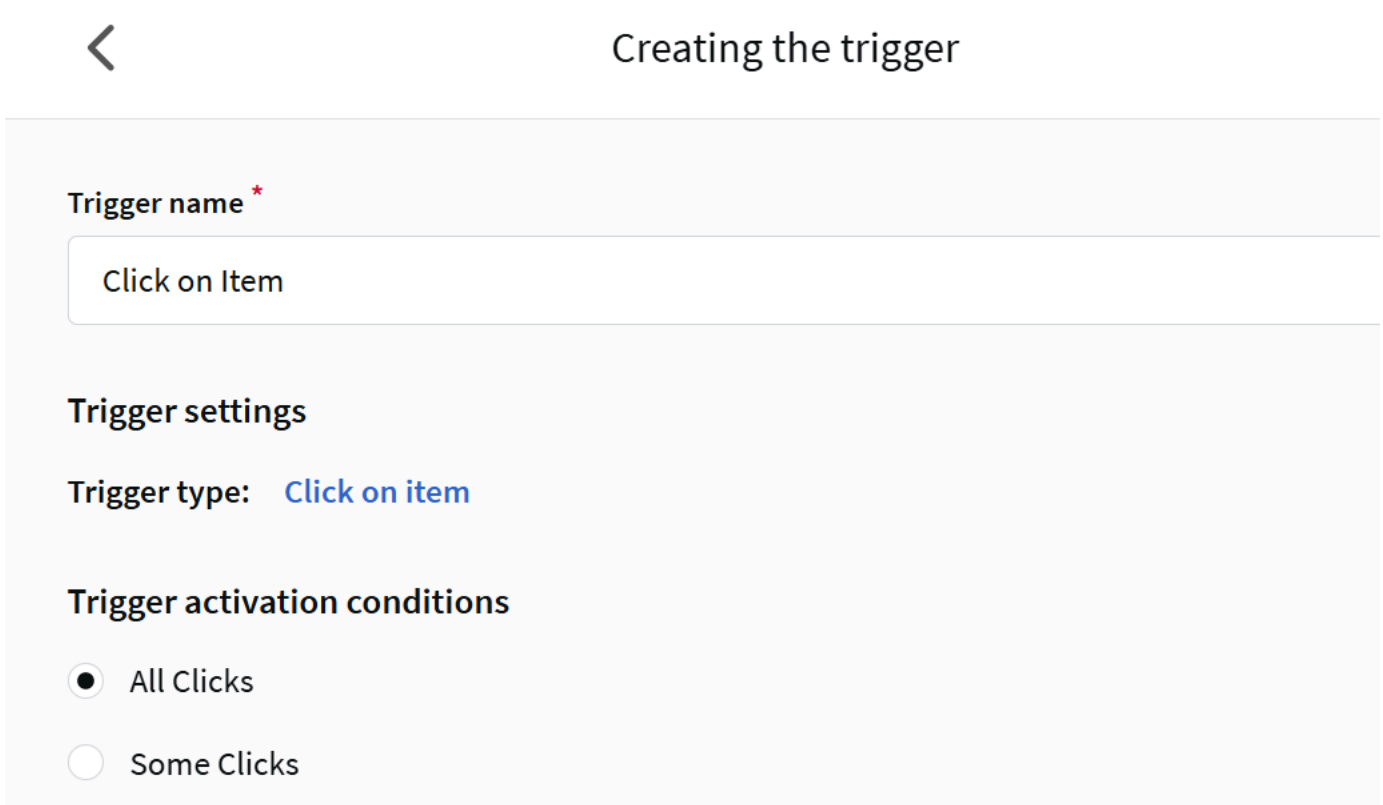
6. To set up a new trigger, click + Create:



7. Enter the name of the trigger and select its type.

TIP

Read more about trigger types in [this article].

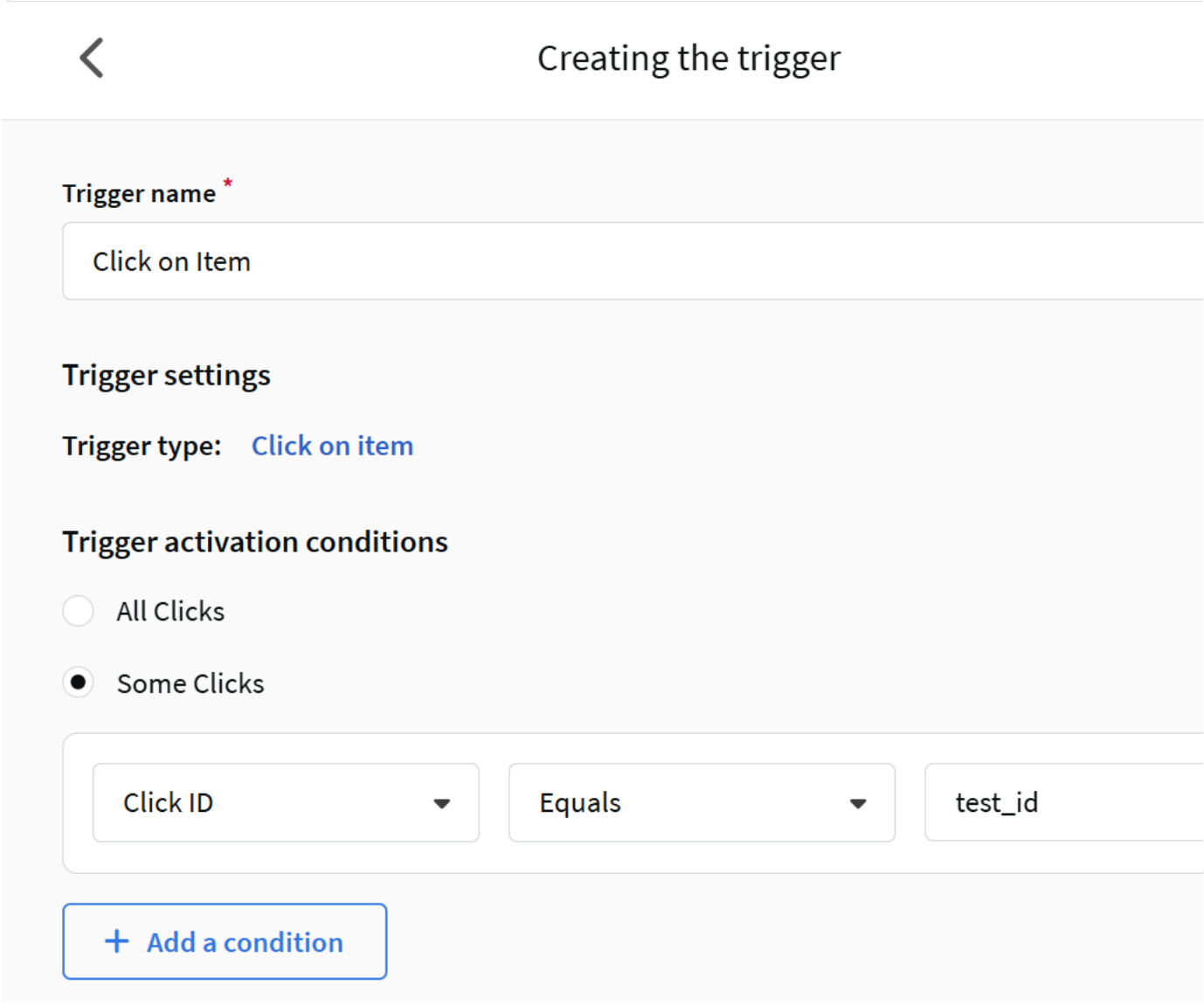


8. By default, triggers fire in response to all events of a certain type. To set additional trigger activation this, in the Trigger activation conditions block, select Some events. After configuring the filter, save the of the page.

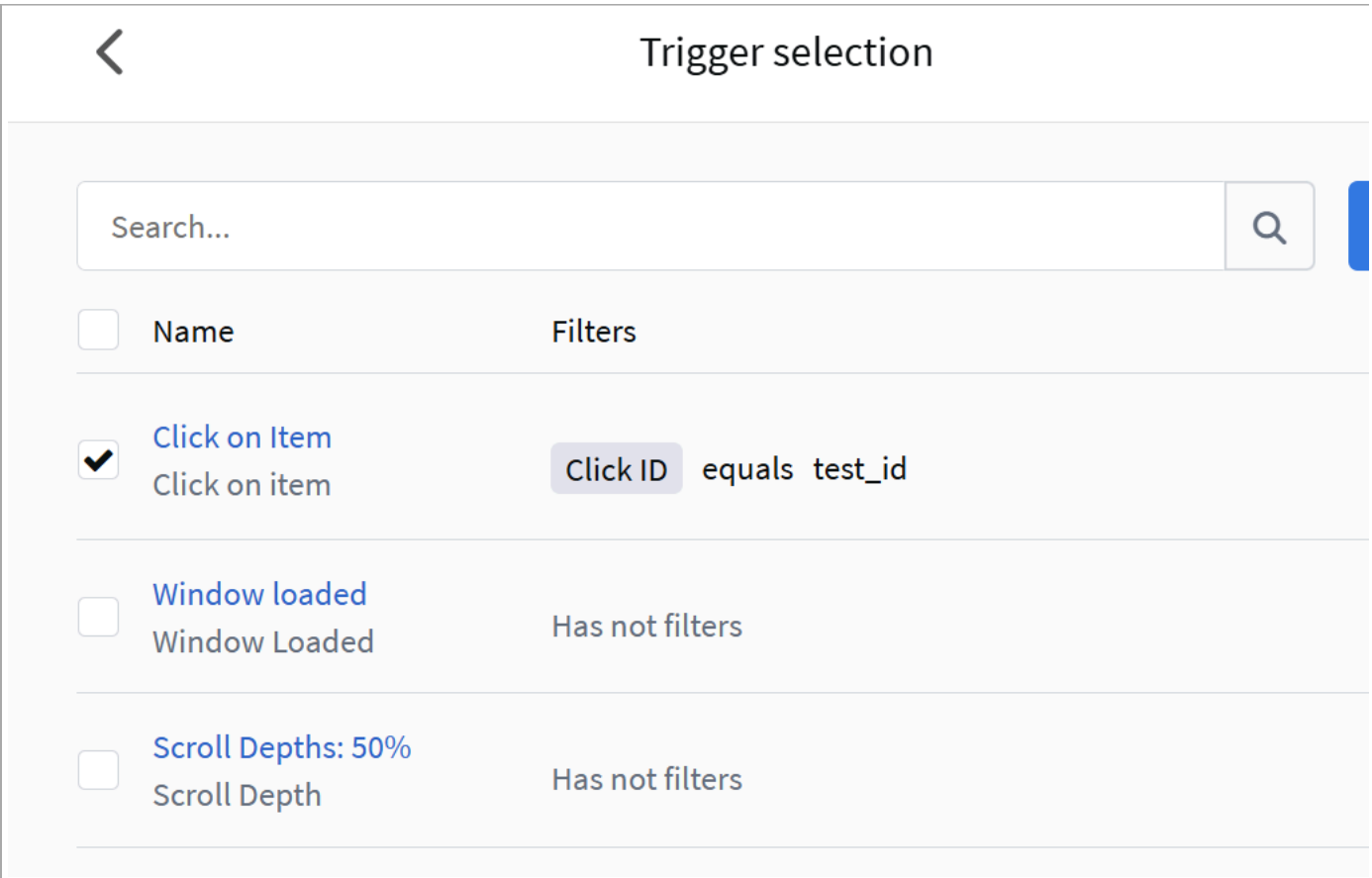
INFO

A filter consists of a variable, an operator, and a value. Select a [variable] (built-in or custom) and an ope a text value for the variable.

Each event on the site is associated with a set of variables that contain information about it. When an e specific variable. Tag Manager compares the value of this variable with the value specified in the filter. the parameters specified in the filter, the trigger is activated and fires the tag.
Thus, using a filter, you can limit the audience for which the tag will be activated.



9. In the Trigger selection window, select one or more triggers and click the Select button at the bottom triggers, the tag will be activated when any of them fires.

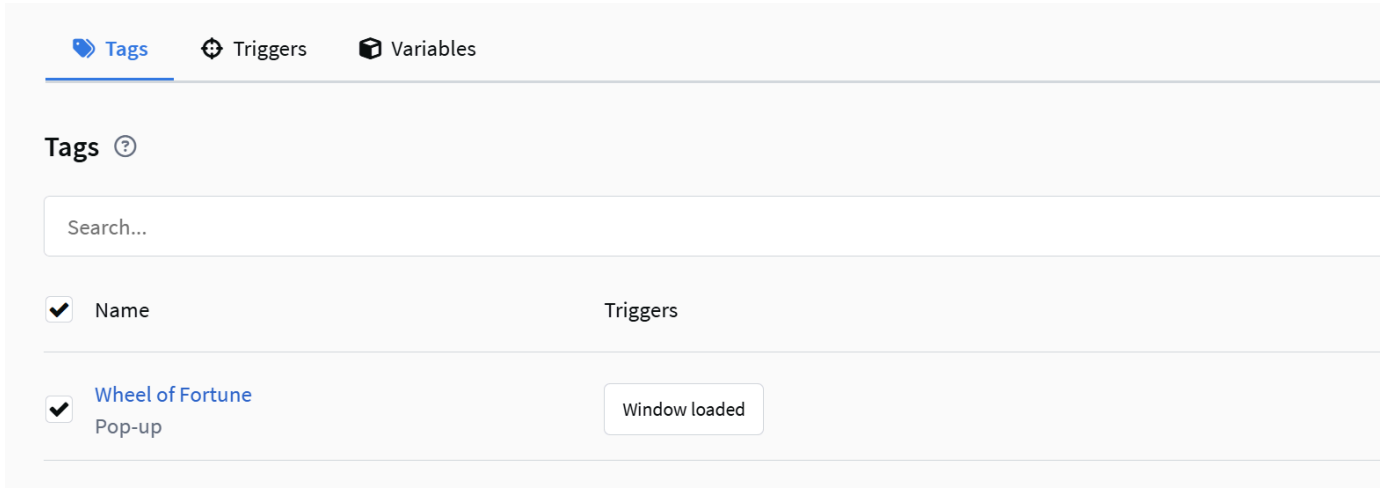


10. In the Tag creation window, check the settings and click Save.

The created tag appears on the Tags tab. To change its settings later, click its name. To temporarily sto the tags and click

, and then publish a new container version. You can also remove unnecessary tags from the container us

.



Step 3. Publishing a container version

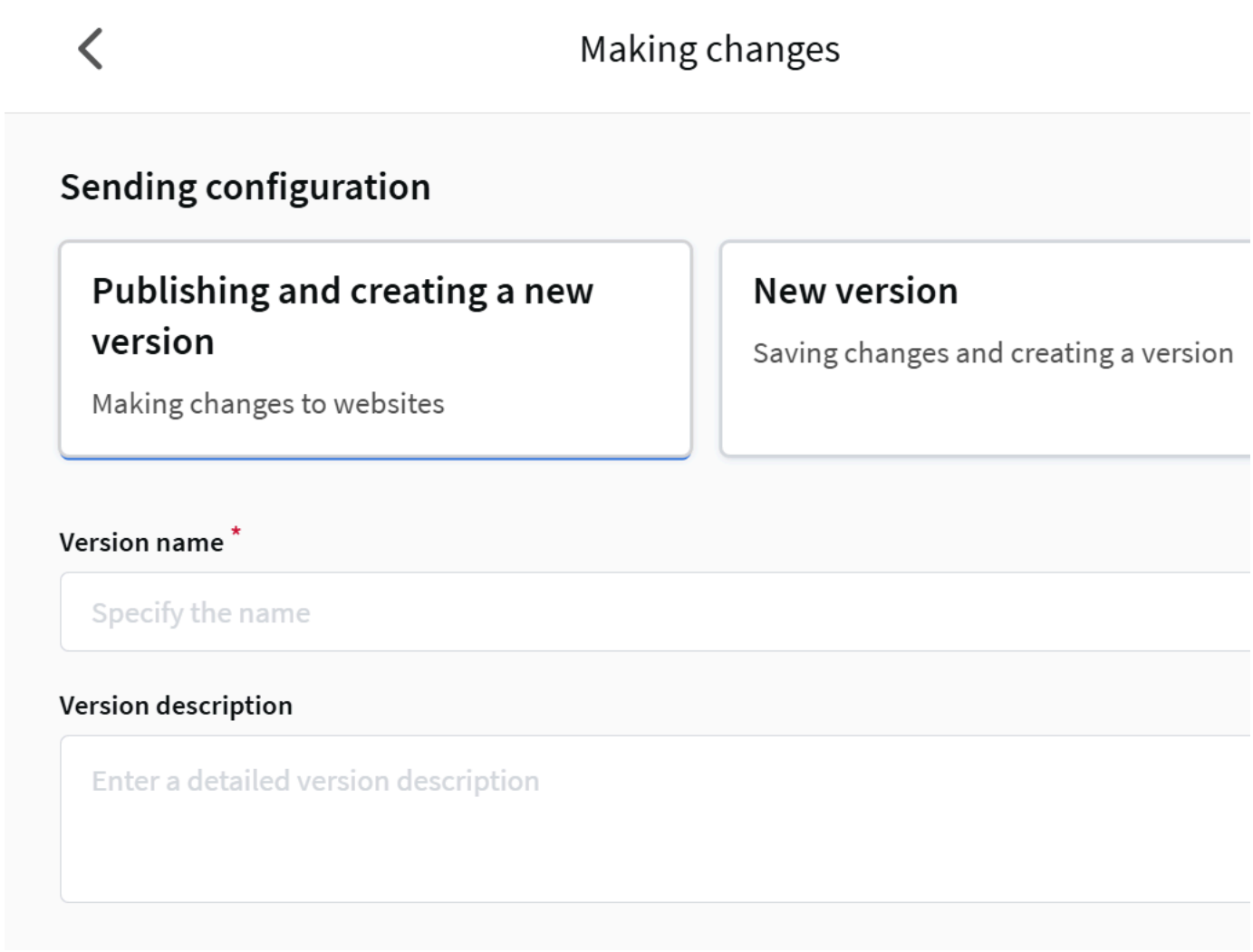
A version is a saved copy of the container at a specific point in time. If you make changes to the contain can save these changes as a version. To do this, in the Container setup window click the Publish button. available:

- **Publish and create a new version — a new version of the container is created and all changes are ap**
- **New version — a new version is created and all changes are saved, but the container is not publishe site.**

CAUTION: Checking variables before publishing

Before publishing, the system checks that all variables used in HTML tags exist in the container. If unkı example, removed or renamed ones), publishing is canceled and an error message is shown.

It is recommended to provide a name and description for the version so you can see exactly what change by clicking Publish at the bottom of the page.



To open the container version list, use the

button in the quick actions menu. The table shows which user created or published each version and wh You can switch to another version or publish it instead of the current one. To do this, open the additional

and click Select as the latest version:

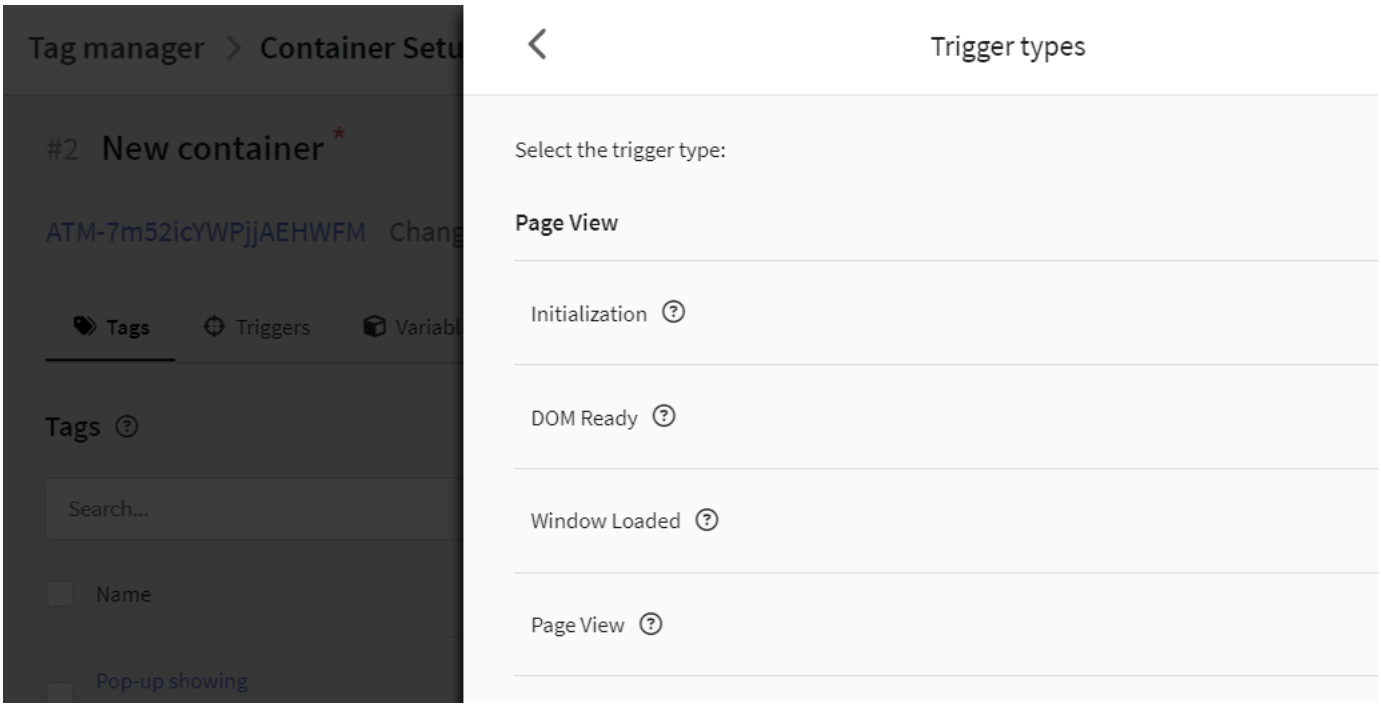
Versions

Status	Version name	Version description	Creation date	Published
			29 Sep 2025 at 14:40	29 Sep 2025 at 14:
			29 Sep 2025 at 14:47	29 Sep 2025 at 14:
			29 Sep 2025 at 14:58	29 Sep 2025 at 14:
			29 Sep 2025 at 15:00	29 Sep 2025 at 15:

The first version of the container has been published and Altcraft Tag Manager has been installed on you tag is working correctly.

Trigger types

Page view



Initialization

The trigger fires before all other triggers fire. Use this trigger to have it fire the tags you want before oth

DOM ready

The trigger fires after the browser has completed processing the HTML code of the page.

The browser has fully loaded the HTML and the DOM tree has been built, but external resources such as have been loaded yet.

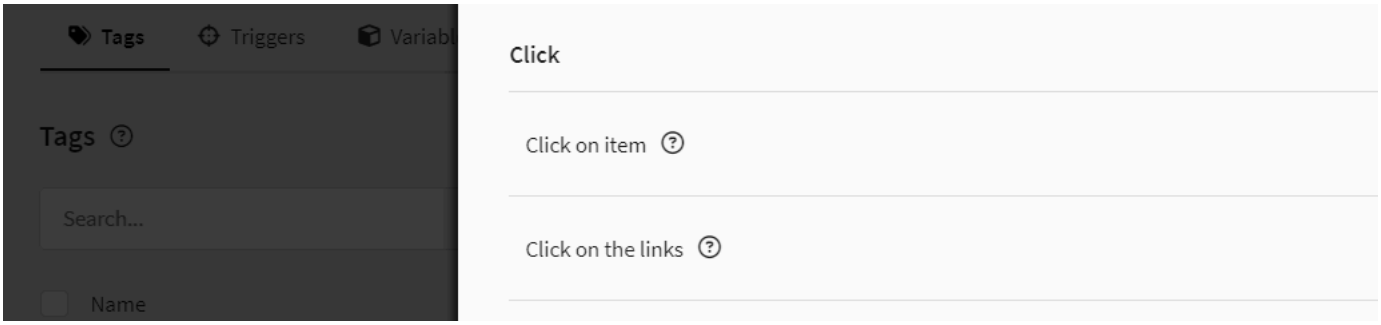
Window loaded

The trigger fires after the page has fully loaded, including images, scripts, and other embedded objects.

Page view

The trigger fires when the page begins to load. Used to obtain data only on page impressions.

Click



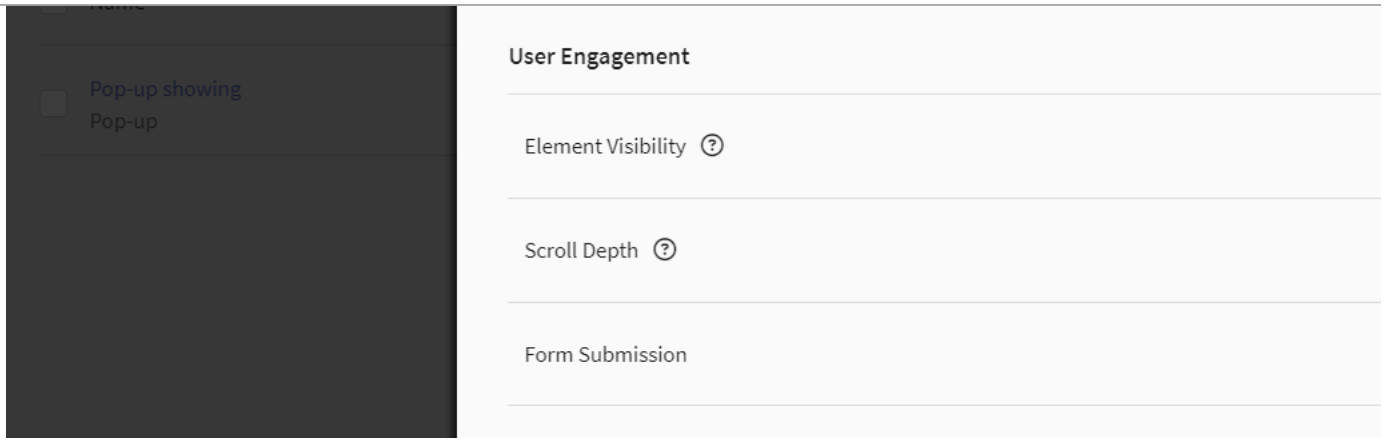
Click on item

The trigger fires when you click on any page item, including links, images, buttons, etc.

Click on the links

The trigger fires when HTML links are clicked, for example, `<a hrefs="https://example.com">Link </a>`

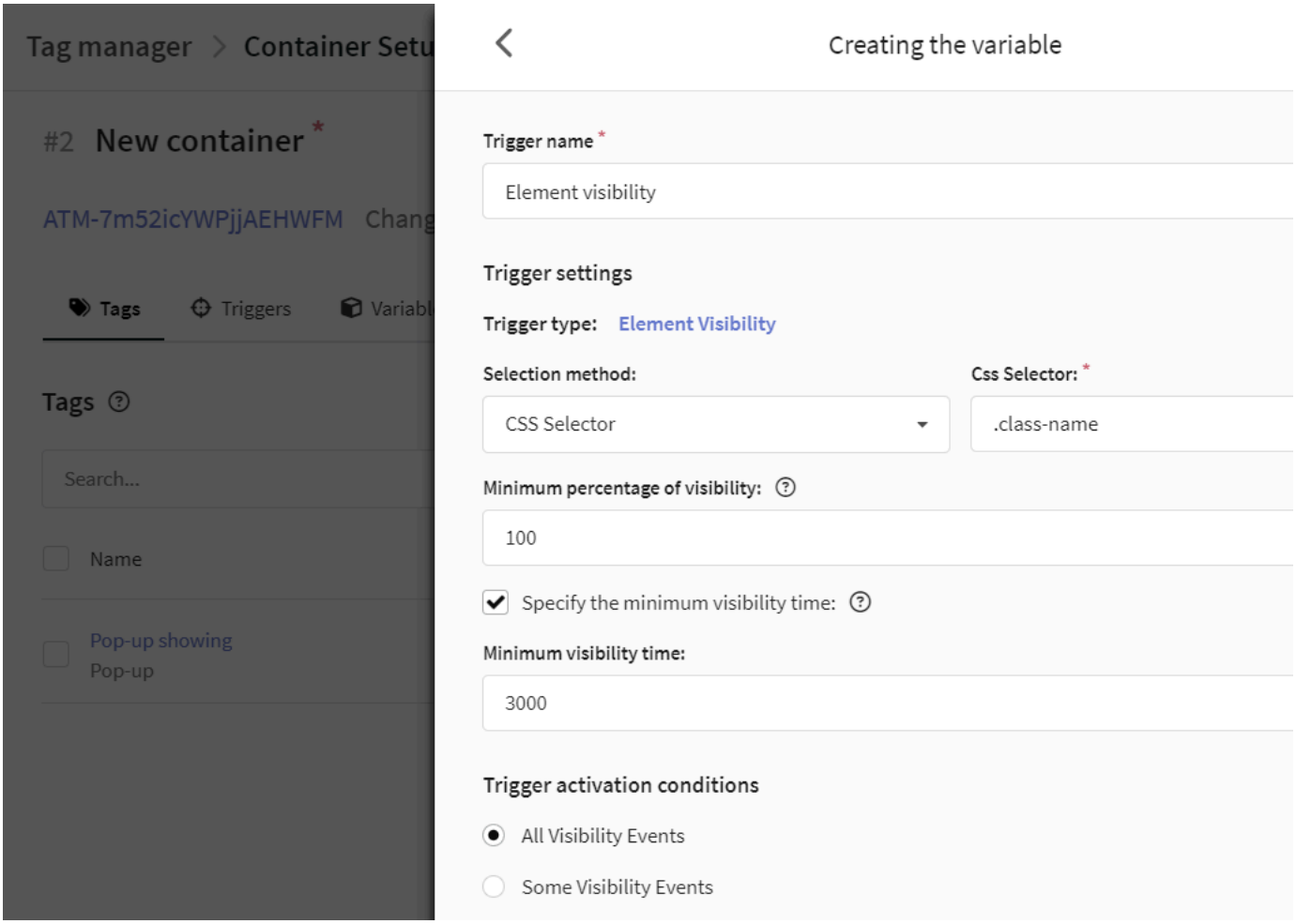
User engagement



Element visibility

The trigger fires when a specific element becomes visible in the viewport.

The visibility of an element can change as a result of various actions: loading the page, scrolling, switch changing the position of the element through code, etc.



Selection method:

- **Identifier is used to track the visibility of a single element with a specific `id` attribute value. For exa element with `id="test"` will be selected.**
- **CSS selector is used to track the visibility of multiple elements that match a given CSS selector patt select elements on the page with `class="class_name"`.**

Minimum percentage of visibility - the percentage of the selected element that must be visible on the us default is 100%, which means the element should be completely displayed on the screen.

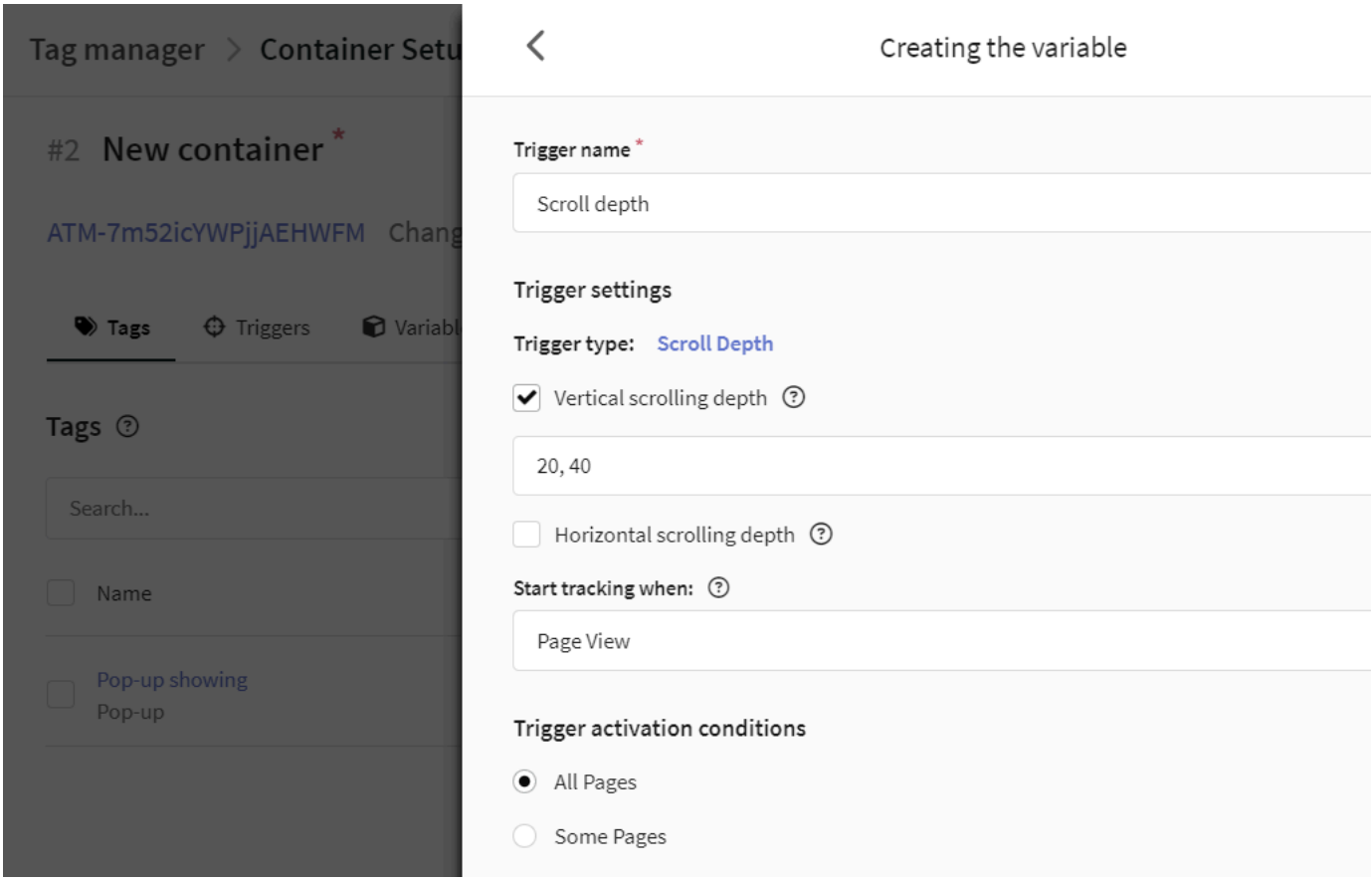
You can also specify the minimum visibility time. The minimum visibility time is the period of time an ele screen for the trigger to fire.

TIP

Please note that time periods are cumulative. If an element was visible for 5000 ms, then hidden and vi visibility time would be 8000 ms.

Scroll depth

The trigger fires depending on how far the user scrolls the page down or to the right.



To configure the trigger, you must select at least one of the scrolling options:

- **Vertical scrolling depth — the trigger is activated based on how far the user has scrolled down the p**
- **Horizontal scrolling depth — the trigger is activated based on data about how far the user has scrolle**

You can use both options at the same time. The scroll depth value is specified in pixels or as a percenta you specify multiple values separated by commas, the trigger is activated when each of them is reached

You can also specify at what stage of the page load you want to start tracking the event:

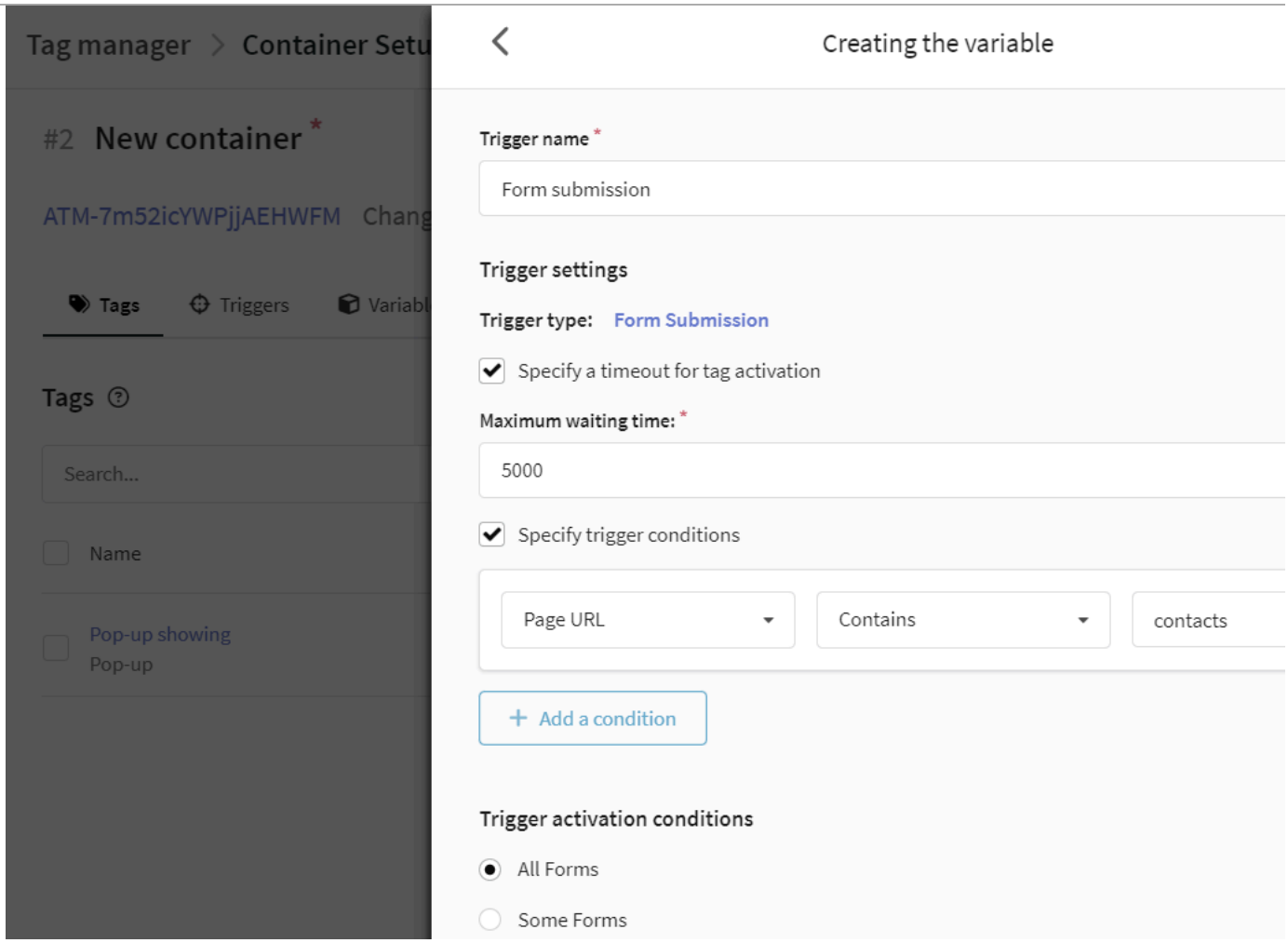
- **Window loaded;**
- **DOM model is ready;**
- **Page view.**

TIP

If the specified scroll depth enters the viewport when the page loads, the trigger fires without scrolling

Form submission

The trigger fires every time the user tries to submit the form.



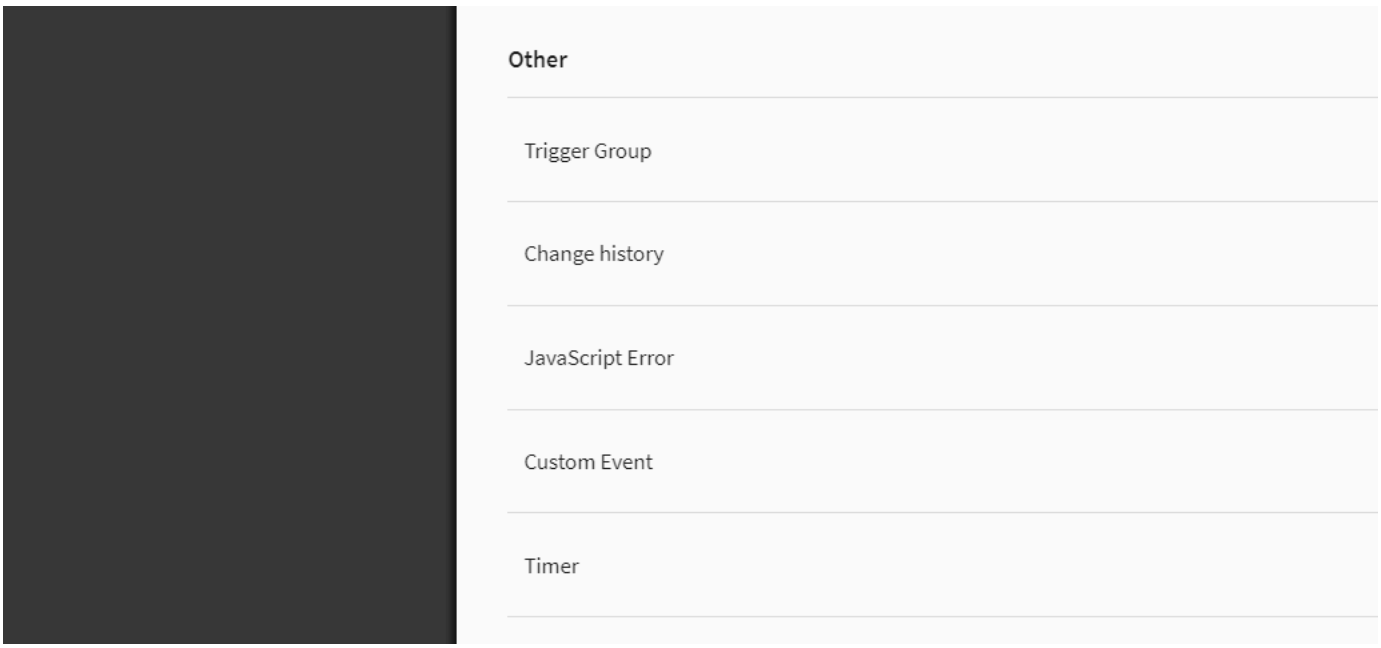
In trigger settings you can:

- **Specify a timeout for a tag activation: form submission is blocked until the specified timeout has exp this trigger have been activated. The wait time is specified in milliseconds (5000 ms = 5 sec).**
- **Specify trigger conditions: use variables to select under what conditions the trigger should monitor e**

TIP

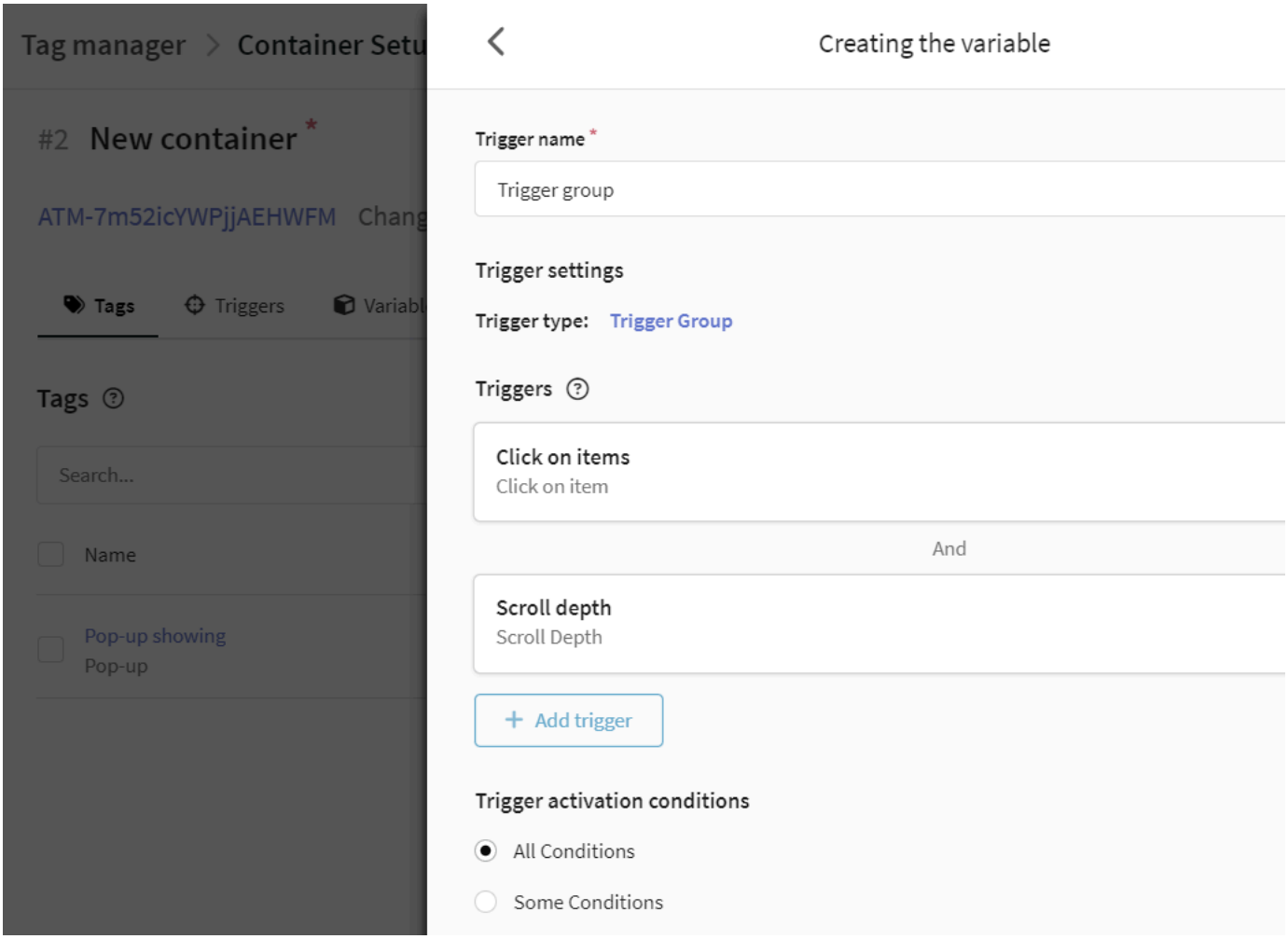
It is recommended to use a filter in the trigger indicating the URL of the page on which the form is loca

Other



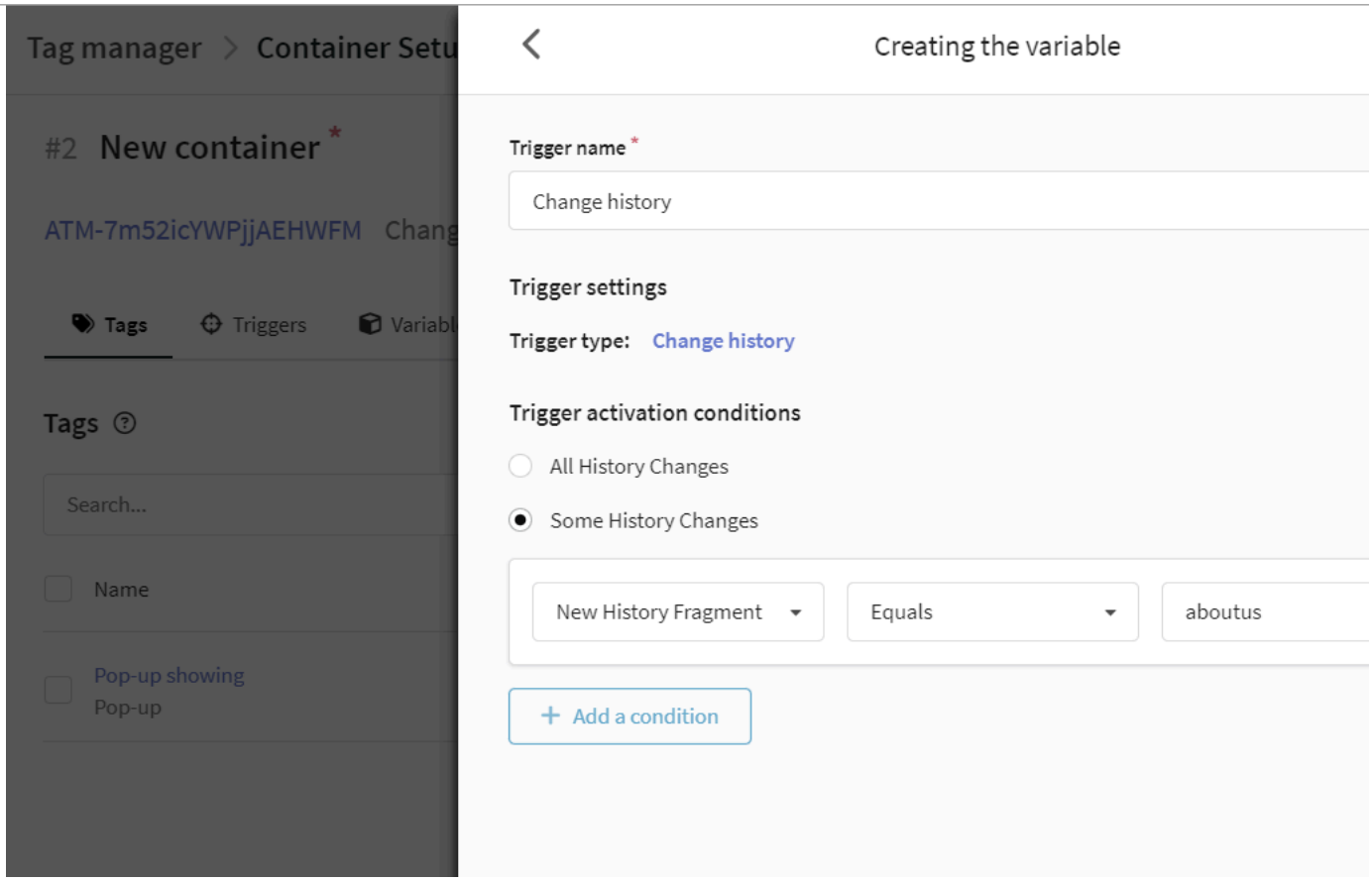
Trigger group

The trigger fires after all selected tags have been activated at least once.
Use the Add Trigger button to select the required triggers. The number of triggers in a group is not limite



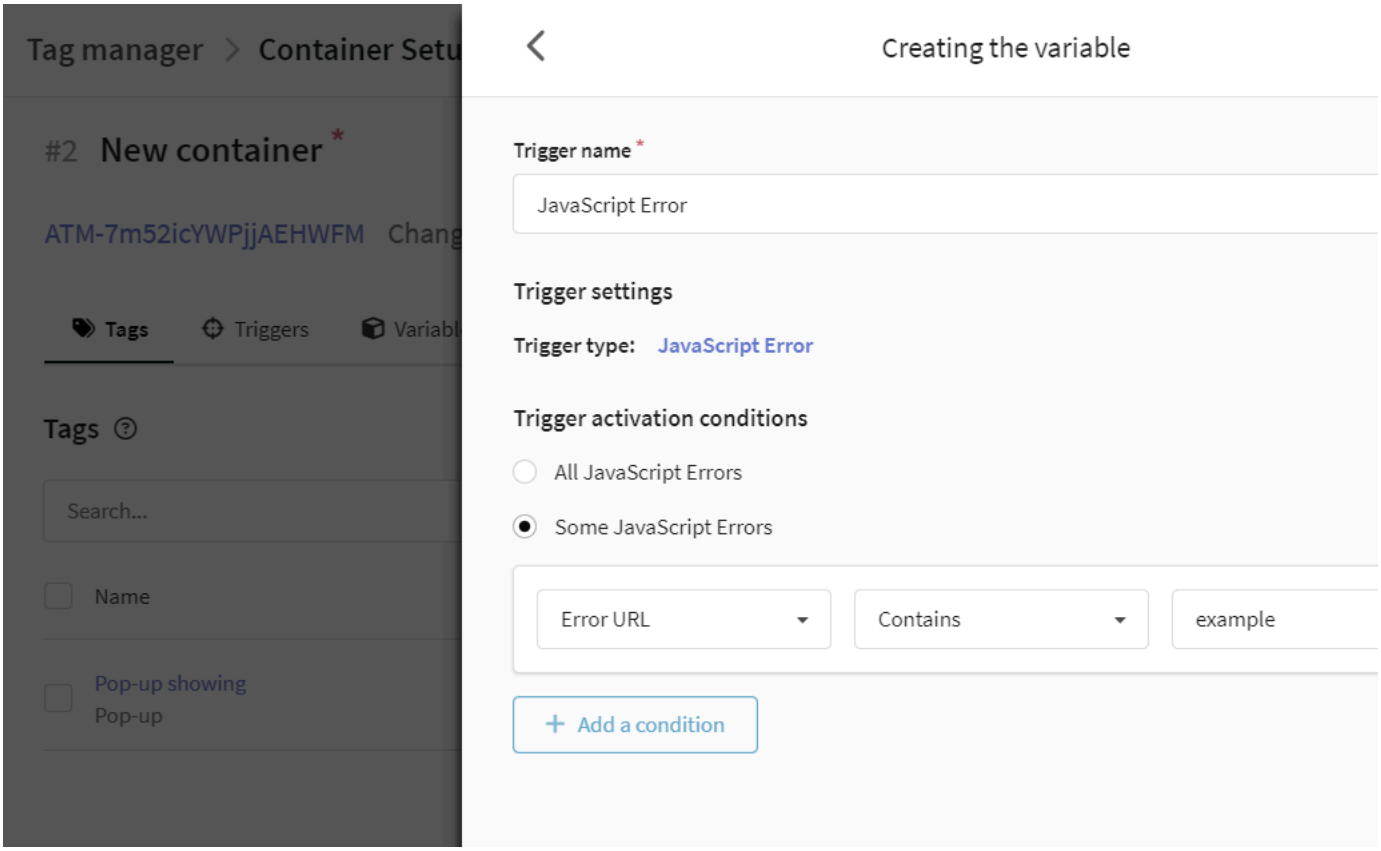
Change history

The trigger fires when there is a change in the URL fragment (hash) or when the pushState HTML5 API is



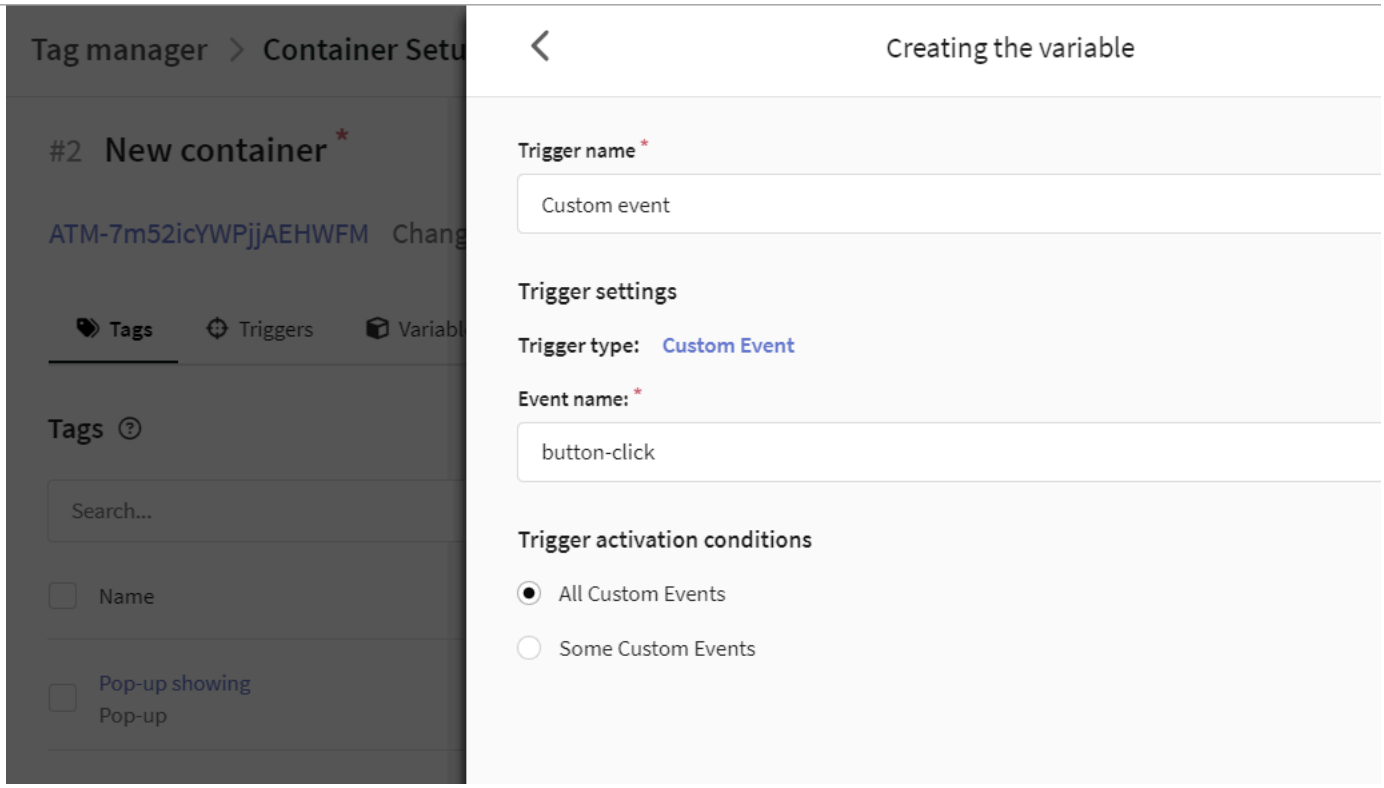
JavaScript error

The trigger fires when a JavaScript error occurs on the site that was not caught by the window.onError e



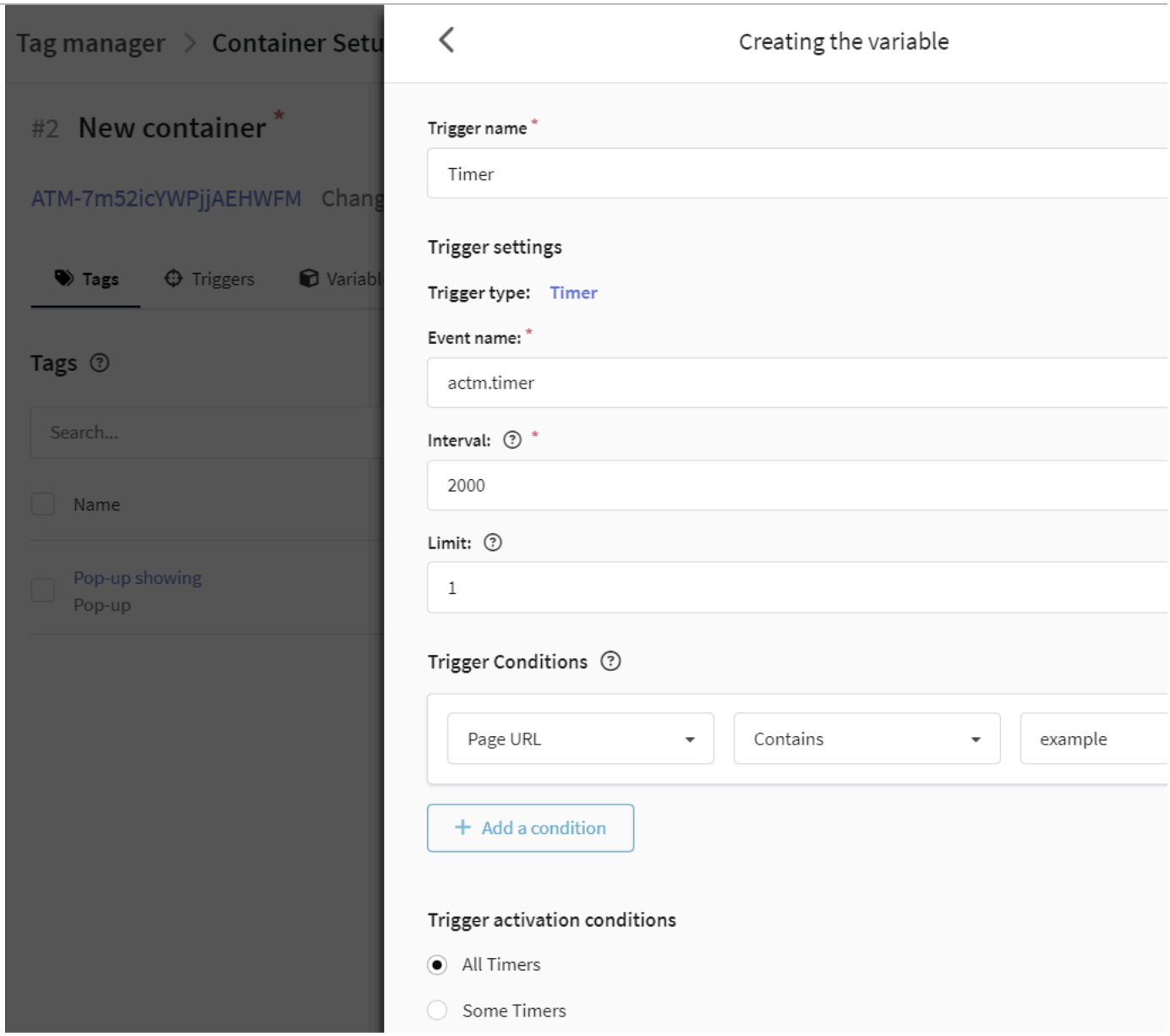
Custom event

A trigger allows you to track interactions with a site or application for which standard methods are not s
track the submission of a form for which you have overridden the behavior of the “submit” event in the b
To configure a trigger, you need to specify the name of the event, for example, `button-click`**.**



Timer

The trigger fires after a specified time interval.



Trigger settings include:

- **Event name by default — `actm.timer`. In most cases, you do not need to change the name. You can sp trigger based on a custom event.**
- **Interval — the time between event firings (in milliseconds). For example, if you set it to 2000, the eve first firing will also occur after the specified time.**
- **Limit — the maximum number of firings. If you leave this field empty, the trigger will fire until the use**

TIP

If you set the interval to 2000 ms and the limit to 1, the first and last firing will occur after 2 seconds.

Variables types

Description

A variable is an object that is given a name and can take on different values during code execution on th variable name is constant, but the value changes depending on what actions occur on the site. The varia contents of a page. For example, the “Page URL” variable returns the URL of the current page.

Variables are used in both triggers and tags:

- **In triggers variables are used to set up filters that determine when certain tags will be triggered. In targeting — by setting conditions for activating a tag, they limit the audience to whom certain conte example, a pop-up.**
- **In tags variables are used to obtain dynamic values.**

To manage container variables, go to the “Variables” tab. Built-in and user’s variables are available in Ta configurations, built-in variables are sufficient. However, you can also create your own variables to suit y

> **CAUTION**
>
> **Please note that when setting up a trigger, it is not recommended to use the same variable several time**

Using variables in tags

Built-in and user’s variables can be used in HTML and JavaScript tag code:

HTML

```
<div class="banner">
  <h3>{{Promotion name}}</h3>
  <p>Special offer for {{City}}</p>
</div>
```

JS

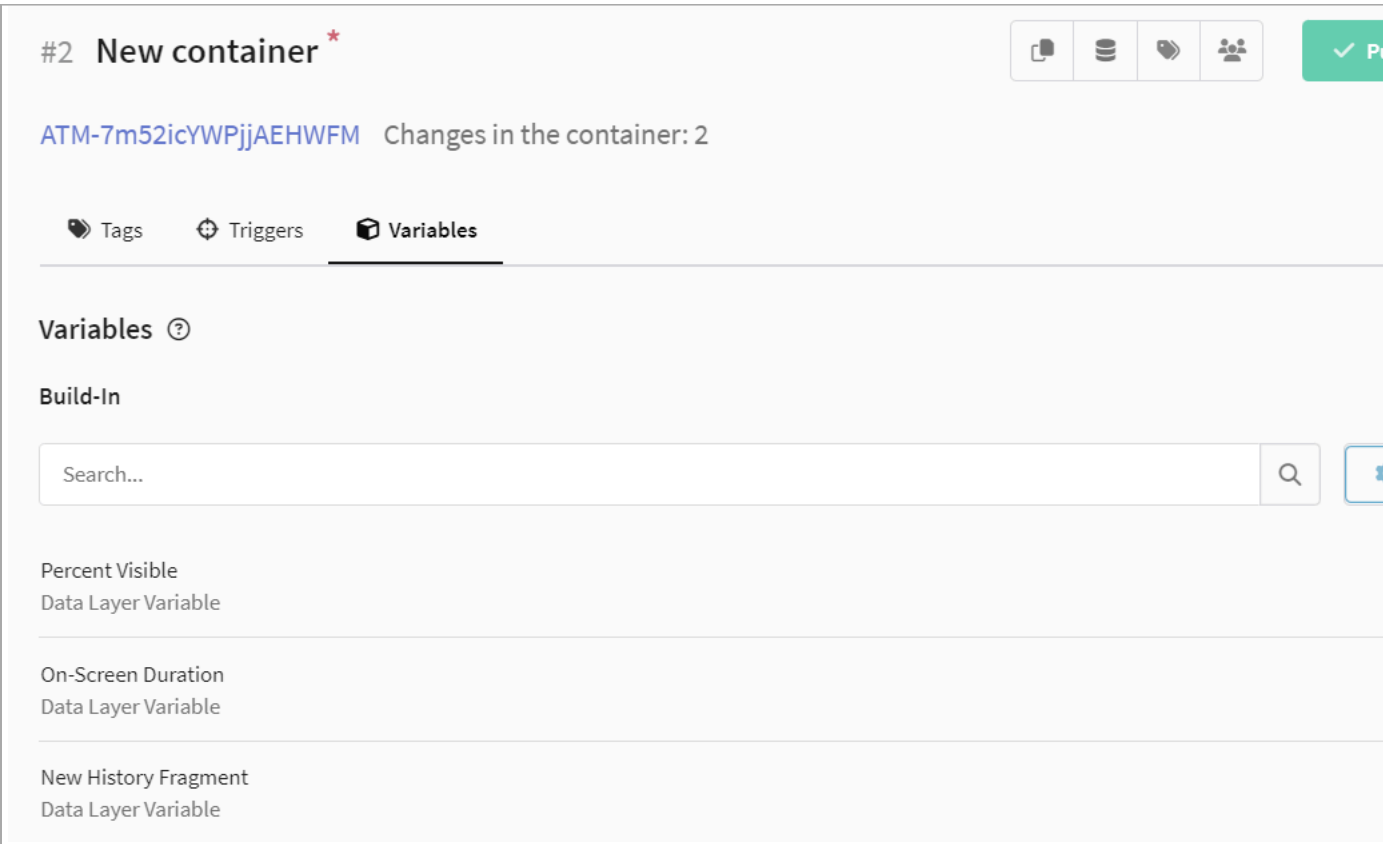
```
<script>
console.log('Current user:', {{User ID}});
analytics.track('page_view', { url: {{Page URL}} });
</script>
```

When you publish a container, the system checks that all used variables exist. If unknown variables are f deleted or renamed), the publication is canceled and an error message is shown.

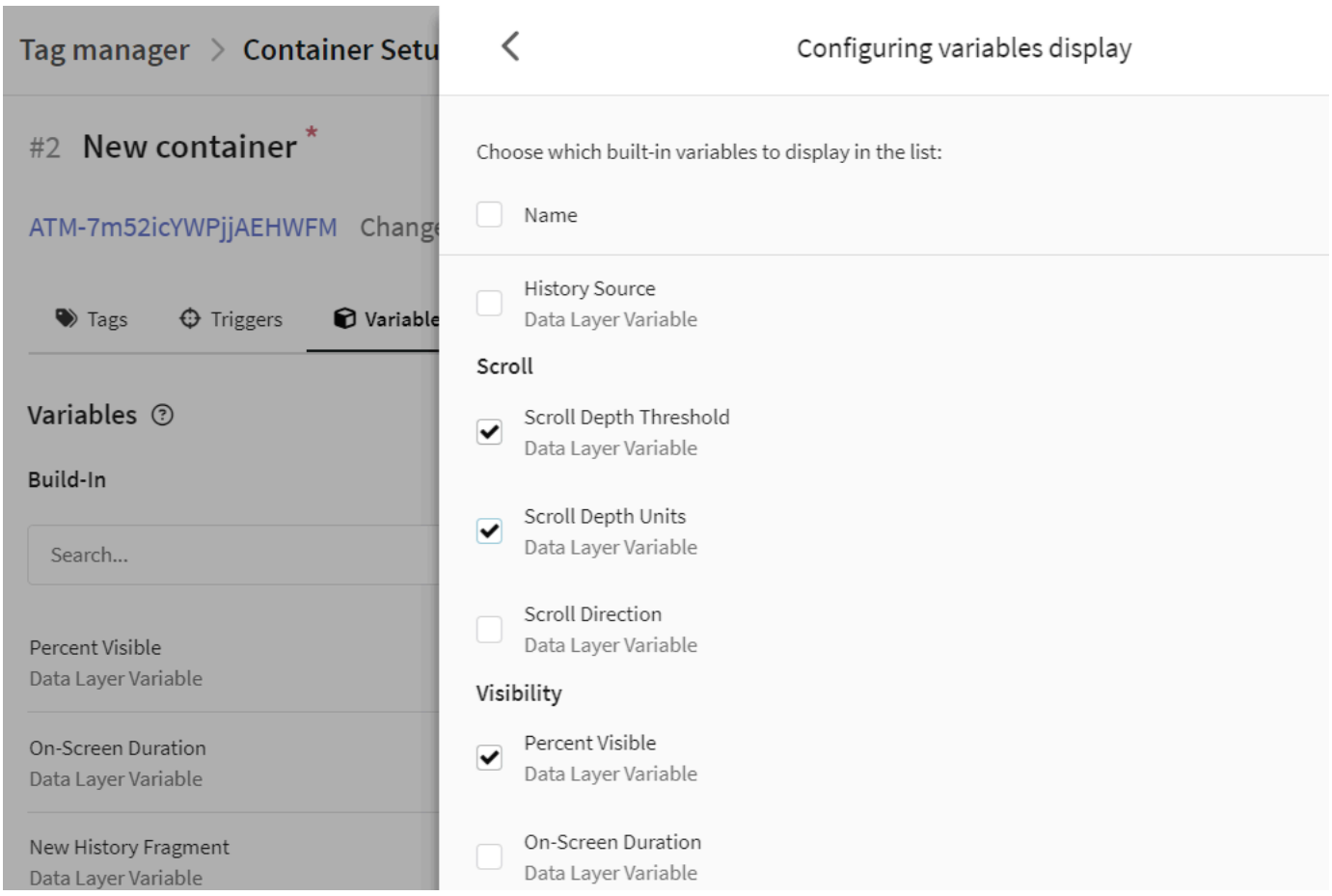
> **CAUTION**
>
> **If you change a variable name, you must update it in all tags where it is used.**

Built-in variables

Built-in variables are ready-made variables that are automatically added to a new container. Such variab



By default, all built-in variables in a container are active. If you do not want to display a variable in the li the window that opens, uncheck the box next to its name. Then click the Select button at the bottom of



Types of built-in variables

Pages

- **Page URL. The variable returns the full URL of the current page, but without the hash. For example, i `https://example.com/test?param=value#hash`, then the variable will return the value `https://example.com/test?pa`**
- **Page Hostname. The variable returns the domain name in the URL of the current page. For example, `https://example.com/test?param=value`, then the variable will return the value `example.com`.**
- **Page Path. The variable returns the path to the page in the current URL without taking into account user loaded the page `https://example.com/test?param=value`, then the variable will return the value `/test`.**
- **Referrer. The variable returns the full URL of the page from which the user navigated to the current p**

Utilities

- **Container ID. The variable returns the container number in the tag manager. For example, ATM-4ynTV**

Errors

- **Error Message. The variable returns the error message sent by the JavaScript error trigger.**
- **Error URL. The variable returns the URL of the script in which the error was detected.**
- **Error Line. The variable returns the line number of the file where the error occurred.**
- **Debug Mode. The variable returns the value “true” if debug mode is activated in the Tag Manager.**

Clicks

- **Click Element. The variable returns the HTML element that the user clicked on.**
- **Click Classes. The variable returns the value of the `class` attribute for the HTML element that was cl**
- **Click ID. The variable returns the value of the `id` attribute for the HTML element that was clicked by**
- **Click Target. The variable returns the value of the `target` attribute for the HTML element that was cl**
- **Click URL. The variable returns the value of the `href` attribute for the HTML element that was clicke**
- **Click Text. The variable returns the `textContent` value of the HTML element that the user clicked on.**

Forms

- **Form Element. The variable returns the JavaScript object for the submitted form. For example, class child elements.**
- **Form Classes. The value of the `class` attribute for the submitted form.**
- **Form ID. The variable returns the value of the `id` attribute for the submitted form.**
- **Form Target. The variable returns the value of the `target` attribute for the submitted form.**
- **Form URL. The variable returns the value of the `action` attribute for the submit.**
- **Form Text. The variable returns a text from the form, which was sent, and from form’s elements.**

History

- **New History Fragment. The variable returns a new hash value (#) of the website’s URL after the user committed.**
- **Old History Fragment. The variable returns the previous value of the URL hash before the user event**
- **History Source. The variable returns a string containing the event that led to the change of the histo**

Scroll

- **Scroll Depth Threshold. The variable returns the numeric value of the scroll threshold at which the S percentage or pixels).**
- **Scroll Depth Units. This variable returns the unit of measure (pixels or percentages) for the threshol Depth trigger.**
- **Scroll Direction. The variable returns a parameter indicating the scroll direction (“vertical” or “horizo**

Visibility

- **Percent Visible. The variable returns a numeric value between 0 and 100 that indicates what percen when the Item Visibility trigger was fired.**
- **On-Screen Duration. The variable returns a numeric value that indicates how long the selected elem Visibility trigger was fired.**

Geo-variables

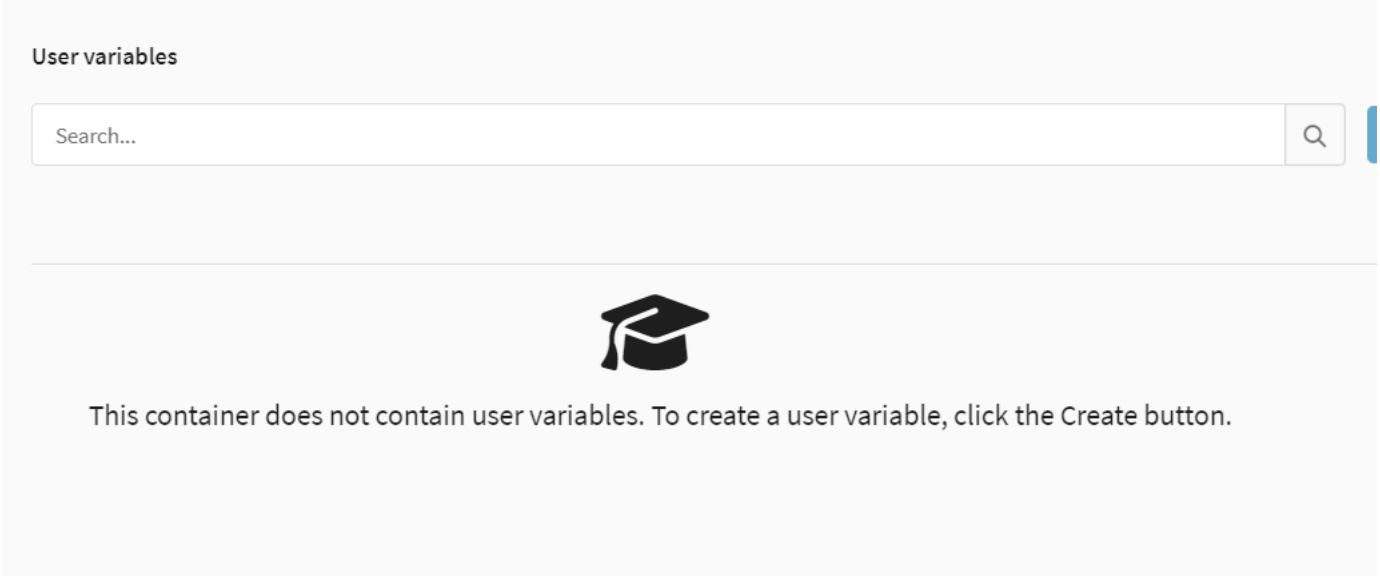
- **Country. The variable returns the country of the profile.**
- **City. The variable returns the profile city.**

Agent

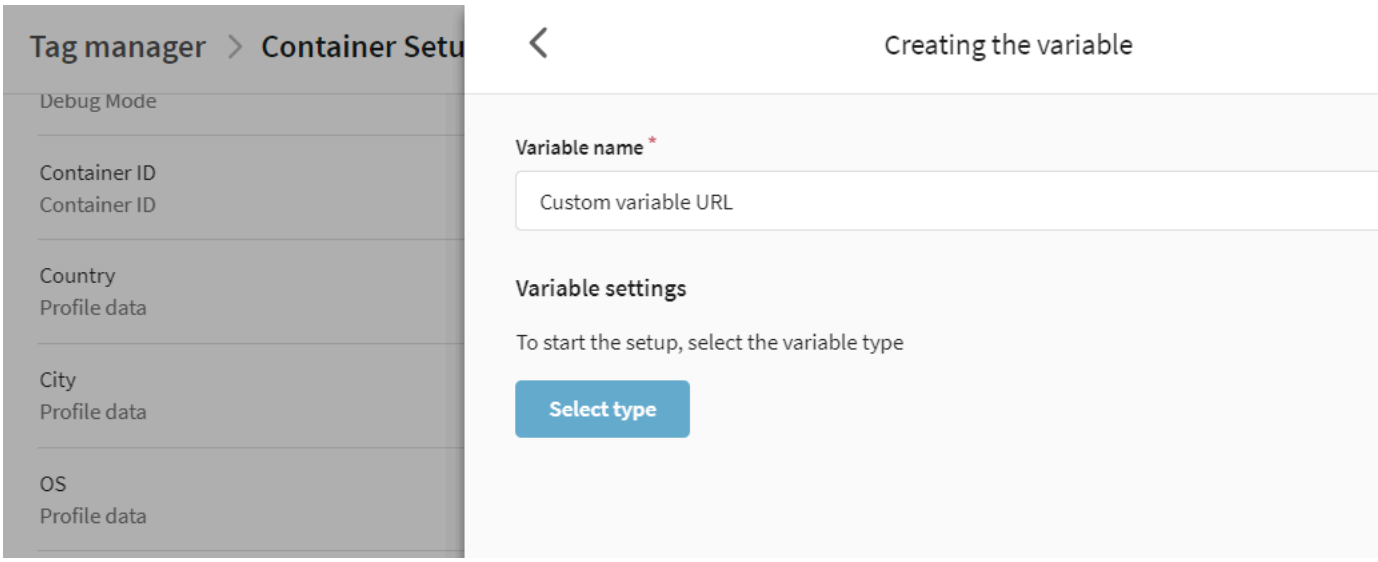
- **Device. The variable returns the profile device.**
- **Browser. The variable returns the profile browser.**

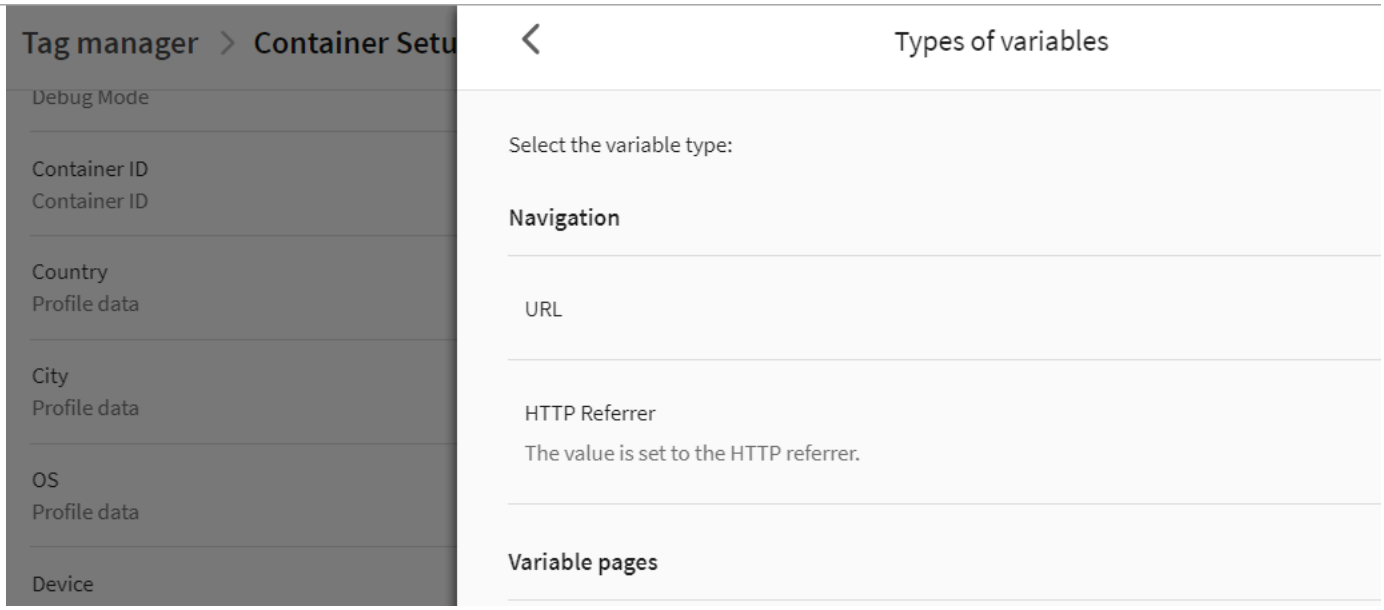
User variables

To create a new variable in the Container Setup window, go to the Variables tab. Scroll to the User varia button:

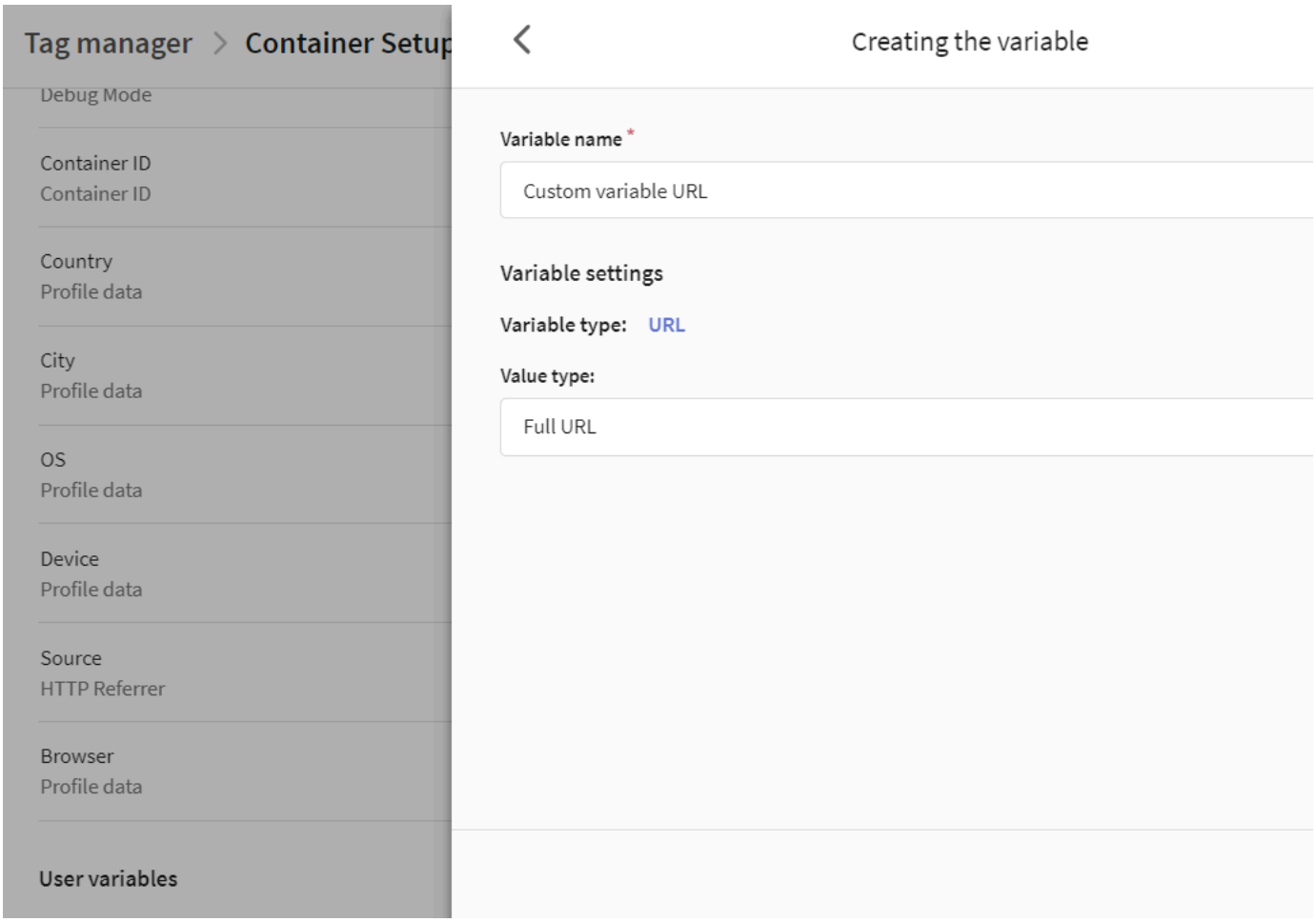


In the window that appears, specify a name for the variable and select its type:



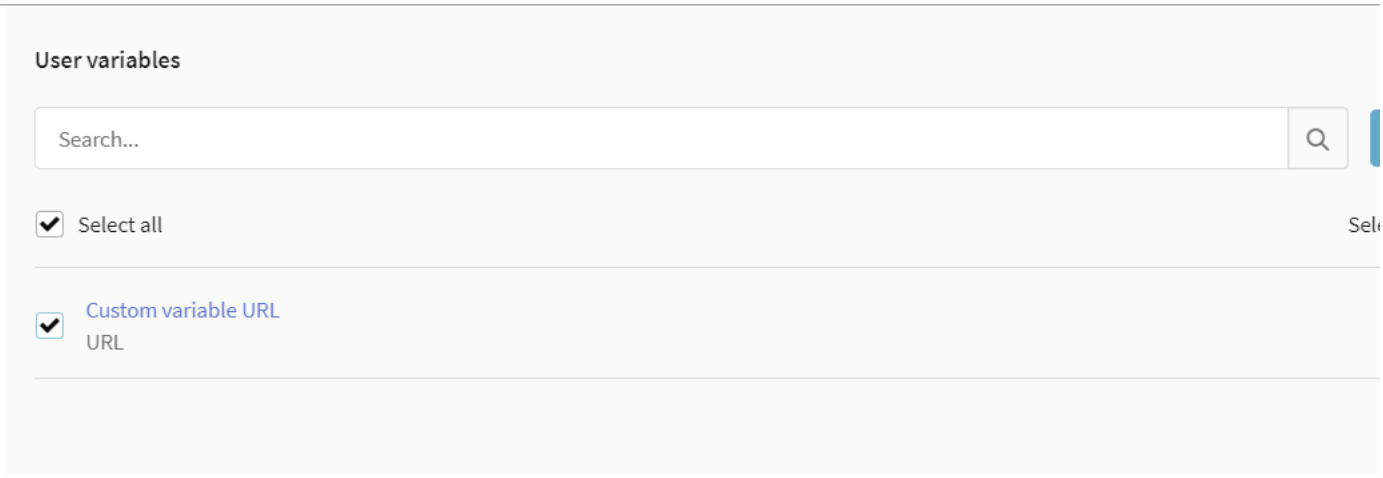


Set variable settings and save using the button at the bottom of the page.



After saving, the variable will appear in the list. To change the settings of a variable, click on its name. Y variables: select them and click

:

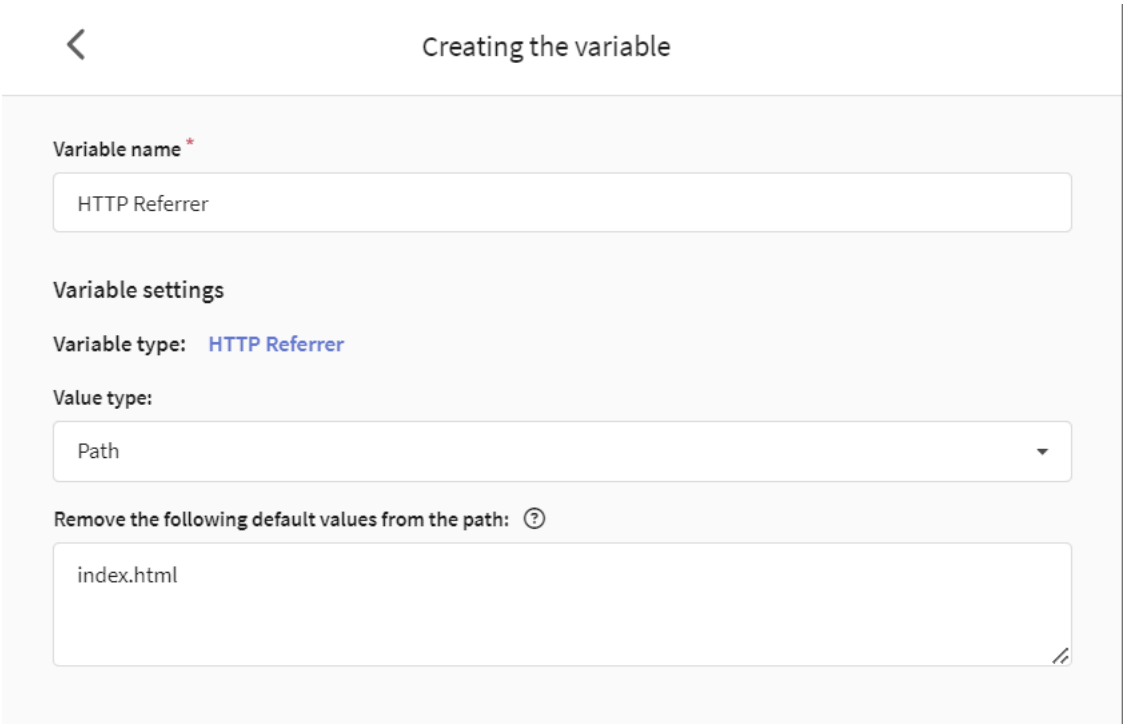


Types of user variables

Navigation

This type of variable allows you to access URL components:

- **URL — Analogue of the built-in variables Page URL, Page Hostname, Page Path. The input value for t page the user is on. The value is retrieved from `document.location`**
- **HTTP Referrer — Analogous to the built-in Referrer variable. The input value for this type of variable value is retrieved from `document.referrer`.**



Value type (using URL example: `http://example.com:80/path/index.html?param1=value1&param2=value2#hash`):

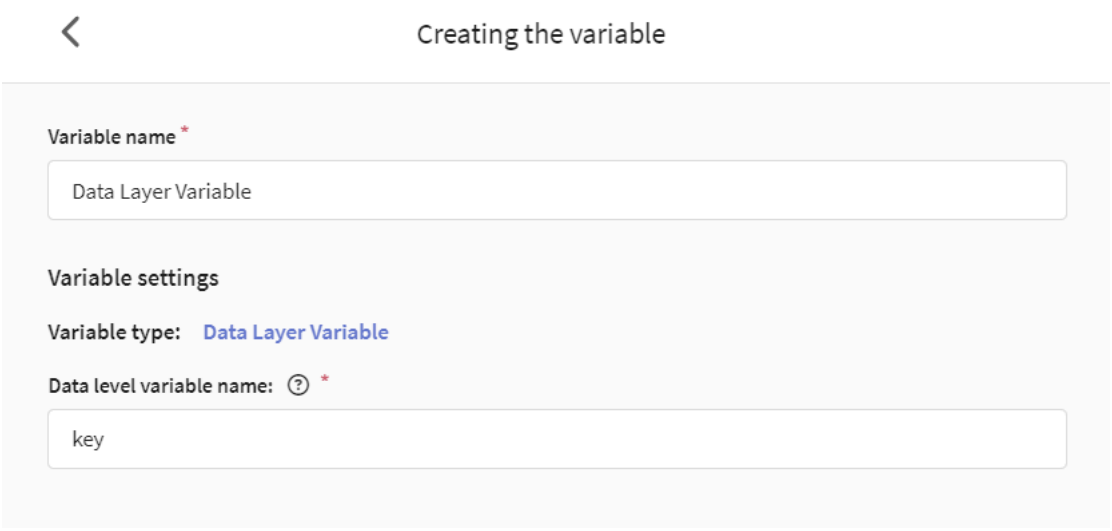
Value	Example	Description
Full URL	**`http://example.com:80/path/index.html?param1=value1&param2=value2`**	**The variable returns the full URL without the hashing fragment.**
Hostname	**example.com**	**The variable returns the hostname of the URL without the port number.** If you enable the "remove www. from hostname" field, the variable will return the same value for www.example. com and example.com. </td>
Request	**param1=value1&param2=value2**	**The variable returns the query parameters (without the "?").** If you specify a query key, the variable will return only the value of that key. for example, for the key <code>param1</code> the variable will return the value <code>value1</code>. </td>
Path	**/path/index.html**	**The variable returns the path in the URL.** In the field "Remove the following default values from the path" you can specify the name of the index file - <code> index.html </code>. in this case, the pages <code> http://example.com/path/index.html </code> and <code>http://example.com/path/</code> will be considered as one page. You can also specify multiple default values - each value on a separate line. in this case, if the last segment of the path matches the default and is not a directory, it will be removed. </td>
Protocol	**http:**	**The variable returns the URL protocol.**

Port	**80**	**The variable returns the port number.**
File extension	**html**	**The variable returns the file extension.**
Fragment	**hash**	**The variable returns the hash fragment (without the "#").**

Page variables

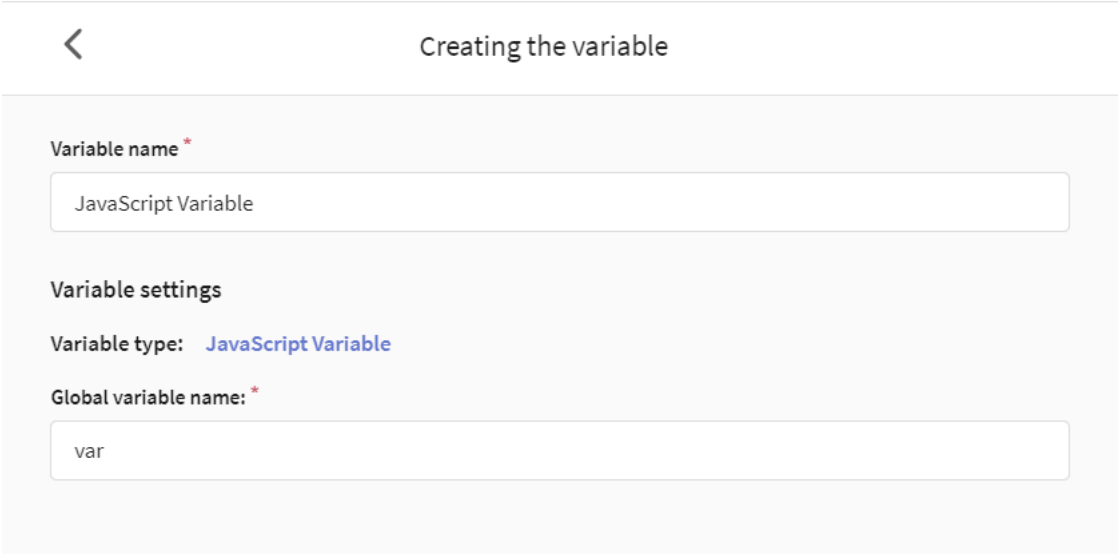
Data Layer Variable

The variable takes a value when information is passed to the data layer using the `acDataLayer.push({'var':'` transmitted in the `key:value` format, where `key` is the name of the variable, which is specified in the "Dat `value` is the value being transmitted.



JavaScript Variable

The variable takes the value of a JavaScript global variable, the name of which is specified in the "Globa variable does not exist, the value undefined will be returned.

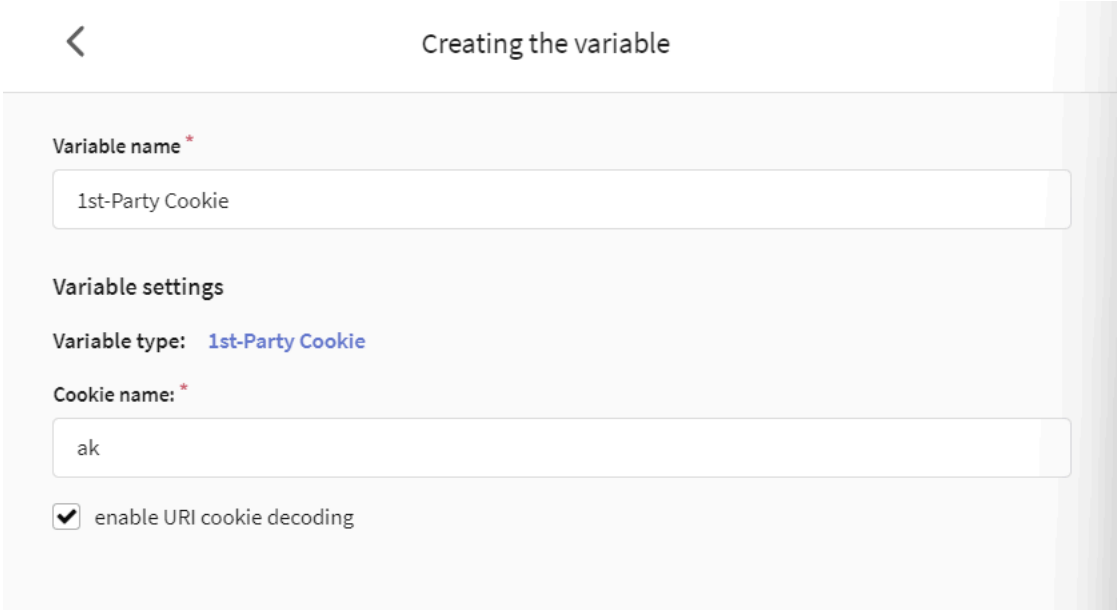


Custom JavaScript

The variable takes the value that is returned by the anonymous JavaScript function.

1st-Party Cookie

The variable takes the user's cookie value available for the current site. The value is retrieved from `docu` name of the cookie whose value you want to transfer to the variable. For example, if the website has the then in the "Cookie name" field you can specify `ak`.
The cookie value is encoded - enable the Enable URI cookie decoding field to decode.



Page elements

Element Visibility

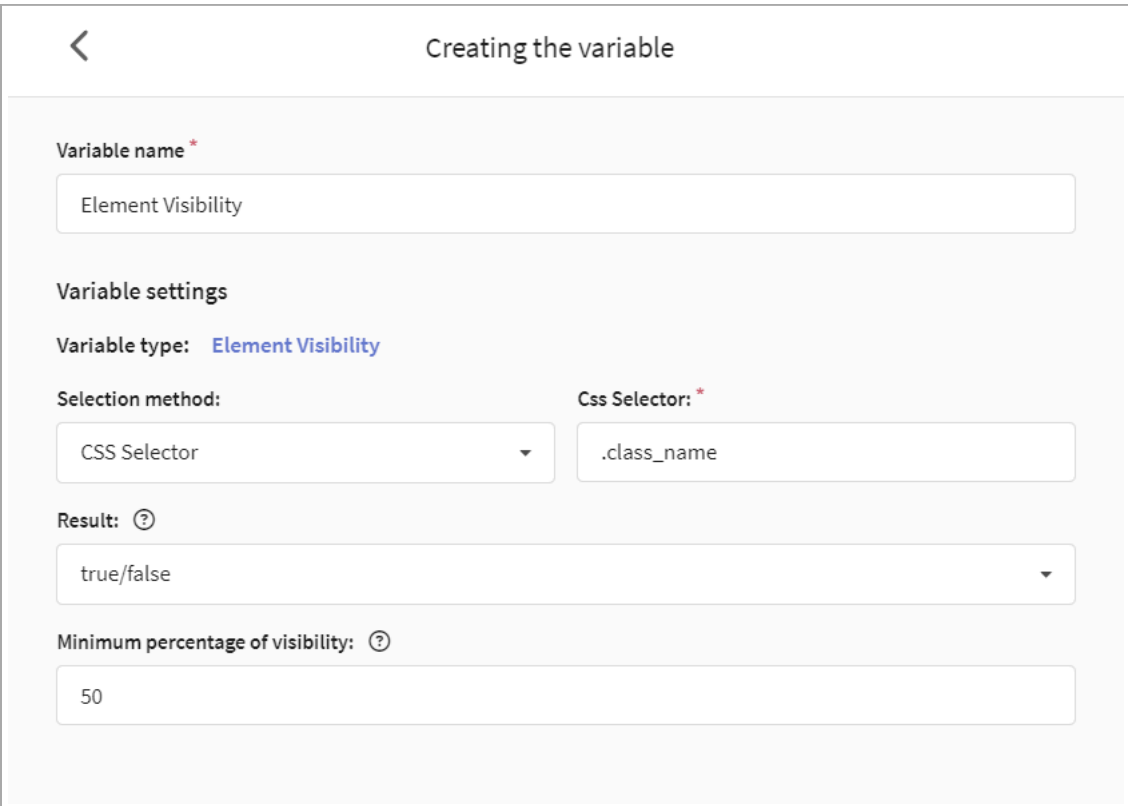
The variable allows you to determine which element was visible in the user's browser. First specify the s

- **Identifier - tracks the visibility of a single element with a specific `id` attribute value. For example, if `id="test"` will be selected.**
- **CSS selector - tracks the visibility of multiple elements that match a given CSS selector pattern. For elements on the page with `class="class_name"`.**

Result defines the return value:

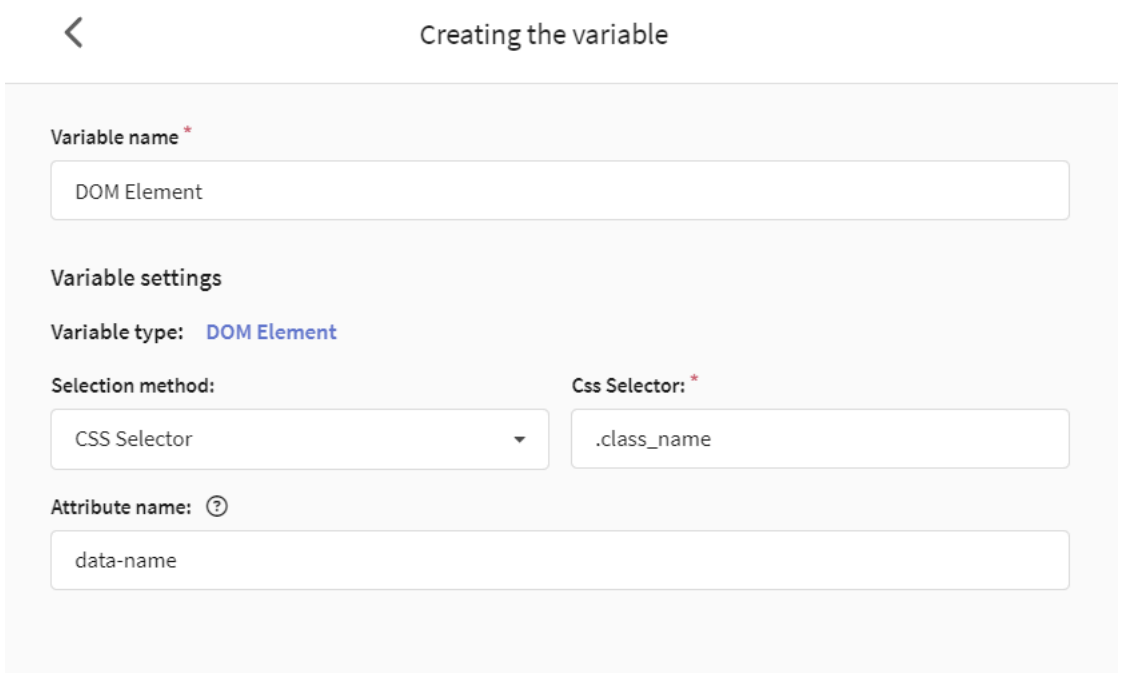
- **True/false - the variable will take the value true if the specified minimum element visibility percenta**
- **Visibility percentage - the variable will take a value equal to the visibility percentage.**

In the example below, the variable will return `true` when half of the element with class `class_name` is disp



DOM Element

The variable is used to retrieve the text content of any DOM element or to retrieve the value of any attrib which elements the variable should track. To do this, specify a selector method and specify an element I If you specify attribute name, the variable will be assigned the attribute value (for example, `data-name="val` specified, the variable will return the text of the DOM element.

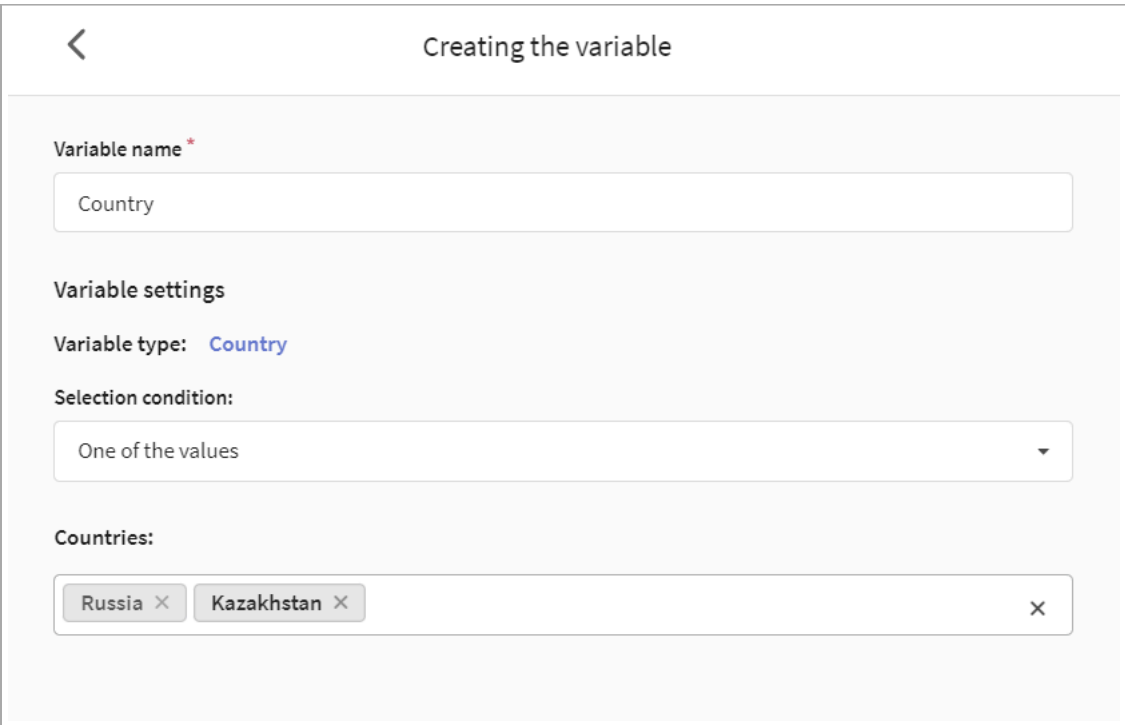


Other

Country

The variable takes the value `true` if the user's country matches the specified selection condition, and `fa` Selection condition:

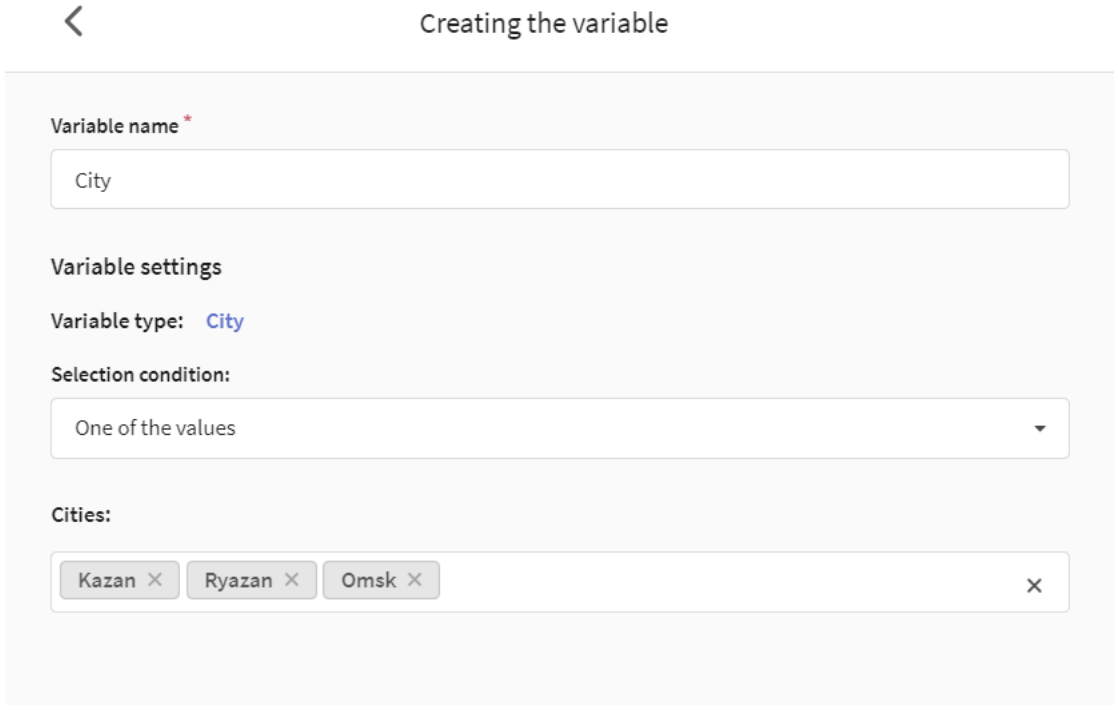
- **One of the values - the variable will return true if the user's country matches one of the selected one**
- **None of the values - the variable will return true if the user's country does not match any of the sele**



City

The variable takes the value `true` if the user's city matches the specified selection condition, and `false`
Selection condition:

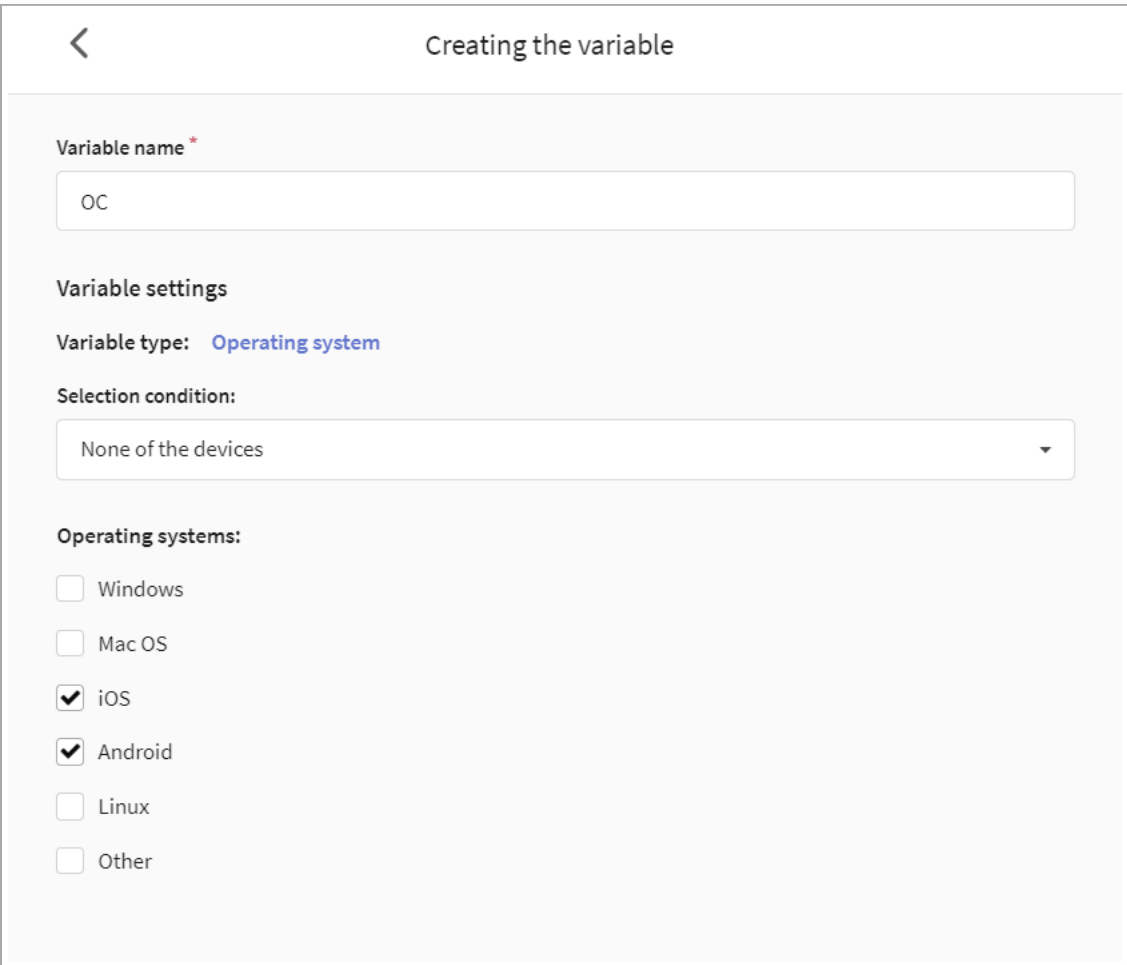
- **One of the values - the variable will return true if the user's city matches one of the selected ones.**
- **None of the values - the variable will return true if the user's city does not match any of the selected**



Operating system

The variable takes the value `true` if the user's OS matches the specified selection condition, and `false` i
Selection condition:

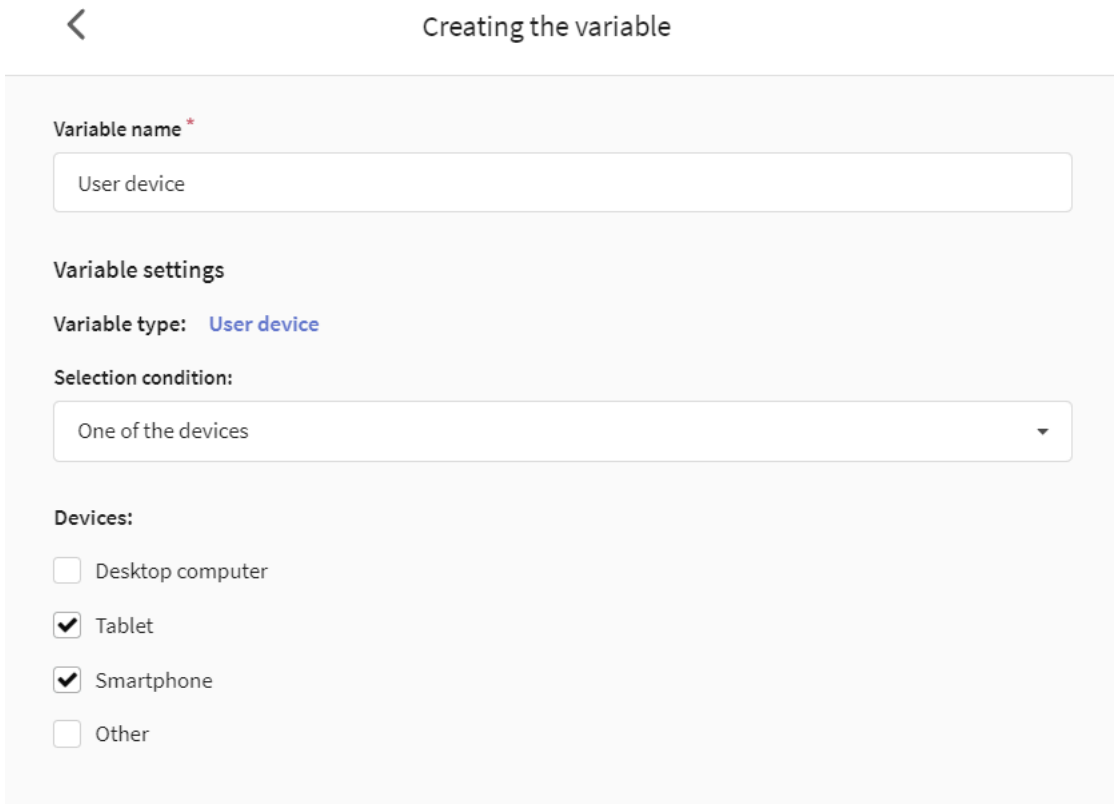
- **One of the devices - the variable will return true if the user is using one of the selected OS.**
- **None of the devices - the variable will return true if the user does not use any of the selected OS.**



User device

The variable takes the value `true` if the user's device matches the specified selection condition, and `fal`
Selection condition:

- **One of the devices - the variable will return true if the user is using one of the selected devices.**
- **None of the devices - the variable will return true if the user does not use any of the selected device**

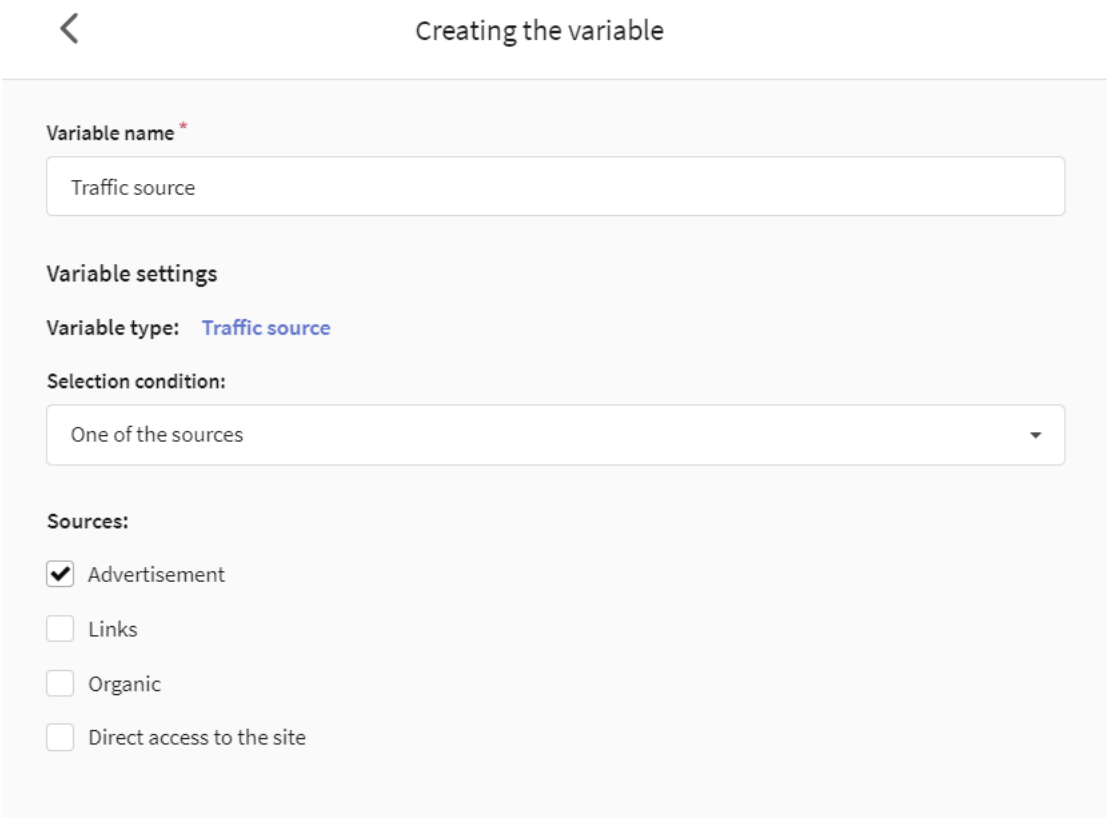


Traffic source

The variable takes the value `true` if the source of the user's transition to the site matches the specified does not.

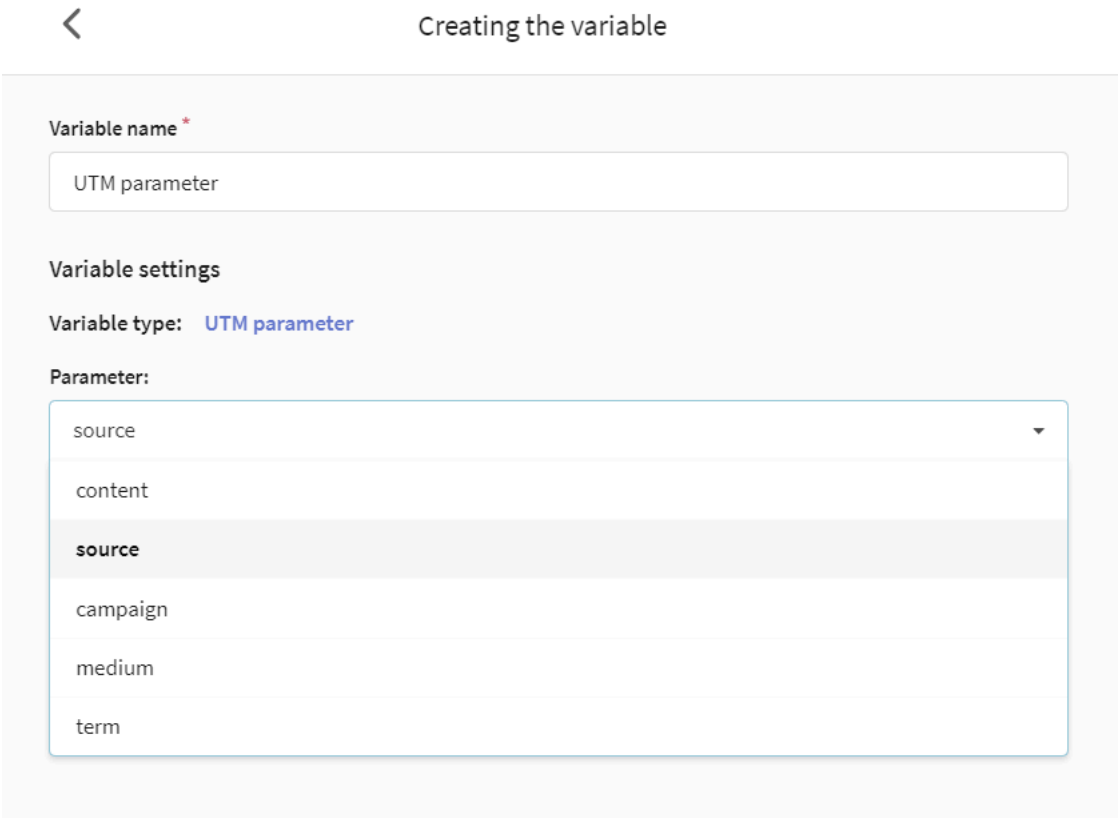
Selection condition:

- **One of the devices - the variable will return true if the traffic source matches one of the selected on**
- **None of the devices - the variable will return true if the traffic source does not match any of the sele**



UTM parameter

The variable takes the value of the selected UTM parameter.



Linking a pixel and the Tag manager

Tag manager is a convenient way to place tracking pixels on your site.

When you use pixels as a standalone tool, you have to make changes to the site's source code every tim avoided by using Tag manager.

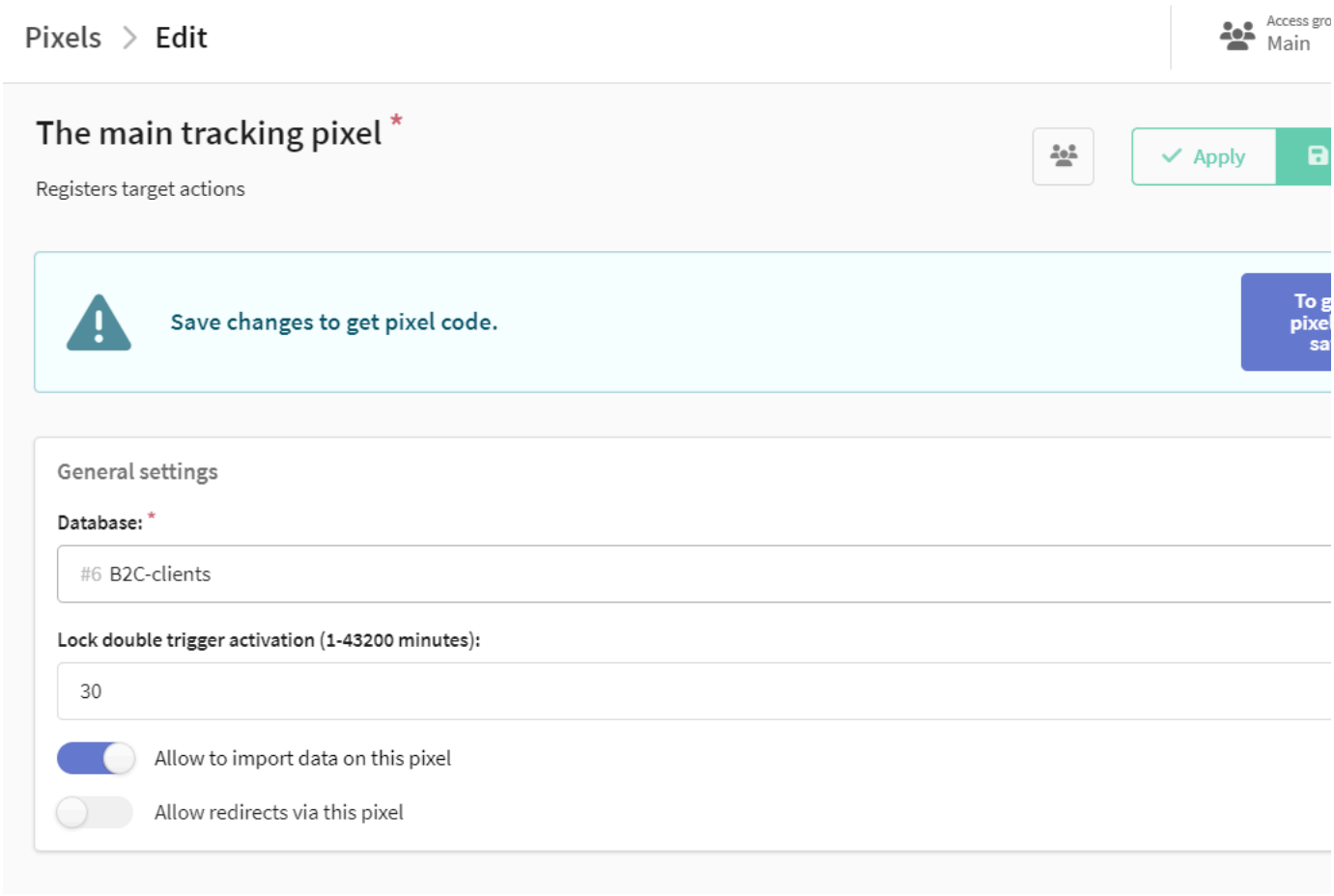
You create a container, add a tag with the HTML code of the pixel and triggers that determine the condit place the container code on the site pages. If in the future you want to make some changes to the opera or suspend the operation of an existing one), it will be enough to edit the tag in the platform interface.

This guide will help you link Tracking Pixels and Tag manager in the Altcraft system.

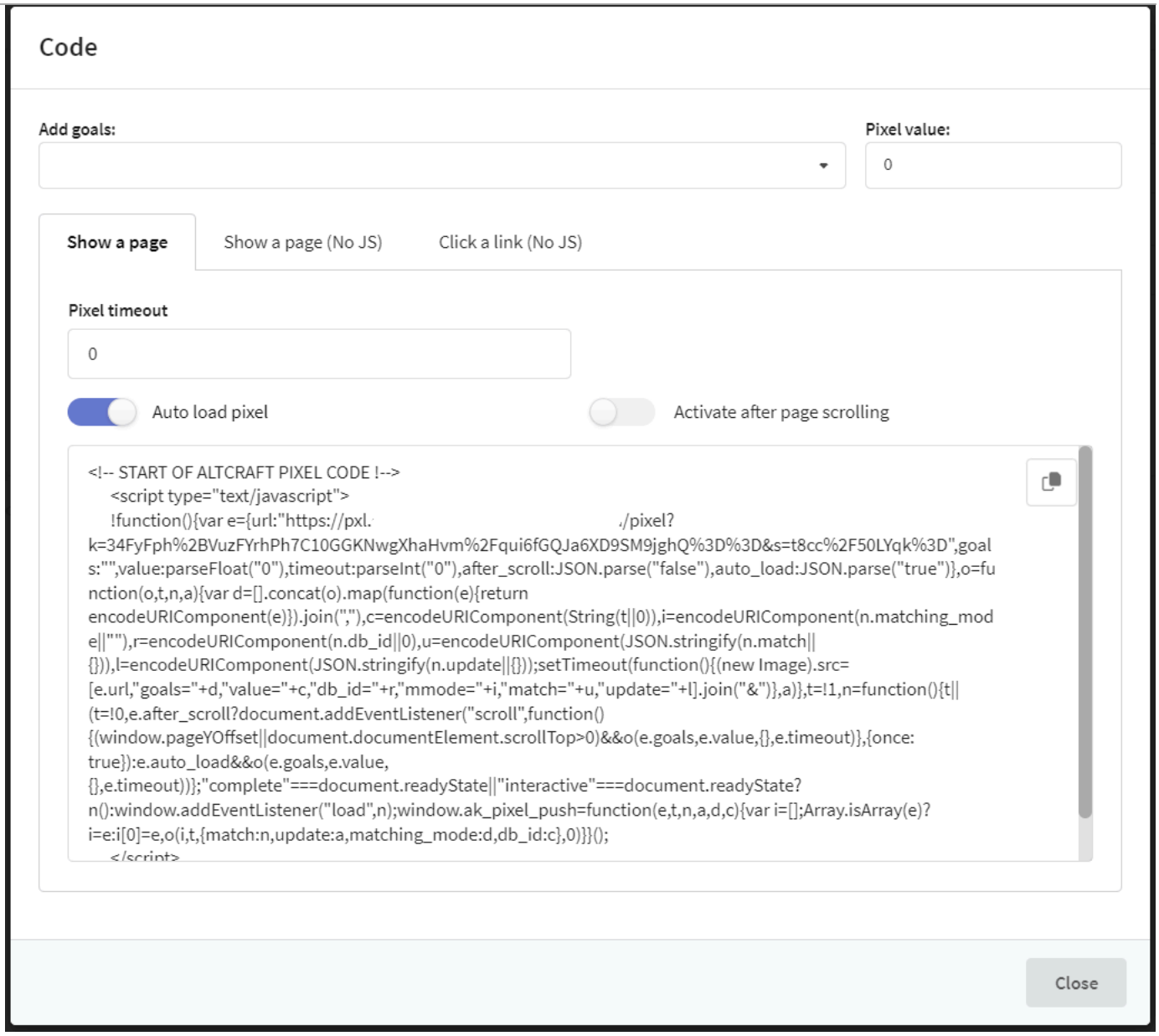
Step 1: Create a pixel

1. Go to the Pixels section and click Create.

2. Set up and save the pixel.



3. Click Get pixel code, add goals and copy the code.



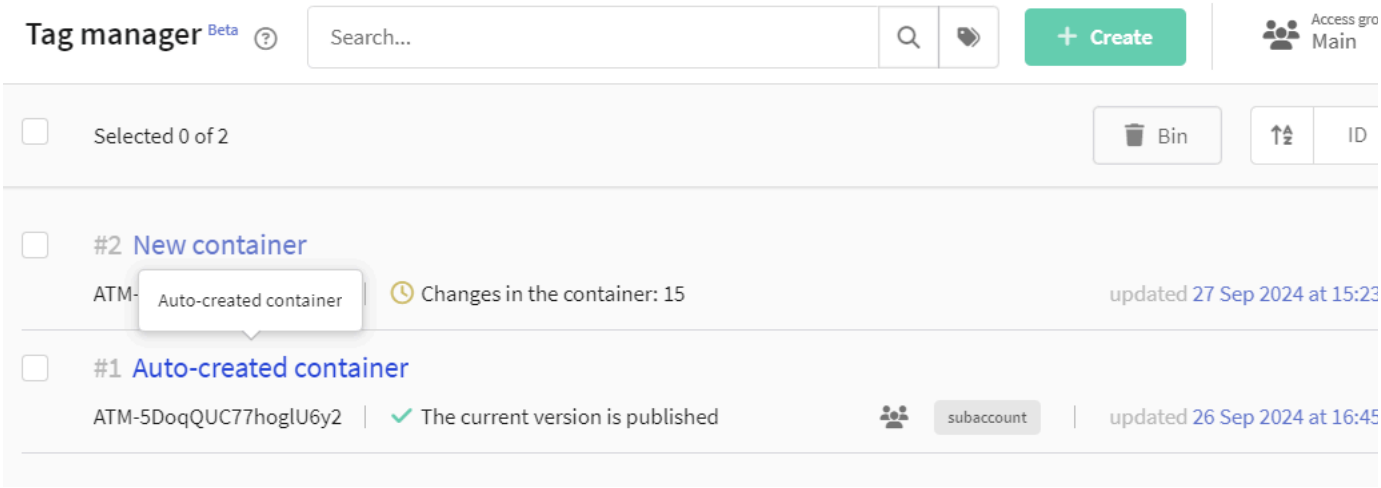
TIP

You can find more information about creating and configuring a pixel in [this article](#).

Step 2: Create a tag in a container

1. Go to Weblayer —> Tag manager.

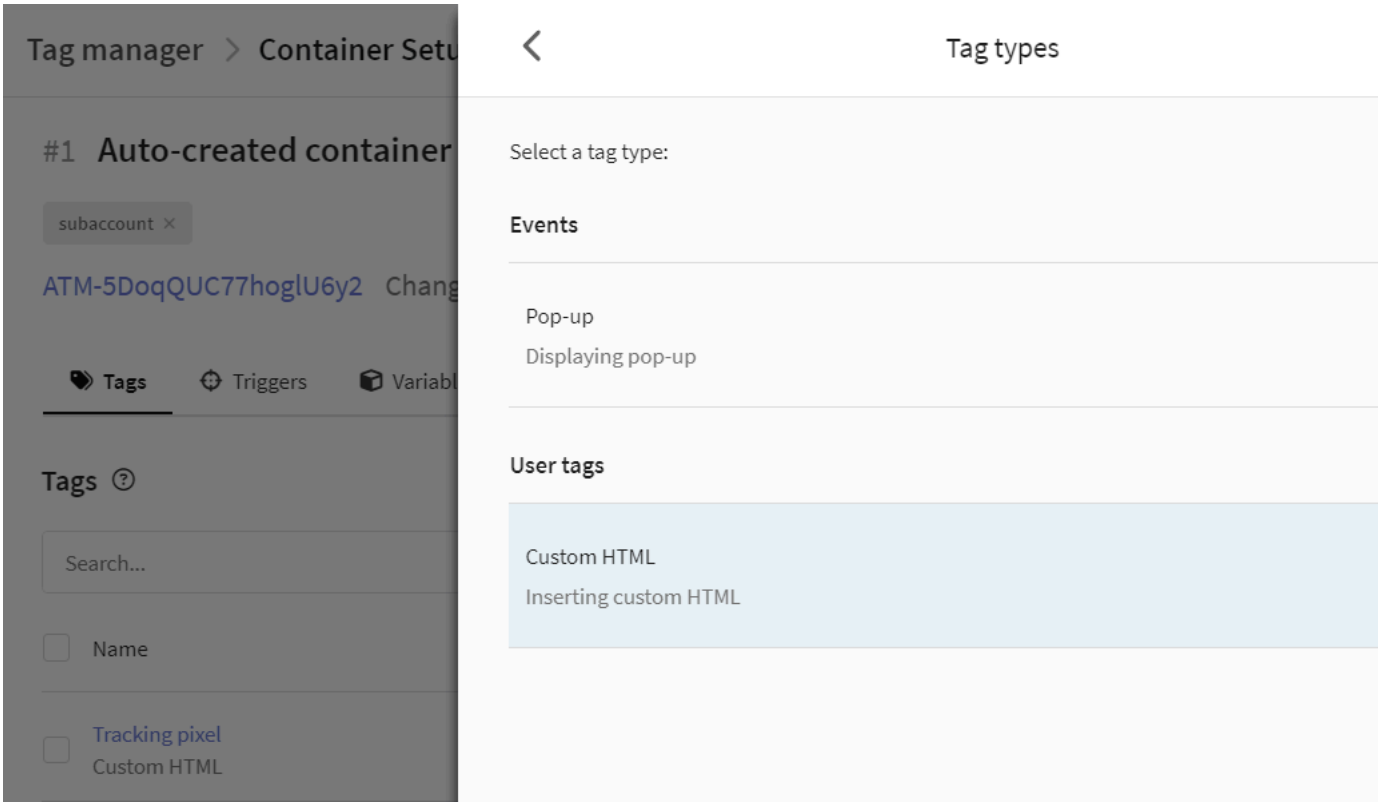
2. Open the container:



3. On the Tags tab, click + Create.

4. Enter a name of a tag.

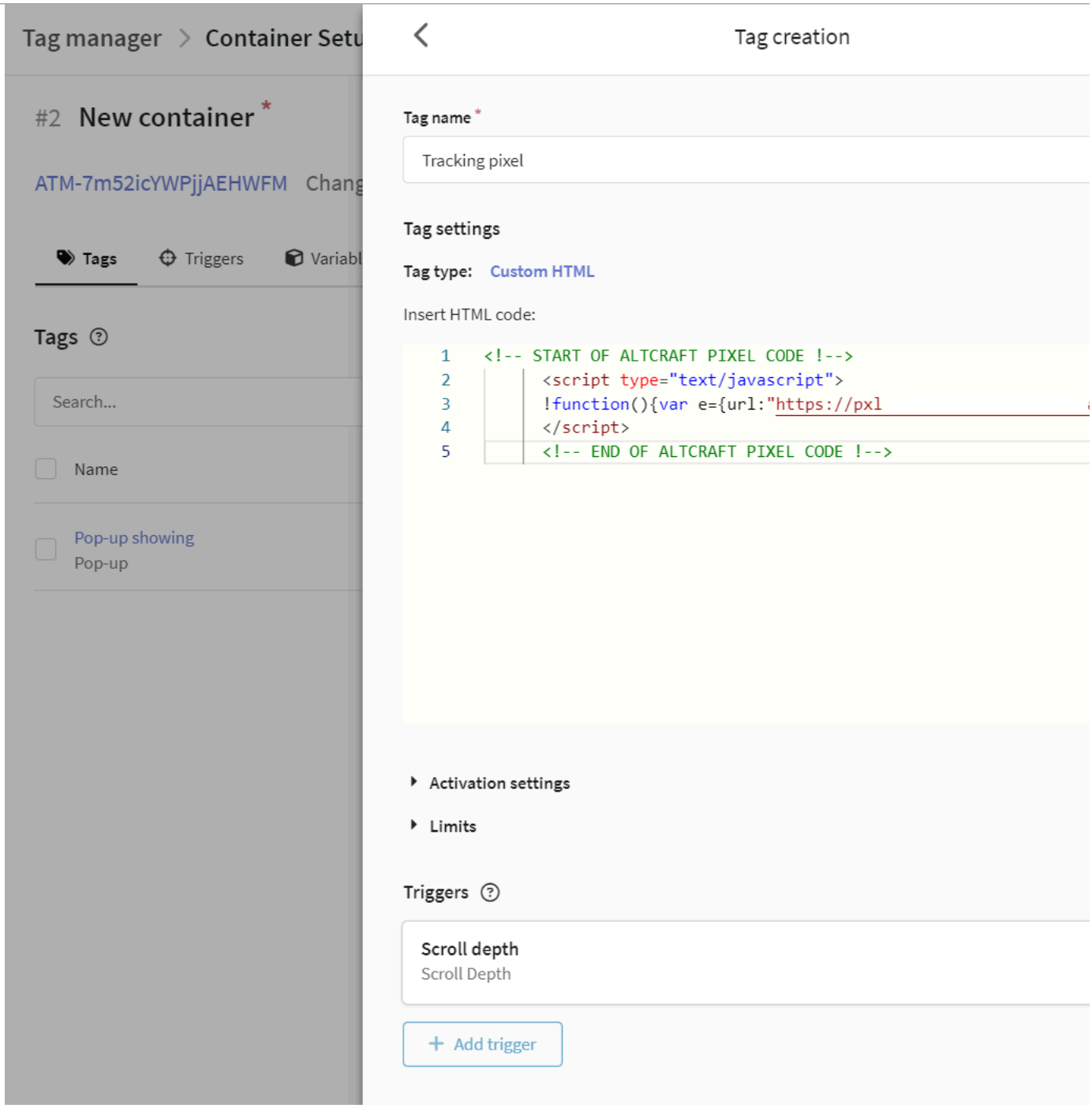
5. Select Custom HTML as the Tag Type.



6. Paste the pixel code into the editor window.

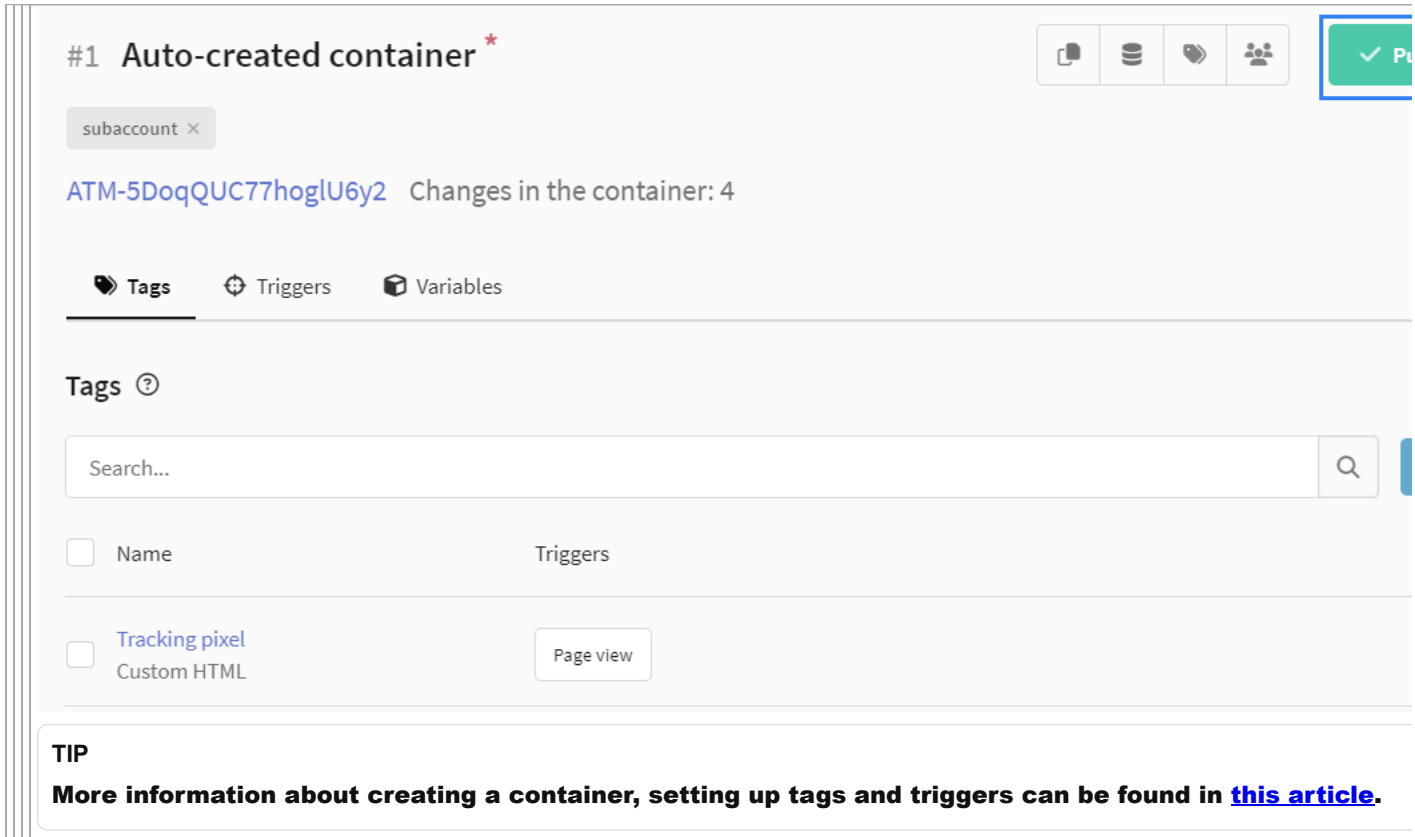
7. Set activation settings and limits for the tag.

8. Select and set up trigger. The tag will be activated when the conditions specified in the trigger occur. users for whom the trigger will work, additionally set up a filter.



9. Save the tag.

10. Click Publish —> Publishing and creating a new version. All changes to the container will be made to tracking user actions.



TIP

More information about creating a container, setting up tags and triggers can be found in this article.

Settings

Settings menu

> **NOTE**
>
> **Altcraft Platform account master-users have access to Settings section of the main menu containing th**

Section contents

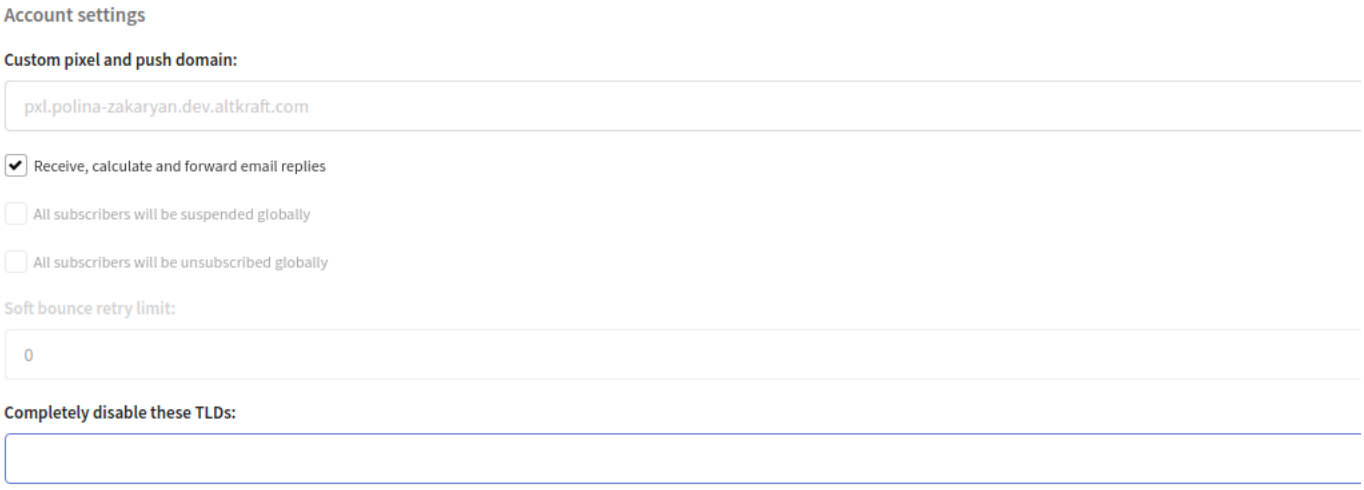
Account settings

Description

![img] (/img/attachments/37850647/37850655.png)	The "Account settings" section includes configuration settings that are applied globally: to all processes and objects in the acc

Administration

In this tab, you'll find the settings of email sending and subscriptions. You also can specify your custom

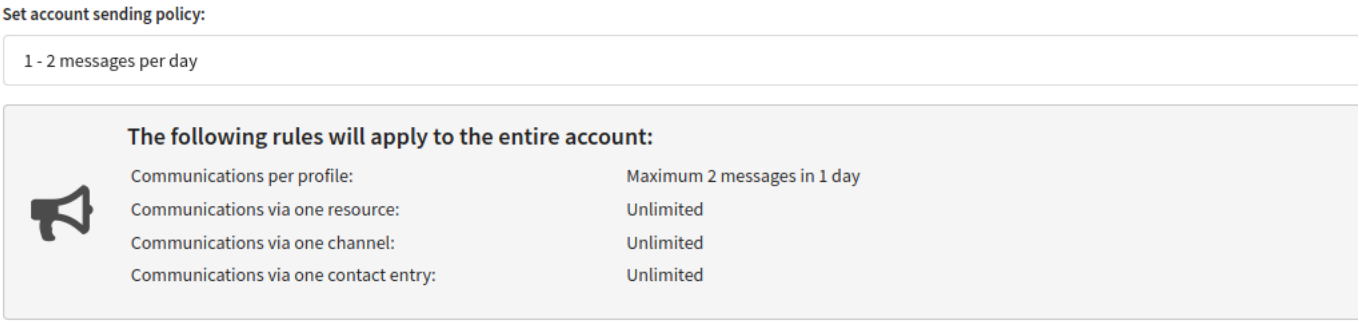


Report settings

In this tab, you can set the day of the week and hour for sending reports to email of platform users. The each user separately. Read more in [this article].

Global sending policy

In the next tab, you can set the global [sending policy]. It will apply to all resources and mailings that do n



Additional options

At the bottom of the page there are options for proxying tracking requests and auto-setting utm tags wh

Custom links

Description

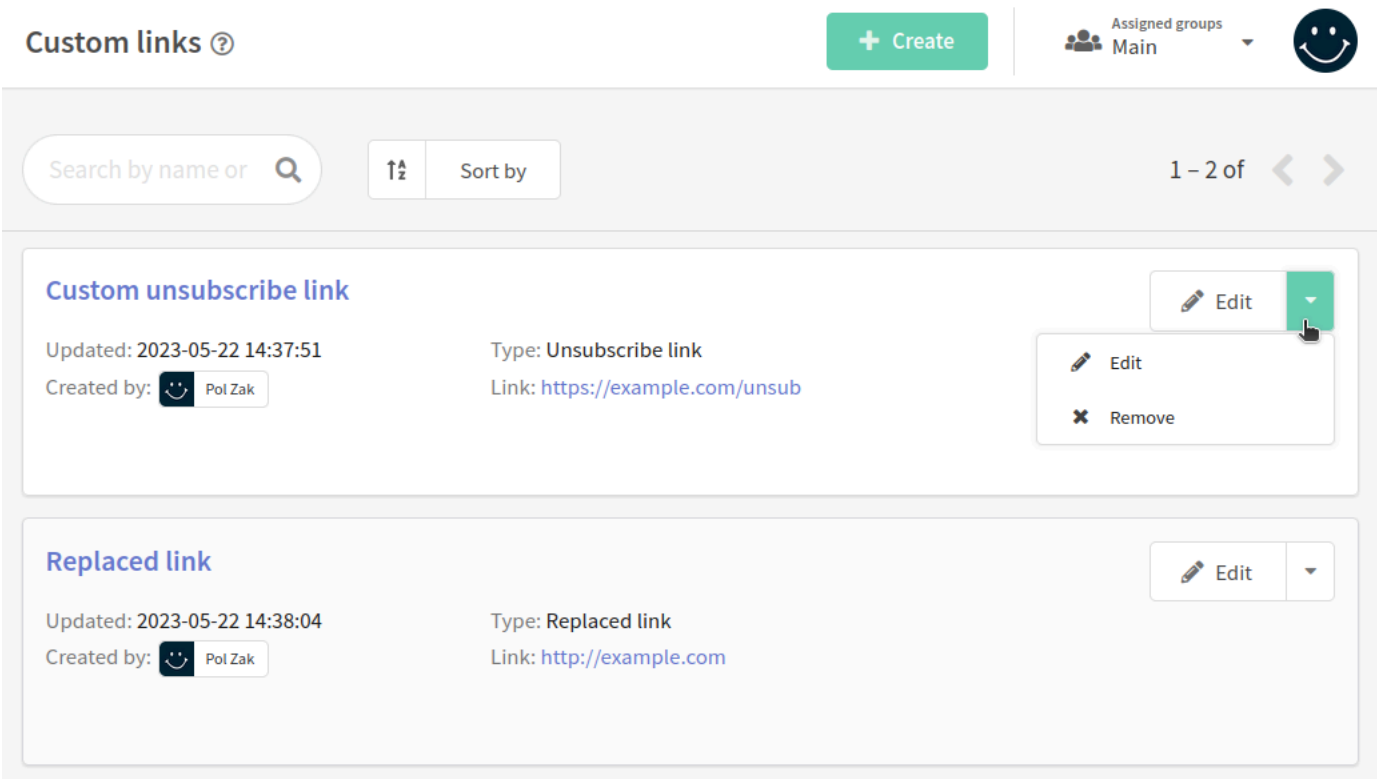
There are two types of custom links available: unsubscribe link and replaced link. With a custom unsubs recipients to their personal account on your site or to any web page when they click on the “Unsubscribe replace the URLs of links in messages that have already been sent.

Managing custom links

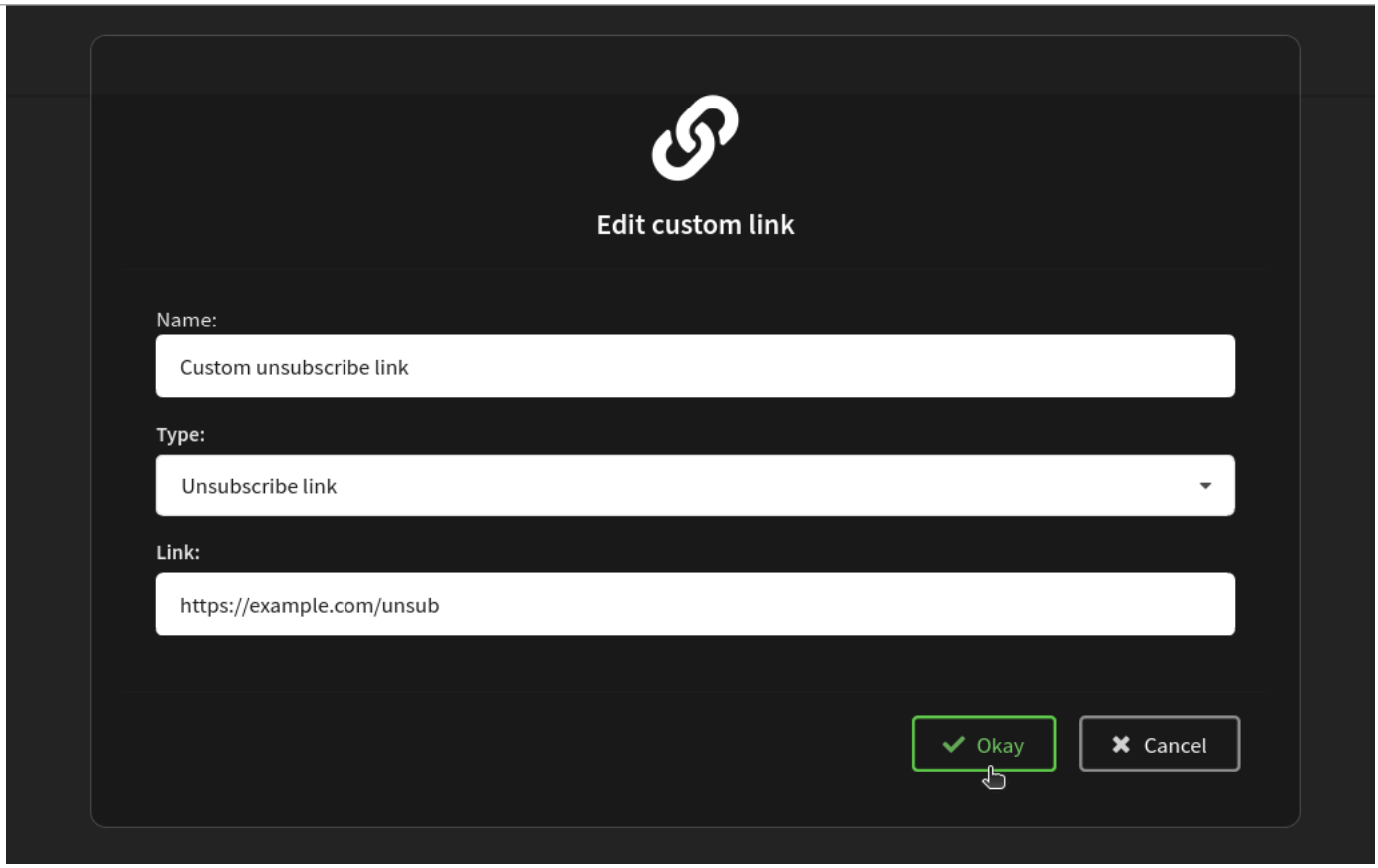
CAUTION

To create custom links you would like to have master useraccount.

In the main menu select Settings —> Custom links and start with + Create button:

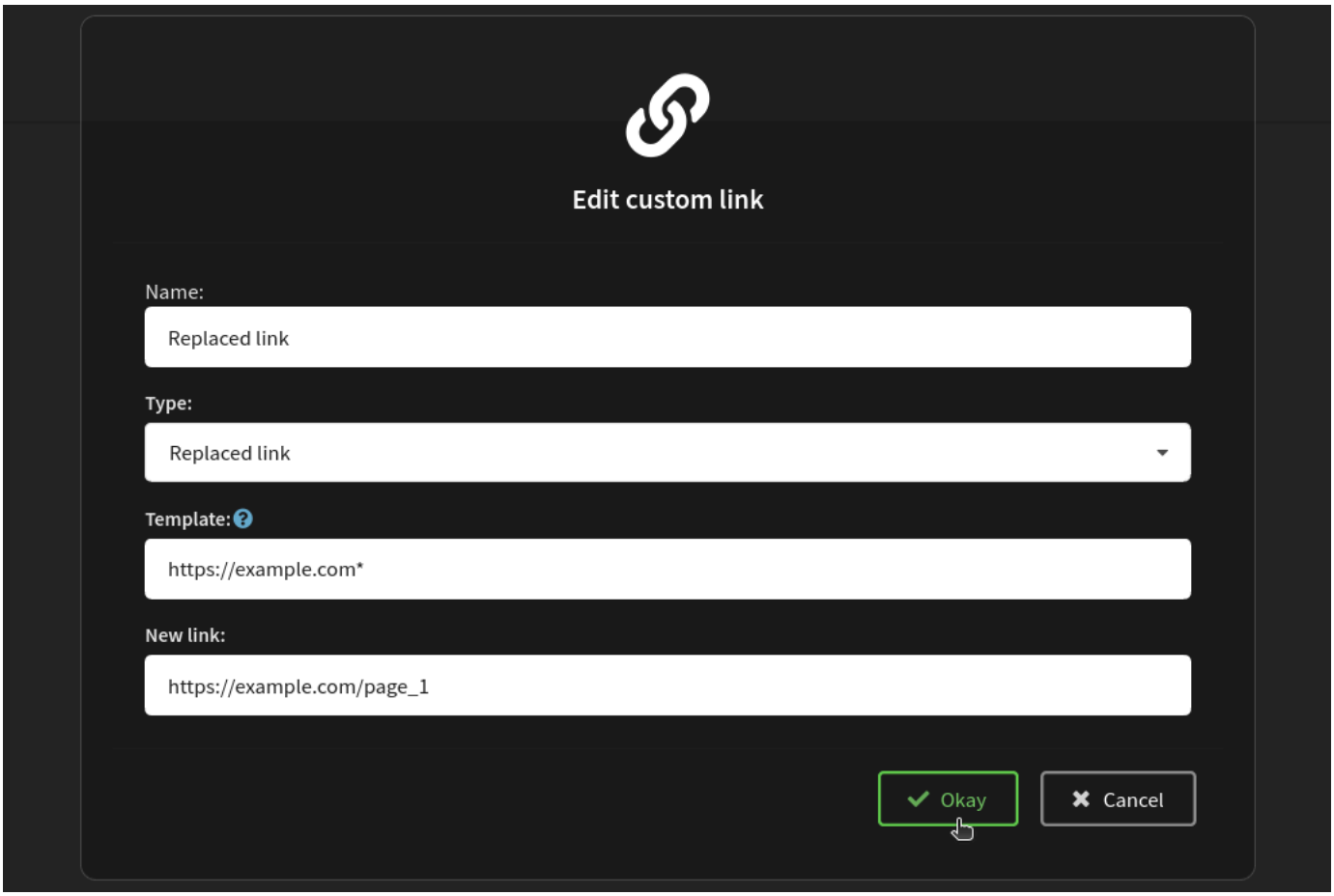


In the editing form enter the new link name, select its type (Unsubscribe or Replaced) and enter the URL



For a replaced link specify:

- **Template — an old link or a part of a link. Wildcard (symbol *) can be used to search for multiple URL example, if you specify `https://example.com/*`, the platform will replace all links in message templates**
- **New link — new redirect address for your link.**



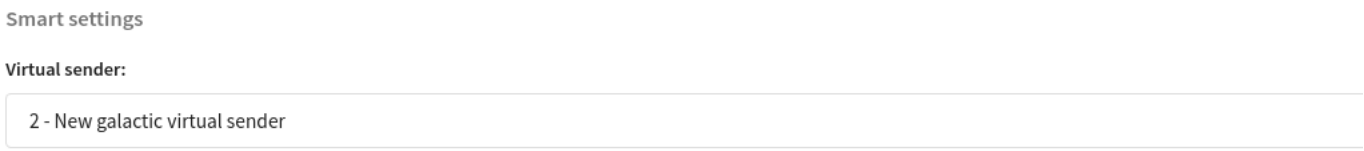
Virtual senders

Overview

A virtual sender is a set of rules for real senders and a powerful tool for fine messaging tuning.
Your delivery rate can greatly depend on it — especially with large amounts of messages to be sent.
These rules specify real sender usage proportions and custom From and Reply-to email fields for messag

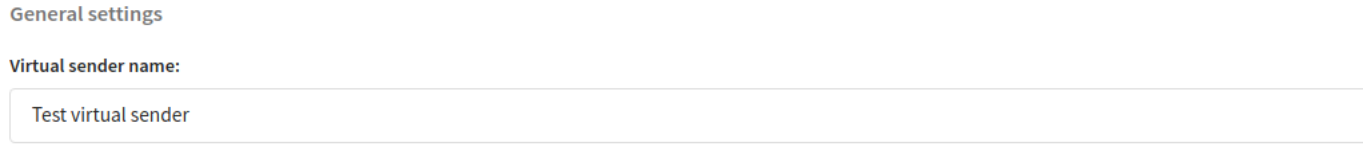
> NOTE
> **Virtual senders can be created and managed only by a master user or an Altcraft platform administrato**

If you need to manage the virtual senders of your account, in the main menu, select Settings —> Virtual
You can enable virtual senders for mailings if Smart settings are available for your account. If Smart set
virtual sender will be used.

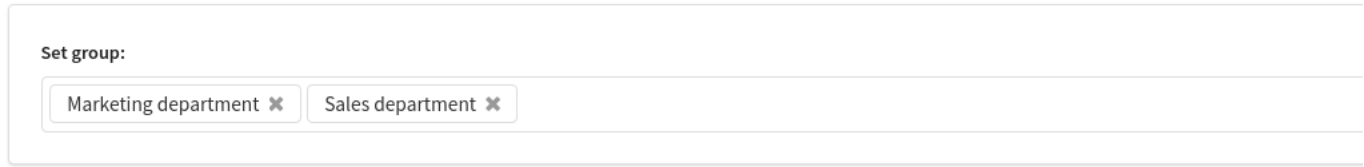


General settings

First of all, you need to enter a unique name for each sender:



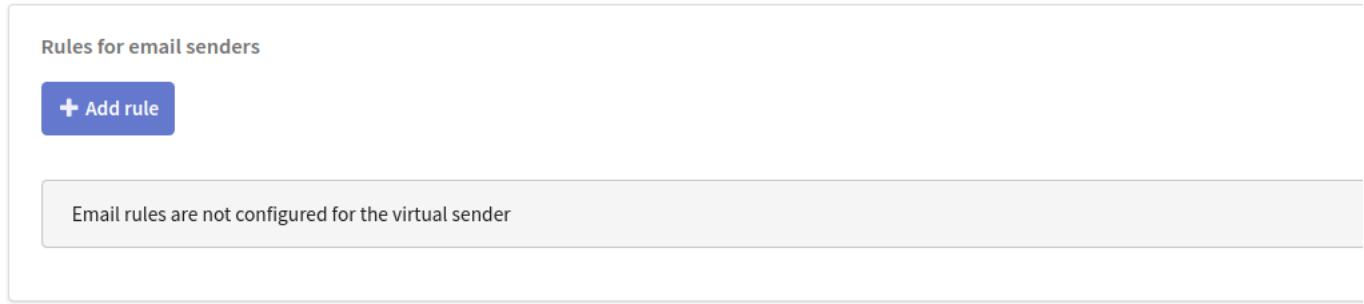
You can set groups in the bottom of the page. These groups will get access to the virtual sender.



Sending rules for different communication channels

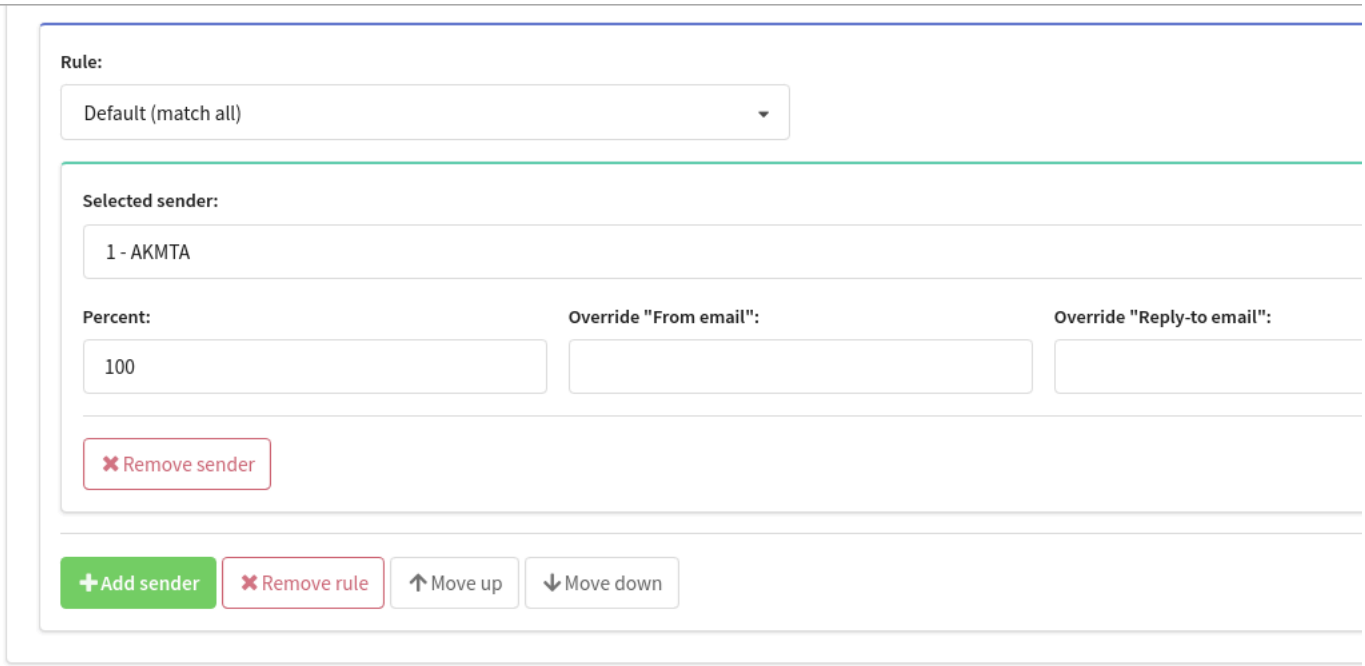
Rules for email senders

The rules for email senders determine which real senders and in what proportions will be used under ce
Add Rule.



Select the appropriate condition from the drop-down menu When the rule is applied:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Match by "From domain" — the rule is applied when sending from a specific domain.**
- **Match by "To domain" — the rule is applied when sending to specific email domains.**
- **Match by "From email" — the rule is applied when sending from a specific email address.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by def sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position or Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move U

> **CAUTION**
>
> **The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases tha**

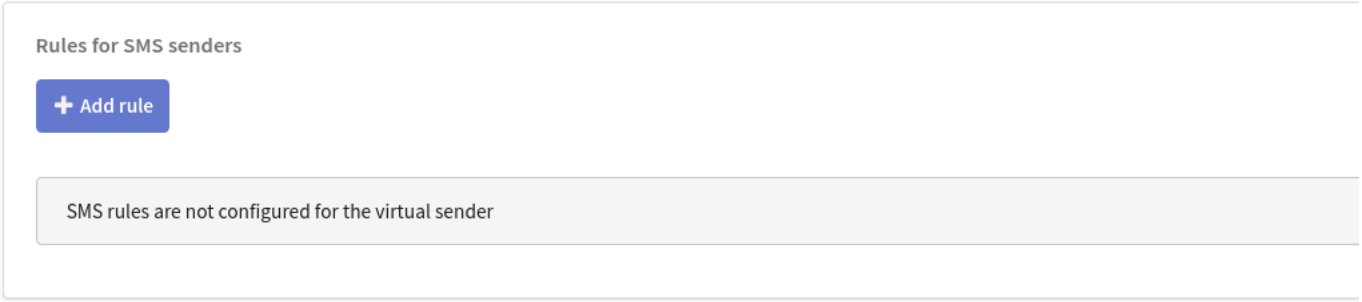
Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule for each sender:

- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From Email" — override the "From email value" for outgoing messages.**
- **Override "Reply-to Email" — override the "Reply-to email" value for outgoing messages.**

> **NOTE**
>
> **For one rule the sum of senders usage percent must be 100.**

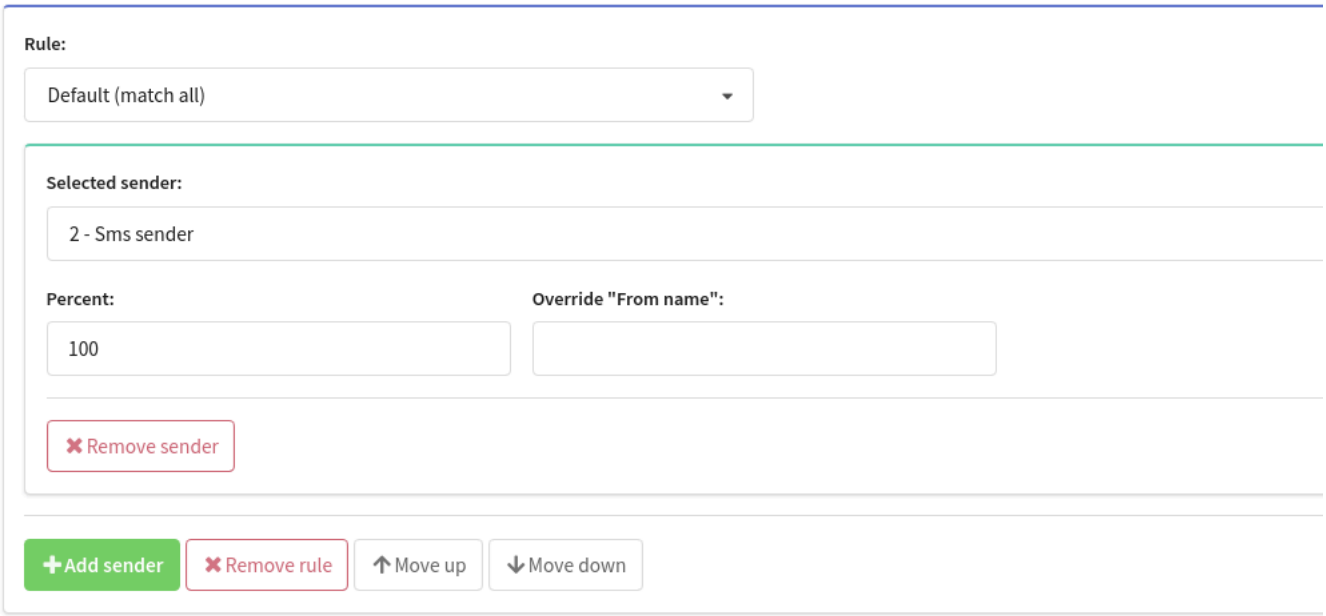
Rules for SMS senders

The rules for SMS senders determine which real senders and in what proportions will be used under cert Add Rule.



Select the appropriate condition from the dropdown menu with rules:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by def sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position or Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move U

> CAUTION
>
> **The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases tha**

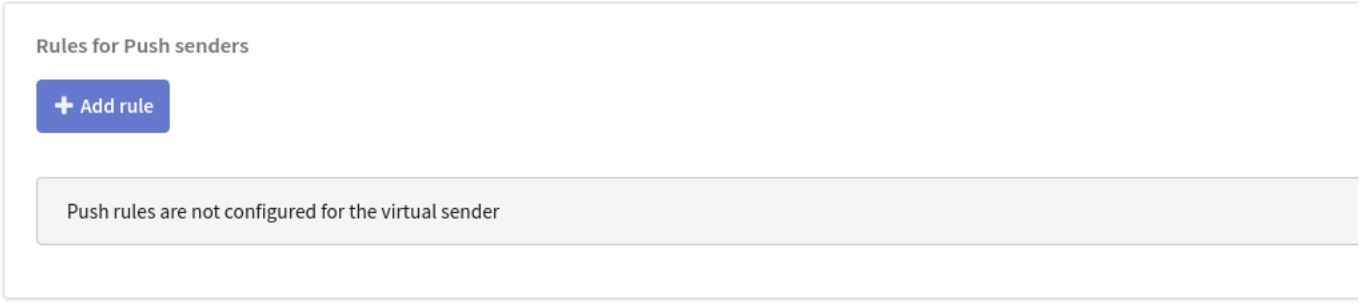
Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule for each sender:

- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From name" — override the "Caller_ID" value for outgoing messages.**

> NOTE
>
> **For one rule the sum of senders usage percent must be 100.**

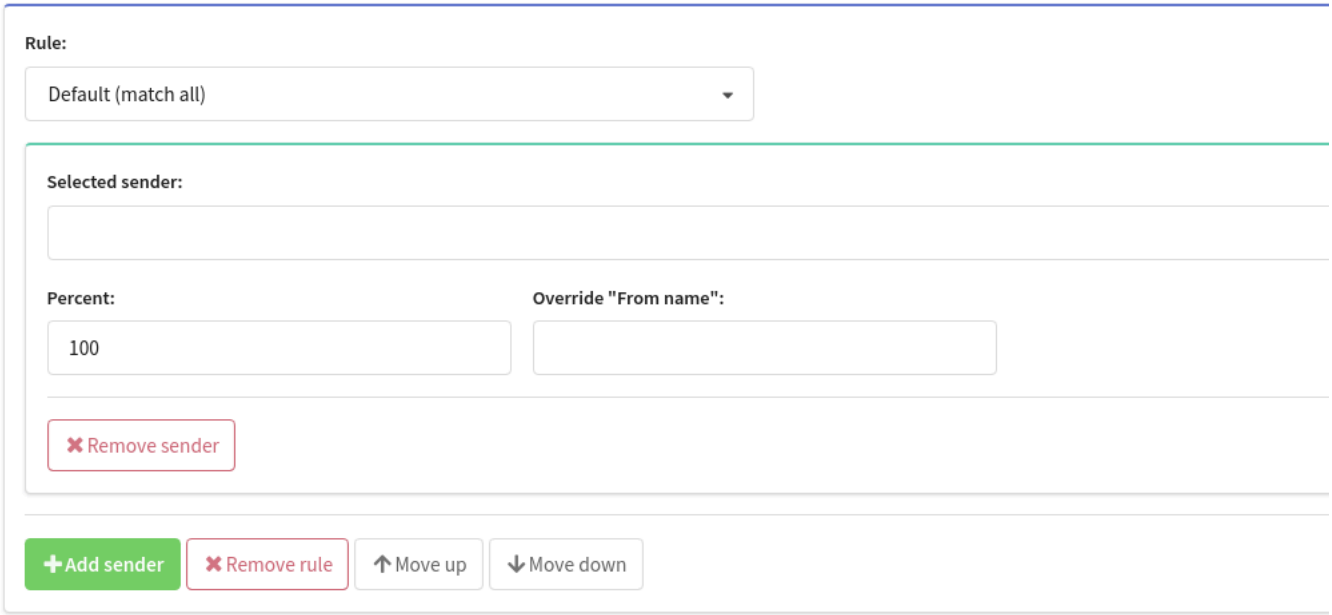
Rules for push senders

The rules for SMS senders determine which real senders and in what proportions will be used under cert Add Rule.



Select the appropriate condition from the dropdown menu with rules:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by def sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position or
Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move U

> CAUTION
>
> **The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases tha**

Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule for each sender:

- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From name" — override the "From name" value for outgoing messages.**

> NOTE
>
> **For one rule the sum of senders usage percent must be 100.**

Sending policies

Overview

Sending policies prevent from sending too much messages to a single recipient. Such policy can be appli

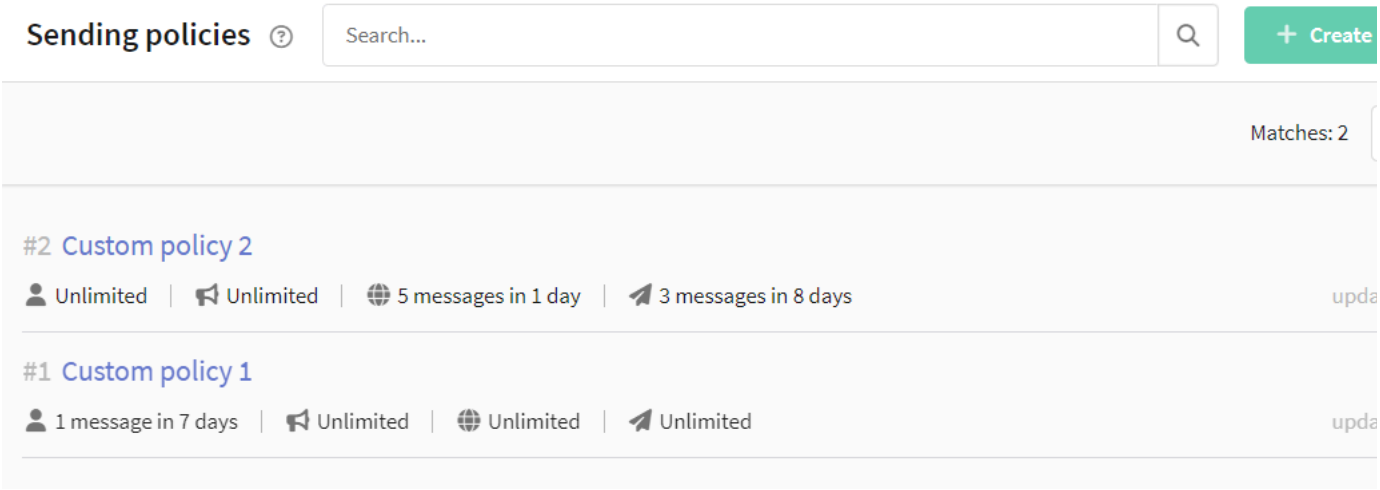
- **Account — a policy will be applied globally to all account mailings.**
- **Resource — a policy will affect all mailings within a Resource, overriding Account policy.**
- **Mailing — a policy will affect only the mailing it was applied to, overriding both Account and Resourc**

> CAUTION
>
> **The sending policy cannot be set for a specific profile. The policy is configured and valid for all profiles**

Managing sending policies

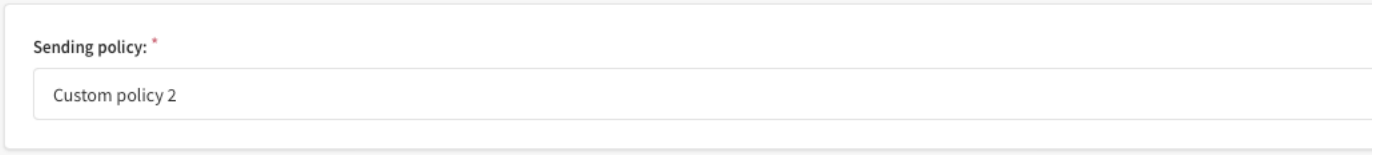
> CAUTION
>
> **Only master-users can create, edit or delete sending policies.**

To start managing sending policies in the main menu select Settings —> Sending policies. The list you w policies:



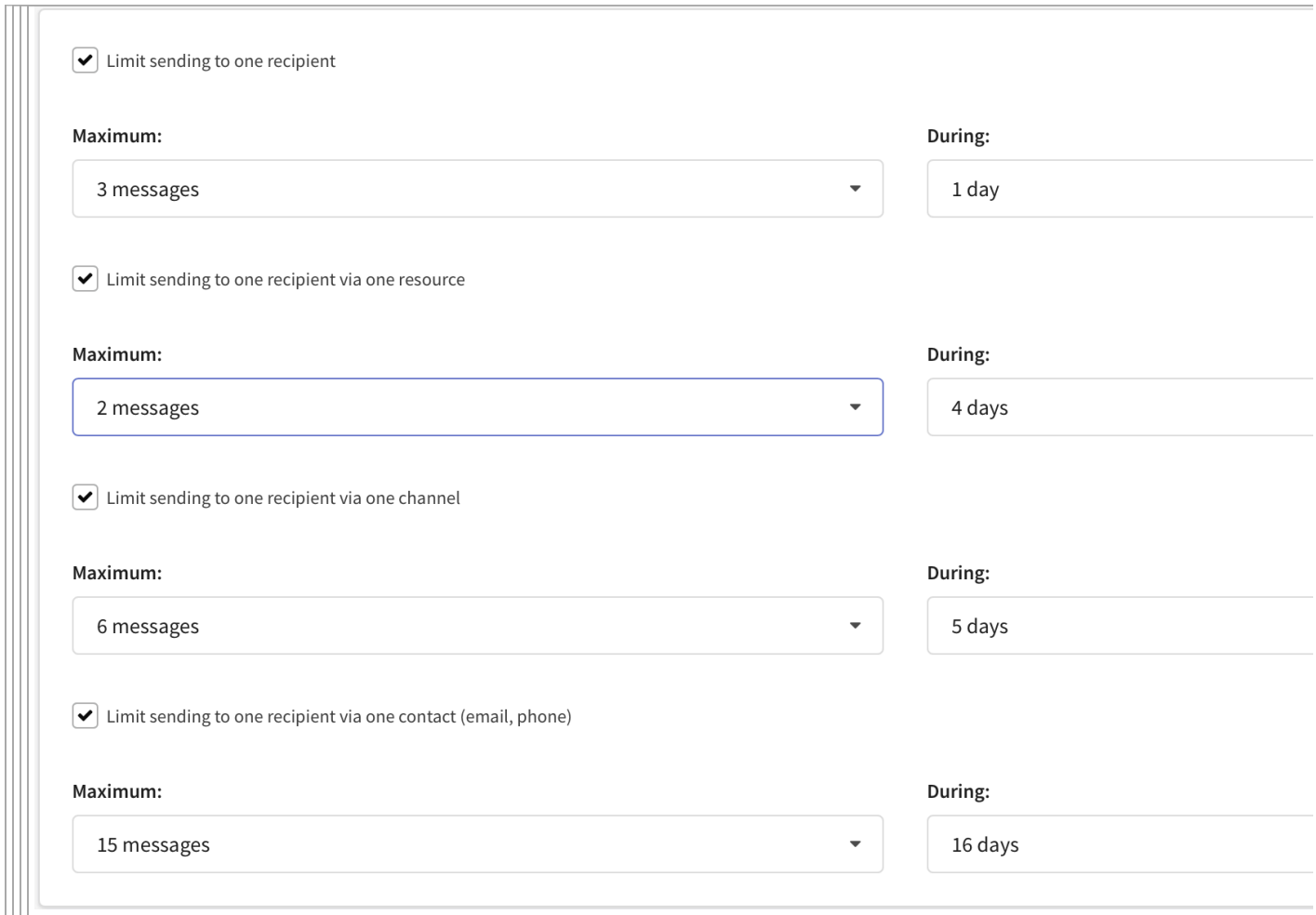
You can clone or delete a policy in a context menu on the right side of the list. To edit a policy you can a

To create a new policy click + Create in page header. Every policy has three sections of settings. In the f



In the second section specify messaging limitations. Set maximum number of messages between 1 and days:

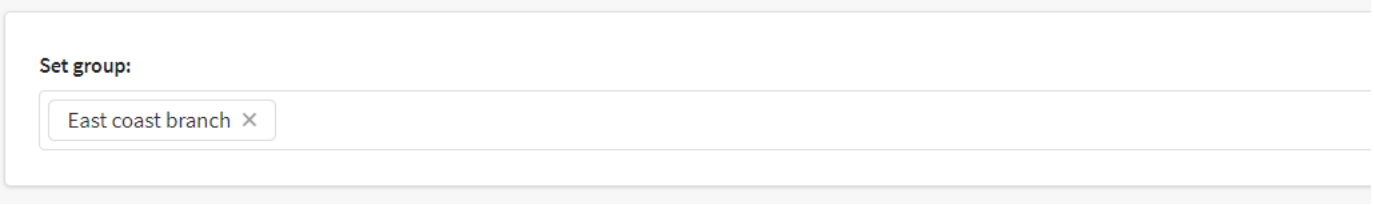
- **Limit sending to one recipient — will limit messaging from all resources and channels for one subscr**
- **Limit sending to one recipient via one resource — will limit messaging from all channels of any singl**
- **Limit sending to one recipient via one channel — will limit messaging via one channel (email, sms, pu subscriber profile.**
- **Limit sending to one recipient via one contact (email, phone) — will limit messaging for a subscriptio The subscription can be associated with multiple subscriber profiles.**



TIP

A sending policy with no restrictions specified is unlimited. Thus you can set restrictions for the whole a single mailing or a resource applying an unlimited policy.

In the third section you can optionally set access group to the sending policy. Thus it will be available or groups you specify:

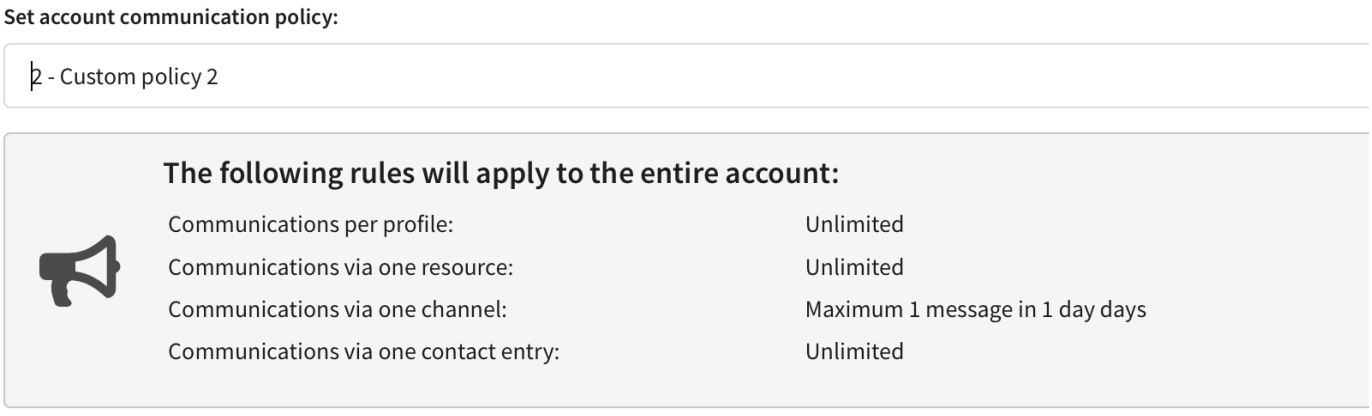


Applying sending policies

You can apply a sending policy to the whole account, to a specific resource - or to a specific mailing.

Global sending policy

To apply a global account policy open Settings —> Account settings and select a sending policy in this d

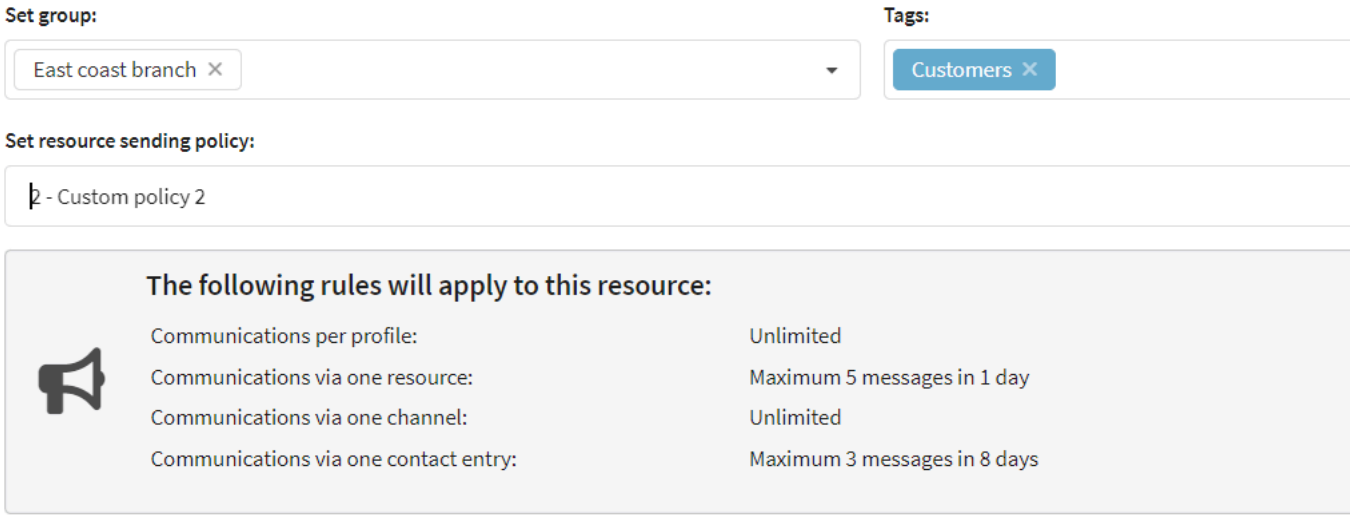


CAUTION

Only master-users have access to Account settings and thus only they can set global sending policies.

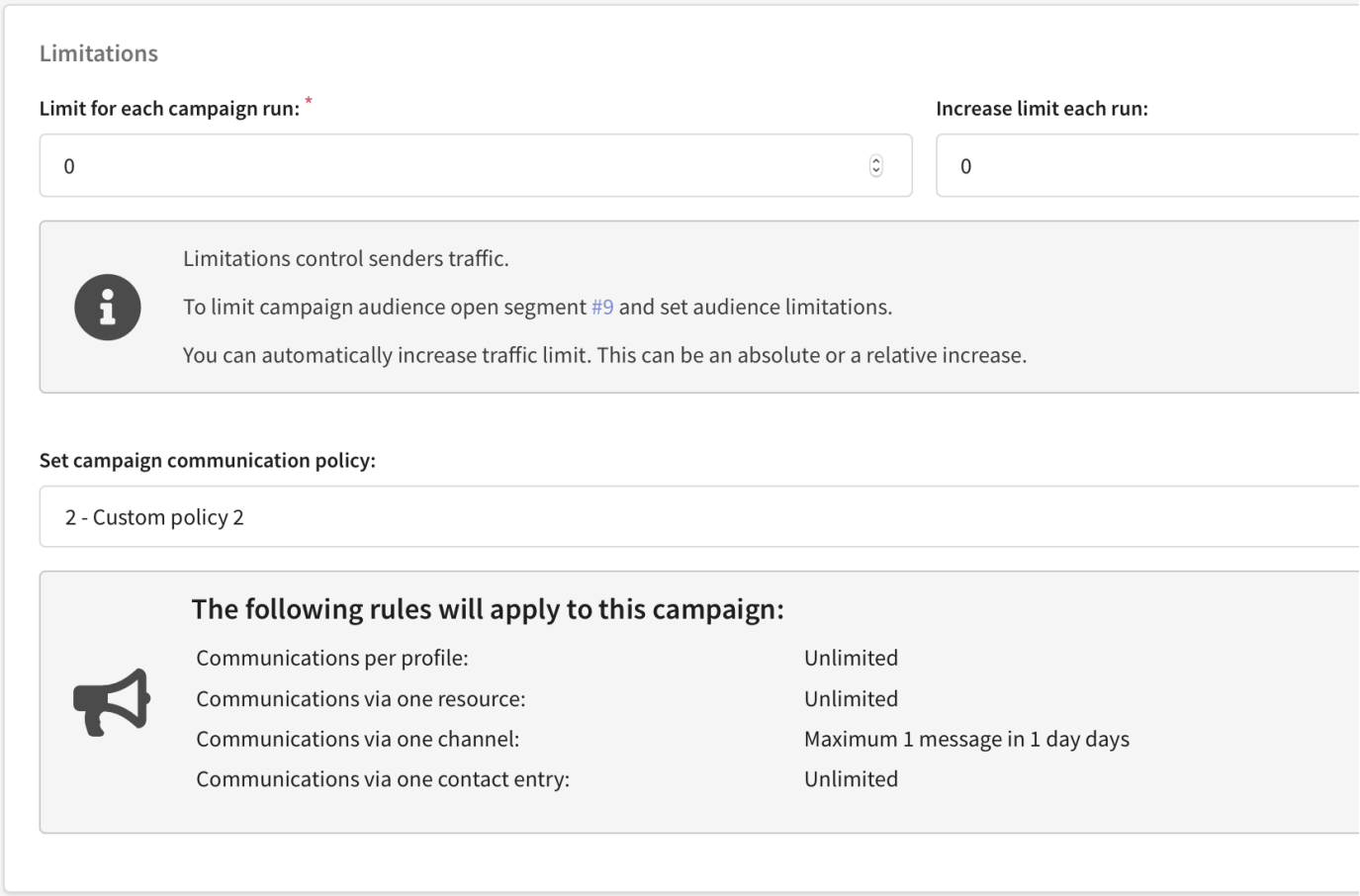
Resource sending policy

A policy applied to a Resource will override global Account settings. Open resources in the main menu a resource select a sending policy:



Mailing sending policy

A policy applied to a specific mailing will override both account and resource sending policies. Open a m Limitations section (Set mailing communication policy).



INFO

Messages that fall under restrictions will form entries in the mailing log of the type Sending is restricte

Audit journal

Description

Audit journal is an important security and monitoring element that records user actions in the platform. (have access to the audit. To go to the audit, open Settings → Audit journal.

User action audit is a very important security and monitoring feature, capturing Altcraft platform user ac privileges can access audit log by opening Settings —> Audit journal.

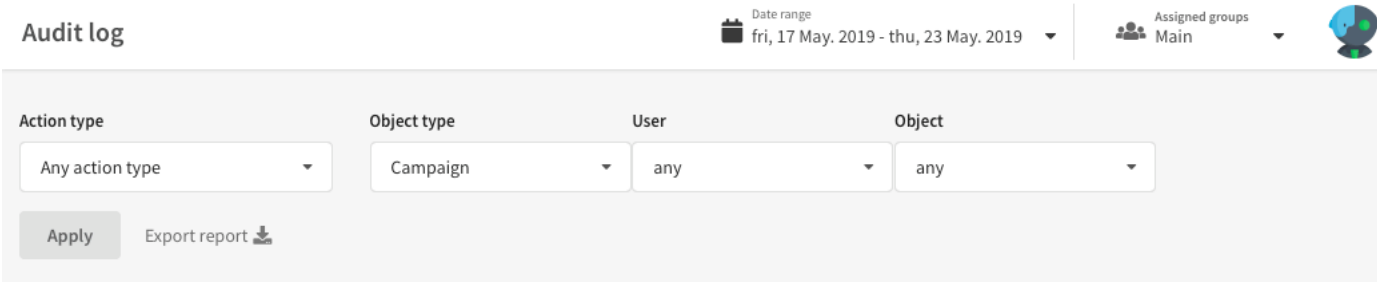
The audit log is stored for 90 days. To change this value, you need to add the entry `"AUDIT_LOG_TTL" : 90` to changing the value as needed.

Audit journal settings

You can configure the audit query according to your current tasks:

- **Period — time frame for displaying recorded actions.**
- **Action type — type of event recorded by the system. Actions are related to authorization in the platf of objects in the account.**
- **Object type — allows you to view actions related to a specific type of entities in Altcraft. These can databases and segments, resources and mailings, campaigns, templates and their parts, as well as**
- **Object — you can select a specific account object and get information about actions with it.**

To view the audit report, click Apply. To export the audit report to .XLS or .CSV, click Export report:

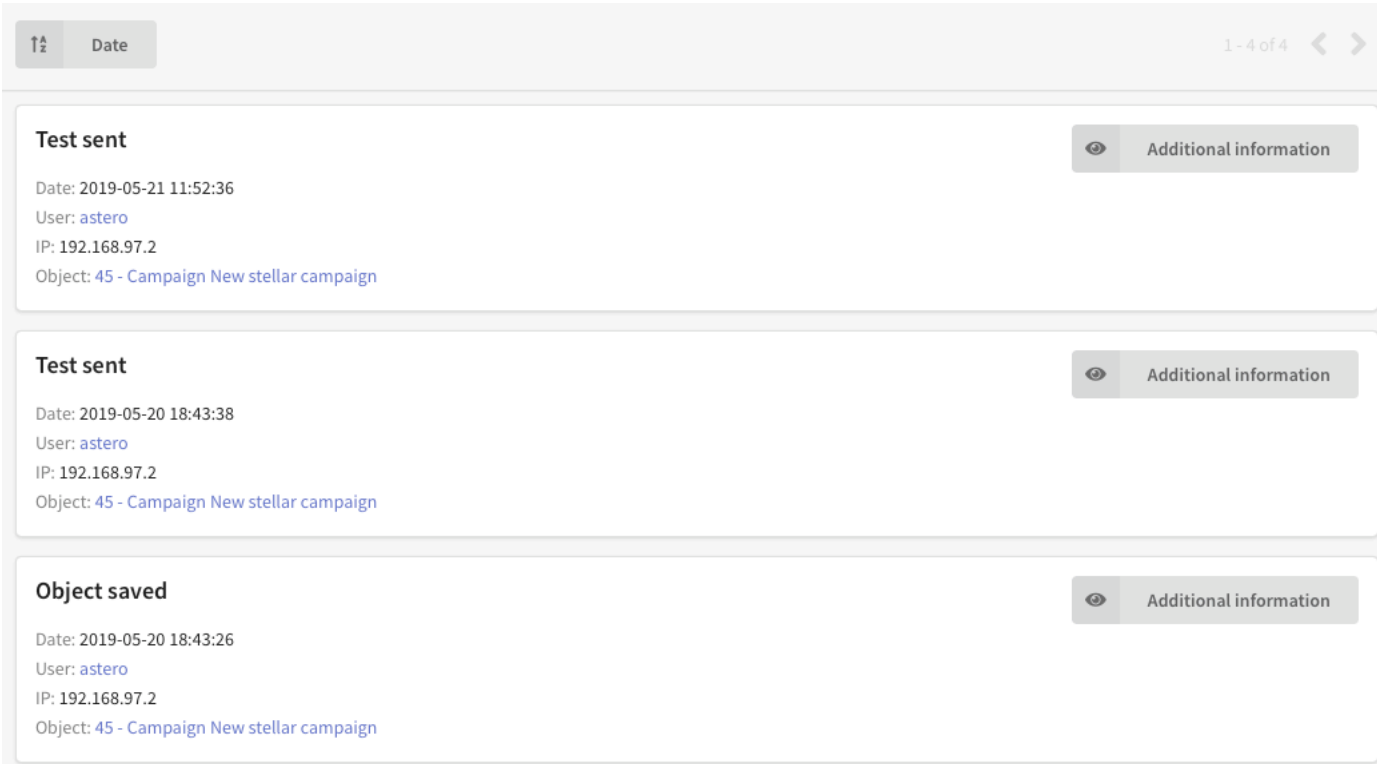


Viewing audit events

Actions that meet the query conditions are displayed as a list of tiles. The title of each tile indicates the action, the date, the user who performed the action, their IP address and the object associated with the

Additional information contains a more detailed description of the event, for example, email addresses t

Actions in the report can be sorted by date, user, IP address or object.



Campaigns audit actions

The system records the following actions with campaigns (combinations of mailings and scenarios):

Basic campaign operations

- **Campaign creation — creating a new campaign**
- **Campaign update — changing campaign settings**
- **Campaign deletion — deleting a campaign**
- **Campaign activation — launching a campaign**
- **Campaign deactivation — stopping a campaign**
- **Campaign completion — final completion of campaign work**

Local control groups (LCG) management

- **LCG creation — setting up a local control group**
- **LCG update — changing LCG parameters**
- **LCG deletion — deleting a local control group**

Marketing activities management

- **Activity binding — adding a mailing or scenario to a campaign**
- **Activity unbinding — removing a mailing or scenario from a campaign**

Segments management

- **Segment binding — assigning a segment to a campaign**
- **Segment unbinding — removing a segment from a campaign**

Loyalty programs operations

- **Loyalty program creation — creating a new loyalty program**
- **Loyalty program update — changing loyalty program settings**
- **Loyalty program deletion — deleting a loyalty program**
- **Loyalty program activation — launching a loyalty program**
- **Loyalty program deactivation — stopping a loyalty program**

All the above actions are also available for points types and loyalty levels.

Partner operations in loyalty programs:

- **Loyalty program partner creation — creating a new loyalty program partner**
- **Loyalty program partner update — changing loyalty program partner**
- **Loyalty program partner deletion — deleting a loyalty program partner**

Promotion operations:

- **Promotion creation — creating a new promotion**
- **Promotion update — changing promotion settings**
- **Promotion deletion — deleting a promotion**
- **Promotion activation — launching a promotion**
- **Promotion deactivation — stopping a promotion**
- **Promotion completion — final completion of promotion work**

The audit journal also stores information about attaching and detaching loyalty programs to subscriber p

TIP

When working with mailings and scenarios within campaigns, the system also records standard actions with the parent campaign ID specified.

Searching tags FAQ

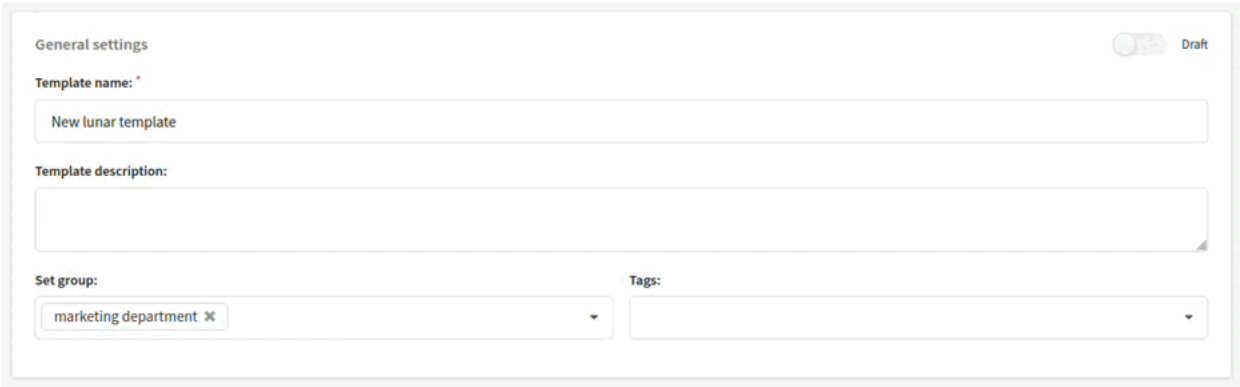
Tag is a label that can be assigned to some platform objects to quickly find these objects later. Tags usu common theme. For example, mailings that are launched as part of one promotion, or templates targetin

Which objects can be tagged:

- **databases**
- **resources**
- **segments**
- **templates**
- **scheduled imports**
- **suppression lists**
- **mailings**
- **campaigns**
- **scenarios**
- **forms**
- **products and orders**

How to create a tag

The tag is created directly in the object to which you want to assign this tag. Go to the object editing wi field and enter the text, then press Add or ↵ Enter.



To unpin a tag from an object, click the cross symbol.

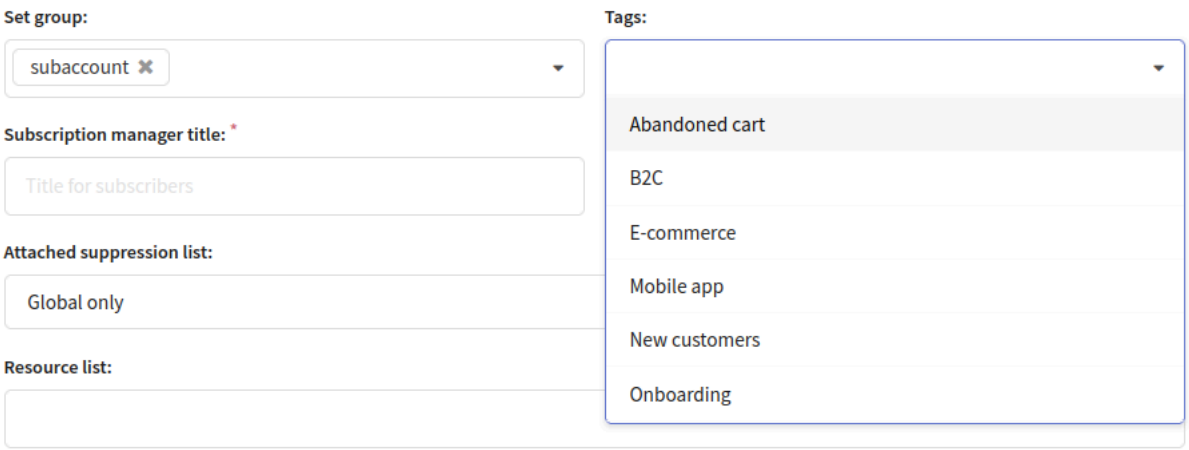
You can assign any number of tags to an object. Tag names are not limited in any way:

- **any number of characters**
- **Cyrillic and Latin**
- **spaces**
- **special symbols**

Tag examples: `new2023, брошенная корзина, abandoned_cart.`

Where I can see my tags

When you save an object with an assigned tag, the tag is also saved. There is no separate window where when editing another object, all previously created tags will be available in the drop-down list:



While a saved tag can be detached from an object, it cannot be completely removed from the platform. A

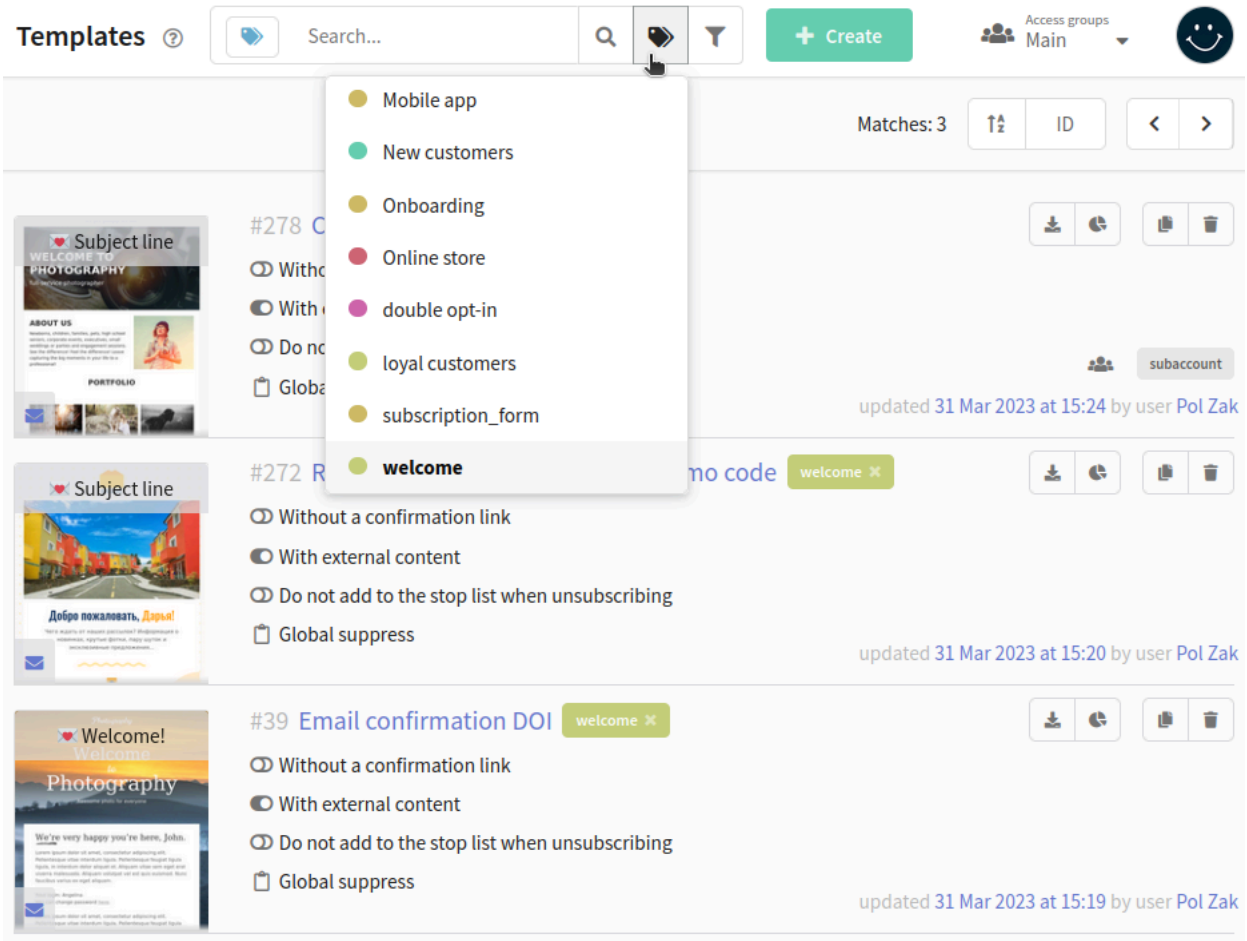
If a tag is not attached to any platform object, it will be automatically deleted.

Where to use tags

1. Search for objects in the general list

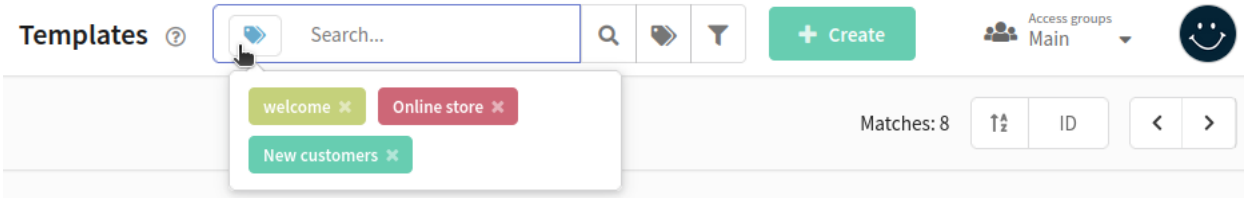
Go to the desired section (for example, Templates) and click the icon

to the right of the search bar. Select one or more tags.



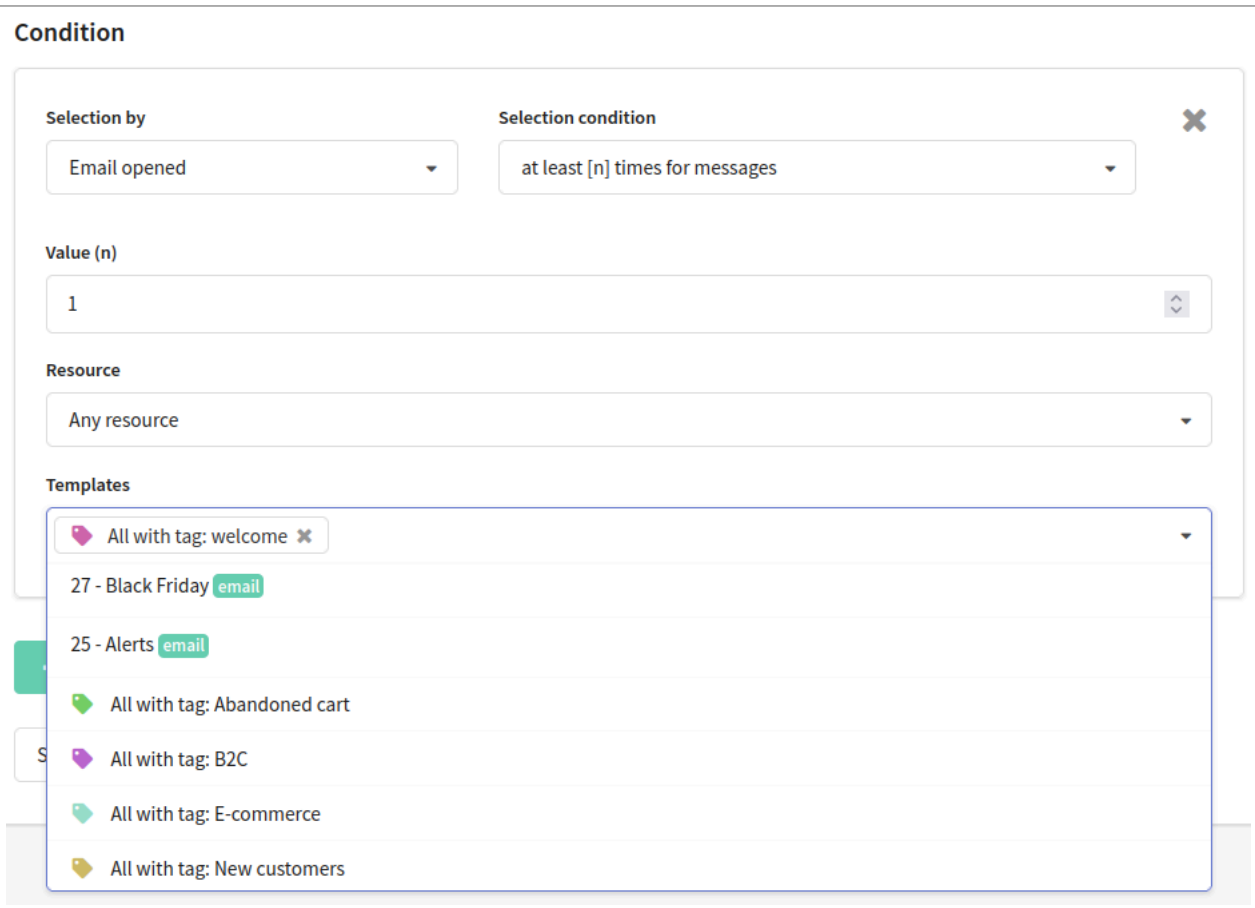
To see what tags have been added to the search, hover over

:



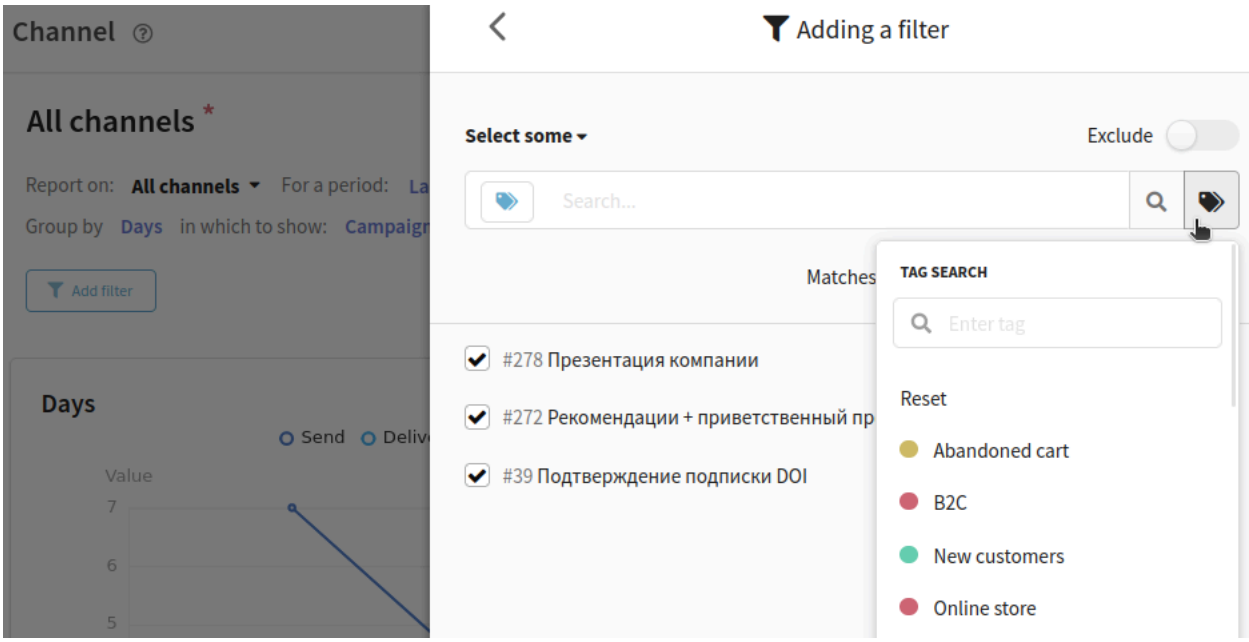
2. Segmentation of profiles by events in mailings or templates (sends, deliveries, opens, clicks)

When you use one of these selection conditions, you can select all mailings or all templates with a speci

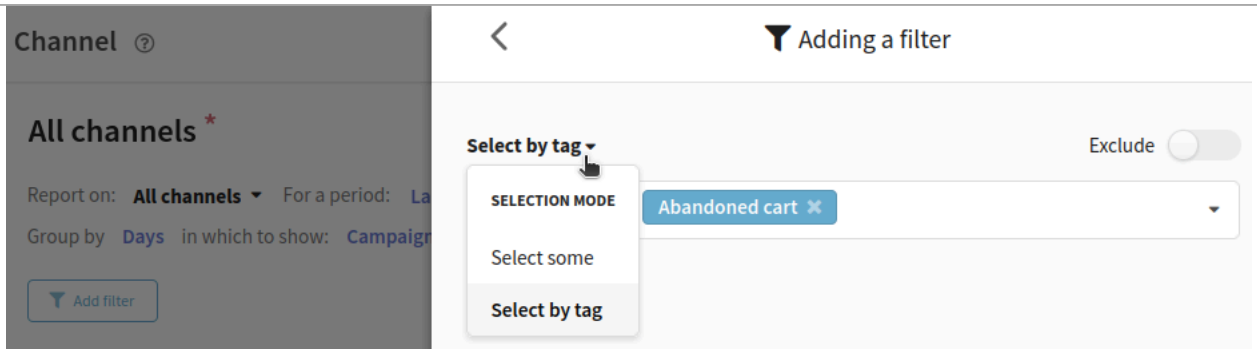


3. Analytical reports

In the channel report and in the traffic report, you can use tags in filters. For example, to quickly find the



You can also select all objects of the same type that have one or more tags assigned at once. To do this, selection of campaigns —> Selection of campaigns by tag. Then select tags in the drop-down list and ap


Channel
All channels *
Report on: All channels For a period: La
Group by Days in which to show: Campaign
Add filter
Adding a filter
Select by tag
Exclude
SELECTION MODE
Select some
Select by tag
Abandoned cart

Attribute settings

What are entity attributes

Attributes are custom user-defined fields that can be added to platform entities. They represent a specif values. Attributes can be used not only to tag specific groups of entities for search, but also as segment into API requests. Attributes can be assigned to campaigns, mailings and scenarios.

> TIP: Attribute inheritance
>
> **Attributes are inherited hierarchically through the chain: Campaign → Scenario → Mailing. This means t for a mailing, the value from the scenario will be used, and if not set there either — the value from the c**

Creating and configuring attributes

To create and configure attributes, go to Settings — Attribute configuration. This opens the attribute ma attribute by clicking the corresponding button.

A configuration window will appear. First, specify the name and the API name for the attribute. Then sele available for — campaigns, mailings or scenarios (at least one type is required).

Next, configure the necessary parameters:

- **Required attribute — the attribute must be filled in for any entity where it is available. Without it, a c be saved.**
- **Disable attribute editing in active entities — prevents changes to the attribute's value once the entit**
- **Limit character count — sets the maximum length of the attribute value.**
- **Use a regular expression — validates the attribute value against a regular expression.**

The next step is to define the attribute values. You can also allow users not only to select from predefine editing the entity. Each attribute can have up to 30 values.

> TIP
>
> **Attributes are inherited hierarchically through the chain: Campaign → Scenario → Mailing. This means t for a mailing, the value from the scenario will be used, and if not set there either — the value from the c**

> INFO: Important
>
> **When deleting an attribute, the platform will warn the user if the attribute is used in any entities. The u deletion of the attribute or cancel the operation and unlink the attribute from the entities manually.**

Later, values can be not only deleted, but also archived. Archived values cannot be assigned to new enti ones and can be used in filters and search.

If needed, set a default value for the attribute. Then save and activate it. After saving, you can deactivat list.

> CAUTION: Displaying changes
>
> **Even saved changes to attribute settings may be displayed incorrectly until the page is reloaded.**
>
> **We recommend refreshing the editing page after each time you save changes.**

Using attributes in the platform

Search and filtering

Attribute values are used to search and filter platform entities.

Profile segmentation

When segmenting by subscriber actions, you can use attribute values to select campaigns, mailings and

GCG exclusions

Users can disable global control groups for scenarios, mailings and campaigns with specific attribute va

LCG exclusions

In local control group (LCG) settings, you can filter profiles by attribute values of campaigns, scenarios a configuration of profile exclusion rules from the control group.

Variable substitution

You can insert the attribute value into templates and scenarios using the `{attribute_value("attribute_name")}` **the attribute's API name.**

Usage examples:

- `{attribute_value("test_attribute")}` — will substitute the value of “test_attribute” attribute
- `{attribute_value("campaign_type")}` — will substitute the value of “campaign_type” attribute
- `{attribute_value("priority")}` — will substitute the value of “priority” attribute

API requests

Parameters with the attribute name and value are available in some API requests to the platform. For mo

Connections

Description

**Connections let you authorize in external systems to expand Altcraft Platform capabilities with the funct
usually need an account in an external system with API access rights to create such a connection.**

Connection settings

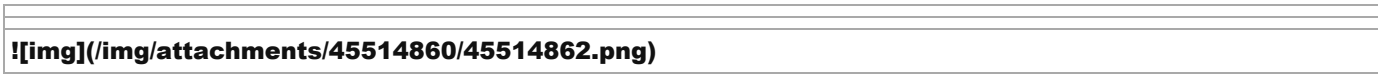
In the main menu click “Settings” —> “Connections”.

Available connections

Connection to Facebook Ads

A connection to Facebook* Ads is required to configure the corresponding integration.

Step 1. Creating a Facebook App



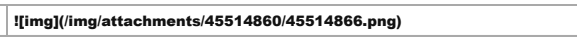
CAUTION

The app must have ads_management rights. In most cases, standard access will be enough (see gettin

Step 2. Creating a Connection in Altcraft

Step 3. Pre-configuring the Facebook Connection

The API version can be viewed in the Settings —> Advanced section on the app dashboard.



Step 4. Initializing the Facebook Connection

Step 5. Choosing a Facebook Ad Account

The connection is ready for use. Proceed to setting up the Facebook Ads Manager integration.

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a

Connection to Google Ads

The connection links a Google Ads account with Altcraft Platform. The Google Ads integration can then static segments with Google Ads audiences.

Step 1. Creating a Google app

TIP

Instructions for creating a Google Ads manager account — follow this link.

Step 2. Creating a connection in Altcraft

![img](/img/attachments/google-ads/google_connection.png)

![img](/img/attachments/google-ads/76808351.jpg)

![img](/img/attachments/google-ads/76808352.jpg)

![img](/img/attachments/google-ads/76808354.jpg)

![img](/img/attachments/google-ads/76808355.jpg)

TIP

More about obtaining the token — in the Google documentation.

Step 3. Initializing the connection

![img](/img/attachments/google-ads/76808356.jpg)

Step 4. Selecting a Google Ads account

![img](/img/attachments/google-ads/76808357.jpg)

The connection is ready to use. Proceed to configuring the Google Ads integration.

Connection statuses

Status	Description
Ready to work	The connection is configured and can be used to upload static segments to Google Audiences.
Connecting	The connection process is in progress. Possibly, the Google authorization was not completed and access rights were not granted.
Connected	The connection is not fully configured; additional steps are required. For example, a Google account must be selected.
Error	An error occurred while working with the API. Make sure the user has access rights.
Connection error	An attempt to configure the connection ended with an error.

Connecting to Yandex.Audience™

Connecting to Yandex.Audience is required to configure the corresponding integration.

Step 1. Register a web application in Yandex

![img](/img/attachments/new-yandex/en/1.png)

![img](/img/attachments/new-yandex/en/2.png)

`https:///connections/init/`, where `` should be replaced with the URL used to access the Altcraft platform.	![img](/img/attachments/new-yandex/en/5.png)

![img](/img/attachments/new-yandex/en/6.png)

![img](/img/attachments/new-yandex/en/email.png)

TIP: Application already exists

If an application has already been created in Yandex.OAuth, update its settings according to the param

---

Step 2. Create a connection in the Altcraft platform

![img](/img/attachments/new-yandex/en/7.png)

![img](/img/attachments/new-yandex/en/8.png)

	![img](/img/attachments/new-yandex/en/9.png)

![img](/img/attachments/new-yandex/en/10.png)

Yandex.Audience integration.

Connection to 360dialog

The 360dialog connection is required to integrate the platform with the WhatsApp* messenger. After c
integration.

> TIP
>
> **First, submit an account registration request with 360dialog via this form.**

Step 1. Create the connection in Altcraft

![img](/img/attachments/whatsapp-connection/360select.png)

![img](/img/attachments/whatsapp-connection/360gialog_general_settings.png)

Step 2. Get an API token

![img](/img/attachments/whatsapp-connection/76808553.jpg)

Step 3. Enter the token and verify

![img](/img/attachments/whatsapp-connection/89326021.png)

Step 4. Save the connection

Click Save.
The connection is ready to use. Proceed to setting up the WhatsApp integration.

***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared**

Connection to Edna

**The connection to Edna is required for integrating the platform with the WhatsApp* messenger. After t
integration can be configured.**

> TIP
>
> **Submit an application for Edna account registration in advance through this form.**

Step 1. Creating a connection in Altcraft

![img](/img/attachments/whatsapp-connection/ednaselect.png)

Step 2. Getting the API token

![img](/img/attachments/whatsapp-connection/89326022.jpg)

Step 3. Entering the token and checking

![img](/img/attachments/whatsapp-connection/89326019.png)

![img](/img/attachments/whatsapp-connection/89326020.png)

Step 4. Saving the connection

Specify the connection name and description at the top of the page. Click Save.
The connection is ready to use. Proceed to configuring the WhatsApp integration.

***The Meta organization, which owns Instagram, Facebook and WhatsApp products, is recognized as ex
of the Russian Federation.***

Connection to Devino Telecom

A connection to Devino Telecom is required to integrate the platform with the Viber*™ messenger. Afte
integration.

Step 1. Register with Devino

Register an account with the service Devino Telecom.

Step 2. Create a Connection in Altcraft

![img](/img/attachments/viber-connection/Devino_en.png)

![img](/img/attachments/viber-connection/devinosettings_en.png)

Step 3. Enter Authorization Data

Select an authorization method:

- **API key — a generated API key.**
- **Login and password — a combination of the Devino.Online account login and password in Base64 e**

TIP

For more details, see the Devino Telecom documentation.

Enter the authorization data.

Step 4. Verify and Save

![img](/img/attachments/viber-connection/en/viber_connection_validation.png)

The connection is ready to use.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Connection to SMSTraffic

A connection to SMSTraffic is required to integrate the platform with the Viber*™ messenger. After the configure the Viber integration.

Step 1. Registering in SMSTraffic

Register an account in the SMS Traffic service.

Step 2. Creating a connection in Altcraft

![img](/img/attachments/viber-connection/SMSTraffic_en.png)

![img](/img/attachments/viber-connection/smssettings_en.png)

Step 3. Verifying and saving

![img](/img/attachments/viber-connection/CheckSuccess_en.png)

The connection is ready to use.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Connection to VK Ads™

A VK Ads™ connection is required to set up the corresponding integration.

INFO

To set up the connection, an account in VK Ads™ with the “Advertiser” status that has passed moder account yet, register via the link. Documentation — here.

Step 1. Creating a connection in Altcraft

![img](/img/attachments/vk-connection/connections_vk_en.png)

![img](/img/attachments/vk-connection/vk_create_en.png)

Step 2. Setting up authorization

Cloud version of the platform:

![img](/img/attachments/vk-connection/accesspermit.jpg)

On-premises version of the platform:

![img](/img/attachments/vk-connection/api_access_request.png)

![img](/img/attachments/vk-connection/API_access_en.png)

Step 3. Connecting and saving

![img](/img/attachments/vk-connection/success_en.png)

INFO

Each successful connection attempt creates an access token. The number of tokens cannot exceed 5 creating another connection, enable the Revoke previously issued access tokens option.

The connection is ready to use. Proceed to setting up the integration with VK Ads.

Connection to MTS OmniChannel

An MTS OmniChannel connection is required to integrate the platform with the Viber™ messenger. Afte integration.

Step 1. Registering in MTS OmniChannel

Register an account on the MTS OmniChannel platform.

Step 2. Creating a connection in Altcraft

![img](/img/attachments/mts/182222.png)

![img](/img/attachments/mts/182434.png)

Step 3. Checking and saving

![img](/img/attachments/mts/1244555.png)

The connection is ready to use. Proceed to configure the Viber integration.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Connection via OAuth 2.0

OAuth 2.0 is an authorization protocol that grants one service rights to access a user's resources on a login and password.

Where it is used:

- **Node "API call" — transfer data to the connected service or request API content.**
- **Action Hooks — transfer events and profile data to an external service.**
- **Scheduled data import in the Market — access to a protected product feed.**

Step 1. Create a connection

![img](/img/attachments/oauth20/oauth_connection.png)

![img](/img/attachments/oauth20/89326069.png)

Step 2. Configure the URI and application data

![img](/img/attachments/oauth20/89326070.png)

Step 3. Connect and save

![img](/img/attachments/oauth20/89326176.png)

Connection settings can be changed at any time by clicking Edit.

***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared**

Basic Authentication connection

Basic Authentication is an HTTP authentication protocol that uses a login and password to access pro

Where it is used:

- **Scenario node “API call” — transferring data to the connected service or requesting API content.**
- **Action Hooks — transferring events and profile data to an external service.**
- **Scheduled data import in the Market — access to a protected product feed.**

Step 1. Creating a connection

![img](/img/attachments/basic/basic-1-en.png)

Step 2. Entering data

![img](/img/attachments/basic/basic-3-en.png)

Step 3. Saving the connection

Click Save. The settings can be changed at any time by clicking Edit.

Token Authentication connection

Token Authentication is an authentication protocol that uses a Bearer token. This method is more secu and password.

Where it is used:

- **Scenario node “API call” — transfer data to the connected service or request API content.**
- **Action Hooks — transfer events and profile data to an external service.**
- **Scheduled data import in the Market — access to a protected product feed.**

Step 1. Creating a connection

![img](/img/attachments/token/token-1-en.png)

Step 2. Entering data

![img](/img/attachments/token/token-3-en.png)

Step 3. Saving the connection

Click Save. The settings can be changed at any time by clicking Edit.

Custom Authentication Connection

Custom Authentication is a connection type with custom HTTP headers for API requests, including the header.

Where it is used:

- **Scenario node “API call” — transferring data to the connected service or requesting API content.**
- **Action Hooks — passing events and profile data to an external service.**
- **Scheduled data import in the Market — connecting to an external product feed.**

Step 1. Creating a connection

![img](/img/attachments/custom/custom-1-en.png)

Step 2. Entering data

![img](/img/attachments/custom/custom-3-en.png)

TIP: Forbidden headers

The following system headers are forbidden: “Host”, “Connection”, “Keep-Alive”, “Accept-Encoding”, “ Encoding”, “Trailer”, “Upgrade”, “Date”, “Location”, “Server”.

Step 3. Saving the connection

Click Save. Settings can be changed at any time by clicking Edit.

Connecting to MAX

Connecting to the MAX messenger is required to configure MAX integrations — the MAX Bot and MAX C

INFO

Create a bot in MAX beforehand. Bot creation is available only for accounts that passed verification fo entrepreneurs. Learn more — in the MAX documentation.

CAUTION: MAX API changes

By July 19, 2026, HTTP requests to the MAX API must be redirected to the domain platform-api2.max. api.max.ru). You must also add the MinTsifra certificate to your list of trusted certificates. More detai

---

Step 1. Getting the bot API token

![img](/img/attachments/max-connection/en/integration-token.png)

Step 2. Creating the connection in Altcraft

![img](/img/attachments/max-connection/en/select.png)

Step 3. Entering the token and checking

![img](/img/attachments/max-connection/en/check.png)

Step 4. Saving the connection

Click Save.

The connection is ready to use. Proceed to setting up MAX integrations.

Connection to Notify

A Notify connection is required to configure the corresponding integration.

INFO

New access to Notify is not provided. Support for current users is available via MAX Business tech su

TIP

The connection is only available to those who already have a Notify account.

Step 1. Creating a connection in Altcraft

![img](/img/attachments/notify-connection/select_en.png)

![img](/img/attachments/notify-connection/name_en.png)

Step 2. Entering parameters and verification

![img](/img/attachments/notify-connection/fields_en.png)

Step 3. Saving the connection

Click Save.
The connection is ready for use. Proceed to Notify integration setup.

Connection to Rapporto

A connection to Rapporto is required to integrate the platform with the Viber™ messenger. After conne configured.

> TIP
>
> **Read the Rapporto documentation to obtain the service ID and password.**

Step 1. Creating a connection in Altcraft

![img](/img/attachments/rapporto-connection/133325.png)

Step 2. Entering data and verification

![img](/img/attachments/rapporto-connection/134016.png)

Step 3. Saving the connection

Click Save.
The connection is ready to use. Proceed to Viber integration setup.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Users, groups and access

In Altcraft, you can restrict the functionality of user accounts as well as tokens used for authorization determined by user or token groups and roles:

- **Groups define the visibility of objects such as databases, templates, campaigns and scenarios, etc**
- **Roles define the set of available actions with visible objects.**

> CAUTION
>
> **Managing account users and access control is only available to users with master privileges.**

Access groups

Groups in Altcraft account allow you to restrict user access to certain account objects, such as templ segments. A group can be thought of as a container for objects. Thus users will work within their cont intentionally affect objects in other groups.

Groups are convenient when you need to organize the work of several departments within one account conditions for third-party specialists to work, for example - designers for working with templates or we debugging integration.

> INFO
>
> **If you need to more strongly separate the areas of responsibility of several company departments, us Accounts can use shared or dedicated resources and components. A common email suppression list i platform.**

Main group

The Main group is created by the system and contains all account objects, including objects belonging custom group, the objects in it are not deleted, but are transferred to the Main group. You can work in other, if division into groups is not required.

> CAUTION
>
> **Master access implies full access to the account's Main group regardless of additionally assigned gro is access to delete profiles. The role for deleting profiles is assigned separately.**

Create / assign group

Groups are assigned and created dynamically when editing account objects - in the General settings bl select it from the dropdown list in the Set group field. For example, for a Message template.

> CAUTION
>
> **Only Master users can create and set groups.**

To create a new group, enter its name in the same field for any account object and press Enter on the k the dropdown list.

> CAUTION
>
> **Each object created by a user is assigned to the group they were in at that moment.**

Switch / delete group

Groups available to the user are displayed in the menu in the upper right corner of the application, nex switch from group to group from any account menu.

> NOTE
>
> **Group deletion is available in the same list - only for master users.**

Access roles

An access role defines what actions with account objects are allowed for a user. To go to role manage the main menu.

The opened list displays existing roles. You can clone or delete a role from the context menu on the rig edit a role, click on its name.

To create a new role, click + Create at the top of the page.

Three parameters are configured for each role:

- **Name, which master users see in role lists.**
- **Groups on which the role restrictions will apply.**
- **Access rules to objects of the selected Groups.**

CAUTION

If the Main group is included in a role, this role will apply to all account objects. Master users are not restrictions. The exception is the ability to delete profiles, which must be activated independently, in

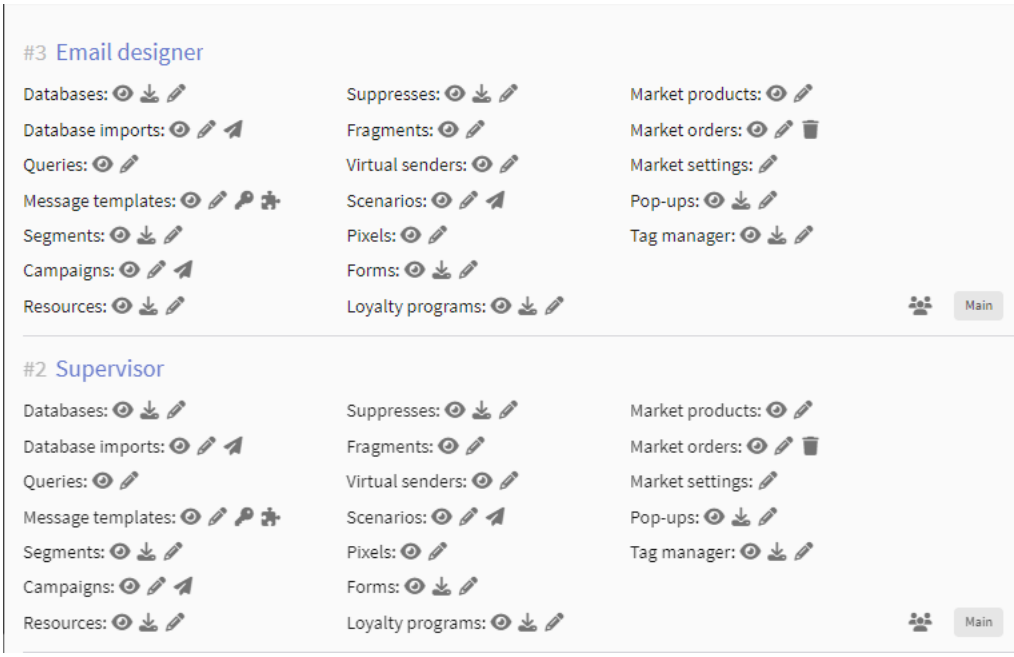
For all account objects, you can allow View, as well as Create and edit. For databases, segments, supp and export is also available - downloading data lists.

NOTE

A role without the Moderation option in the Message access block will only allow saving message tem be available in mailings only after activation - by a user with appropriate access rights. Such a role is accounts creating mailing templates.

For Mailings and Scenarios you can prohibit Launch: users with such a role will only be able to prepare launching them will be done by a user with sufficient access rights.

A role defines which actions are permitted to a user or an API token within the assigned access groups menu select Settings —> Roles:



The list you will see contains the existing access roles. You can edit, clone or delete a role in a contex To edit a role you can as well click on its name. To create a new role click + Create in page header.

Any role has 3 parameters:

- **Role name. A master user can assign roles to users by their unique names.**
- **Groups where the role permissions are applied.**
- **Permitted actions with account objects.**

CAUTION

If Main group is selected in a role, role permissions will be applied to all account objects. Master-use role restrictions. The only exception is the ability to delete profiles, it must be activated independentl Campaigns separate access rights are available:

- **Campaigns - View object — view campaigns and their settings**
- **Campaigns - Create and edit — create and modify campaigns**
- **Campaigns - Launch — activate, deactivate and complete campaigns**

TIP: Permission dependencies

Working with Campaigns requires additional access rights:

- **Campaigns - View object requires view rights for: Mailings, Scenarios, Segments, Databases**
- **Campaigns - Launch requires launch rights for: Mailings and Scenarios**

Roles can restrict user actions with the following platform objects:

- **Resources — tool for managing customer subscriptions to various content through specific commu**
- **Databases — database for storing customer data.**
- **Scheduled database imports — automatic synchronization of Altcraft profile database with externa SQL database).**
- **Relations — used to record interactions between your customers, or customers and company mana**
- **Queries — SQL queries to external databases. The role regulates user work with template queries**
- **Message templates — content you send to subscribers as part of a mailing.**
- **Message fragments — part of a template that can be used in other templates.**
- **Segments — part of a database that includes profiles meeting certain conditions. The role regulate segments.**
- **Suppression lists — list of email addresses, domains and phone numbers excluded from mailing.**
- **Mailings — broadcasts, regular and trigger mailings, multivariate tests, placements.**
- **Campaigns — combination of mailings, scenarios and other activities into a single strategy.**
- **Scenarios — automated chain of conditions and actions that follow when they are met.**
- **Tracking pixels — HTML or JavaScript code placed on a website or in an application, designed to t**
- **Forms — element on a website page through which the user enters and submits their data to you.**
- **Loyalty programs — module for managing promotions and promo codes.**
- **Virtual senders — set of rules for sending messages for real email and SMS sending modules.**
- **Market — module for working with e-commerce product and order data.**

User accounts

Management of Altcraft account user accounts is available both in the user interface and in the Admin interface functionality. To go to user account management, select Settings → Users in the main menu. user accounts.
Go to changing user settings by clicking on the record name. If you want to delete a user, click on the

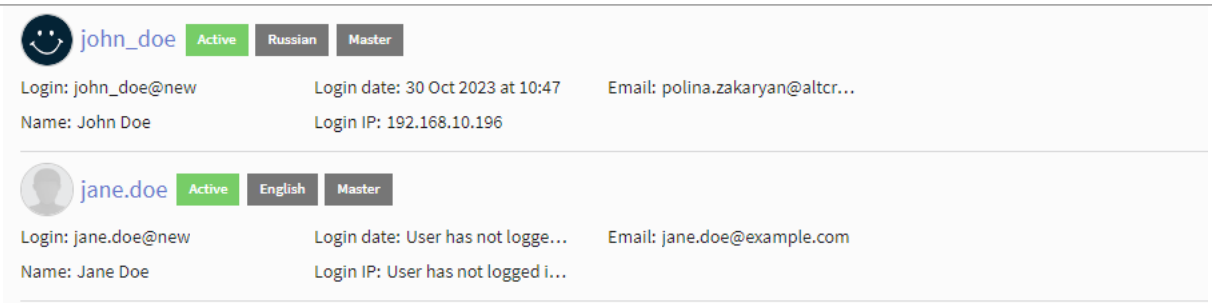
How to create a new user account

Roles can restrict user actions with the following platform objects:

- **Resources is a tool for managing customer subscriptions to various content via specific communic**
- **Databases is a database for storing customer data.**
- **Database imports (scheduled import) is automatic synchronization of the Altcraft profile database example, corporate CRM or SQL database).**
- **Relations are used to record interactions between your clients, or clients and company managers.**
- **Queries are SQL queries to external databases. The role regulates user's actions with templates q queries.**
- **Message templates is the content that you send to subscribers as part of the campaign.**
- **Fragments are parts of the template that can be used in other templates.**
- **Segments are part of a database that includes profiles meet certain conditions. The role regulates dynamic segments.**
- **Suppression lists is a list of email addresses, domains and phone numbers excluded from the maili**
- **Campaigns - broadcasts, regular and trigger campaigns, multivariate tests, placement campaigns.**
- **Scenarios is an automated chain of conditions and actions that will follow when conditions are me**
- **Pixels is HTML or JavaScript code placed on a website or application that is designed to track use**
- **Forms is an element on the page of the website through which the user input and sends you their d**
- **Loyalty is a module that is used to generate and assign promotional codes to profiles.**
- **Virtual senders is a set of rules for real senders and a powerful tool for fine messaging tuning.**
- **Market is a module for working with data about products and orders of an online store.**

User profiles

Managing Altcraft user profiles is available both in User interface and in Administrative panel. This arti User interface procedure. To manage user accounts in the main menu select Settings —> Users:



The list you will see contains the existing account users. You can edit or remove a user in a context m edit a user you can as well click on the record's name. To create a new user click + Create in page hea

General settings

In general settings, specify the following data:

- **Username — account name. Username is used to log into the account, together with the account a**
- **First and last name of the user.**
- **Contact email address — address to which access data will be sent.**
- **Contact phone — user's phone number.**
- **Interface language — choose between English and Russian.**

> **TIP**
>
> **If platform account access is configured via LDAP protocol, through the organization's directory servi account alias will be used for platform authorization: user@company**@account**. The password in th directory service.**

Notification settings

Enter the email address to which notifications will be sent to the user. Also select system notifications

- **Limit exhaustion — notification will be sent if the account tariff has limits configured, and the num messages has reached the maximum;**
- **Subscriber complaints — the percentage of complaints about messages has exceeded the allowab**
- **Hard bounces — the percentage of hard bounce has exceeded the allowable value.**
- **Soft bounces — the percentage of soft bounce has exceeded the allowable value.**
- **Send daily/weekly reports — a report containing main mailing metrics, as well as data on audience be sent to the user's email with a certain frequency.**

Security settings

Set a password for the user to log into the account. The password must contain from 8 to 32 character characters. For more details on password requirements, login blocking after failed attempts, and notifi and login security.

After creating a new user, the password can be automatically transmitted to the owner by checking th To make editing this field possible, make any changes to the password. For example, you can add a pe

If you activate the User must change password on first login field, the user will have to click the "Forge enter the provided username and create a new password.

> **NOTE**
>
> **If platform access is configured only via LDAP, password management in the interface is not used.**

Access to account objects

In the Access block, it is configured which actions the user can perform with objects of certain Groups

- **Master — a user with this setting will have access to any actions in the account's Main group. Acc all objects of all groups. At the same time, role restrictions will not apply in any way. Master users in the main menu, where they can manage other user accounts: delete users, edit their access, do change notification and security settings.**
- **Roles — previously created access roles.**
- **Access groups — groups of account objects that the user will be able to interact with.**

For a user to be able to log into the account, their account must be active, and at least one role and at The chart on the right shows the interaction of user groups and roles:

> **CAUTION**
>
> **If a role does not have a group available to the user, the role rules will not work, and the user will not group's objects.**

If a role has a group that is unavailable to the user - it will remain unavailable.

After completing the setup, click the Save or Apply button at the top of the page:

The Deactivate button will make the account inactive. Users with this setting will not be able to log int

API tokens

API tokens are used for accessing certain platform functions via API. A master user can view, create, e Tokens:

The opened list displays existing tokens. You can go to token editing by clicking on its name. If you wa on the right. To create a new token, click + Create at the top of the page.

For a new token, choose a suitable name or leave the system one. The token itself will become availab token, as for a user account, it is necessary to set groups of account object visibility and related roles

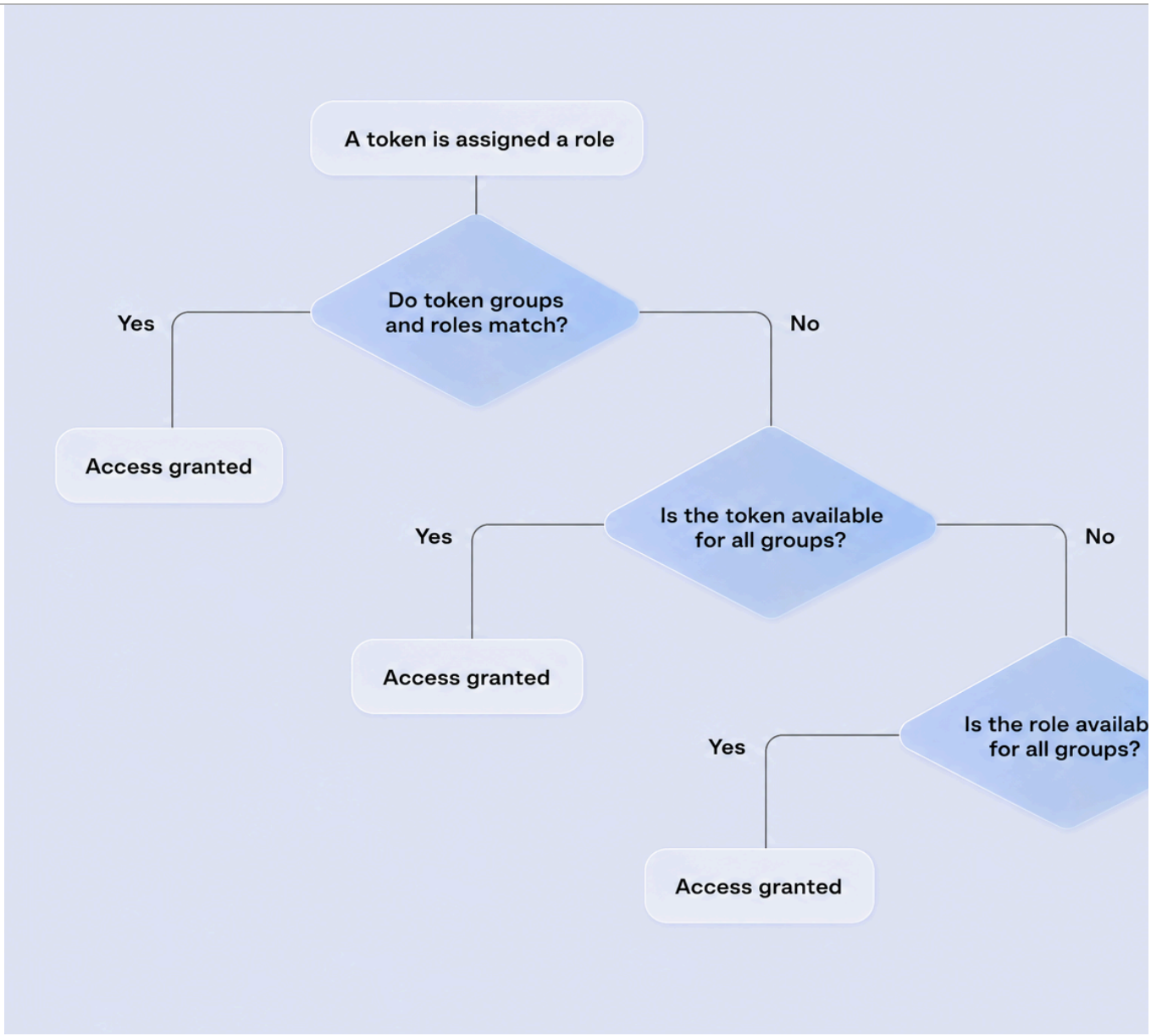
> CAUTION
>
> **If a token is given access to the Main group, all account objects will be available to it, including those**

How token groups and roles combine

When configuring an API token, the group specified in its parameters is matched against the group set token will be able to use the role's permissions if one of these three rules is met:

- **Token group and role group match — for example, both the token and the role belong to the "Marke**
- **Role applies to all groups — "All groups" is selected in the role settings. Such a role applies to toke**
- **Token applies to all groups — "All groups" is selected in the token settings. Such a token can use r**

> INFO: Selecting "All groups"
>
> **"All groups" is not a separate group with its own set of objects. This selection means that the group r side. If "All groups" is set for a token — it works with roles of any group. If set for a role — it is availal**



Examples

Token group	Role group
Marketing	**Marketing**
Marketing	**All groups**
All groups	**Marketing**
Marketing	**Sales**

Password and login security

Altcraft Platform protects user accounts from unauthorized access: passwords must meet specific req trigger temporary blocking, and users receive email notifications about failed login attempts.

Password requirements

The account password must be 8 to 32 characters long and include numbers and special characters. T when it is set or changed and does not allow:

- **reusing a password that was previously set for the account;**
- **using user data in the password;**
- **setting a password that is too simple.**

When requirements are not met, the platform reports the reason: “Password must contain numbers”, “I characters”, “User data cannot be used in password”, “The password you specified has already been u password.”, “Password must be different from the previous one!”.

If the User must change password on first login option is enabled when the user is created, the user w first login to the platform. Creating user accounts is described in the article Users and access separati

Changing the password

The password can be changed in the profile: My profile → Security. In the Password block, enter the cu password and confirm it.

If the current password is incorrect, the platform reports “Invalid current password!”; if the new passw match, it reports “Confirmation password does not match your new password. Please try again.”. A suc the “Password updated successfully!” message.

Password recovery

If the password is lost, use the “Forgot password?” link on the login page. An email with a recovery lin email address, allowing the user to set a new password.

Login blocking after failed attempts

To protect against password brute-forcing, the platform tracks failed login attempts by the IP address counter is shared for an IP address: failed login attempts under different logins from the same address

When the failed attempt limit is exceeded (by default, 5 attempts within 60 seconds), further login atte temporarily rejected, and the platform reports: “The limit of the number of authorization attempts has

> INFO
>
> **The block lasts for 60 seconds after the last failed attempt. Each new failed attempt resets the count another 60 seconds. If no attempts are made within 60 seconds, login from the IP address becomes a**

The attempt limit and the period are set in the platform configuration file and cannot be changed in the Configuration file for details.

Failed login attempt notification

When a login attempt fails, the platform sends the user an email “Failed login attempt” to the contact e the date of the attempt, the browser, and the IP address.

> CAUTION
>
> **If a notification arrives when no login attempt was made, the password should be changed and the su**

Login history

Information about account logins is displayed in the Login history block (My profile → Security). The bl login attempts: login date, device, browser, location, and IP address. The location is determined by the from.

Two-factor authentication

Two-factor authentication provides additional account protection — it is described in the article Two-fa

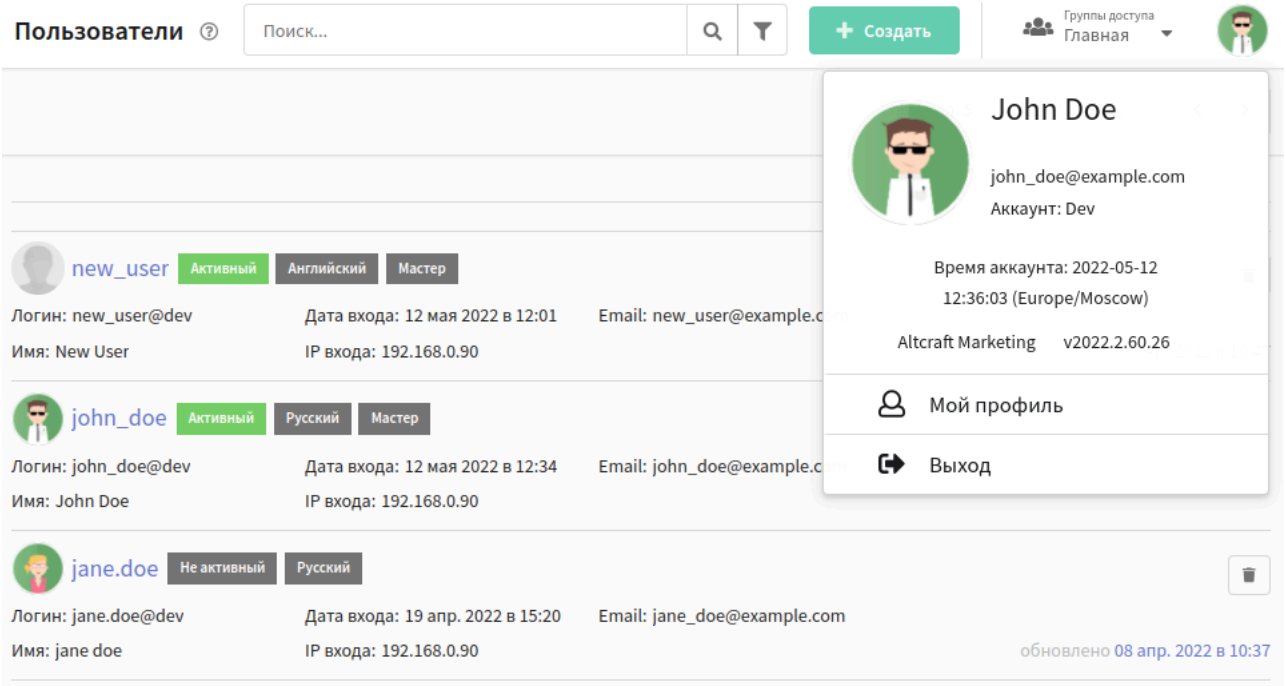
Two-Factor Authentication (2FA)

Description

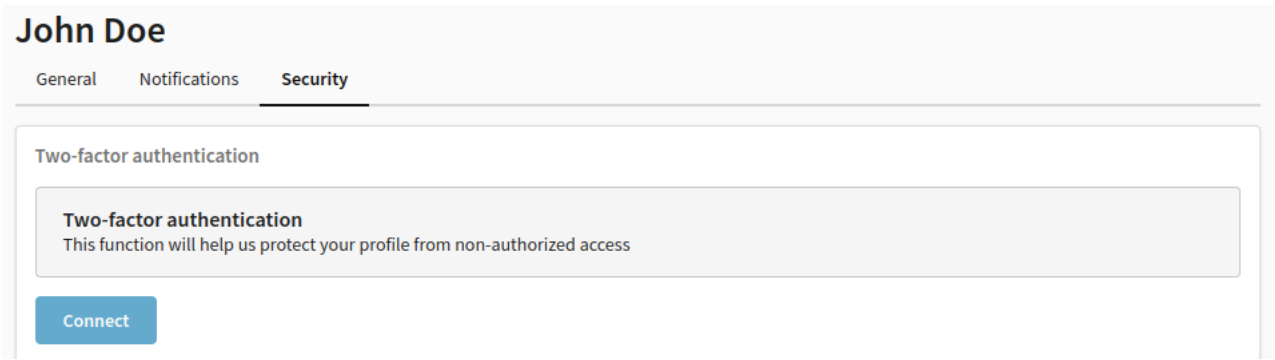
Two-factor authentication (2FA) is a security system that requires not only your username and passwo you. To authenticate, you will need a one-time code generator. For example, Google Authenticator. Onc six-digit code will become a mandatory step every time you log in to the platform.

How to Enable 2FA for an Individual User

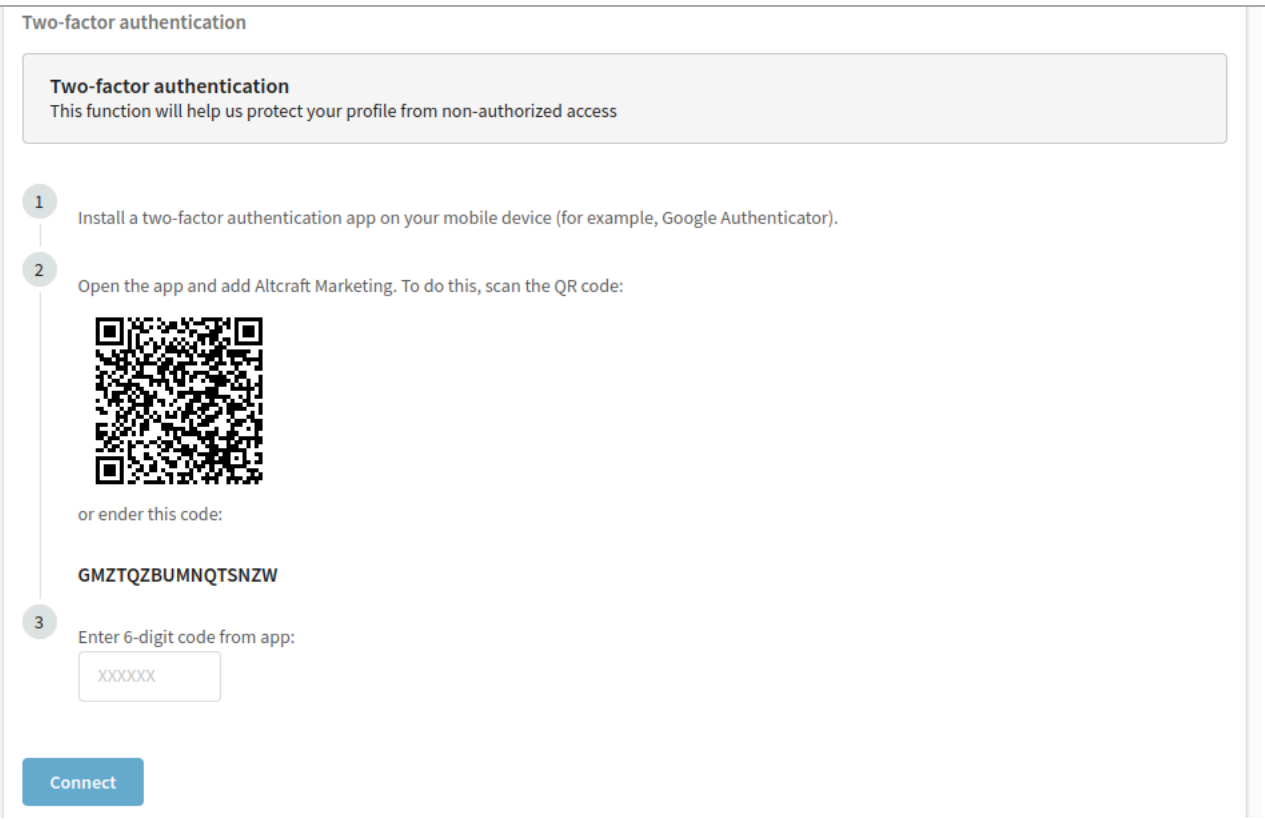
1. Go to your profile settings. To do this, click on the profile picture in the upper right corner. Next —> I



2. Go to Security settings. In the Two-factor authentication block, click “Connect”:



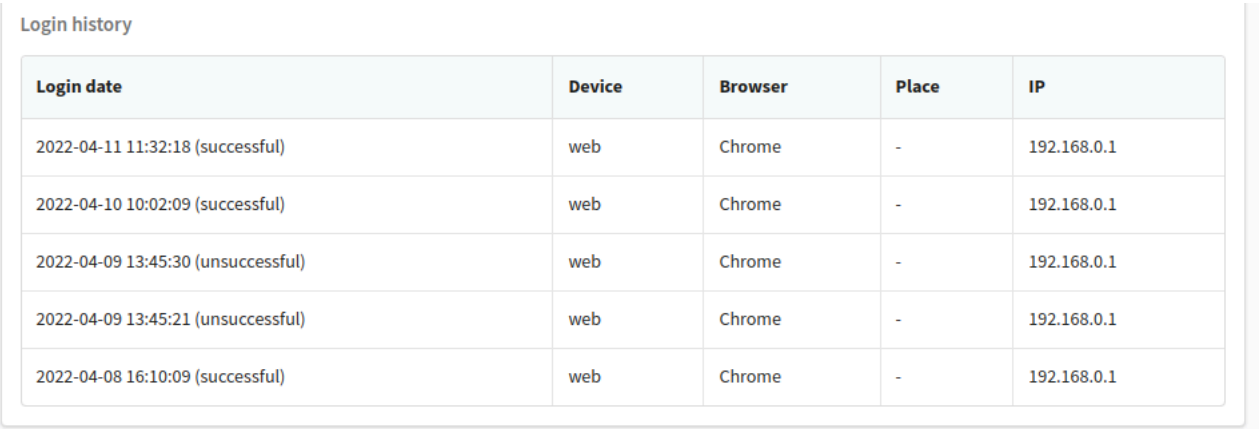
3. The instruction will open. Follow the steps: download the application, add Altcraft Platform in it and Then click “Connect”.



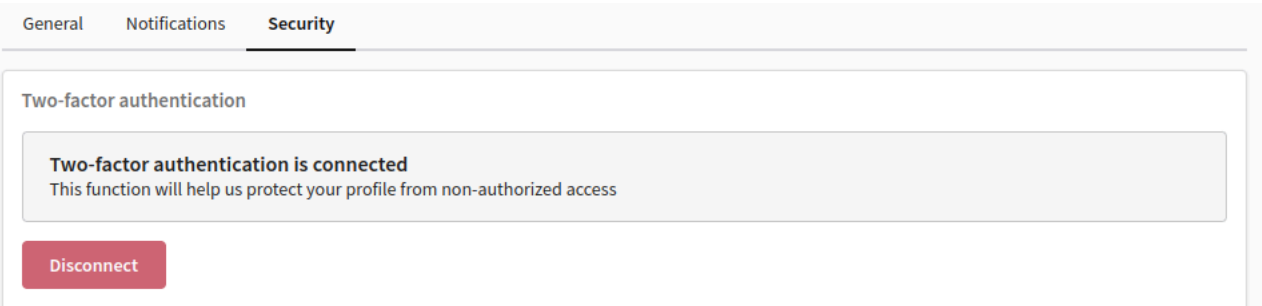
4. A list of recovery codes will appear on the page. Use them to restore access to your account in case 2FA application is installed. Each recovery code can only be used once. It will be impossible to open th in an accessible place.

The 2FA setup is complete. Now, every time the user logs in to the platform, they will have to enter not but also a six-digit code sent to them in the authentication application.

Information about successful and failed login attempts will be displayed in Login history:

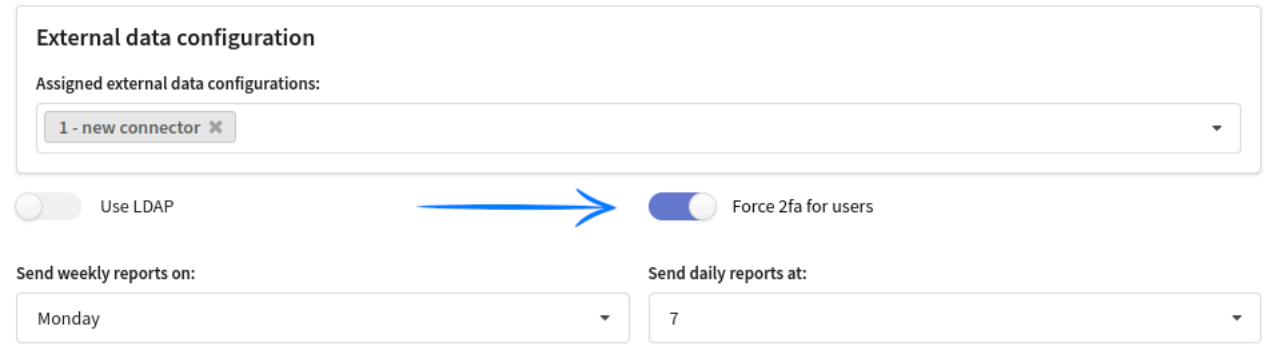


If you no longer need two-factor authentication, click on “Disconnect”:

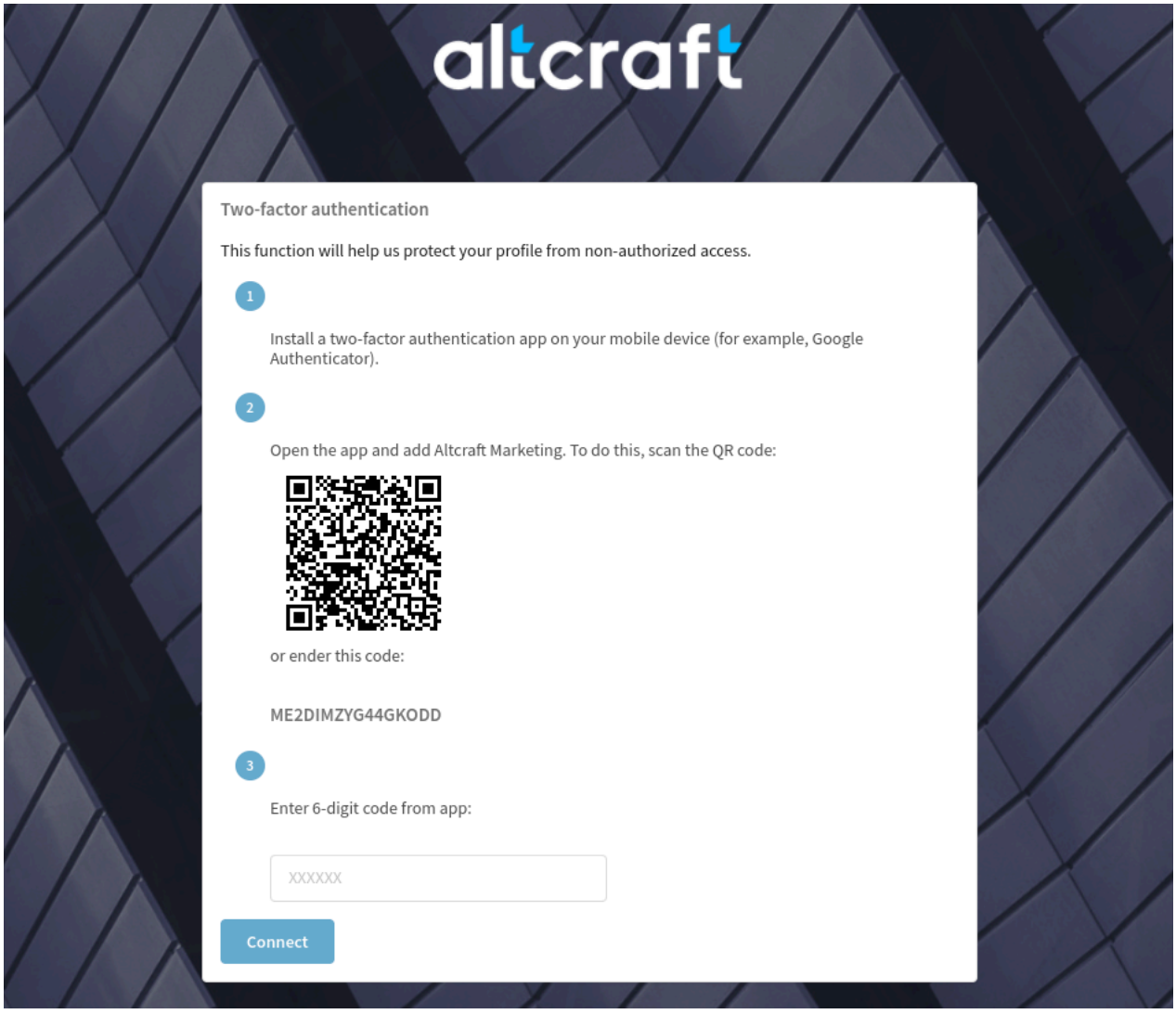


How to Enable 2FA for All Account Users

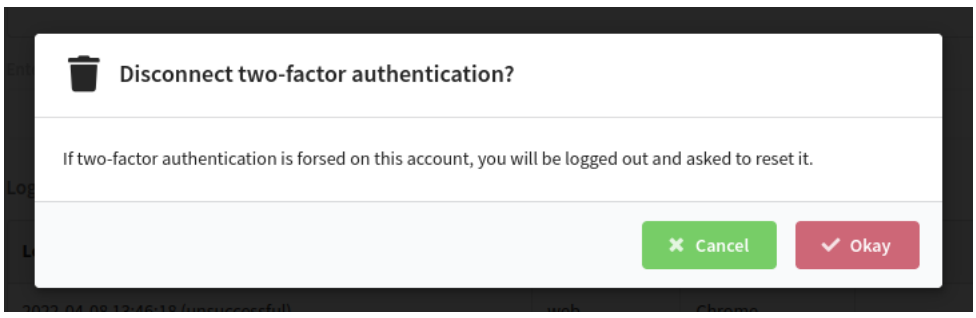
To enable mandatory two-factor authentication for all accounts, the platform administrator must activa account settings in the Administration Panel:



Once you've activated mandatory 2FA, any user will have to set up two-factor authentication to log in t will open when the user enters the correct username and password:



If two-factor authentication is mandatory for all accounts, the user won't be able to disable it. If they c window will pop up:



If, despite the warning, the user disables 2FA, he will be logged out from the platform. To log in, he will

Facebook Ads Manager™

The Facebook*™ Ads Manager integration allows synchronizing static segment data of Altcraft Platfor Facebook*™.

TIP

A connection must be set up for the integration. Instructions — here.

Step 1. Creating the integration

![img](/img/attachments/41484416/95715597.png)

Step 2. Setting up the integration

![img](/img/attachments/41484416/41484424.png)

![img](/img/attachments/41484416/41484425.png)

Step 3. Data type for profile search

![img](/img/attachments/41484416/41484427.png)

Data type for search	Description
Contact data	**Email and phone numbers (including from subscriptions)**
Geo data	**Country, city**
Personal data	**First name, last name, birth date, gender**
External ID	**Matching on the selected profile field**

By default — contact data. Selecting only geo data or personal data is not allowed: contact data or an together with them.

External ID

![img](/img/attachments/41484416/41484431.png)

The external ID must be set up both in Altcraft (custom field) and in Facebook*™ (External ID). More de documentation.

TIP

Sometimes multiple profiles are created in a Facebook*™ audience for one subscriber — if the profile resources, each subscription is transferred separately.

Step 4. Saving the integration

Click Save. The integration is ready to work.

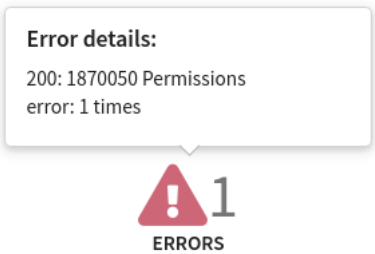
Facebook™ integration statistics

TIP: Integration schedule

Resynchronization starts automatically when new profiles appear in the linked segment.

![img](/img/attachments/41484416/41484433.png)

Synchronization errors — click the icon to view details:



The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared a

Google Ads Audiences

The integration syncs clients from Altcraft static segments with Google Ads audiences for advertising Display Network.

> TIP
>
> **The integration requires a configured connection. Instructions — here.**

Step 1. Creating the integration

![img](/img/attachments/9407286/sql-type.png)

Step 2. Configuring the integration

![img](/img/attachments/76808315/76808361.png)

![img](/img/attachments/76808315/76808362.png)

> TIP
>
> **Google Ads cannot serve ads for a segment with fewer than 100 profiles.**

![img](/img/attachments/76808315/76808320.png)

Step 3. Saving the integration

Click Save. The integration is ready to use.

Synchronization statistics

![img](/img/attachments/76808315/76808328.png)

WhatsApp

WhatsApp* integration allows:

- **send personalized messages in the messenger;**
- **segment the audience by user behavior;**
- **use the WhatsApp channel in scenarios;**
- **collect statistics and analyze data.**

CAUTION

User consent is required to send a message. Learn more.

The integration works through providers: Edna or 360dialog.

TIP: Connections

Register an account with the provider, then set up a connection:

- **Connection to Edna**
- **Connection to 360dialog**

TIP

In addition to the provider's services, the sent messages themselves are also paid for. Rates — here.

Step 1. Opening the integration

![img](/img/attachments/76808817/76808827.png)

![img](/img/attachments/76808817/89326047.png)

Setup via Edna

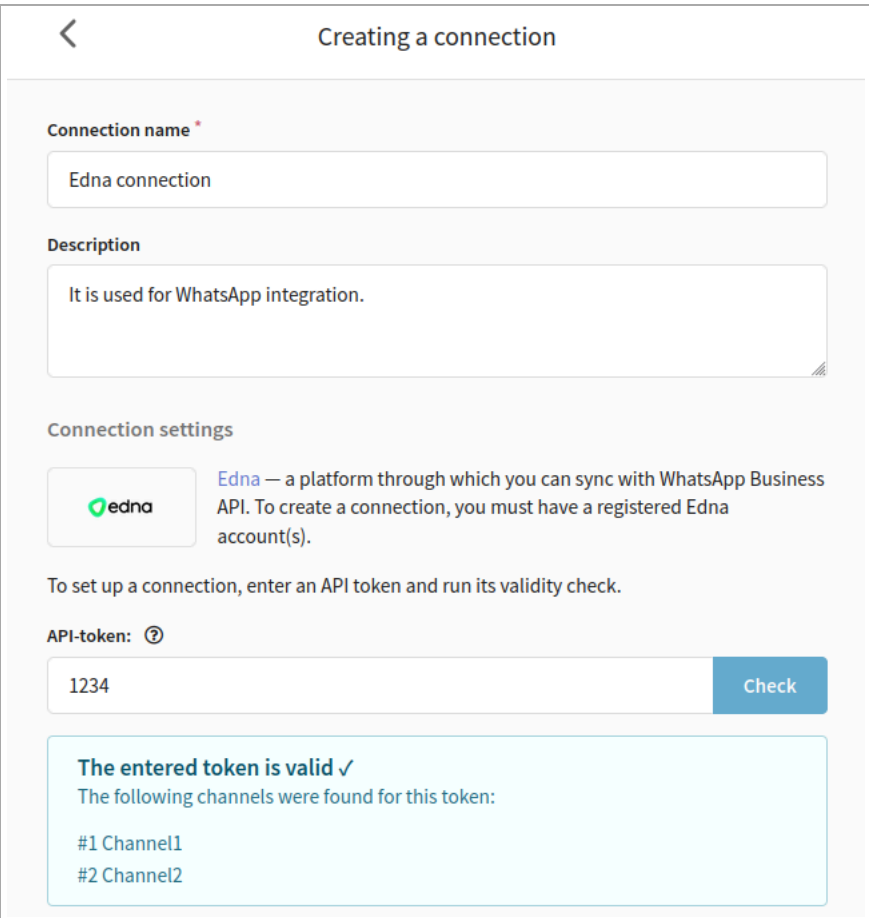
Step 2. Activating and adding a channel

Enable the activation toggle. Click Add channel. Select an available connection:

![img](/img/attachments/76808817/89326049.png)

No available connections

Click Create. Enter a name, description, and API token from the Edna personal account. Run a check. I
For details — connection to Edna.



Step 3. Selecting a channel and cascade

Select an active channel (status “Works correctly”). Specify the cascade — a scenario for sequential s

CAUTION

Only cascades with a single channel are available.

If necessary, enable event capture in Edna — message status data will be sent to Altcraft.

Click Add:

![img](/img/attachments/76808817/89326048.png)

![img](/img/attachments/76808817/89326051.png)

Click Save.

Setup via 360dialog

Step 2. Activating and adding an account

Enable the activation toggle. Click Add account:

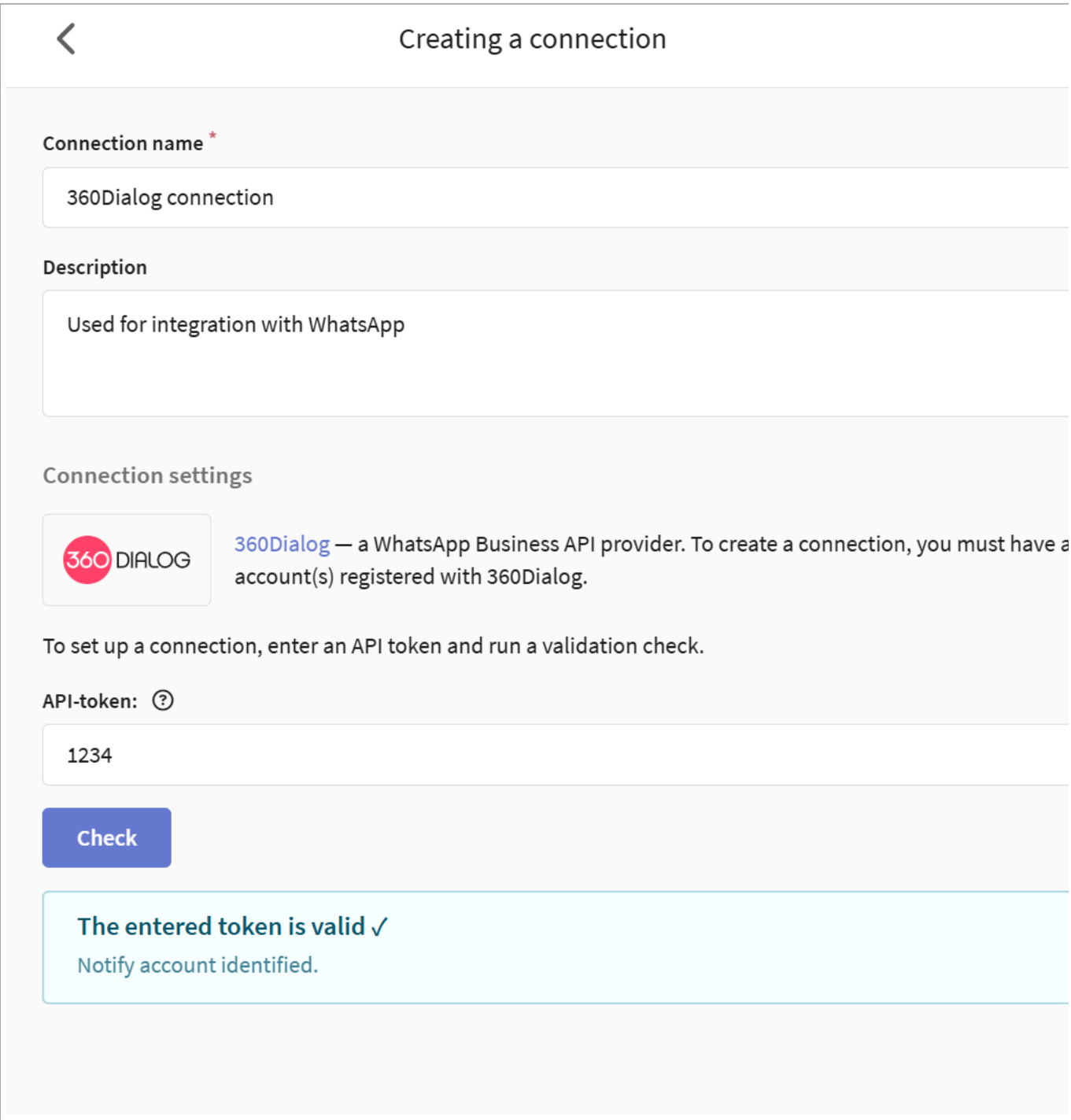
![img](/img/attachments/76808817/89326053.png)

![img](/img/attachments/76808817/89326055.png)

No available connections

Click Create. Enter a name, description, and API token from the 360 Client Hub. Run a check. If the tok

For details — connection to 360dialog.



Step 3. Filling in the Namespace

![img](/img/attachments/76808932/en/namespace_2.png)

![img](/img/attachments/76808932/360Dialog.png)

If necessary, enable event capture in 360dialog.

Click Add. Multiple accounts can be added. To remove — click

:

![img](/img/attachments/76808817/89326056.png)

Click Save.

After setting up the integration, add a WhatsApp channel to a resource.

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Static segment synchronization

Data from [static segments] can be synchronized with external SQL tables. Updates occur on the followi

- **Adding/removing profiles to/from a segment**
- **Clearing a segment**
- **Creating/updating/deleting a segment**

> TIP: SQL connector
>
> **Create a connector to the external SQL storage beforehand. Instructions — here.**

Step 1. Create an Integration

![img](/img/attachments/9407286/sql-type.png)

Step 2. Set the Integration Scope

![img](/img/attachments/9407286/9407291.png)

Step 3. SQL Queries for Event Synchronization

![img](/img/attachments/9407286/9407361.png)

Adding and Removing Profiles

> CAUTION
>
> **Queries for adding and removing profiles are mandatory.**

Export profiles one by one or in batches. For batches, set the Batch size and Timeout (default — 60 sec

Example of table structure

```
mysegment

+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| segId    | int(11)      | YES  |     | NULL    |       |
| clientId | varchar(255) | YES  |     | NULL    |       |
| updated  | datetime     | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

`segId` **— segment ID,** `clientId` **— client ID,** `updated` **— event time.**

Example of a batch query for adding profiles

```
INSERT INTO mysegment (segId, clientId, updated)
  VALUES
  {for $index, $lead = $batch}
    ({segment_id}, "{$lead.CustomID}", NOW()){autocomma}
  {end}
  ON DUPLICATE KEY UPDATE
    updated = NOW();
```

`{segment_id}` **— segment ID,** `{$lead.CustomID}` **— custom profile field. [Altcraft variables].**

Example of a batch query for removing profiles

```
DELETE FROM mysegment
WHERE segment_id = {segment_id}
AND clientId IN ({batch_in "CustomID"})
```

Example of a query for clearing a segment

```
DELETE FROM mysegment
WHERE segment_id = {segment_id}
```

Adding/Updating and Deleting a Segment

Example of table structure

segments

```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(255) | YES  |     | NULL    |       |
| segment_id  | int(11)      | YES  |     | NULL    |       |
| updated     | datetime     | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

Example of a query for adding/updating a segment

```
INSERT INTO segments (segment_id,name,updated)
  VALUES
     ({segment_id}, "{segment_name}", NOW())
  ON DUPLICATE KEY UPDATE
    name = "{segment_name}",
    updated = NOW();
```

Example of a query for deleting a segment

```
DELETE FROM segments
WHERE segment_id = {segment_id}
LIMIT 1
```

INFO

Single quotes `'` in profile data are escaped:

Database	Sym
MySQL, MSSQL, PostgreSQL, Oracle	`'`
ClickHouse	`\`

Step 4. Save the Integration

Click Save. The integration is ready to use.

Synchronization Statistics

![img](/img/attachments/9407286/9407374.png)

Viber™

The Viber™ integration allows:

- **sending personalized messages in the messenger;**
- **segmenting the audience based on user behavior;**
- **using the Viber channel in scenarios;**
- **collecting statistics and analyzing data.**

The integration works through providers: Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto.

TIP: Connections

Register an account with a provider, then set up a connection:

- **Connection to Devino Telecom**
- **Connection to SMS Traffic**
- **Connection to MTS OmniChannel**
- **Connection to Rapporto**

INFO

Message costs are determined by the provider. Altcraft does not charge additional fees.

Step 1. Opening the integration

![img](/img/attachments/9407286/sql-type.png)

Setup via Devino

Step 2. Activation and adding a connection

Enable the activation switch. Click Add connection:

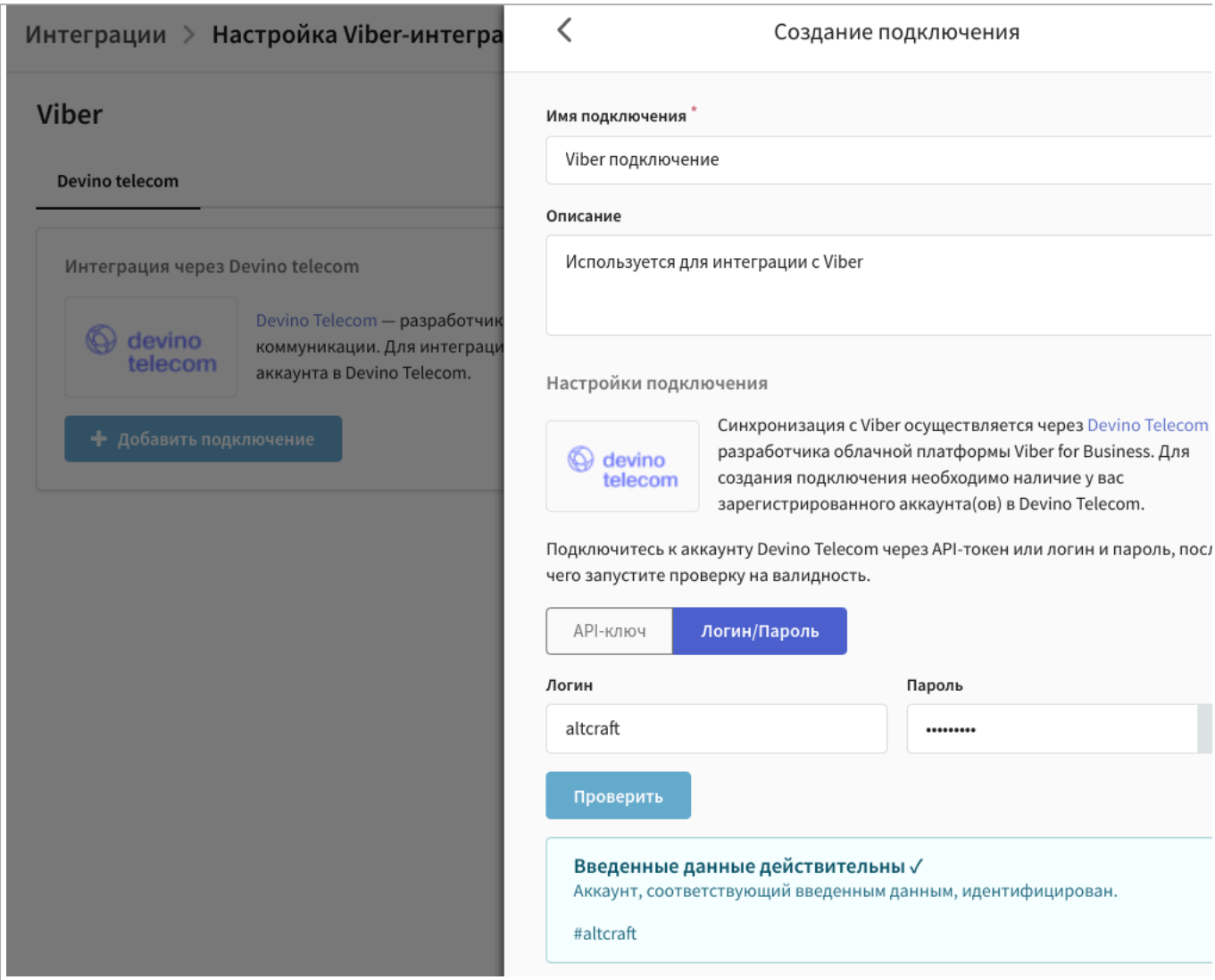
![img](/img/attachments/viber-integration/190116.png)

![img](/img/attachments/viber-integration/select_viber_connection.png)

No connections available

1. **Click Create.**
2. **Enter the name and a description.**
3. **Enter the authorization data (API key or login/password).**
4. **Click Check.**
5. **If the data is valid — click Create.**

Learn more — connection to Devino Telecom.



	![img](/img/attachments/viber-integration/active_viber_integration.png)

Click Save.

Setup via SMSTraffic

Step 2. Activation and adding a connection

Enable the activation switch. Click Add connection:

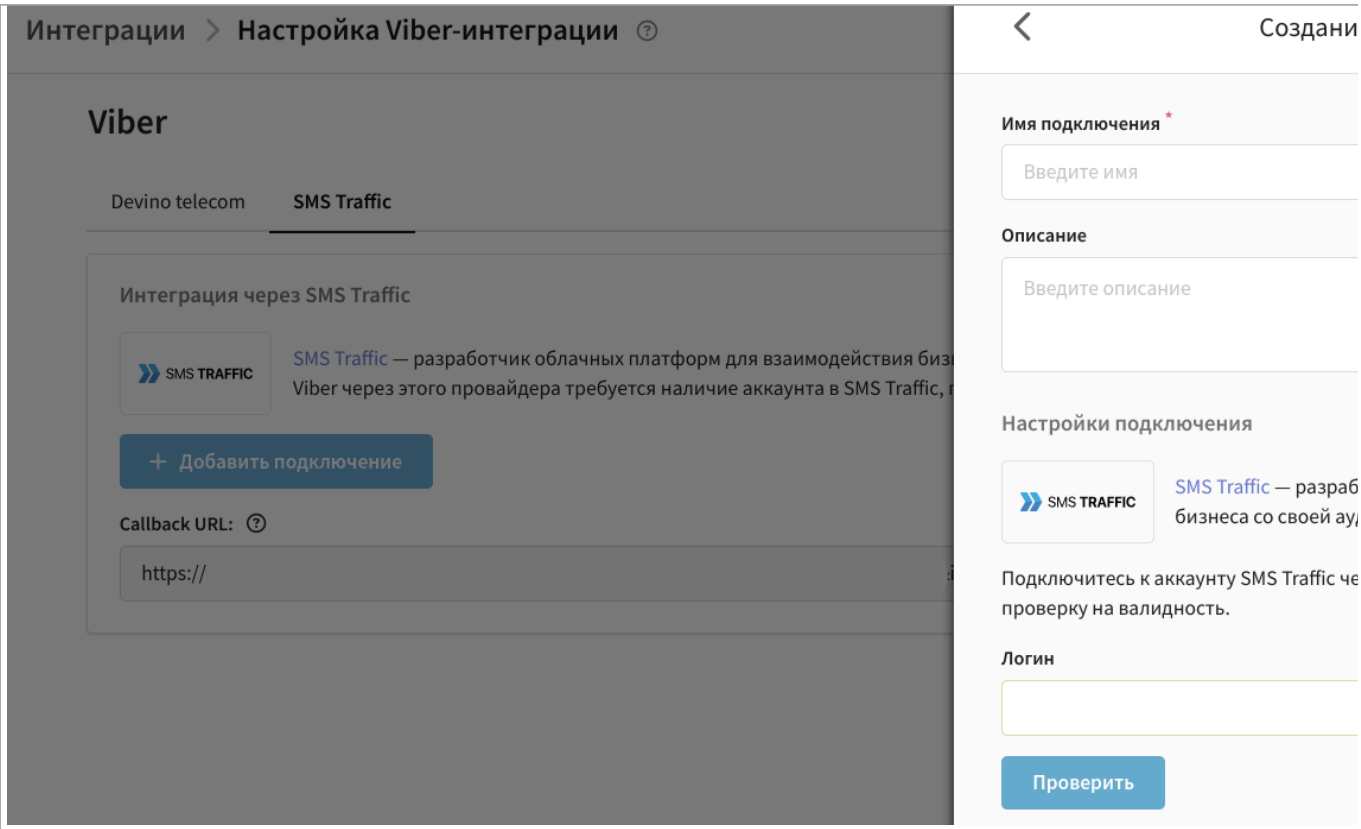
![img](/img/attachments/viber-integration/190111.png)

![img](/img/attachments/viber-integration/smschooseconnection.png)

No connections available

1. Click Create.
2. Enter the name and a description.
3. Enter the login and password.
4. Click Check.
5. If the data is valid — click Create.

Learn more — [connection to SMS Traffic](#).



![img](/img/attachments/viber_guide/callback.png)

	![img](/img/attachments/viber-integration/smsokgood.png)

Setup via MTS OmniChannel

Step 2. Activation and adding a connection

Enable the activation switch. Click Add connection:

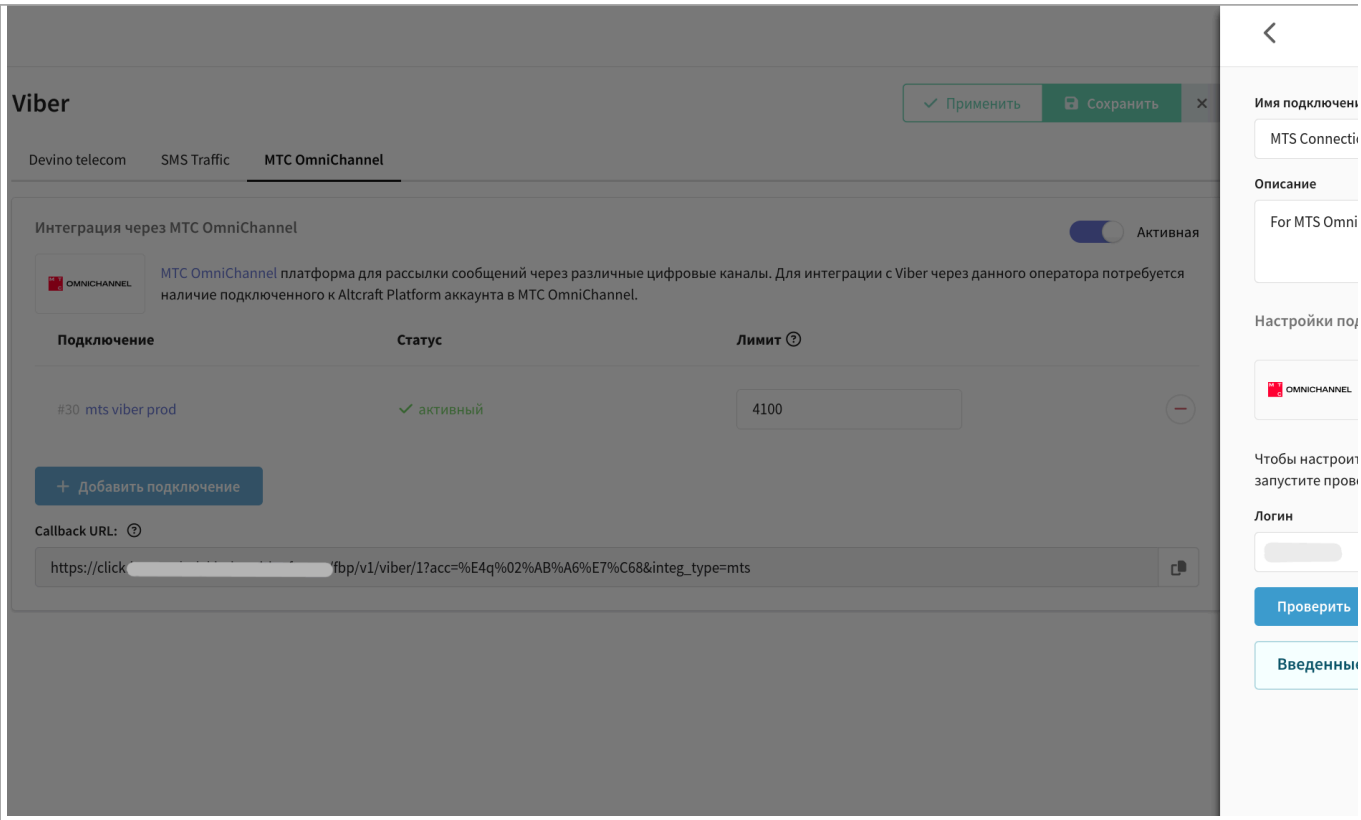
![img](/img/attachments/viber-integration/190105.png)

![img](/img/attachments/viber-integration/1235444.png)

No connections available

1. **Click Create.**
2. **Enter the name and a description.**
3. **Enter the login and password.**
4. **Click Check.**
5. **If the data is valid — click Create.**

Learn more — [connection to MTS OmniChannel](#).



![img](/img/attachments/viber-integration/en/mts_omni.png)

Click Save.

Setup via Rapporto

Step 2. Activation and adding a connection

Enable the activation switch. Click Add connection:

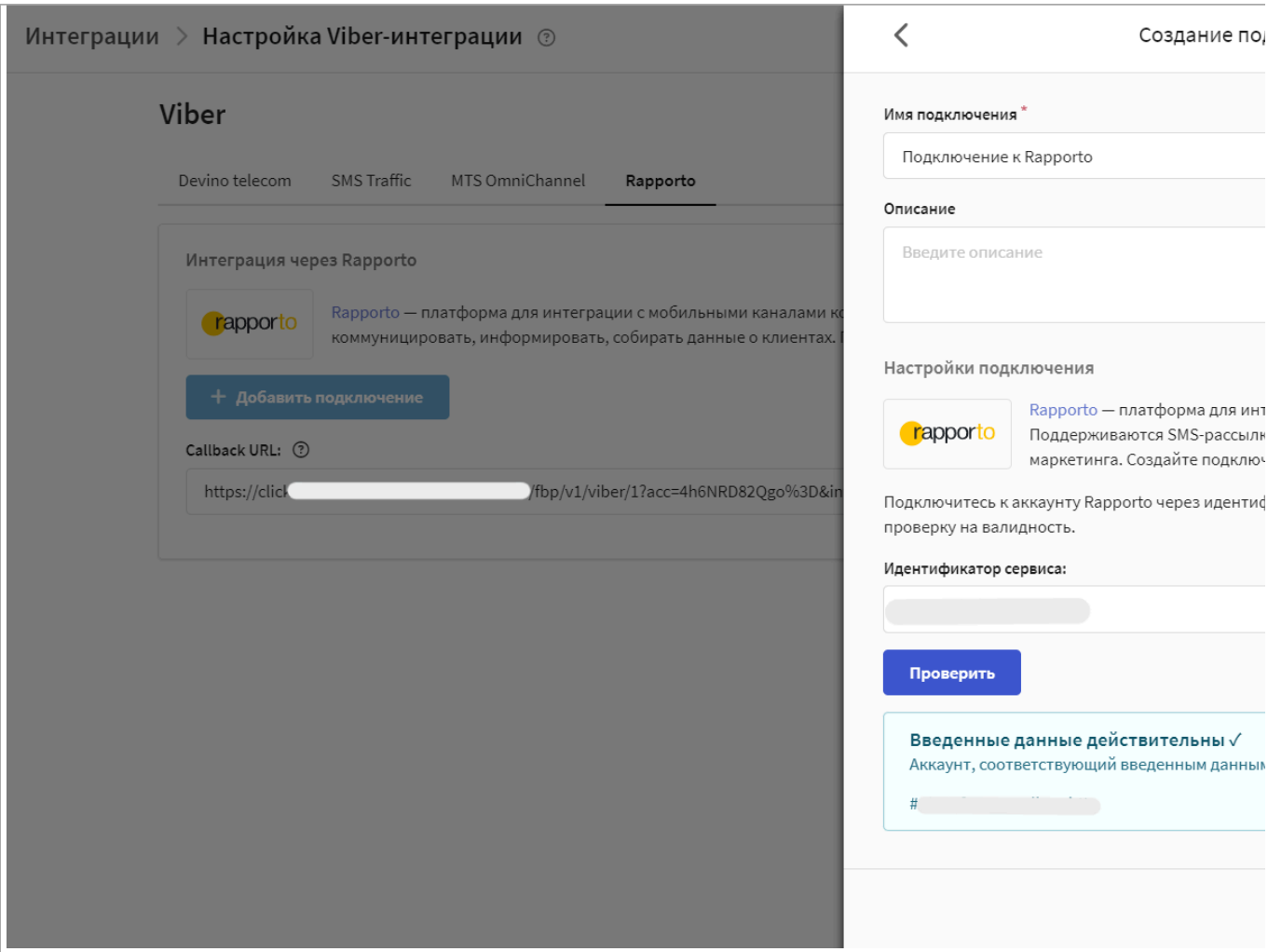
![img](/img/attachments/viber-integration/153148.png)

![img](/img/attachments/viber-integration/153743.png)

No connections available

1. **Click Create.**
2. **Enter the name and a description.**
3. **Enter the service ID and password.**
4. **Click Check.**
5. **If the data is valid — click Create.**

Learn more — [connection to Rapporto](#).



	![img](/img/attachments/viber-integration/154858.png)

Click Save.

Finishing the setup

After setting up the integration, add the Viber channel to the resource.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

VK Ads™

The integration synchronizes static segments of Altcraft with audiences in VK Ads. Transfer segments finds the right people and launches targeted advertising.

TIP

To integrate, set up a connection first. Instructions — here.

Step 1. Create the integration

![img](/img/attachments/9407286/sql-type.png)

Step 2. Configure the integration

![img](/img/attachments/vk-integration/conn_en.png)

Instructions for creating a connection — in this article.

![img](/img/attachments/vk-integration/main_en.png)

![img](/img/attachments/vk-integration/integration_settings_en.png)

CAUTION

A segment with fewer than 100 profiles cannot be transferred. The integration uploads data only from

---

![img](/img/attachments/vk-integration/schedule_en.png)

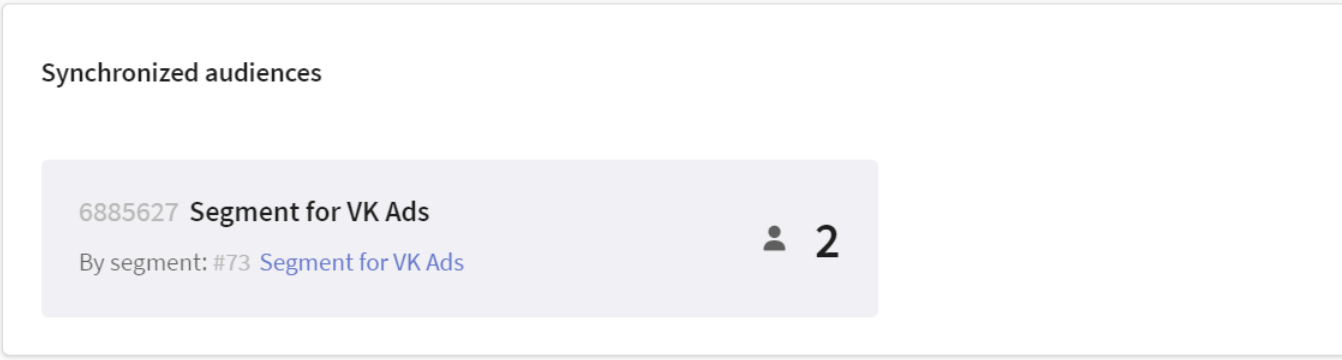
Step 3. Save the integration

Click Save. The integration is ready to use.

Synchronization and statistics

![img](/img/attachments/vk-integration/syn_en.png)

The Resynchronize button deletes all synchronized audiences and re-uploads segments from the scope



TIP

The integration transfers only profiles that appeared after it was launched. For a full transfer, run Res integration.

Integration statistics contains information about synchronizations over the week, actions, and errors:

Integration statistics for: Fri, 24 May 2024

User data

1926
Contacts transferred

1924
Profiles processed

Actions with audiences

1
Audiences created

0
Audiences removed

Transfer of data for the last week

TIP

The Contacts transferred item counts each communication channel of a profile. If a profile has a phor contacts appear in the statistics.

Notify

The Notify integration allows using the service as a communication channel for sending transactional a

INFO

New access to Notify is not provided. Support of current users is provided through MAX Business tecl

TIP: Connecting to Notify

Pre-configure the connection. Instructions — here.

Step 1. Creating the integration

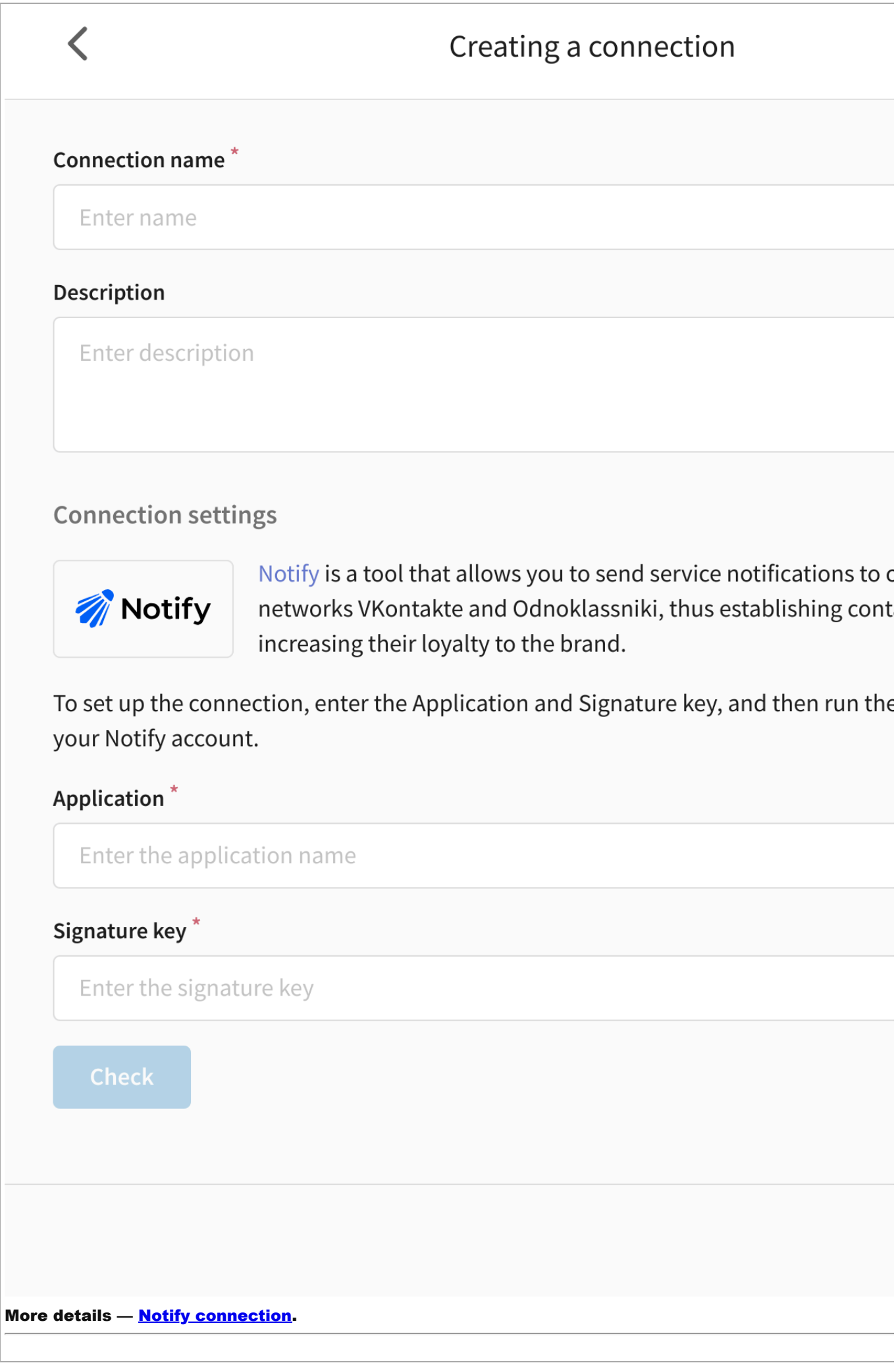
![img](/img/attachments/9407286/sql-type.png)

Step 2. Adding the connection

![img](/img/attachments/notify-integration/sel_conn_en.png)

No available connections

Click Create. Enter the name, description, and Application and Signature key parameters. Click Check.



More details — [Notify connection](#).

Step 3. Integration parameters

![img](/img/attachments/notify-integration/fields_en.png)

Click Add. If the data is incorrect, the platform will ask to verify it.

Step 4. Activating and saving

![img](/img/attachments/notify-integration/fin_en.png)

After configuration, add the Notify channel to the resource.

Integration with the MAX messenger

Integration with MAX enables using the messenger as a communication channel. Two types of integrat

- **MAX Bot — Messages via a bot.**
- **MAX Group — Messages to chats (groups).**

> CAUTION
>
> **Take into account the internal rules of the MAX messenger.**

> TIP: MAX connection
>
> **Set up the connection in advance. The instruction is here.**

Step 1. Opening the integration

![img](/img/attachments/max-integration/en/select.png)

MAX Bot integration

Step 2. Adding a connection

![img](/img/attachments/max-integration/en/bot-connection.png)

Step 3. Activation and saving

![img](/img/attachments/max-integration/en/bot-save.png)

The integration is ready. The next step is configuring the resource.

MAX Group integration

Step 2. Adding a connection

![img](/img/attachments/max-integration/en/group-connection.png)

Step 3. Entering the chat_id

![img](/img/attachments/max-integration/en/group-chat-id.png)

> TIP: How to find the chat_id
>
> **Open web.max.ru, go to the desired chat, and look at the URL in the address bar. The number after th**

Step 4. Activation and saving

![img](/img/attachments/max-integration/en/group-save-new.png)

The integration is ready. The next step is configuring the resource.

Yandex AppMetrica™

Yandex AppMetrica™ is an analytics platform for mobile applications with push notification delivery.
Website: https://appmetrica.yandex.ru
Altcraft can capture events and installations from AppMetrica for:

- **registering target actions and their value;**
- **importing or updating subscriber profiles.**

Step 1. Connecting to the AppMetrica™ API

![img](/img/attachments/9404929/9404931.png)

![img](/img/attachments/9404929/9408042.png)

Step 2. Setting up event captures

![img](/img/attachments/9404929/9404940.png)

Event fields

The integration supports all fields from the AppMetrica Logs API:

- **Events — documentation**
- **Installations — documentation**

> INFO
> **Enter additional parameters from event_json manually in Field name, separating levels with a dot: `eve` `event_json.contacts.email`. Altcraft supports up to 10 nesting levels.**

Step 3. Selecting the action type

Goal registration

![img](/img/attachments/9404929/9404946.png)

To attach a goal to a profile, enable Attach to profile. Select a database and a matching mode:

- **By email from profile**
- **By email from profile or subscriptions**
- **By AppMetrica ID — by `appmetrica_device_id`**
- **By custom field — by an additional profile field**

Profile import

![img](/img/attachments/9404929/9404948.png)

> CAUTION
> **It is not recommended to change the matching mode from email/custom to `appmetrica_device_id` within a**

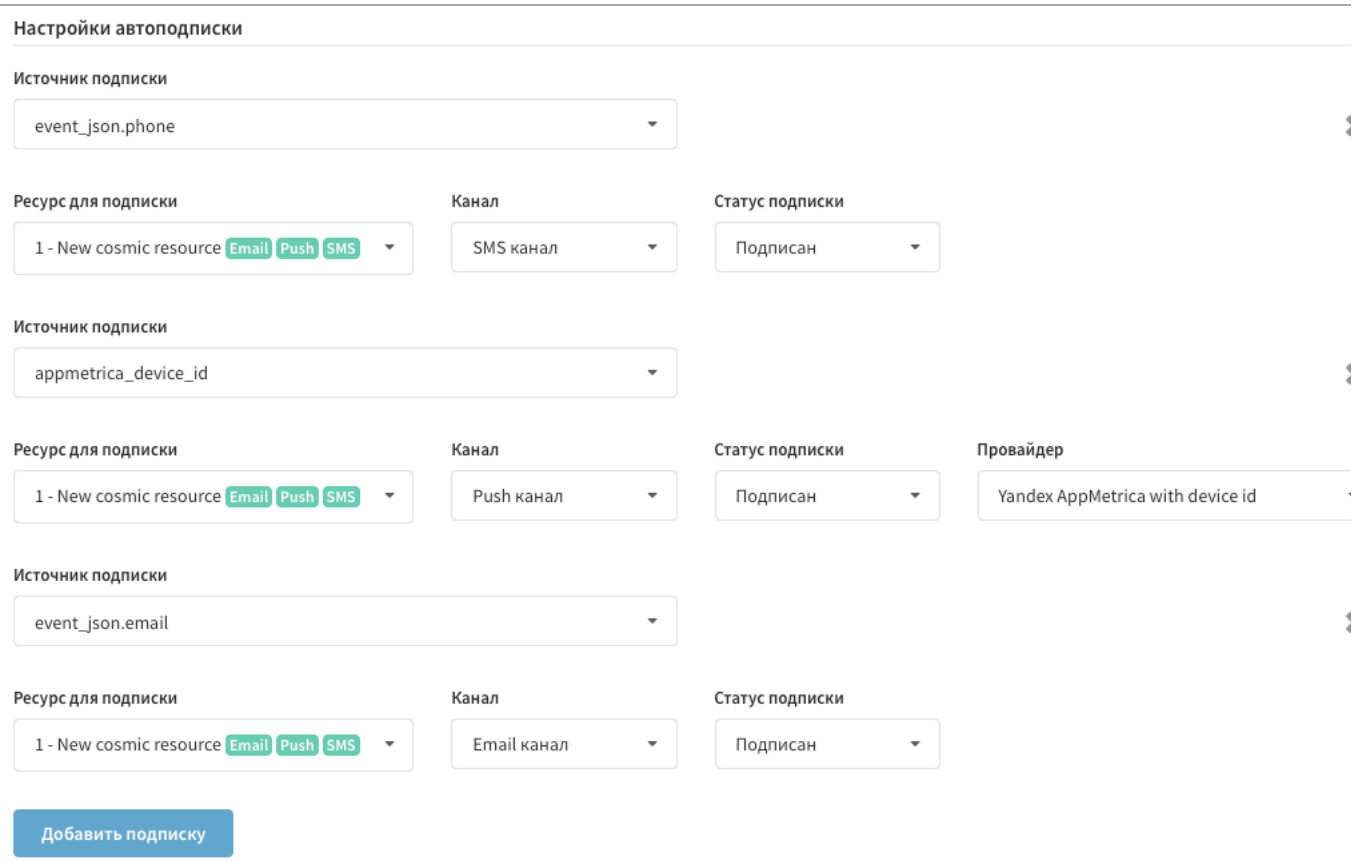
Auto-subscription

![img](/img/attachments/9404929/9404951.png)

For Push from AppMetrica:

Provider	
AppMetrica with device id	**appmetrica_device_id**
AppMetrica with iOS IFA	**ios_ifa**
AppMetrica with Google AID	**google_aid**

For other providers and Email/SMS — pass the data via `event_json`.



For each subscription, a [status] can be set: “Subscribed” (single opt-in) or “Not confirmed” (double opt-i

Data matching

![img](/img/attachments/9404929/9404953.png)

Step 4. Saving the integration

Click Save. The integration is ready to use.

Yandex.Audience™

The integration synchronizes static segments from Altcraft with segments in Yandex.Audience. Segme so that the service finds the right people and launches targeted advertising.

> TIP
>
> **A connection must be set up before the integration. See the instructions.**

Step 1. Creating the integration

![img](/img/attachments/9407286/sql-type.png)

Step 2. Configuring the integration

![img](/img/attachments/45514795/45514800.png)

![img](/img/attachments/45514795/45514802.png)

> TIP
>
> **A segment with fewer than 100 profiles cannot be built in Yandex.Audience.**

![img](/img/attachments/45514795/45514803.png)

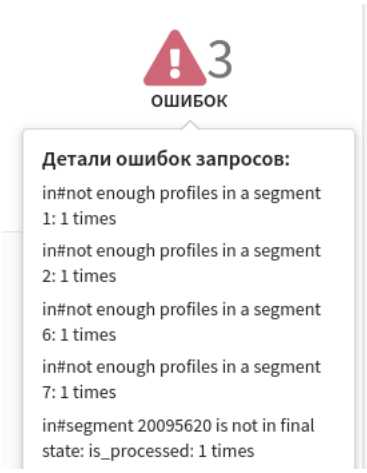
Step 3. Saving the integration

Click Save. The integration is ready for use.

Synchronization statistics

![img](/img/attachments/45514795/45514826.png)

To view synchronization errors, click the icon:



Current segment statuses in Yandex.Audience:

Информация об аудиториях

Сегмент	Аудитория	Количество	Статус	Последняя синхронизация
#91 seg 1 ya.Au	seg 1 ya.Au (20054702)	772	Готов	2021-07-21 08:59:21
#92 seg 2 ya.Au	seg 2 ya.Au (20054703)	806	Готов	2021-07-21 08:59:22
#93 seg 3 ya.Au	seg 3 ya.Au (20054704)	877	Готов	2021-07-21 08:59:23

Lpgenerator™

Description

Imports or updates customer profile.
At first create an input form for customer profiles data fields in the landing page editor. It can contain a

Request URL

Method: POST

`https://example.com/api/v1.1/integrations/lpgenerator/profile_import`

Request parameters

Lpgenerator default fields. You cannot redefine field values.

Parameter	Type	Example	Required	
id	**string**	**“777777”**		
meta_data	**JSON object**	**“meta_data” : {...}**	**Yes**	**Metadata like IP and browser inf**
utm_data	**JSON object**	**“utm_data” : {...}**	**Yes**	**Predefined utm data (can be emp**

Hidden fields

Parameter	Type	Example	Required	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
db_id	**string**	**“1”**	**Yes**	**Database ID**
matching	**string**	**“phone”**	**No**	**Profiles matching mode: email/phone/cu By default – email**
field_name	**string**	**“custom_ID”**	**No**	**Sets a custom field for profile matching**
resource_id	**string**	**“3”**	**No**	**Defines subscription resource ID**

Profile fields

Parameter	Type	Example	Required	
lead_data	**JSON object**	**“lead_data” : {...}**		**Altcraft datal**

`db_id` , `token` , `resource_id` , `matching` , `field_name` **must be predefined in LPGenerator web hook settings**
Use Altcraft API URL as LPGenerator web hook URL: `<your.domain>/api/integrations/lpgenerator/profile_import`
API tokens are created by Altcraft platform master users.

Request examples

Email matching profile import:

```
{
        "id": 77777777,
        "lead_data": {
                "email": "qwerty@dron.ff",
        },
        "meta_data": {
                "HTTP_USER_AGENT": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/53
                "LANGUAGE": "en-US",
                "form_name": "form 1",
                "from": "Page 107570 (Variant 1)",
                "page_id": 1642169,
                "user_ip": "96.123.123.123",
                "variant_id": 1935651
        },
        "utm_data": {}
    "db_id": "2",
    "resource_id": "2",
    "matching": "email",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

Profile fields update with a resource subscription (custom field profile matching by `custom_ID` field):

```
{
         "id": 27773744,
         "lead_data": {
                  "email": "ytrewq@dron.ff",
        "custom_ID": "Bagdasaryan"
         },
         "meta_data": {
                  "HTTP_USER_AGENT": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/53
                  "LANGUAGE": "en-US",
                  "form_name": "form 1",
                  "from": "Page 207571",
                  "page_id": 1642169,
                  "user_ip": "96.123.123.123",
                  "variant_id": 1935651
         },
         "utm_data": {}
    "db_id": "2",
    "resource_id": "2",
    "matching": "custom",
    "field_name": "custom_ID",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```


Response example


```
{
  "error": 0,
  "error_text": "Successful operation"
}
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Tilda™

Tilda profile import integration

In Altcraft Platform, you can transfer the data of a user who filled out a form on your website designed use the Webhooks tool. You can read more about Tilda Webhooks in [this article](#).

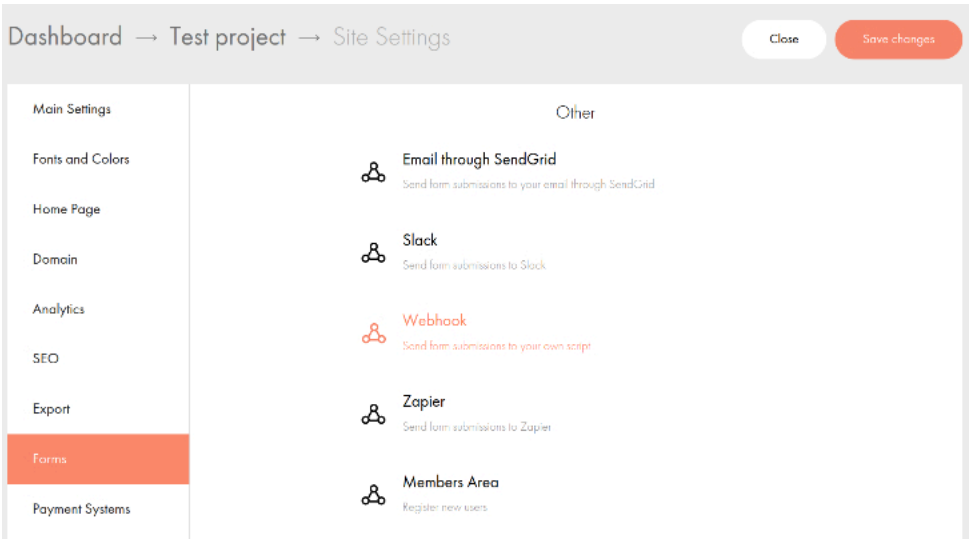
> NOTE
>
> **Please note that integration with Tilda allows you to integrate only user subscriptions to email and SI**

For this integration you will need the following data.

- **Altcraft service instance URL. For example, `https://ru.altcraft.com`.**
- **API token: tokens are created by [platform master users](#).**
- **Database ID and Resource ID. If there is no need to subscribe to a resource, you need only the dat**

Step 1: Create a Tilda webhook

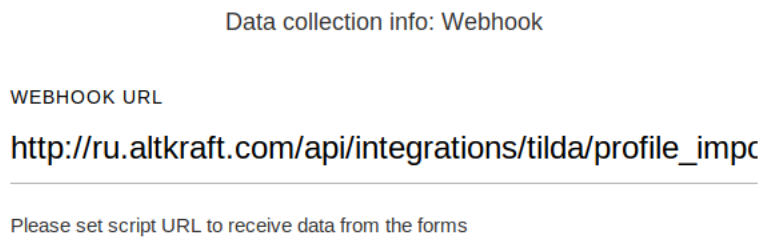
Go to the section: Dashboard —> Your project —> Site settings —> Forms. At the very bottom you will f



Fill in the WEBHOOK URL field as follows:

`<your.domain>/api/integrations/tilda/profile_import/?token=<api_token>`

Here you need to replace <your.domain> with the URL of the Altcraft platform service instance and <ap you:

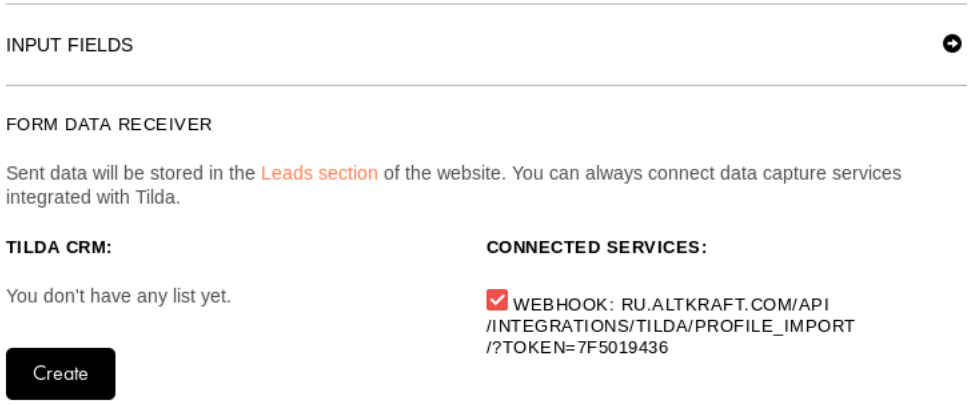


Click “Add”. After that, you will be prompted to assign this data collection service to all forms on the w the Close button. So, Webhook is created.

Step 2. Set up a Tilda form

Open the site page editing window and find the form from which you want to transfer data to Altcraft. T

Find the FORM DATA RECEIVER section and enable your Webhook:



There are two types of form in Tilda:

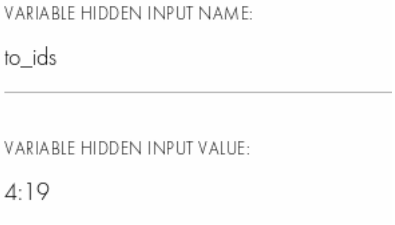
- **a simple form with a button,**
- **form with multiple fields.**

The settings for each of them are slightly different.

Forms with one input

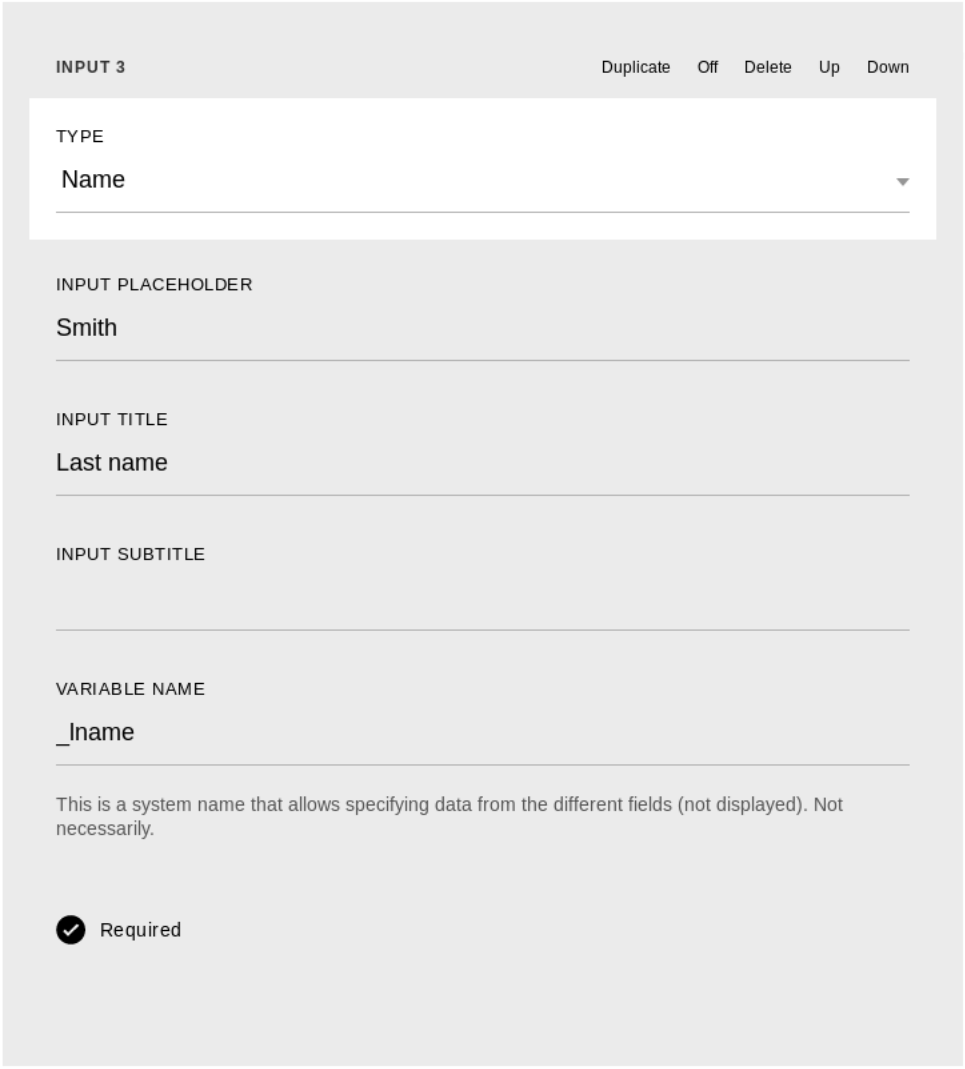
For a single input field form you will need block “Options” to set variable names.

Field	Description
Variable name	**If you want this field to get into Altcraft, then its name must correspond to the same field in the database. The variable name must be emai** **The name is not case sensitive, for example, if the Altcraft database has an “email” field, and Tilda’s field is called “EMAIL”, the data from t**
Variable name for a hidden field	**Name it** `to_ids`
Variable value for a hidden field	**Enter a colon-separated database ID and resource ID. If you do not need to subscribe to the resource, then you need only the database ID. I database ID,** `19` **is the resource ID.**

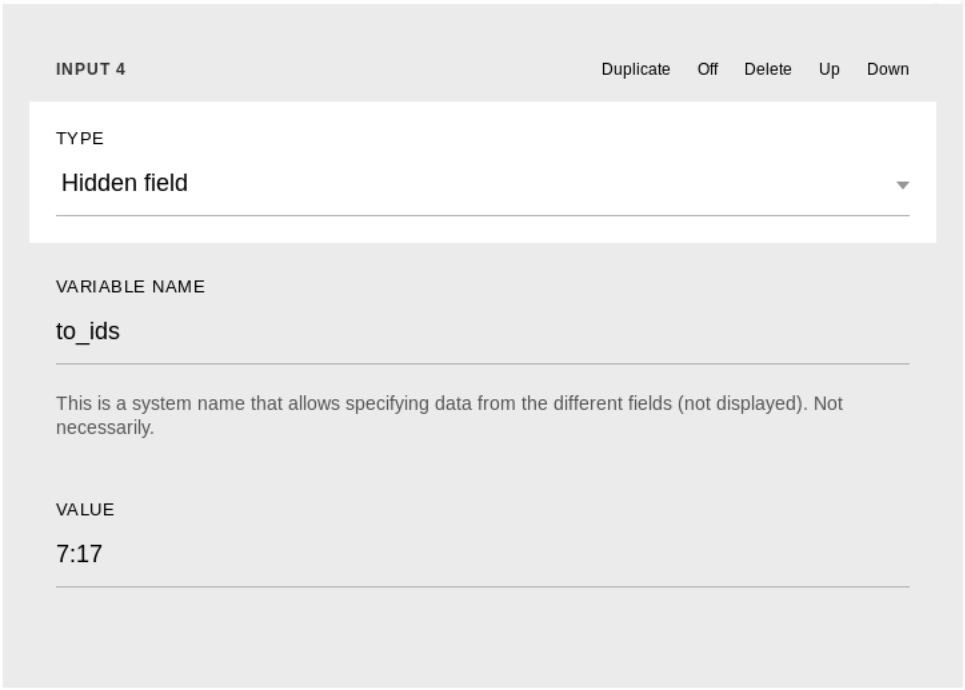


Form with several inputs

If you are working with a form that contains multiple fields, you need to open the “Content tab. There, Pay attention to the names of all the variables that the form contains. If you want a field to get into Alt the same field in the database.



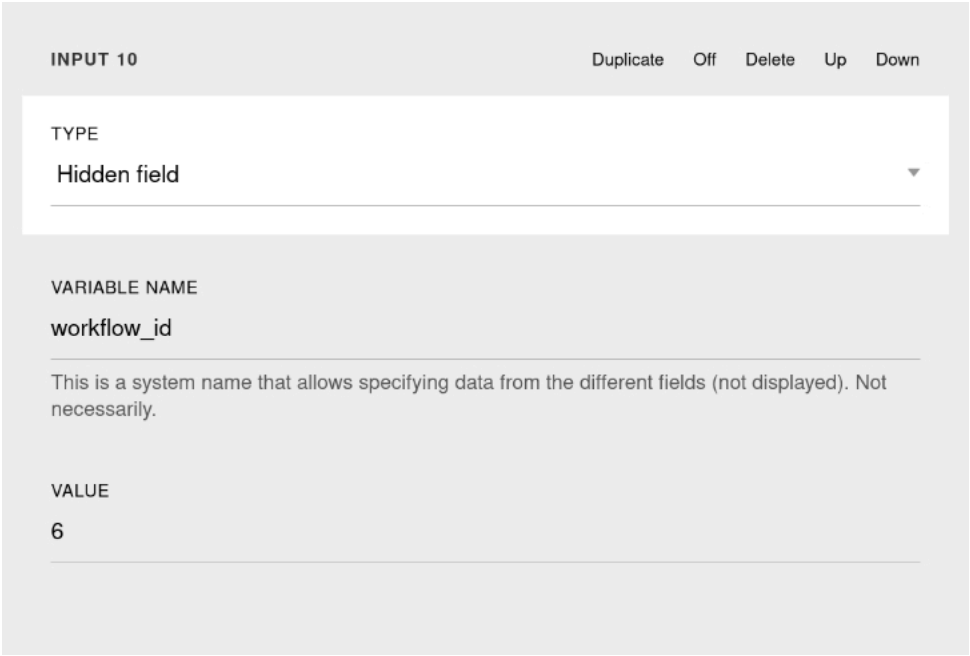
The contact data fields should be named email or phones.The variable name is case insensitive, e.g. El
After setting up the visible form fields, add another field. Type is hidden. Name it to_ids . Enter a colon-
resource ID. If you do not want to subscribe to the resource, enter only the database ID.
In the example in the screenshot below, 7 is the database identifier, 17 is the resource identifier.



Integration with Tilda allows not only importing new profiles into Altcraft databases, but also immedia
example, you can run a trigger mailing right after importing a new profile to send them a welcome ema
scenario.

To do this, you need to add additional hidden fields in the form:

Action after import	Variable name	
Launch a trigger mailing	**trigger_id**	**Trigger maili**
Add a profile to a scenario	**workflow_id**	**Scenario ID**



Passing a subscription status

When you import a profile from a multi-field form, you can pass a specific subscription status to the res displayed in the digital profile of the subscriber in Altcraft Platform.

To pass profile status, you need to add one more hidden field in your form settings. You must specify st choose one of the statuses reserved by Altcraft Platform as the value.

Available statuses:

Status	Status_s
Subscribed	**0**
Unsubscribed	**1**
Complaint	**2**
Hardbounced	**3**
Unconfirmed	**4**
Suspended	**5**
Invalid	**6**

Step 3. Test a form

After you set up the fields for the form, publish the page. Then test the page by submitting test data to

TIP

Tilda won't update the form right away, so it's best to test it a couple of minutes after posting.

Next, open Your project —> Leads.

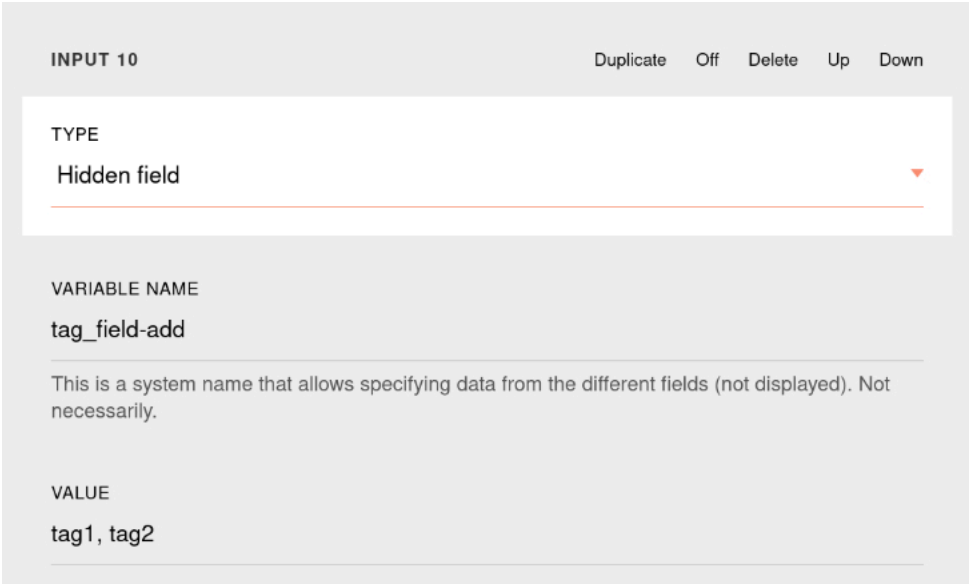
The screenshot shows an example of a successfully sent application. If you see an error, you need to c

- **The URL in the Webhook is correct and contains the correct API Key**
- **The database exists and it is selected in the resource you are using to subscribe.**
- **The form has an email or phone field**

Extra options

You can set different variable parameters as hidden form fields:

- **Profile matching mode by any of available. Default matching is Email.**
- **You can use `db_id` and `resource_id` instead of `to_ids` variable**
- **You can set a custom value for `_vendor` variable. By default it will send form ID.**
- **When updating a profile field with the “tags” type in the database, the field is completely overwritt ones, add the `-add` to the variable name when setting up a field in Tilda. For example, to update a t `tag_field-add`.**



Altcraft action hooks

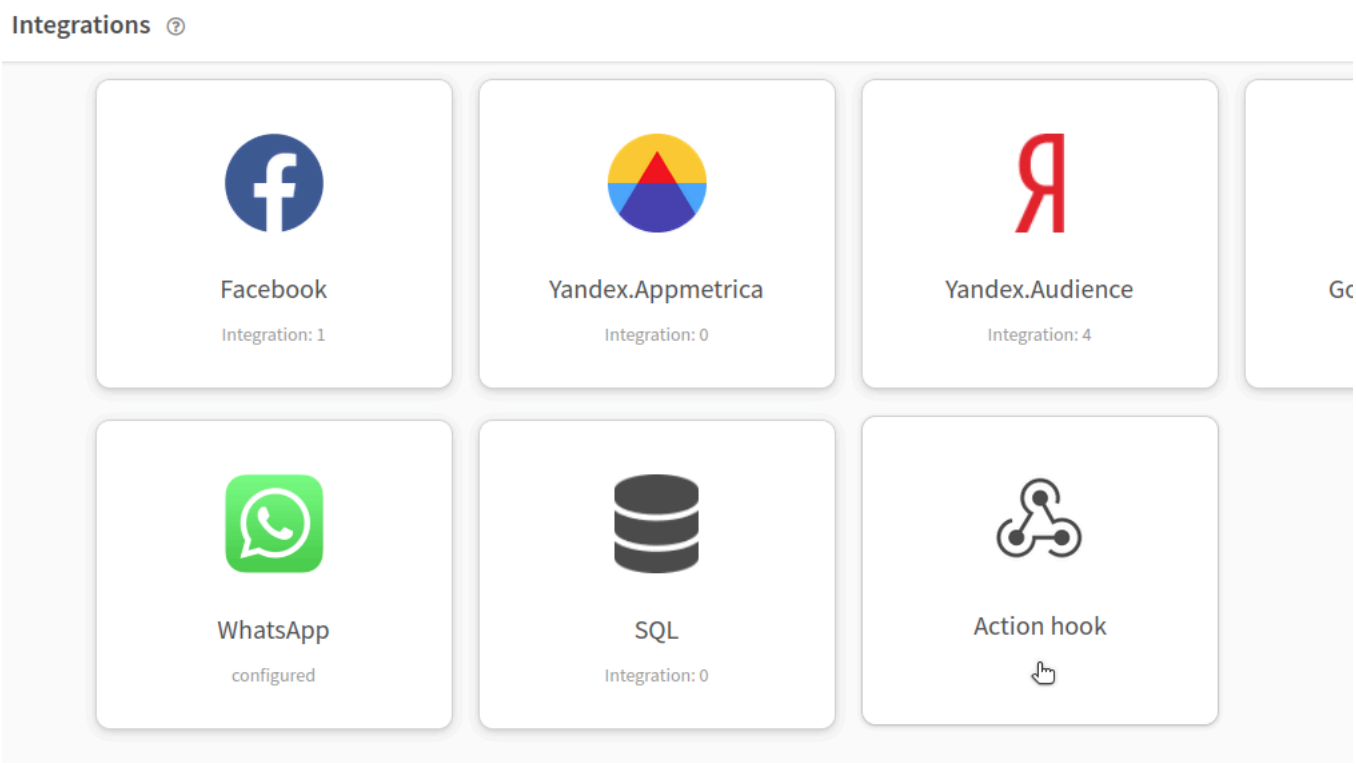
Overview

Action Hooks is a mechanism for notifying external systems of platform events. These can range from delivery or importing subscriber profiles, to customer actions tracked by Altcraft platform pixels on yo

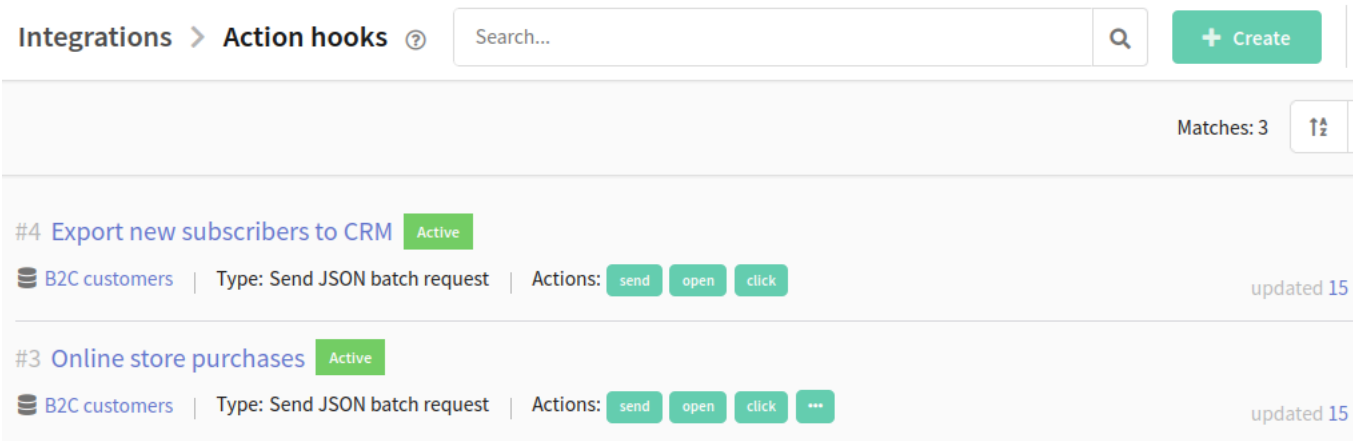
Action Hooks can work with all databases or any specific one. Captured events can be transmitted bat sent to the broker queue asynchronously.

Configurable by master user user for a specific database or for all databases at once.

From the main menu, select Integrations → Event Capture to get started:

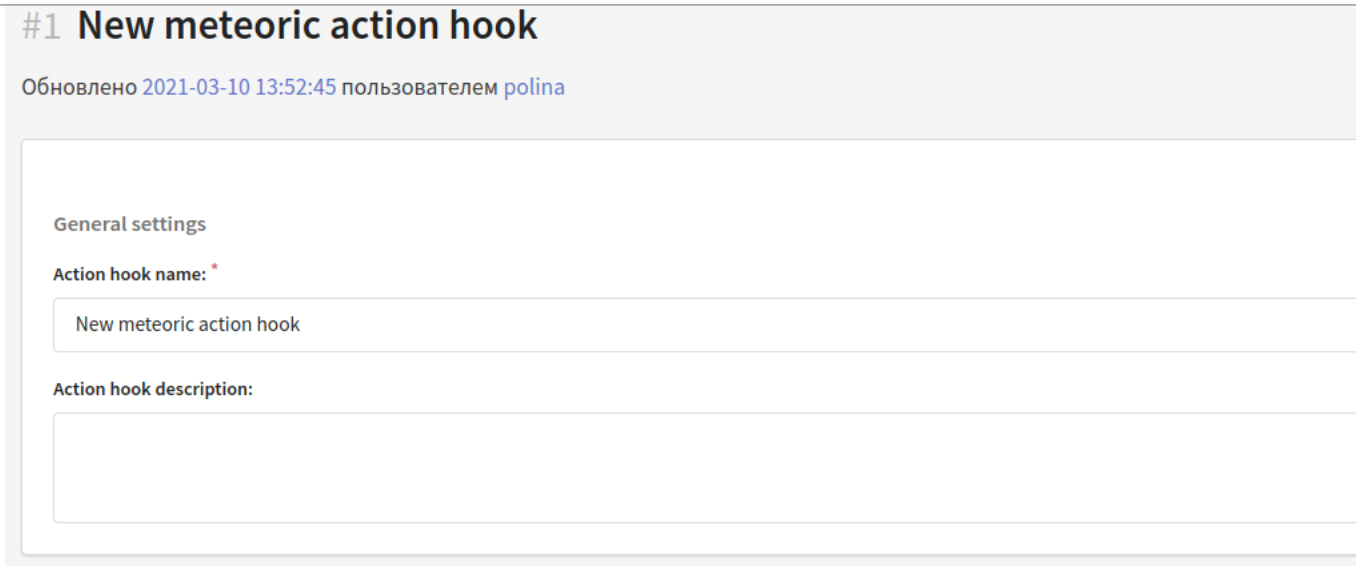


You can activate or stop existing action hooks, clone or delete them using the context menu. If you wa To add a new one, click + Create at the top of the page:



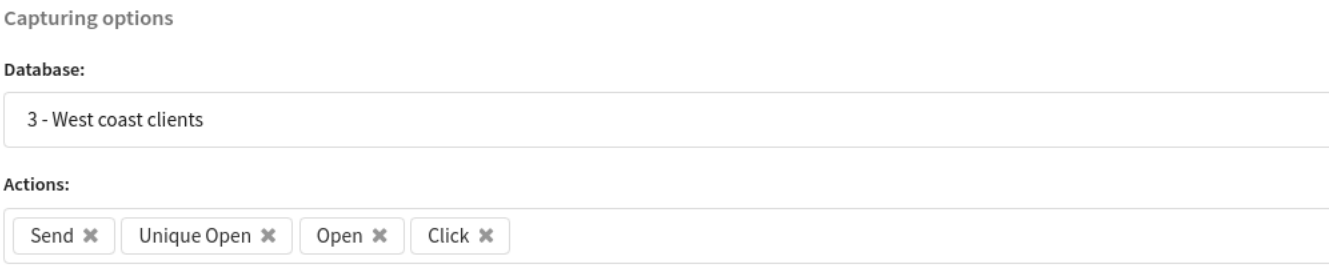
Action hook general settings

In General settings enter a name for the new hook. Here you will also find a switch to activate the ever



Capturing options

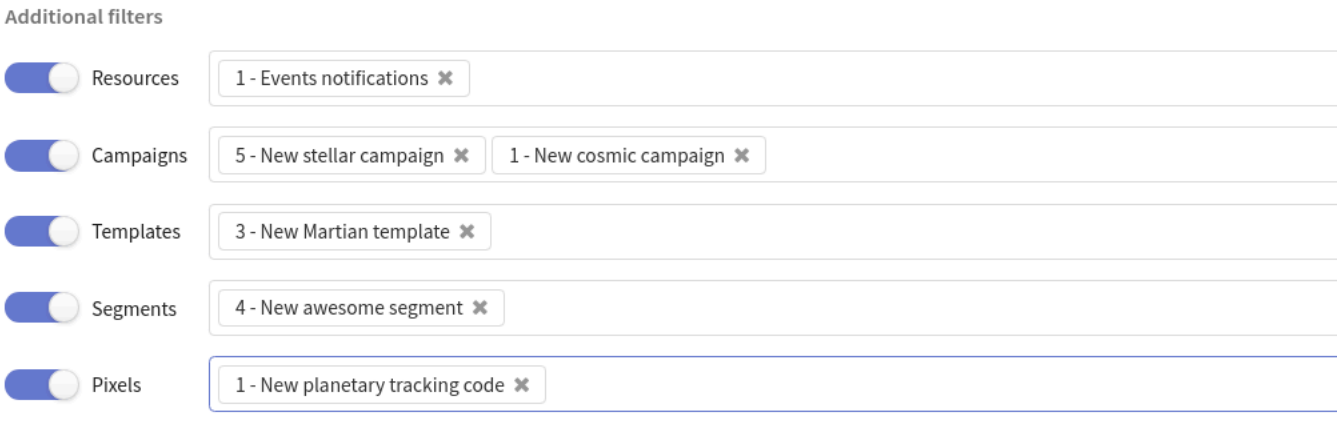
Action hook can work with all Profile Databases at once or with one specific one. In the Capture Settin from the Database drop-down menu:



In the Actions field, select from the drop-down menu the events to be sent to external systems. Event t menu and in the hook request, and descriptions can be found in this article.

By default, Action hook logs all events of the types you select. You can limit the events to get a narrow Filters: block.

- **Resources - events related to communications within certain [resources](/en/user-guide/profiles-an captured.**
- **Mailing - events from specific marketing [mailings](/en/user-guide/campaigns/) will be captured.**
- **Templates - events from mailings using specific [templates](/en/user-guide/message-templates/mar**
- **Segments - events related to subscribers entering certain [segments](/en/user-guide/segmentation/**
- **Pixels - actions of subscribers on external resources, registered by certain pixels, will be captured**

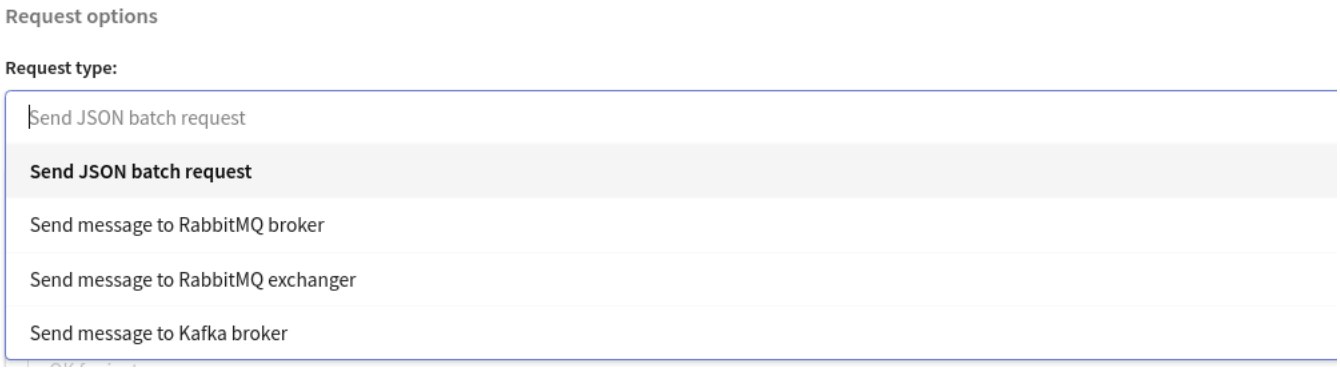


Request options

The following request types are available:

1. **Send a JSON batche request. Read more.**
2. **Send a message to the RabbitMQ queue. Read More.**
3. **Send a message to the RabbitMQ exchange. Read More.**

4. Send a message to Kafka broker. [Read more](#).



INFO

Retry

If for some reason an error occurs during sending a message (e.g. network failure), the message will
period of time. The number of repeated sends and their frequency are set in the system configuration

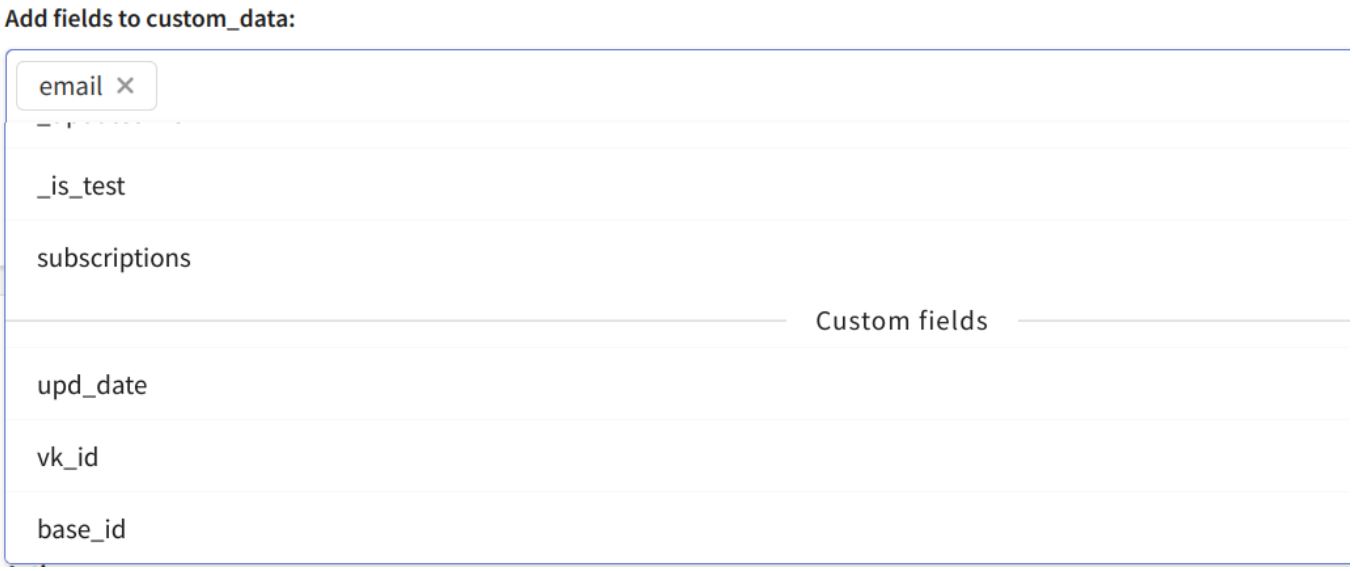
`"HOOK_RETRY_TIMES"` **: set maximum number of attempts (default 4)**

`"HOOK_RETRY_PERIOD_SEC"` **: periods between attempts, in seconds (default 15)**

Transmit additional event data in the capture

Client profile data in capture

Along with information about the captured event, you can pass values of standard and additional profil
additional fields depends on the selected database. Select from the drop-down list the profile fields yo
custom_data:



The template custom_data can be set below. The platform will use the value from this template in the
`custom_data` **field in the [API request](#) to start the trigger.**

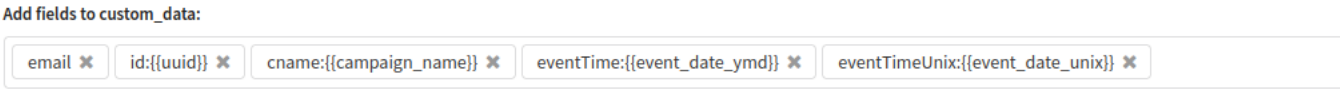
Additional data in the hook

In addition to profile data, you can transmit any data in `mykey:myvalue` format to the `custom_data` object. T and value and then press Enter ↵.

Template options available:

Parameter	Description
`{{uuid}}`	**Generates new UUIDv4**
`{{event_date}}`	**Event date in the format "2006-01-25T15:04:05.999999Z"**
`{{event_date_ymd}}`	**Event date in the format "2006-01-25"**
`{{event_date_rfc3339}}`	**Event date in the format RFC 3339**
`{{event_date_unix}}`	**Event date in unix time, in string**
`{{event_date_unixnano}}`	**Event date in unix time (nanoseconds), in string**
`{{campaign_name}}`	**Retrieves the mailing name if it is relevant to the event, otherwise empty**
`{{campaign_subid}}`	**Retrieves the mailing SubID if it is relevant to the event, otherwise empty**
`{{database_name}}`	**Retrieves the database name if it is relevant to the event, otherwise empty**
`{{segment_name}}`	**Retrieves the segment name if it is relevant to the event, otherwise empty**
`{{message_name}}`	**Retrieves the template name if it is relevant to the event, otherwise empty**
`{{message_subject}}`	**Retrieves the template saubject if it is relevant to the event, otherwise empty**
`{{resource_name}}`	**Retrieves the resource name if it is relevant to the event, otherwise empty**

For example, you add fields to `custom_data`:



The data will be transferred in this way:

```
"custom_data": {
        "cname": "Black Friday",
        "email": "test@example.com",
        "eventTime": "2021-03-17",
        "eventTimeUnix": "1615988820",
        "id": "dc876134-87a4-4437-aabb-d2fc8a3d4213",
    }
```

The `custom_data` field can contain any arbitrary data. Fields passed inside it do not have to exist in the p within the event itself. In the event, they appear in the following format:

```
"custom_data": {
    "test_field": "test"
}
```

It is important to note that fields in `custom_data` are not directly tied to the structure of the profile datab capture, it may seem that these fields must exist in the database because they are included in the API The "Add fields to custom_data" function is used to append additional data—for example, to include val - even if those fields were not originally present in the request.

Message content in hook

In order to transmit the content of all messages along with events, activate the option Add messages

The content will be passed to the `"content"` field. The field structure for each channel is described in th

> **CAUTION**
>
> **Please note that this will store all sent messages in storage for a fixed amount of time. This will requ default retention time is 72 hours; it is set by the global setting `HOOK_CONTENT_CLEANUP_HOURS` in the main.js be transmitted without any content associated with them.**

Exporting Historical Events for a Period

This function allows you to export to an integration the data about events that have already occurred c recovery if event capturing was not configured in time or if a failure occurred.

> **NOTE**
>
> **The following event types are not available for historical export: `campaign_launch`, `profile_merge`, `profile_`**

Export Setup

To start exporting historical events, enable and save the event capture with the desired settings. Then option Capture historical events for a period, specify the start and end dates, and click Start:

> **INFO: Note**
>
> **The period for exporting historical events is limited: the minimum date is 01-01- 1970, and the maximu date cannot be selected.**

To verify the export operation, use the testing tool — the system will generate an example event of the

Managing the Process

After starting the export, you can monitor progress and manage execution:

- **Suspend — temporarily stop the export**
- **Resume — continue the export**
- **Abort — completely cancel the export**

Execution Statuses

- **In Progress — `The export of historical events for the selected period is in progress`**
- **Paused — `The export process of historical events has been paused`**
- **Completed — `The export of historical events for the period has been completed`**
- **Error — `An error occurred during the export of historical events`**

> **NOTE**
>
> **If you deactivate event capture during the export of historical events, the process will be paused unti**

Action hooks event types

Interface name	API name	Description
Email channel events		
Send	**send**	**Email is sent to a recipient.**
Reply	**reply**	**Subscriber's reply to received email.**
Unique Open	**uopen**	**Unique email message opening.**
Open	**open**	**Any email opening.**
Unique Click	**uclick**	**Unique email link click. It creates open and uopen events when *open* event is not captured.**
Click	**click**	**Any email link click.**
Confirm	**confirm**	**Confirmation link click in a regular message (creates click and uclick, if it is the first click for this message).**
Confirm Subscription	**confirmsub**	**Confirmation link click in a double opt-in message.**
Unsubscribe	**unsub**	**Recipient opted out by clicking a message link `{unsubscribe}` or `{globalunsubscribe}`.**
Complain	**complain**	**A recipient complained or reported spam.**
SoftBounce	**sbounce**	**Temporary delivery problems provider message. Soft bounce.**
HardBounce	**hbounce**	**Invalid address or similar delivery problems message. Hard bounce.**
Suppression	**suppress**	**Email was not sent: recipient address is in suppression list.**
Delivered	**deliv**	**Email delivered to a recipient.**
Undelivered	**undeliv**	**Email could not be delivered.**
Read Message	**read**	**Message remained open for 8 seconds or longer.**
Glanced Message	**glanced**	**Message remained open for less than 4 seconds.**
Skimmed Message	**skimmed**	**Message remained open for 4 to 8 seconds.**
Subscriber Profile Import Events		
Manual Import	**import_manual**	**Manual profile import via platform interface.**
File Import	**import_file**	**From file profile import via platform interface.**
API Import	**import_api**	**API profile import.**
SMS channel events		
Send SMS	**send_sms**	**SMS sent to a recipient.**
Click SMS	**click_sms**	**Any SMS link click.**
Unique Click SMS	**uclick_sms**	**Unique SMS link click.**
Delivered SMS	**deliv_sms**	**SMS delivered to a recipient.**
Undelivered SMS	**undeliv_sms**	**SMS could not be delivered.**
Unsubscribe SMS	**unsub_sms**	**Recipient opted out by clicking a message link `{unsubscribe}` or `{globalunsubscribe}`.**
Push channel events		
Unique Open Push	**uopen_push**	**Unique push open.**
Open Push	**open_push**	**Any push open.**
Send Push	**send_push**	**Push sent to a recipient.**
Read Push	**read_push**	**Push read in the notification center.**
Click Push	**click_push**	**Any push link click.**
Unique Click Push	**uclick_push**	**Unique push link click.**
Delivered Push	**deliv_push**	**Push delivered to a recipient.**
Undelivered Push	**undeliv_push**	**Push could not be delivered to a recipient.**
SoftBounce Push	**sbounce_push**	**Temporary delivery failure (network issues, push service unavailability for any reason). This event means tha message.**
Hardbounce Push	**hbounce_push**	**Permanent delivery failure (Invalid subscription id). After receiving such a response, the platform will not att**
Custom (API) opt out		
API unsubscribe	**unsub_api**	**API profile opt out.**
Mailing launch		
Mailing launch	**campaign_launch**	**Mailing launch. It has a custom hook message structure. Not available for [historical events export.](/en/user**
Mailing errors		
General: Error	**camp_error**	**Errors that occurred during the campaign.**
The events that are captured by Pixels in the external resources		
Pixel Open	**pixel_open**	**Pixel script activation. You can find out more about the parameters of this request [there](/en/developer-guid**
Promo code events		
Promocode Attach	**promocode_attach**	**The subscriber is assigned a promotional code.**
Promocode Detach	**promocode_detach**	**The promo code is unlinked from the subscriber's profile.**
Promocode Activate	**promocode_activate**	**The promo code is activated.**
Events related to the automatic creation of a static segment during import		
Static Segment Added	**segs_add**	**The profile entered the static segment.**
Static Segment Removed	**segs_remove**	**The profile exited the static segment.**
Events of merging multiple profiles or splitting one profile		
Profile merge	**profile_merge**	**Multiple profiles are merged into a new or existing profile through an API request. Not available for [historica hooks/37849833#export)**
Profile split	**profile_split**	**A profile is divided into several new or existing profiles through an API request. Not available for [historical e hooks/37849833#export)**
MAX Bot channel events		
MAX bot delivery	**max_bot_deliv**	**The message was delivered to the chat with the MAX bot.**

MAX bot undelivery	**max_bot_undeliv**	**The message was not delivered to the chat with the MAX bot.**
MAX bot send	**max_bot_send**	**The message was sent for delivery to the chat with the MAX bot.**
MAX bot click	**max_bot_click**	**Non-unique click on a link in a message in the chat with the MAX bot.**
MAX bot unique click	**max_bot_uclick**	**Unique click on a link in a message in the chat with the MAX bot.**
MAX bot subscription	**max_bot_subscribe**	**The profile subscribed to the MAX bot.**
MAX bot unsubscribe	**max_bot_unsub**	**The profile unsubscribed from the MAX Bot channel.**
MAX bot complaint	**max_bot_complain**	**The subscriber complained about the message.**
MAX bot sending policy violation	**max_bot_offence**	**The limit on the number of messages was exceeded.**
MAX Group channel events		
MAX group delivery	**max_group_deliv**	**The message was delivered to the MAX channel.**
MAX group undelivery	**max_group_undeliv**	**The message was not delivered to the MAX channel.**
MAX group send	**max_group_send**	**The message was sent for delivery to the MAX channel.**
MAX group click	**max_group_click**	**Non-unique click on a link in a message sent to the MAX channel.**
MAX group unique click	**max_group_uclick**	**Unique click on a link in a message sent to the MAX channel.**
MAX group subscription	**max_group_subscribe**	**The profile subscribed to the MAX Group channel.**
MAX group unsubscribe	**max_group_unsub**	**The profile unsubscribed from the MAX Group channel.**
MAX group complaint	**max_group_complain**	**The subscriber complained about the message.**
MAX group sending policy violation	**max_group_offence**	**The limit on the number of messages was exceeded.**
Telegram Bot events		
Telegram bot send	**telegram_bot_send**	**Message has been sent for delivery to a subscriber in a chat with a Telegram bot.**
Telegram bot delivered	**telegram_bot_deliv**	**The message has been delivered to a chat with a Telegram bot.**
Telegram bot click	**telegram_bot_click**	**A non-unique click on a link in a message in a chat with a Telegram bot.**
Telegram bot unique click	**telegram_bot_uclick**	**A unique click on a link in a message.**
Telegram bot undelivered	**telegram_bot_undeliv**	**The message has not been delivered to a chat with a Telegram bot.**
Telegram bot subscription	**telegram_bot_subscribe**	**A profile has subscribed to a bot by initiating a dialogue.**
Telegram Group events		
Telegram group send	**telegram_group_send**	**Message has been sent for delivery to a subscriber in a chat with a Telegram bot.**
Telegram group delivered	**telegram_group_deliv**	**Message has been delivered in a Telegram group.**
Telegram group click	**telegram_group_click**	**Click on the link in the message in a Telegram group.**
Telegram group undelivered	**telegram_group_undeliv**	**Message has not been delivered to a Telegram group.**
WhatsApp* events		
WhatsApp* send	**whatsapp_send**	**Message has been sent for delivery to a subscriber in a WhatsApp* chat.**
WhatsApp* deliv	**whatsapp_deliv**	**Message has been delivered in a WhatsApp* chat.**
WhatsApp* click	**whatsapp_click**	**A non-unique click on a link in a message in a chat.**
WhatsApp* unique click	**whatsapp_uclick**	**A unique click on a link in a message.**
WhatsApp* undeliv	**whatsapp_undeliv**	**Message has not been delivered in a WhatsApp* chat.**
WhatsApp* read	**whatsapp_read**	**Message has been read by a user.**
WhatsApp* subscription	**whatsapp_subscribe**	**Profile has been subscribed to a WhatsApp* channel.**
WhatsApp* unsubscribe	**whatsapp_unsub**	**Profile has been unsubscribed from a WhatsApp* channel.**
Viber events		
Viber send	**viber_send**	**Message has been sent for delivery to a subscriber in a Viber chat.**
Viber deliv	**viber_deliv**	**Message has been delivered in a Viber chat.**
Viber click	**viber_click**	**Click on a link in the message. If the recipient clicks on the same link multiple times, each click event is rec**
Viber unique click	**viber_uclick**	**Unique click on a link in the message. If the recipient clicks on any link at least once, one unique click event**
Viber undeliv	**viber_undeliv**	**Message was rejected by Viber or Devino.Online.**
Viber read	**viber_read**	**Message was read by the recipient.**
Viber subscription	**viber_subscribe**	**Profile subscribed to the Viber channel, i.e., agreed to receive messages from the business account.**
Viber unsubscribe	**viber_unsub**	**Profile unsubscribed from the Viber channel, i.e., blocked receiving messages from the business account.**
Notify channel events		
Notify delivered	**notify_deliv**	**Message has been delivered.**
Notify undelivered	**notify_undeliv**	**Message has not been delivered.**
Notify send	**notify_send**	**Message has been sent.**
Notify click	**notify_click**	**Click on a link in the message. If the recipient clicks on the same link multiple times, each click event is rec**
Notify unique click	**notify_uclick**	**Unique click on a link in the message. If the recipient clicks on any link at least once, one unique click event**
Notify subscription	**notify_subscribe**	**Profile has been subscribed to a Notify channel.**
Notify read	**notify_read**	**Message has been read by a user.**
Notify unsubscribe	**notify_unsub**	**Profile has been unsubscribed from a Notify channel.**
Notify complain	**notify_complain**	**A recipient complained or reported spam.**
Pop-up events		
Popup: Post	**popup_post**	**The popup form has been completed and submitted.**
Popup: Click	**popup_click**	**Click on any popup element except the "Close" button.**
Popup: Error	**popup_error**	**Any error in the popup.**
Popup: Bounce	**popup_bounce**	**While submitting the popup form, technical problems occurred (for example, a server failure).**
Popup: Close	**popup_close**	**Click on the "Close" button in the popup.**
Popup: Open	**popup_open**	**The popup is shown to the user.**

Popup: Custom	**popup_custom**	**An event configured on the client side was registered.**
Loyalty program events		
Loyalty: Registration	**loyalty_register**	**Profile registration in the loyalty program.**
Loyalty: Unregistration	**loyalty_unregister**	**Profile exclusion from the loyalty program.**
Loyalty: Level Up	**level_up**	**Loyalty level upgrade.**
Loyalty: Level Down	**level_down**	**Loyalty level downgrade.**
Loyalty: Points Accrual	**points_accrual**	**Loyalty points accrual.**
Loyalty: Points Activation	**points_activate**	**Loyalty points activation.**
Loyalty: Points Expiration	**points_expire**	**Loyalty points expiration.**
Loyalty: Points Redeem	**points_redeem**	**Loyalty points redemption.**
Loyalty: Cancel Accrual	**points_cancel_accrual**	**Loyalty points accrual cancellation.**
Loyalty: Cancel Redemption	**points_cancel_redeem**	**Loyalty points redemption cancellation.**
Loyalty: Points Reset	**points_q_reset**	**Loyalty points reset.**
Campaign events		
Campaign: joined audience	**sc_add**	**Profile added to the campaign audience**
Campaign: failed to join	**sc_gcg_suppress**	**Profile not added to the campaign audience because it is in the global control group**
Campaign: joined test group	**sc_test_group_assign**	**Profile assigned to the test group**
Campaign: joined control group	**sc_control_group_assign**	**Profile assigned to the control group**
Campaign: left the audience	**sc_remove**	**Profile removed from the campaign audience**
Added to Campaign Audience	**sc_add**	**The profile has been added to the campaign audience.**
Assigned to Test Group	**sc_test_group_assign**	**The profile has been assigned to a test group.**
Assigned to Control Group	**sc_control_group_assign**	**The profile has been assigned to a control group.**
Removed from Campaign Audience	**sc_remove**	**The profile has been removed from the campaign audience.**
Suppressed by GCG from Campaign	**sc_gcg_suppress**	**The profile was not added to the campaign audience due to being in the GCG (Global Control Group).**
GCG events		
GCG: Add	**gcg_add**	**Registered GCG entry event.**
GCG: Remove	**gcg_remove**	**Registered GCG exit event.**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Action Hook Message Structure

General structure

General hook message structure for most event types:

Parameter	Type	Required	Example	Description
event_type	**string**	**Yes**	**“click”**	**Event name as in the event types table**
event_date	**string**	**Yes**	**“2018-03-04T13:13:19.034315796Z”**	**Event timestamp in RFC 3339 format**
profile_id	**string**	**No**	**“abcdefgh12345678”**	**Recipient profile ID**
_xxh	**string**	**No**	**“ad1850e0d1d9481e”**	**Subscriber email hash used for identification in Altcraft**
_md5	**string**	**No**	**“145bb30f7ddf1184cb560bc4b7705646”**	**MD5 hash of the subscriber’s email**
list_id	**int**	**No**	**111**	**Database ID where the event occurred**
account_id	**int**	**Yes**	**23**	**Account ID where the event occurred**
sends_trying_num	**int**	**No**	**2**	**Number of message sending attempts (present only if the number of attempts is gr**
is_test	**bool**	**Yes**	**false**	**Whether the event was triggered by a test send**
is_static	**bool**	**No**	**false**	**Whether dynamic substitutions were used in the sent content**
subscription_hash_id	**string**	**No**	**“0a4b910312345”**	**Subscription hash**
custom_data	**JSON object**	**No**	`{“alice@example.org”}`	**Subscriber data set configured in the Action Hook editing interface**
event_data	**JSON object**	**Yes**	`{“browser”: “Firefox”}`	**Data set that depends on the event type (there are two types: tracking events and**
subscription	**array**	**No**	`[ { “resource_id”: 123, “channel”: “email”, “email”: “user@example.com”, “phone”: “+79991234567”, “subscription_id”: “abc123”, “provider”: “androidfirebase”, “priority”: 1, “status”: “subscribed”, “ccsub”: { “custom_key”: “custom_value” } } ]`	**Array with subscription details including information about the resource, communi structure is described below.**

INFO: Note

When exporting historical events, message content is not transferred. Events are sent in the same for passed for deleted profiles.

event_data structure

Campaign events

For events of the following types:

- **sc_add — profile added to the campaign audience**
- **sc_gcg_suppress — profile not added to the campaign audience because it is in the global control g**
- **sc_test_group_assign — profile assigned to the test group**
- **sc_control_group_assign — profile assigned to the control group**
- **sc_remove — profile removed from the campaign audience**

Parameter	Type	Required	Example	Description
scampaign_id	**int**	**No**	**1**	**Campaign ID in Altcraft**
segment_id	**int**	**No**	**5**	**ID of the segment related to the campaign audience**
prev_st	**int**	**No**	**1**	**Group ID: 1 — test group, 2 — global control group**

Tracking events

For the following event types:

- **Email: uopen, open, uclick, click, confirm, confirmsub, unsub, read, glanced, skimmed**
- **SMS: uclick_sms, click_sms, unsub_sms**
- **Push: uopen_push, open_push, click_push, uclick_push**

Parameter	Type	Required	Example	For which channe
message_id	**int**	**Yes**	**17**	**Email, SMS, Push**
campaign_id	**int**	**Yes**	**973**	**Email, SMS, Push**
scampaign_id	**int**	**No**	**973**	**For all**
segment_id	**int**	**No**	**0**	**Email, SMS, Push**
resource_id	**int**	**Yes**	**3**	**Email, SMS, Push**
ip	**string**	**No**	**“10.9.0.14”**	**Email, Push**
country	**string**	**No**	**“.us” “.ru”**	**Email, Push**
city	**string**	**No**	**“New York” “Moscow”**	**Email, Push**
postal_code	**string**	**No**	**“101000”**	**Email, Push**

Parameter	Type	Required	Example	For which chann
region	string	No	“Ohio” “Moskovskaya oblast”	Email, Push
tz	string	No	“Europe/Vaduz”	Email, Push
lat	float	No	55.244075	Email, Push
lon	float	No	55.244075	Email, Push
user_agent	string	No	“Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0”	Email, Push
os	string	No	“Linux x86_64”	Email, Push
browser	string	No	“Firefox”	Email, Push
device	string	No	“web”	Email, Push
wick	int	No	1	Email, Push
link_url	string	No	“http://www.example.com”	Email, SMS, Push
launch_id	string	Yes	“00000017000003cd5a9bf01a”	Email, SMS, Push
send_message_id	string	Yes	“5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78”	Email, SMS, Push
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODIKR8-gA”	Email, SMS, Push
email_message_id	string	No	“mzi5c1~+9vqekqmy63mh8xhgt2qjwfkad6lmiafbyn-0te1_uzckfqyi4ry”	Email
extra_uid	string	No	“ew4Sgf41Qwjq8_2_2E_w_k__t.2SDvJv8bvBYQFtSDx”	Push
provider	string	No	“YandexAppMetricaDeviceIDAndroid”	Push
prev_st	int	No	1	For all

Messaging events

For the following event types:

- **Email: send, deliv, suppress, complain**
- **SMS: send_sms, deliv_sms**
- **Push: send_push, deliv_push**

Parameter	Type	Required	Example	For which channel	
message_id	int	Yes	17	Email, SMS, Push	Message ID
campaign_id	int	Yes	973	Email, SMS, Push	Mailing ID in
segment_id	int	No	0	Email, SMS, Push	Segment ID,
resource_id	int	Yes	3	Email, SMS, Push	Resource ID
launch_id	string	Yes	“00000017000003cd5a9bf01a”	Email, SMS, Push	Launch ID
send_message_id	string	Yes	“5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78”	Email, SMS, Push	Sent messa
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODIKR8-gA”	Email, SMS, Push	Full SendMe
email_message_id	string	No	“mzi5c1~+9vqekqmy63mh8xhgt2qjwfkad6lmiafbyn-0te1_uzckfqyi4ry”	Email	Email-safe S
extra_uid	string	No	“ew4Sgf41Qwjq8_2_2E_w_k__t.2SDvJv8bvBYQFtSDx”	Push	Extra UID
provider	string	No	“YandexAppMetricaDeviceIDAndroid”	Push	Provider
workflow_id	int	No	1	For all	Scenario ID,

For events sbounce, hbounce, sbounce_push, hbounce_push, the following parameters are also passed:

Parameter	Type	Required	Example	For which channel
bounce_code	int	Yes	500	Email, Push
bounce_message	string	Yes	“No such email”	Email, Push

For events undeliv, undeliv_sms, undeliv_push, the following parameters are passed:

Parameter	Type	Required	Example	For which chann
undeliv_code	int	Yes	400	Email, SMS, Push
undeliv_message	string	Yes	“Invalid Caller ID”	Email, SMS, Push

Pixel events

For pixel_open events:

Parameter	Type	Required	Example
ip	string	Yes	“10.9.0.14”
country	string	No	".ru "
city	string	No	“Moscow”
pixel_id	int	Yes	41
postal_code	string	No	“390000”
region	string	No	“Ryazanskaya oblast”
tz	string	No	“Europe/Vaduz”
lat	float	No	55.244075
lon	float	No	55.244075
user_agent	string	Yes	“Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0”
os	string	No	“Linux x86_64”
browser	string	No	“Firefox”
device	string	No	“web”
wick	int	No	1

The full list of possible parameters is available in this article.

Popup events

For all popup_post, popup_click, popup_error, popup_bounce, popup_close, popup_open events:

Parameter	Type	Required	Example
popup_id	int	Yes	123
referer	string	Yes	“api.example.com”
smid	string	Yes	“fGmXiWU9Y4NSIpCwl4QQQQEQEAABCRA”
utm_campaign	string	No	“utm_campaign”
utm_content	string	No	“utm_content”
utm_source	string	No	“utm_source”

Some events have their own unique parameters:

Parameter	**Type**	**Required**	**Example**	**Description**
popup_post				
post_uid	string	**Yes**	"7bc142b5-7f38-4272-bc7e-7539f70b25b1"	Unique form submission id
popup_click				
name	string	**Yes**	"main_block"	Name of the element that
url	string	**Yes**	"example.com/page"	Value of the `href` attribut
popup_error, popup_bounce				
error	string	**Yes**	"object_not_found"	Error text
error_category	int	**Yes**	404	Error number

Promocode events

For promocode_attach, promocode_detach, promocode_activate events:

Parameter	Type	Required	Example	De
attached	string	Yes	“2020-10-25T13:28:52Z”	Date and time the promocode was attached to the pr
detached	string	Yes	“2020-10-25T13:28:52Z”	Date and time the promocode was detached
activated	string	Yes	“2020-10-25T13:28:52Z”	Promocode activation date
code	string	Yes	“TEST-XOUH-KBJM-J5K9-24I1”	Promocode
db_id	int	Yes	123456	Profile database ID
loyalty_id	int	Yes	9	Loyalty program ID
profile_id	string	Yes	“000000000000000000000000”	Profile ID

All dates are specified in RFC 3339 format.

Segment creation events during import

For segs_add, segs_remove events:

Parameter	Type	Required	Example	
campaign_id	int	Yes	9	Mailing ID in
launch_id	string	Yes	“”	Mailing launc
message_id	int	Yes	0	Message ID i
segment_id	int	Yes	80	Segment ID, i
send_message_id	string	Yes	“”	Sent messag
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODIKR8-gA”	Full SendMes

Events of merging multiple profiles or splitting one profile

For profile_merge, profile_split events:

Parameter	Type	Required	Example	Description
destination	array	Yes	[{ “custom_data”: { “age”: 25 }, “db_id”: 94, “profile_id”: “608a8960e3224a6be26a71df” }]	If profile merging occurs, it contains the matching data of the profile that w If profile splitting occurs, it contains the matching data of the profiles that
source	array	Yes	[{ “custom_data”: { “age”: 25 }, “db_id”: 94, “profile_id”: “608a8960e3224a6be26a71df” }]	When profiles are merged, it contains matching data about the profiles that When a profile is split, it contains matching data about the profile that must

Channel events

For events of the following channels:

- **Telegram:** `telegram_bot_send, telegram_bot_deliv, telegram_bot_click, telegram_bot_uclick, telegram_bot_undeliv, telegram_group_deliv, telegram_group_click, telegram_group_undeliv`
- **MAX:** `max_bot_send, max_bot_deliv, max_bot_click, max_bot_uclick, max_bot_undeliv, max_group_send, max_group_deli`
- **WhatsApp:** `whatsapp_deliv, whatsapp_undeliv, whatsapp_send, whatsapp_click, whatsapp_uclick, whatsapp_subscribe`
- **Viber:** `viber_deliv, viber_undeliv, viber_send, viber_click, viber_uclick, viber_subscribe, viber_read, viber_unsu`
- **Notify:** `notify_deliv, notify_undeliv, notify_send, notify_click, notify_uclick, notify_subscribe, notify_read, no`

Parameter	Type	Example	Required	
campaign_id	int	9	Yes	Mailing ID in the Altcraft system
launch_id	string	“”	Yes	Mailing launch ID
message_id	int	0	Yes	Message ID in the Altcraft system
resource_id	int	3	No	Resource ID in the Altcraft system
segment_id	int	80	Yes	Segment ID, if available
send_message_id	string	“”	Yes	Sent message ID
full_message_id	string	“fQIQs4FgAAEBIcARADkxODlKR8-gA”	Yes	Full SendMessageID
city	string	“Moscow”	No	Subscriber city
browser	string	“Firefox”	No	Browser and version
country	string	“.ru”	No	Country in TLD format
device	string	“web”	No	Device
ip	string	“10.9.0.14”	No	Subscriber IPv4 address
lat	float	55.244075	No	Geographic latitude
link_url	string	“http://www.example.com”	No	HTTP link the user was redirected to
lon	float	55.244075	No	Geographic longitude
os	string	“Linux x86_64”	No	Operating system
postal_code	string	“390000”	No	Subscriber postal code
region	string	“Ryazanskaya oblast”	No	Subscriber region
tz	string	“Europe/Vaduz”	No	Subscriber time zone
undeliv_code	int	500	No	Delivery failure error code
undeliv_message	string	“Content is required”	No	Error information
workflow_id	int	1	No	Scenario ID, if the mailing was launched

Mailing error events

For camp_error events:

Parameter	Type	Required	Example	
campaign_id	int	Yes	9	Mailing ID in t
error_code	int	Yes	500	Error code
error_text	string	Yes	“LOG_INTERNAL_ERROR”	Error text
launch_id	string	Yes	“”	Launch ID
message_id	int	Yes	0	Message ID in
resource_id	int	Yes	3	Resource ID ir
segment_id	int	Yes	80	Segment ID, if
send_message_id	string	Yes	“”	Sent message
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODlKR8-gA”	Full SendMess

campaign_launch structure

General campaign_launch structure:

Parameter	Type	Required	Example	
event_type	string	Yes	“campaign_launch”	Event type
event_date	string	Yes	“2018-03-25T13:13:19.034315796Z”	Event date in RFC 3339 format
account_id	int	No	23	Platform account ID
is_test	bool	Yes	false	Test event sending flag
is_static	bool	No	true	Whether dynamic substitutions were used in
is_historical	bool	Yes	false	Whether the event is historical
event_data	JSON object	Yes	{}	Object with mailing data

event_data structure for campaign_launch

Parameter	Type	Required	Example	
campaign_id	int	Yes	42	Mailing identifier
workflow_id	int	No	1	Scenario ID, if the campaign was launched
scampaign_id	int	No	1	Campaign ID in the Altcraft system
campaign_sub_id	string	Yes	“x11”	Statistics identifier
campaign_name	string	Yes	“name”	Mailing name
campaign_type	string	Yes	“broadcast”	Mailing type
data_type	string	Yes	“segment”	Data source type
data_id	int	Yes	3	Data source identifier
launch_id	string	Yes	“00000017000003cd5a9bf01a”	Launch ID
content	JSON object	Yes	{}	Information about mailing message templa

Campaign launch event example

```
{
  "event_type": "campaign_launch",
  "event_date": "2016-12-09T10:11:24.934079204Z",
  "is_test": false,
  "account_id": 161,
  "event_data": {
    "campaign_id": 1472,
    "campaign_type": "broadcast",
    "campaign_name": "Campaugn Launch Hook",
    "campaign_sub_id": "x11",
    "data_type": "segment",
    "data_id": 1221,
    "launch_id": "000000a1000005c0584a834c",
    "content": [
      {
        "message_name": "Hello",
        "message_id": 1412,
        "suppress_id": 271,
        "replyto_email": "",
        "replyto_name": "",
        "text": "текстовая версия письма",
        "html": "<html> html версия письма </html>",
        "from_name": "",
        "from_email": "",
        "attach_links": [],
        "subject": "Hello, {your_name}!"
      },
      {
        "message_name": "Hi",
        "message_id": 1413,
        "suppress_id": 272,
        "replyto_email": "monty@altcraft.com",
        "replyto_name": "Monty",
        "text": "текстовая версия письма",
        "html": "<html> html версия письма </html>",
        "from_name": "",
        "from_email": "",
        "attach_links": [
          {
            "name": "text_file.txt",
            "url": "http://test.altcraft.com/api/attach/link/text_file.txt?token=eyJh..."
          },
          {
            "name": "picture_file.png",
            "url": "http://test.altcraft.com/api/attach/link/picture_file.png?token=eyJh..."
          }
        ],
        "subject": "Hi, {your_name}!"
      }
    ]
  }
}
```


Subscription structure

This object contains information about the subscription to the resource through which the message wa

INFO: Required fields

The following fields are required for the subscription structure: `resource_id`, `channel`, `priority`, `status`,

Parameter	**Type**	**Example**	**Description**
Common for all channels			
resource_id	int	12	Resource ID
channel	string	"email"	Communication channel
priority	int	0	Subscription priority
status	string	"subscribed"	Subscription status
ccsub	JSON object	\{"custom_key": "custom_value"\}	Object with custom channel subscription fields. If there are no fiel
For the email channel			
email	string	"john@example.com"	Subscriber email
For the SMS channel			
phone	string	"+79991112233"	Subscriber phone number
For the push channel			
provider	string	"YandexAppMetricaDeviceIDAndroid"	Push provider
subscription_id	string	"123ab667960efec47b1afe64b56bfa386a3"	Provider subscription ID

content structure

If the “Add messages content” option is enabled in the webhook settings, the `content` field is added to structure.

For Push and Viber, content is not transmitted. For Telegram, content fully depends on the template.

Email

```
{
    "content": {
        "amp": "<html amp4email></html>",
        "from_email": "no-reply@altcraft.com",
        "from_name": "Altcraft",
        "html": "<html></html>",
        "replyto_email": "example@example.com",
        "replyto_name": "Altcraft",
        "subject": "Weather alert",
        "text": "Text version",
        "to_email": "profile@example.com",
        "to_name": "Fname Lname",
        "webversion_email_html": "http://click.altkraft.com/0000/path.html",
        "webversion_email_text": "http://click.altkraft.com/0000/path.txt"
    },
}
```

Parameter	Type	Required	Example	
from_email	**string**	**Yes**	**“no-reply@altcraft.com”**	**Sender email**
from_name	**string**	**Yes**	**“Altcraft”**	**Sender name**
html	**string**	**Yes**		**HTML version of the ema**
replyto_email	**string**	**Yes**	**“example@example.com”**	**Reply-to email**
replyto_name	**string**	**Yes**	**“Altcraft”**	**Reply-to name**
subject	**string**	**Yes**	**“Weather alert”**	**Email subject**
text	**string**	**Yes**	**“Text version is important...”**	**Text version of the emai**
to_email	**string**	**Yes**	**“profile@example.com”**	**Recipient profile email a**
to_name	**string**	**Yes**	**“Fname Lname”**	**Recipient profile name**
amp	**string**	**No**		**AMP version of the emai**
webversion_email_html	**string**	**Yes**	**http://click.altkraft.com**	**Link to the web version**
webversion_email_text	**string**	**Yes**	**http://click.altkraft.com**	**Link to the text web ver**

SMS

```
{
    "content": {
        "fromnamesms": "abcdefghijklmnqrstuvwxyz",
        "sms": "Hello, Aaron! See the new items for you: http://click.altkraft.com/s/path",
        "webversion_sms_text": "http://click.altkraft.com/000000/path.txt"
    }
}
```

Parameter	Type	Required	Example	
fromnamesms	**string**	**Yes**	**“abcdefghijklmnqrstuvwxyz”**	**Caller ID**
sms	**string**	**Yes**	**“Follow the link and claim your welcome bonus!”**	**Message te**
webversion_sms_text	**string**	**Yes**	**http://click.altkraft.com**	**Link to the**

WhatsApp

Parameter	Type	Required	Example	
type	**string**	**Yes**	**“template”**	**Message type**
category	**string**	**Yes**	**“marketing”**	**Message categ**
name	**string**	**Yes**	**“welcome”**	**Template name**
language	**string**	**Yes**	**“en”**	**Template langu**
namespace	**string**	**Yes**	**“xxx”**	**Template name**
wa_account_id	**string**	**Yes**	**“123456”**	**WhatsApp acco**
components	**array**	**Yes**	**[...]**	**Message compo**

MAX

Parameter	Type	Required	Example	
text	**string**	**Yes**	**“Hello!”**	**Message t**
buttons	**array**	**Yes**	**[...]**	**Message b**
disable_link_preview	**bool**	**Yes**	**true**	**Disable lin**

Push

```
{
    "content": {
        "app_push_settings": {
            "background_color": "#00CED1",
            "badges": 0,
            "banner": "",
            "critical_push": false,
            "deliver_prior": 0,
            "expir_time": 604800,
            "led": "",
            "media": "",
            "silent_push": false,
            "sound": "",
            "subtitle": "",
            "vibro": false
        },
        "body": "See if our new items are fit to make you even more awesome!",
        "click_url": "https://click.altkraft.com/0000000",
        "custom_push_json": {},
        "icon": "",
        "result_content": {
            "content_available": true,
            "data": {
                "hub_link": {
                    "ack": "https://pxl.altkraft.com/crypto_post_events?/bFOoNcxW7lpDA",
                    "open": "https://pxl.altkraft.com/crypto_post_events?/LbFOoNcxW7lpDA",
                    "read": "https://pxl.altkraft.com/crypto_post_events?/cFOoNcxW7lpDA"
                },
                "image": "",
                "launch_id": "2_3J_4Z2ZTRaQr7p",
                "mediaUrl": "",
                "message": "See if our new items are fit to make you even more awesome!",
                "uid": "fATFSGbgLjinkRE59ZXDOSd0BcAABARnhIxDBBAAAEbAQowRH"
            },
            "mutable_content": true,
            "notification": {
                "body": "See if our new items are fit to make you even more awesome!",
                "click_action": "https://click.altkraft.com/0000000/abcdefghijklmnqrstuvwxyz",
                "color": "",
                "icon": "",
                "sound": "default"
            },
            "title": "Just look, Alex! These are awesome!",
            "priority": "normal",
            "time_to_live": 604800,
            "to": "5dbDUxVXHV"
        },
        "title": "Just look, Alex! These are awesome!",
        "webversion_push_body": "http://click.altkraft.com/000000/path.txt"
    }
}
```


Parameter	Type	Description
body	**string**	**Push message body**
click_url	**string**	**Click-through link**
custom_push_json	**JSON object**	**Custom JSON of the message**
icon	**string**	**Path to the push message icon**
title	**string**	**Message title**
result_content	**JSON object**	**Stores the structure of data sent to the push service**
webversion_push_body	**string**	**Link to the web version of the push message**
app_push_settings	**JSON object**	**Advanced settings for mobile push notifications**
web_push_settings	**JSON object**	**Advanced settings for browser push notifications**

app_push_settings structure

Parameter	Type	Example	Description
background_color	**string**	**#00CED1**	**Push notification background color**
badges	**int**	**0**	**Event indicator**
banner	**string**	**http://example.com/path.img"**	**Path to the banner image**
critical_push	**bool**	**false**	**Enables critical mode in the push notification**
deliver_prior	**int**	**0,1**	**Delivery priority**
expir_time	**int**	**604800**	**Push notification TTL in seconds**
led	**string**	**#00CED2**	**LED indicator color**
media	**string**	**http://example.com/path**	**Path to a local file in the application**
silent_push	**bool**	**false**	**Enables or disables silent mode**

Parameter	Type	Example	Description
sound	**string**	**/sound/sound.mp3**	**Path to a local file in the application**
subtitle	**string**	**“Just for you!”**	**Message subtitle (for iOS)**
vibro	**bool**	**true**	**Vibration when the push notification is received**
soundless	**bool**	**true**	**Indicates whether the push message will be accompanied by sound (used by providers: android firebas**
actions	**array**	[{ “label”: “Open”, “action”: “https://example.com” }]	**Buttons in push notifications**

web_push_settings structure

Parameter	Type	Example	
actions	**JSON object**	`[]`	**The array contains push buttons: click-throug**
button_label	**string**	**“Sure”**	**Action button text (the value is passed only f**
large_image	**string**	**“https://.altkraft.com/284834790/1/664c6da207767c769a/image.png”**	**Path to the banner image**
expir_time	**string**	**604800**	**Push notification TTL in seconds**
large_image	**string**	**“https://click.example.com”**	**Path to the local banner image**

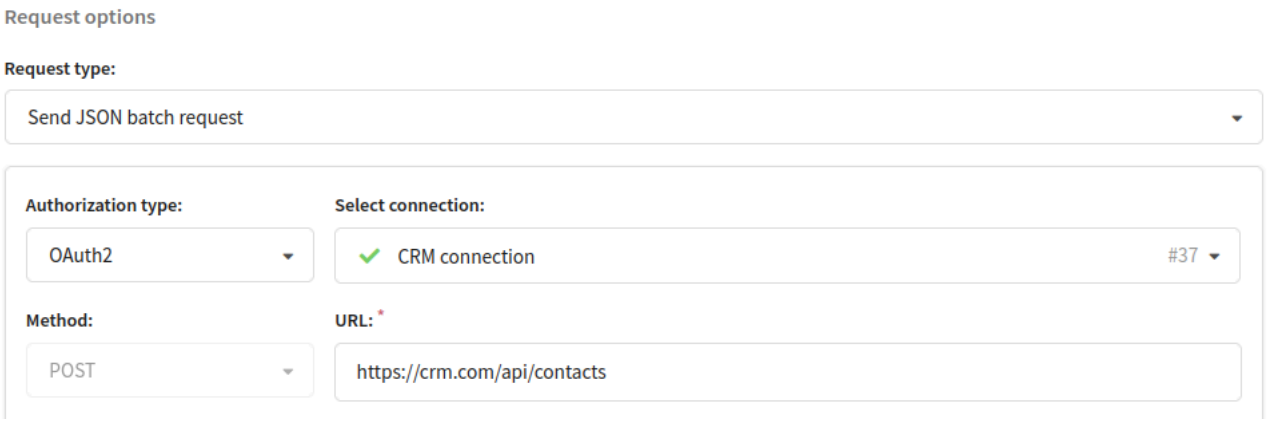
**Meta, which owns Instagram, Facebook, and WhatsApp, has been recognized as an extremist organi
Federation.*
Viber is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.

JSON batch request (HTTP POST action hook)

Batch requests allow you to combine many HTTP requests into one by specifying each request as a se server returns one HTTP response that contains the responses to each submitted request.
When we choose this request type, we send the received events in JSON format in the body of the POS sent, a JSON array with events will be generated.

Request settings

The standard request is sent without authorization. However, if you want to send data to an external s framework as its authorization method, change the authorization type to OAuth2. In this case, you will links Altcraft Platform and the service.



You have to specify:

- **URL of the service that receives the data**
- **Maximum batch size: limit the number of events in JSON array. POST request will be sent as soon a**

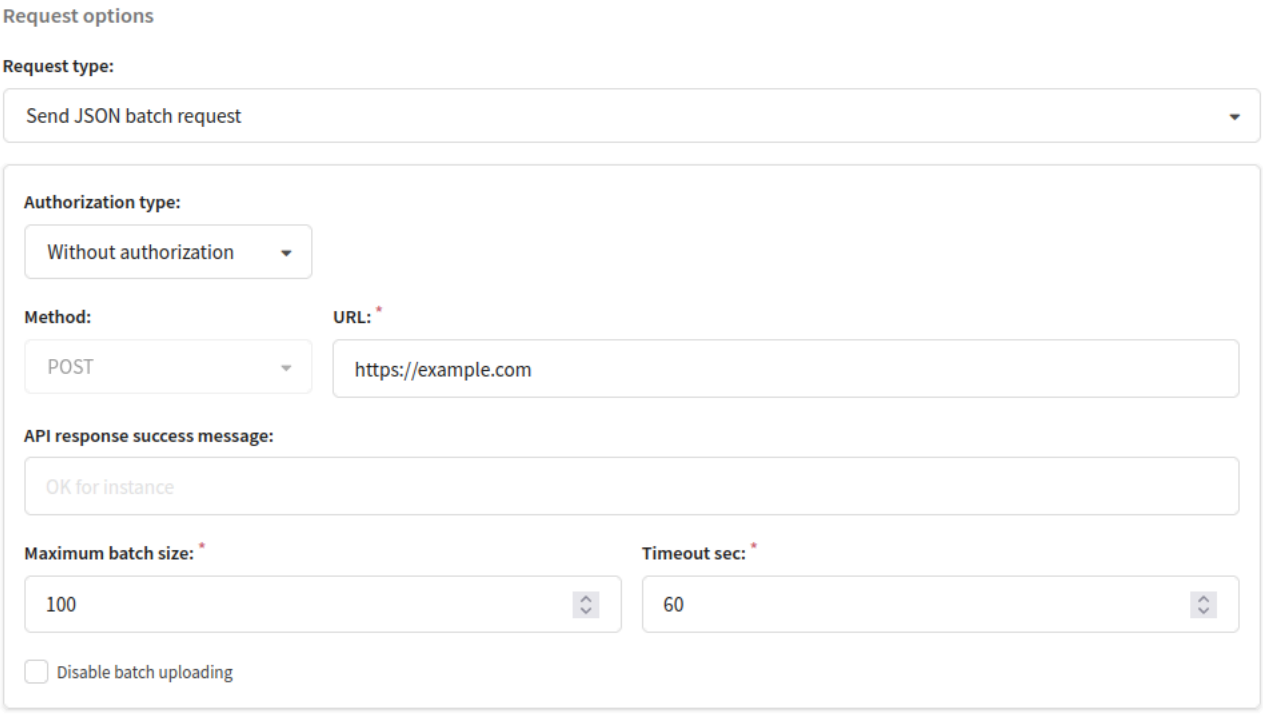
```
[{"event": "send", ... }, { }, ... ]
```

- **Timeout sec: the period of time during which events are accumulated in a batch to be sent to the s maximum batch size is not reached, the POST request is still sent. In this case, the JSON array co have accumulated by the time the timeout expires.**
- **API response success message: the response from the server must go into this Regex, only then th considered successful. For instance:**

```
.*OK.*
```

CAUTION

If you leave field API response success message empty, any response the status code of which is 200

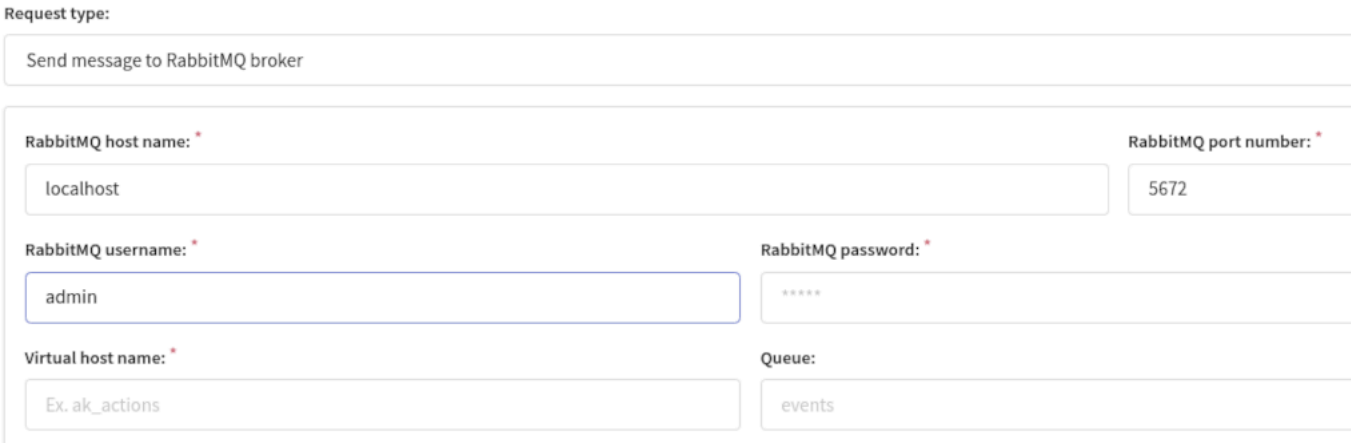


Email delivered message example


```
[
  {
    "event_type": "deliv",
    "list_id": 50,
    "account_id": 161,
    "_xxh": "5ada5e6ea3c6da5f",
    "_md5": "a5656dc1e3f603063a1918d582a10f58",
    "event_data": {
      "message_id": 5,
      "segment_id": 0,
      "campaign_id": 130
    },
    "event_date": "2016-11-15T09:12:28.514388956Z",
    "custom_data": {
      "email": "eddie@test-ukr.net",
      "_fname": "Eddie"
    },
    "is_test": false
  },
  {
    "event_type": "deliv",
    "list_id": 50,
    "account_id": 161,
    "_xxh": "bb42e541c7267fa6",
    "_md5": "593b250db256a764da8d2da4be694b31",
    "event_data": {
      "message_id": 5,
      "segment_id": 0,
      "campaign_id": 130
    },
    "event_date": "2016-11-15T09:12:28.569124547Z",
    "custom_data": {
      "email": "simon@test-gmail.com",
      "_fname": "Simon"
    },
    "is_test": false
  }
]
```

Message to RabbitMQ broker

**Each single event will be sent immediately to the message broker queue. Both a local RMQ, which is pa
located on an external host can be used.**



In the settings, specify the broker's parameters:

- **host name**
- **port number**
- **username**
- **password for authorization**
- **virtual host name**
- **name of the queue into which event messages will be sent.**

CAUTION

**RabbitMQ user must have access to the selected virtual host. Create a virtual host and give the Rabb
in RabbitMQ.**

**You are free to choose whether to set up RabbitMQ queue name: if it is not set - a queue will be named
Email link click message example.**


```
{
  "event_type": "click",
  "event_date": "2018-03-04T13:13:19.034315796Z",
  "list_id": 111,
  "_xxh": "ad1850e0d1d9481e",
  "_md5": "145bb30f7ddf1184cb560bc4b7705646",
  "is_test": false,
  "event_data": {
    "browser": "Firefox",
    "campaign_id": 973,
    "city": "",
    "country": "",
    "device": "web",
    "ip": "10.9.0.14",
    "lat": 0,
    "launch_id": "00000017000003cd5a9bf01a",
    "link_url": "http://www.ford.com",
    "lon": 0,
    "message_id": 17,
    "os": "Linux x86_64",
    "postal_code": "",
    "region": "",
    "segment_id": 0,
    "send_message_id": "5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78",
    "tz": "",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0"
  },
  "custom_data": {
    "email": "alice@example.org"
  },
  "account_id": 23
}
```

Message to RabbitMQ exchanger

Exchange is a point of exchange, in other words, another player between the message sender and the distributes messages to one or more queues.

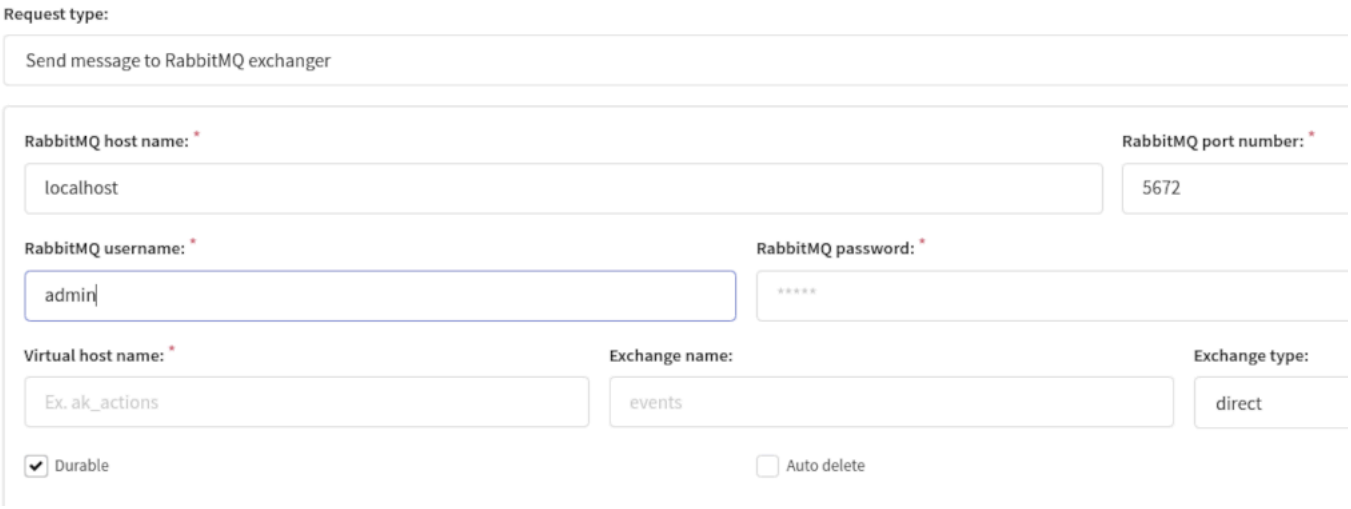
To configure sending a message to exchange RabbitMQ, specify host name, port name, username, pas

Next select an exchange type:

- **Direct type allows sending messages to only one queue subscribed to the corresponding routing k automatically using the pattern:** `<db_id>.<event_type>`**.**
- **Topic type allows sending messages to multiple queues that are subscribed to a routing key patter**

The durable option is enabled by default. This means that the exchange point will be permanently stor survive a server restart. If you disabled this field, the exchange will become temporary and will be del

Also pay attention to the Auto delete field. If you enable this field, then exchange will be deleted as so deleted.

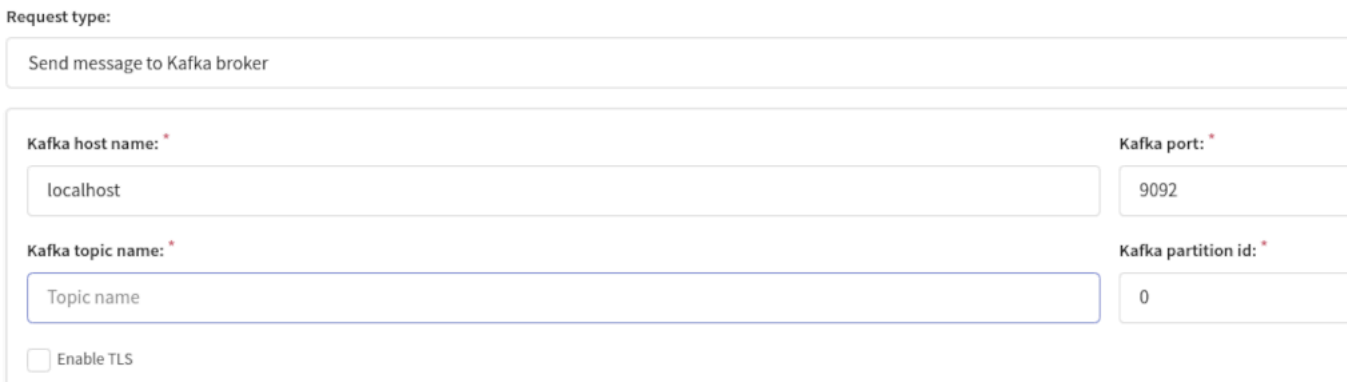


Message to Kafka broker

To configure sending events to Apache Kafka, specify the next parameters:

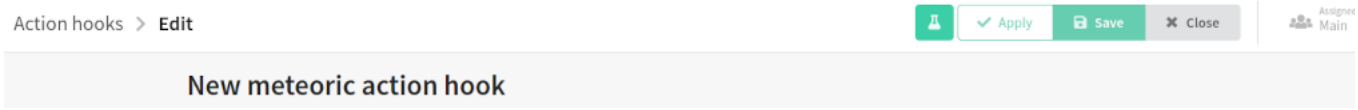
- **kafka host name,**
- **kafka port,**
- **kafka topic name,**
- **kafka partition id.**

Activate TLS if you need. When using a self-signed certificate, if it is not included in the trusted ones, s
PEM format. This also takes into account the global option `ALLOW_INSECURE_REQUESTS` in the main.json confi
The validity of certificates is not checked. If authorization with a client certificate is used, then you mu
key and the certificate, also in PEM format.

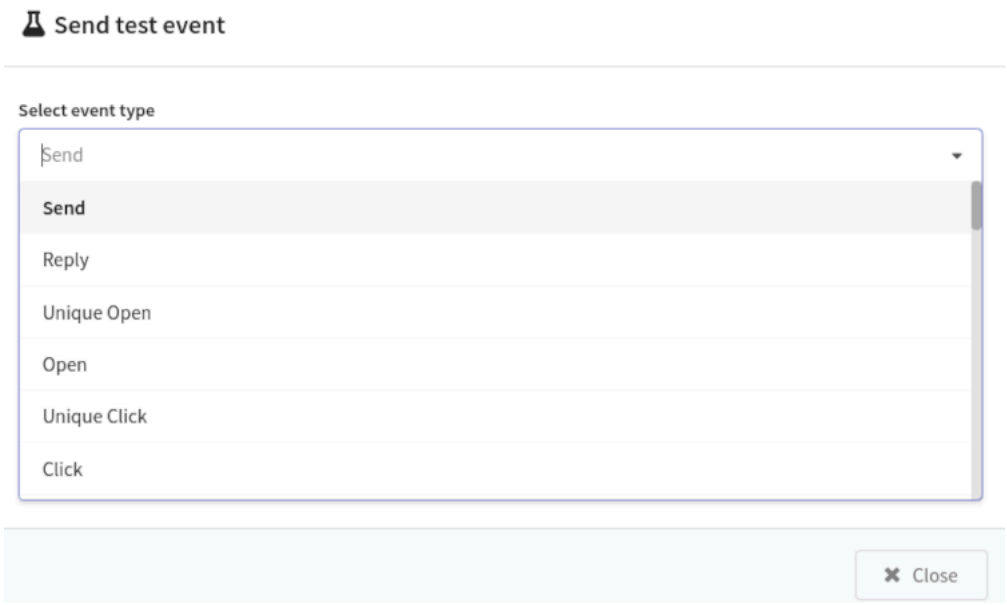


Test event

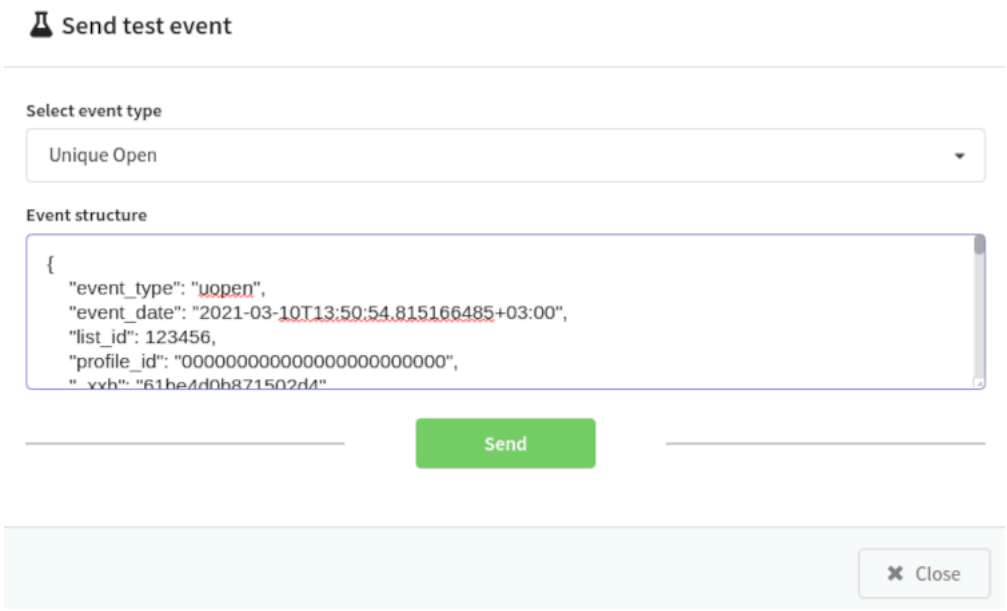
When you are working with action hooks you can test any event using Altcraft Platform interface. Click



Select event type in new window:



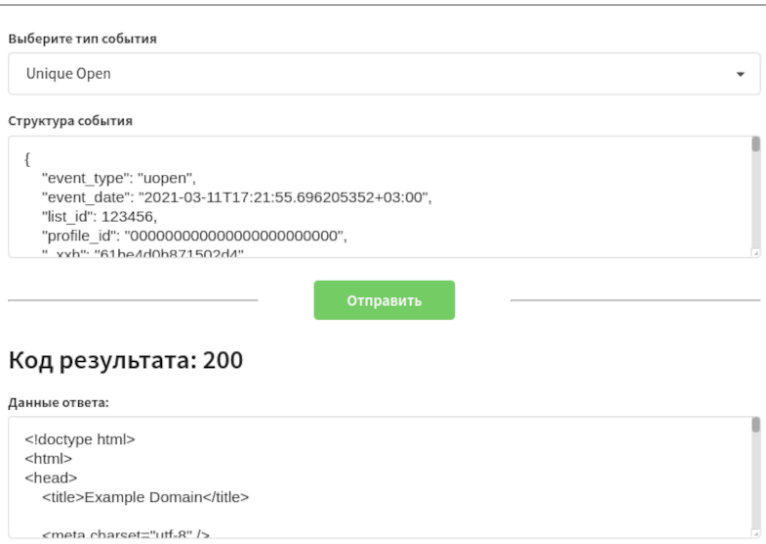
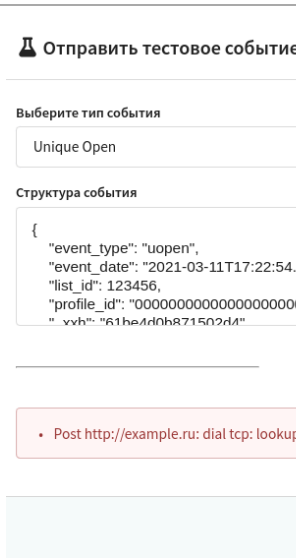
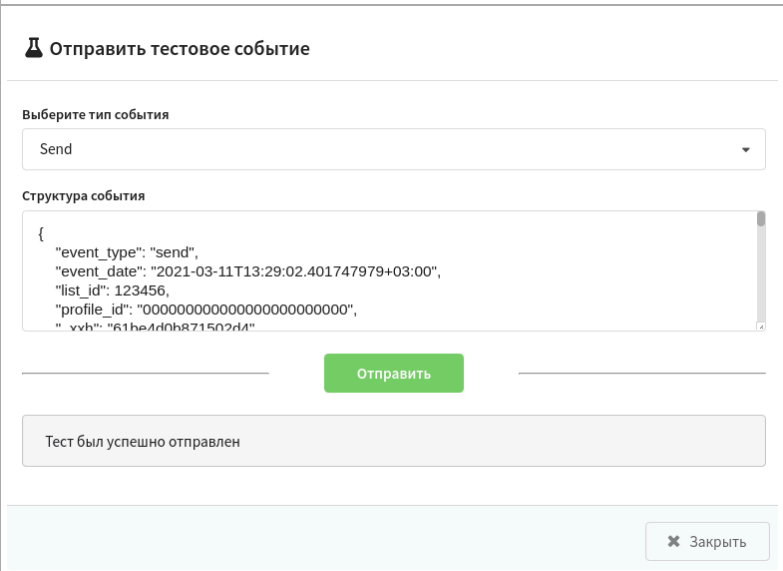
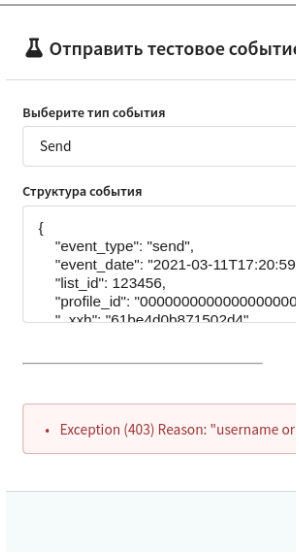
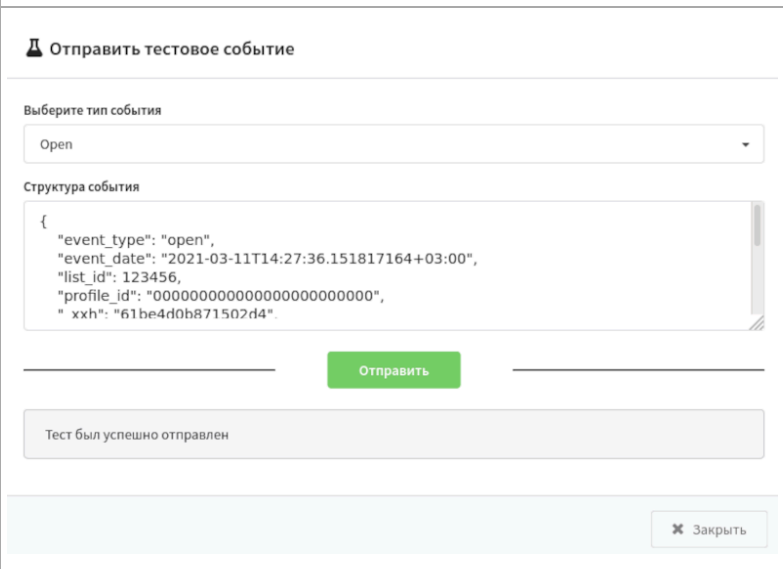
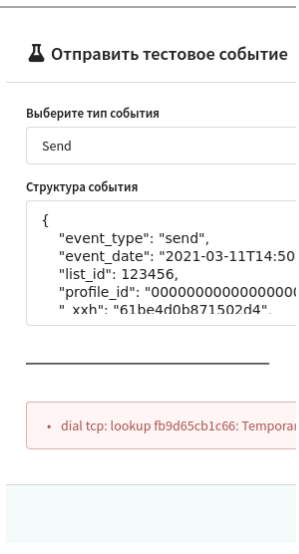
En example of the structure of selected event will be generated automatically. This structure can be c



Click Send to test the event.

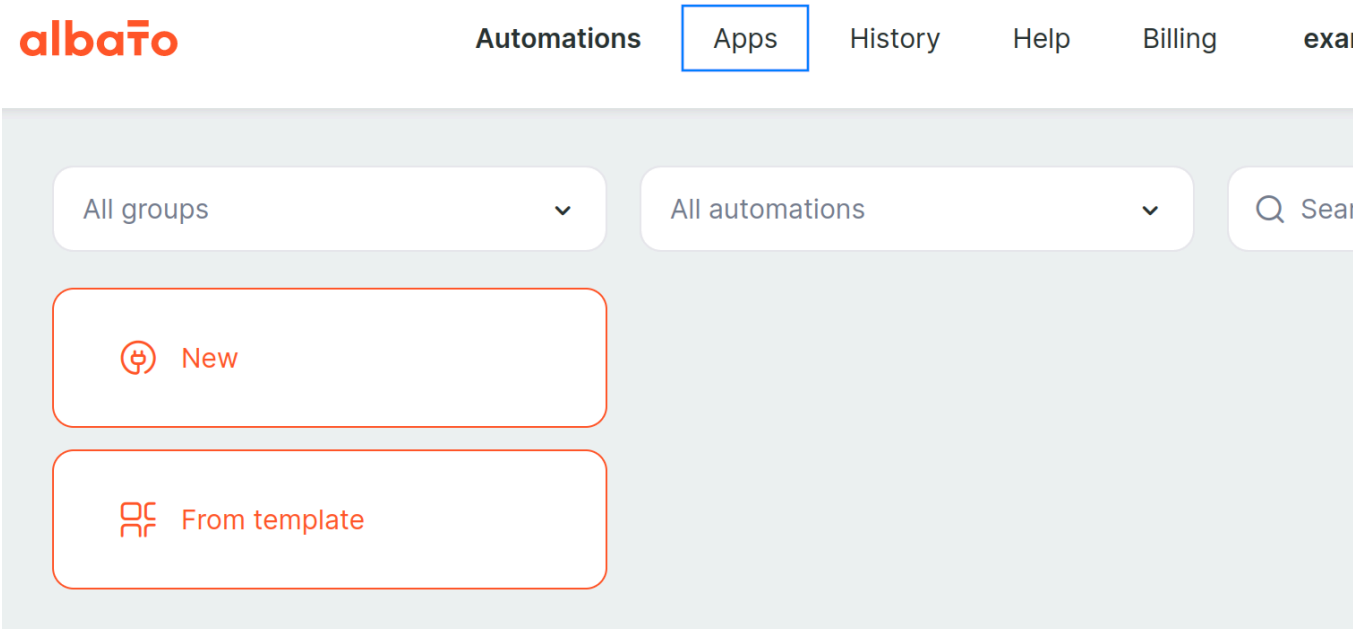
- **If the message was successfully sent via an HTTP request, you will get Result code: 200 and Resp**
- **If the message was successfully sent to the queue or exchanger RabbitMQ or to Kafka broker, ther was sent successfully."**

In other case there will be red field, where you can know the error code and its reason.

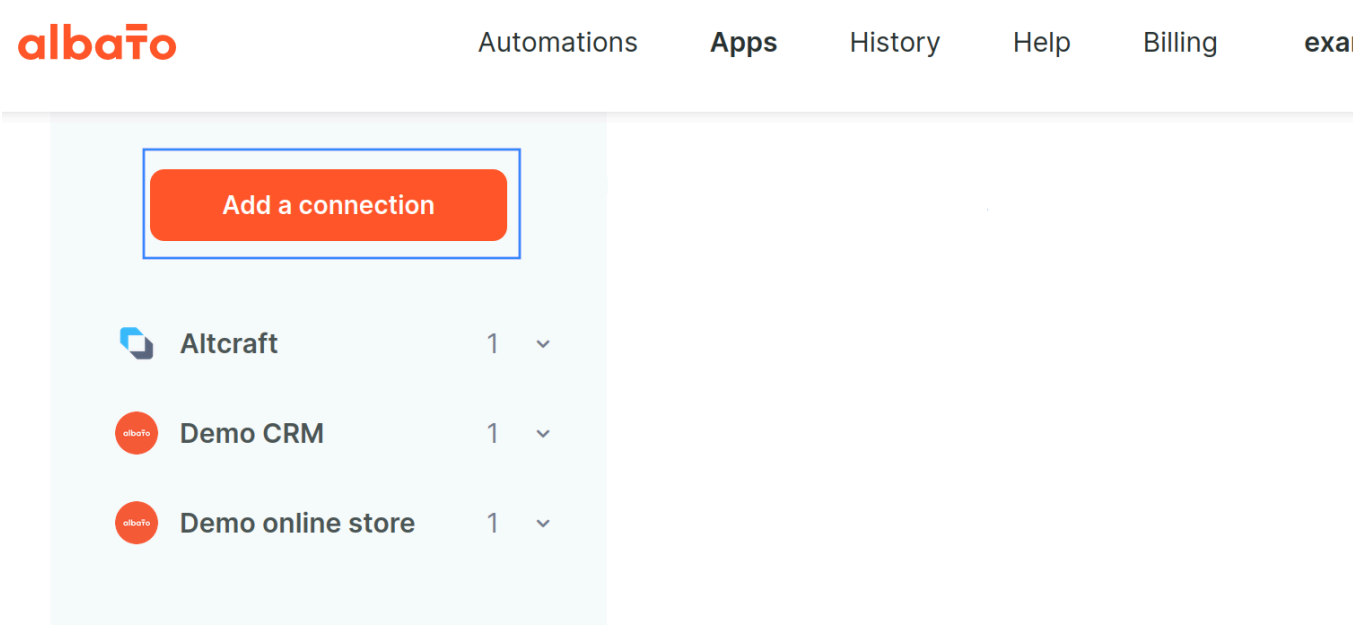
Request type	The message is successfully delivered	
JSON batch request		
Message to RabbitMQ broker / exchanger		
Message to Kafka broker		

Connecting Altcraft to Albato

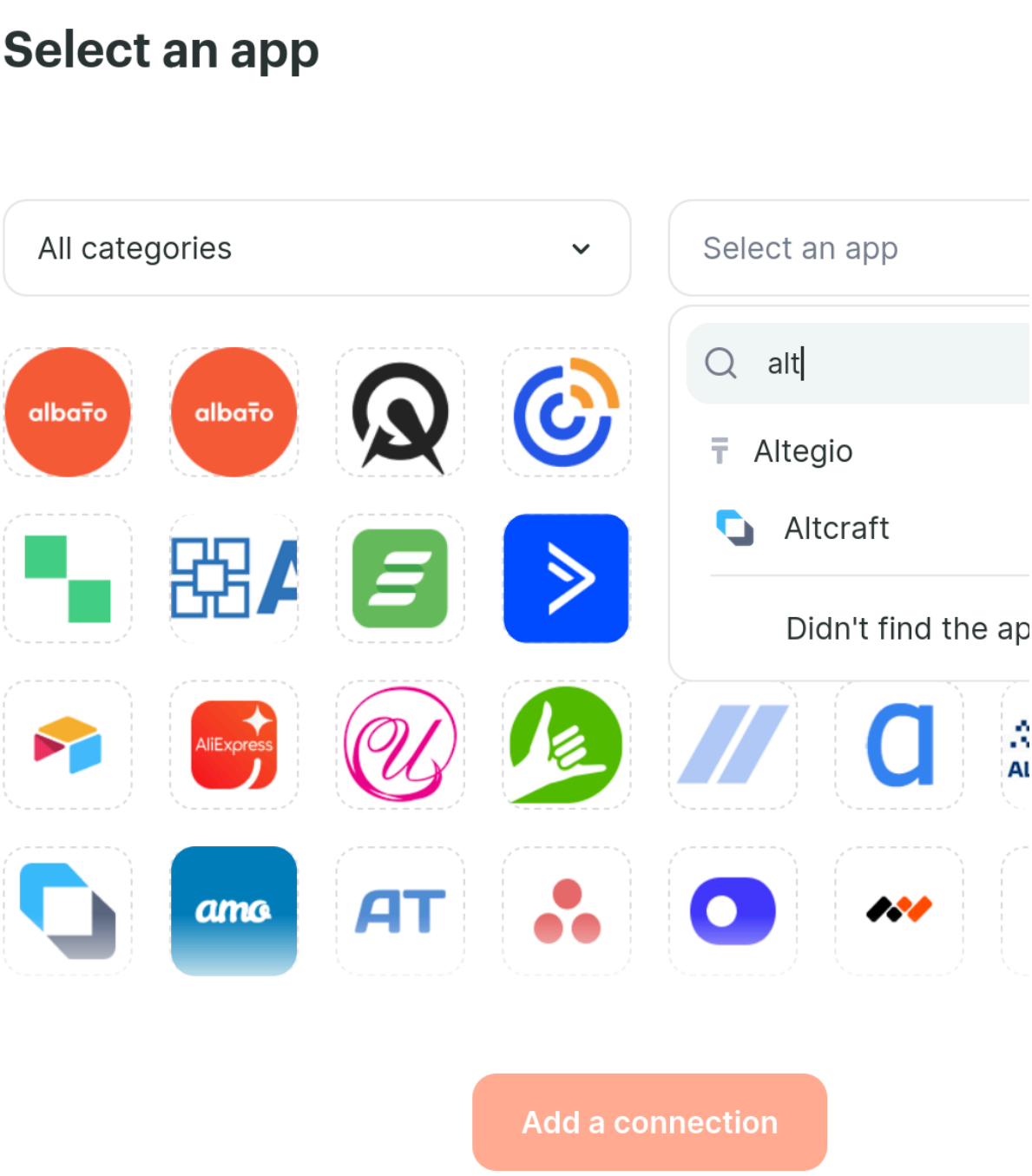
Before you can set up integration with the desired service, you need to connect Altcraft to Albato. Sta account. Then select “Apps” from the top menu.



In the “Apps” section in the menu on the left, click “Add a connection”.



Now you need to select a service. In the “Select an app” field, specify the name of the platform — Altc must confirm the selected service by clicking on the “Add a connection” button.



After adding the connection, in the next window you can specify the connection title (or leave the defa (required field).

TIP

Obtaining an Access Token in Altcraft Platform is described in the documentation.

Creating a connection to Altcraft

Tutorial

Title *

My Account Altcraft

Create your connection name

Access token

••••••••••••••••••••••••••••••••

After adding the Access token, click “Continue”. The following window will open, displaying informatio connection.

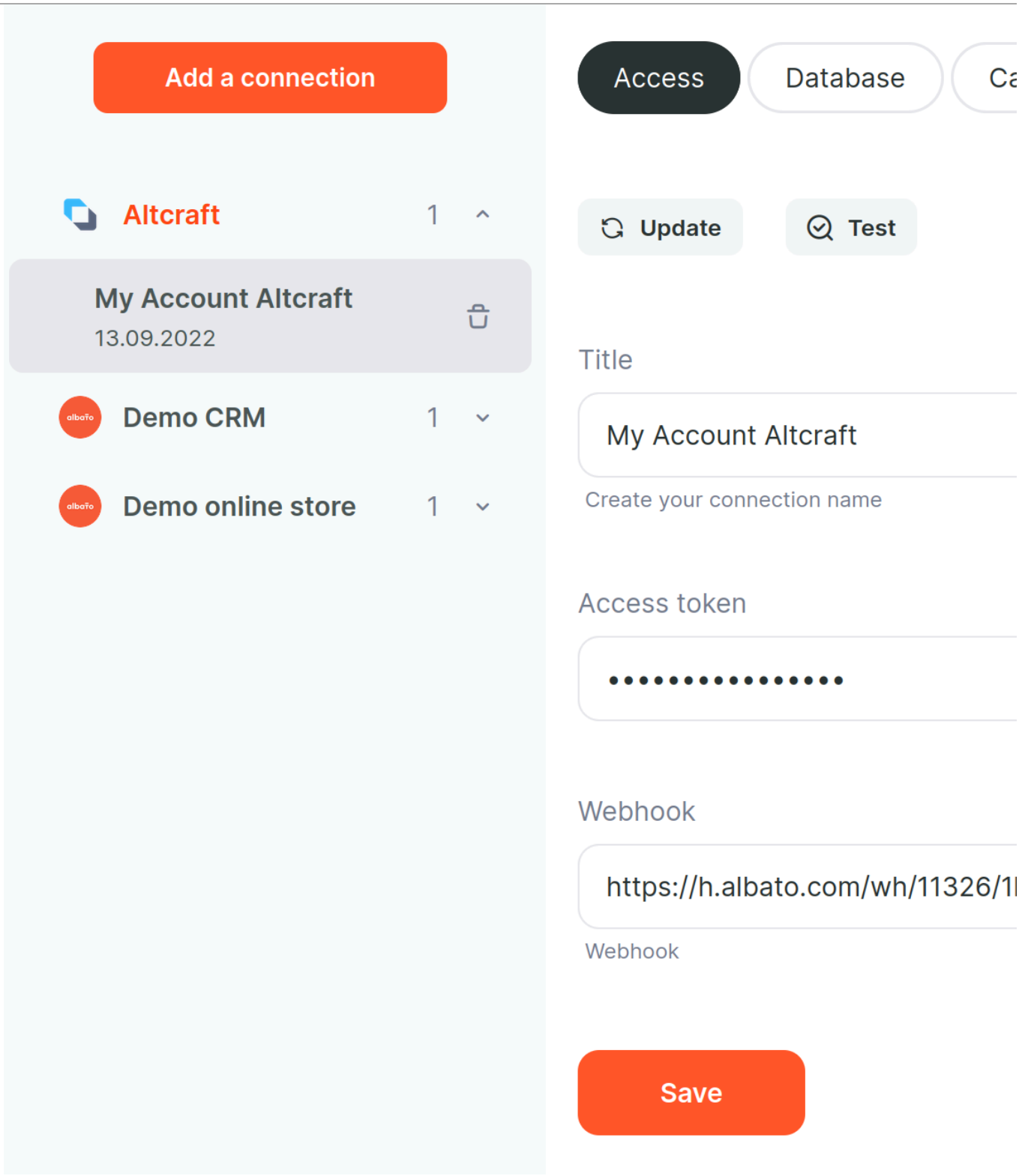
If the connection is successful, in the “Apps” section, in the left menu, Altcraft will be displayed in the

CAUTION

Albato can connect multiple Altcraft users with limited access. To do this, you need to generate a nev necessary rights and create a new connection as described above.

It is also necessary to specify the names for each connection, so that later it will be convenient to vie

To test the connection, you must select a connection from the list (if there are several). Next, you will information.



In this window, you can view the list of available databases (there may be restrictions depending on th well as the list of available trigger mailings. You can also test the connection.

Launching the welcome scenario using Albato

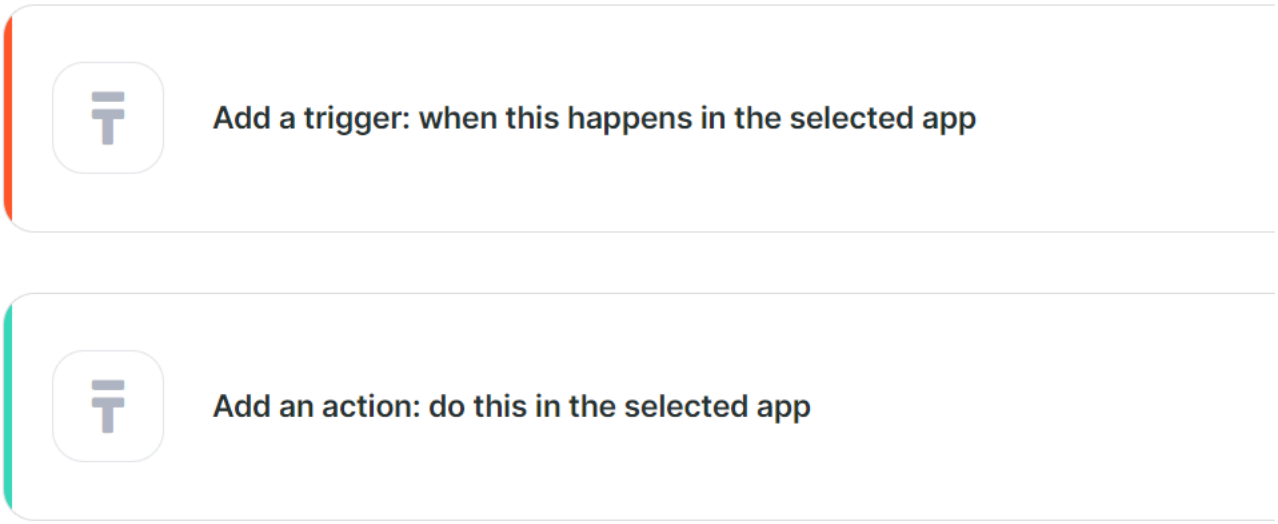
Albato allows you to configure data transfer from an external service to the Altcraft platform. Using a s
information see the [developer guide] section), which comes from Albato, the platform decides what to d
The platform can:

- **update the data in the client's profile by importing**
- **add a profile to a segment**
- **send the client to the scenario**

In this article, we will know how to use Albato to add a client profile to a private scenario.

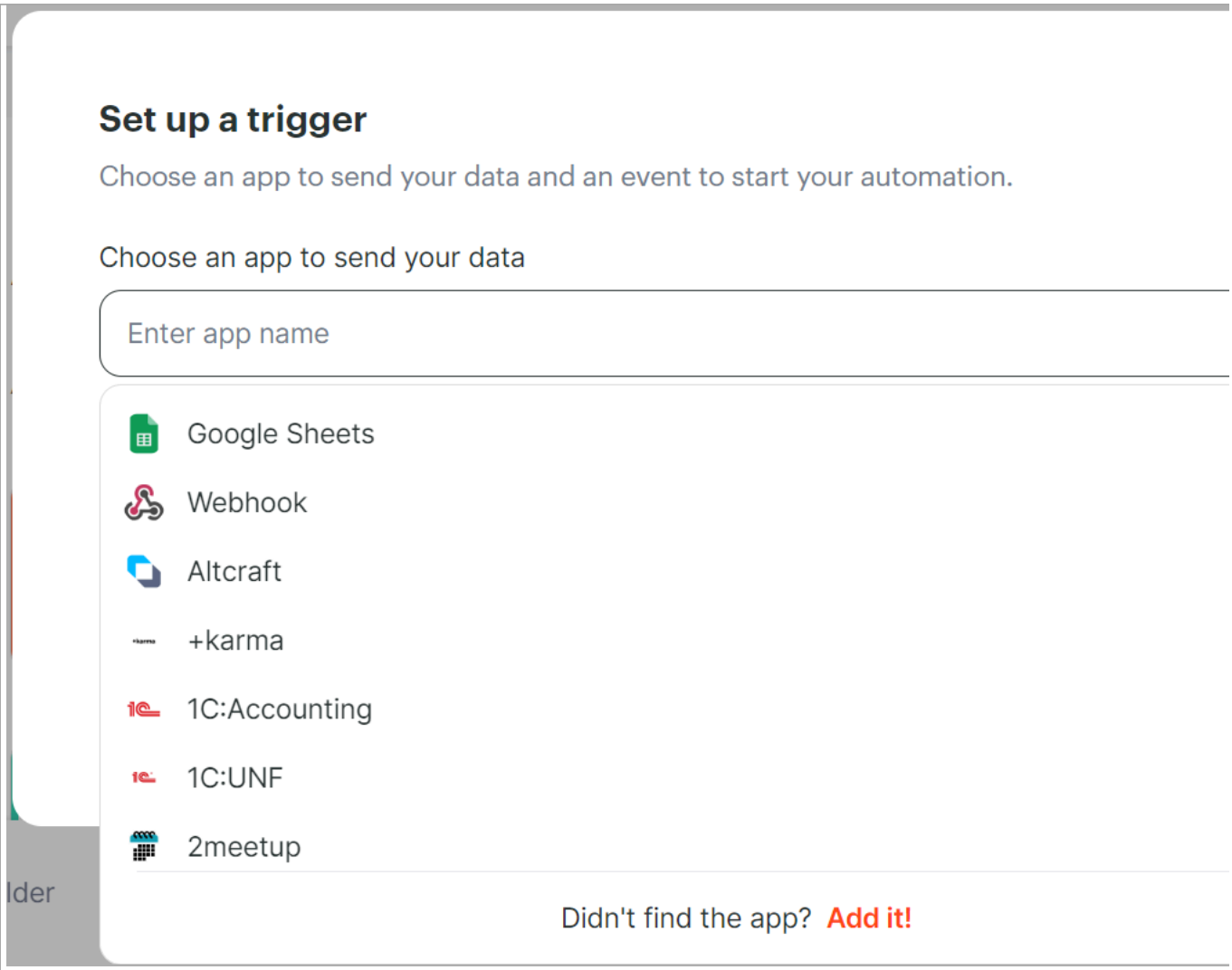
Create a new automation in Albato

A automations in Albato consists of two components: a trigger that start the transfer of data from an e
action that occurs after the trigger is started.

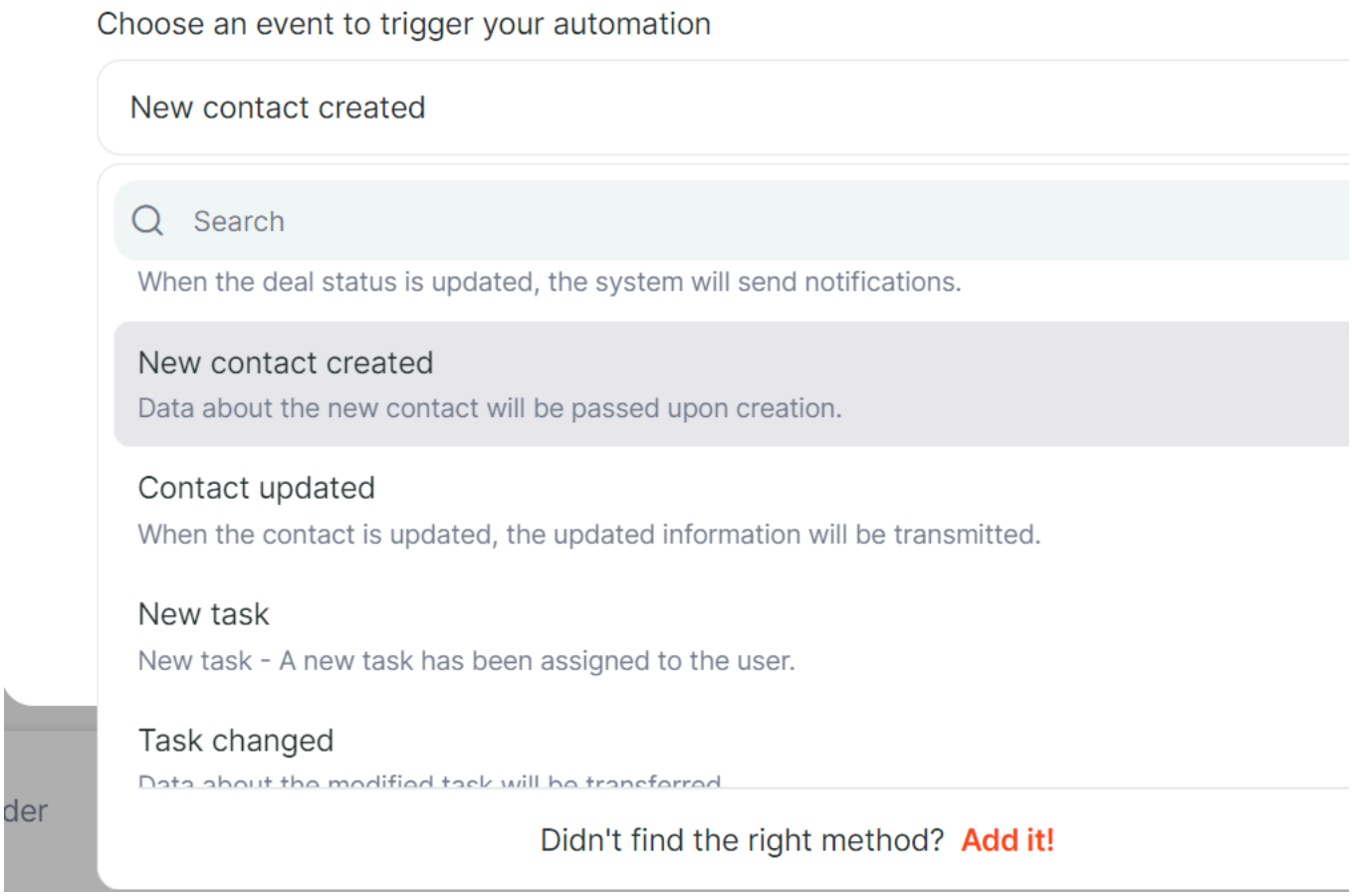


Add a trigger

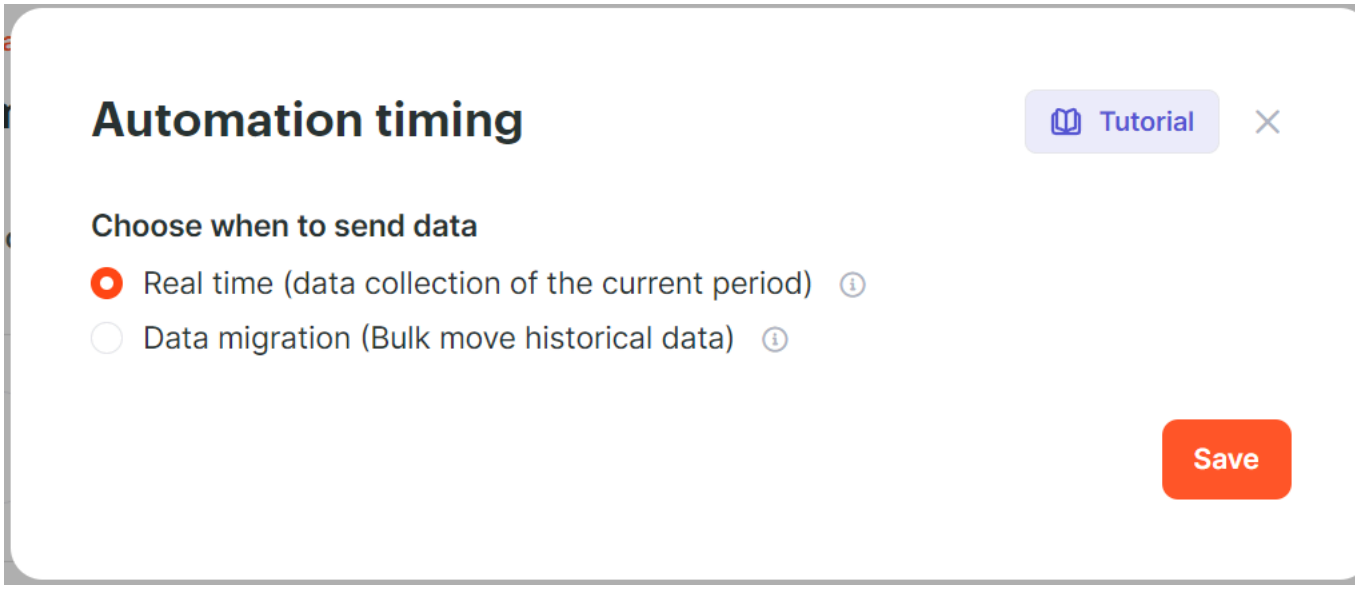
Select the service from which Albato will receive data:



Select the event that will trigger the data transfer. In our case, the action will be “New contact created

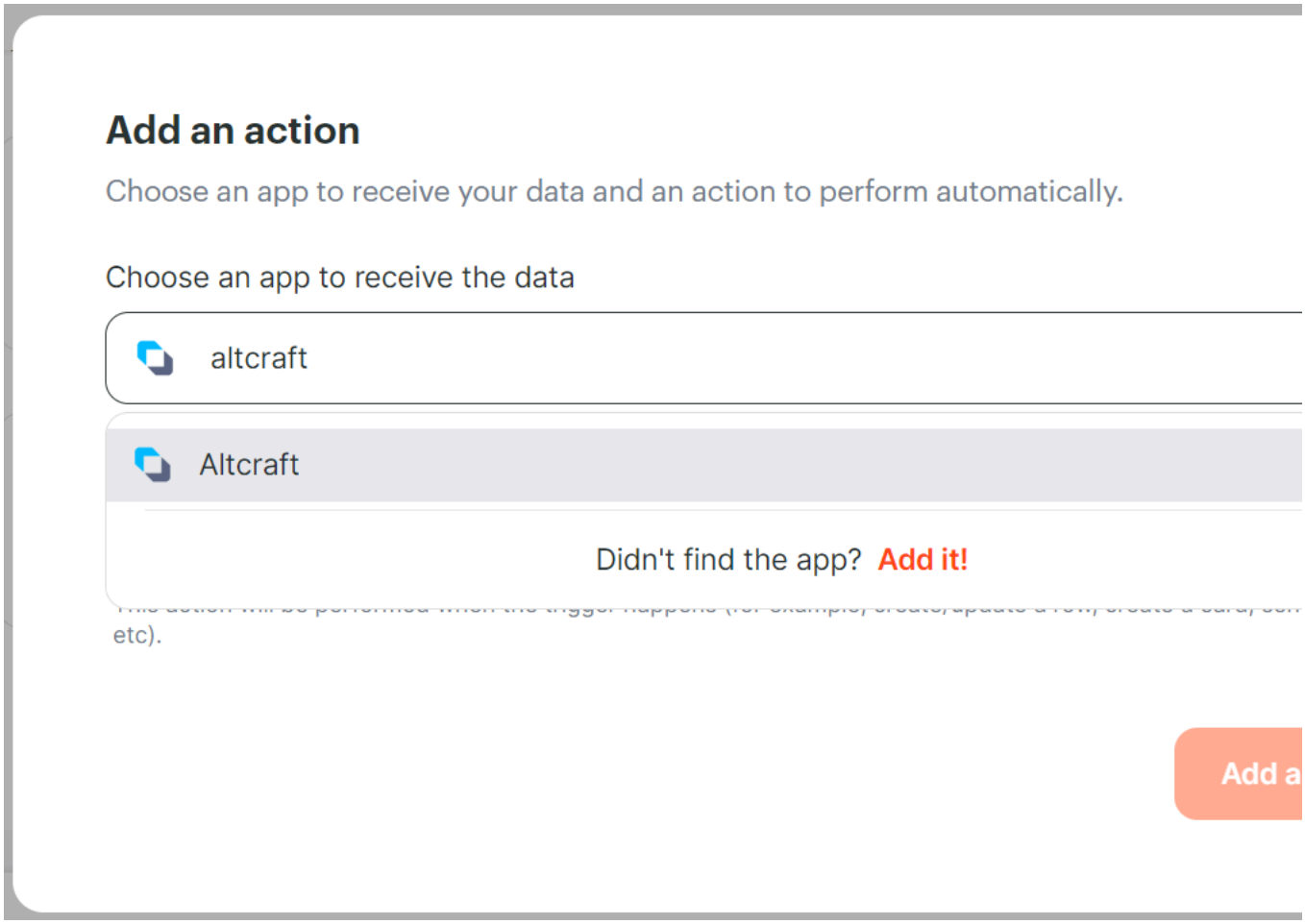


Select the mode of operation of the automations: data can be sent to Altcraft in real time or specify fo

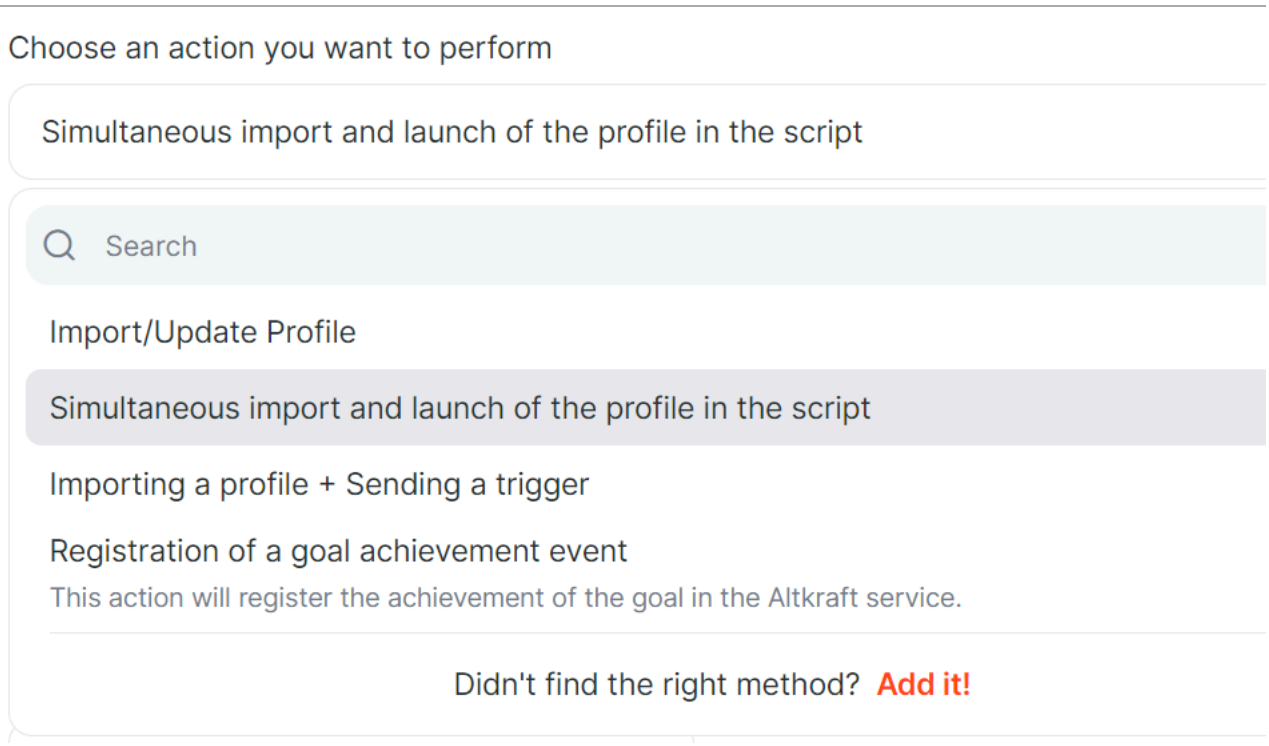


Add an action

Select the Altcraft platform as the server that will receive data from service.



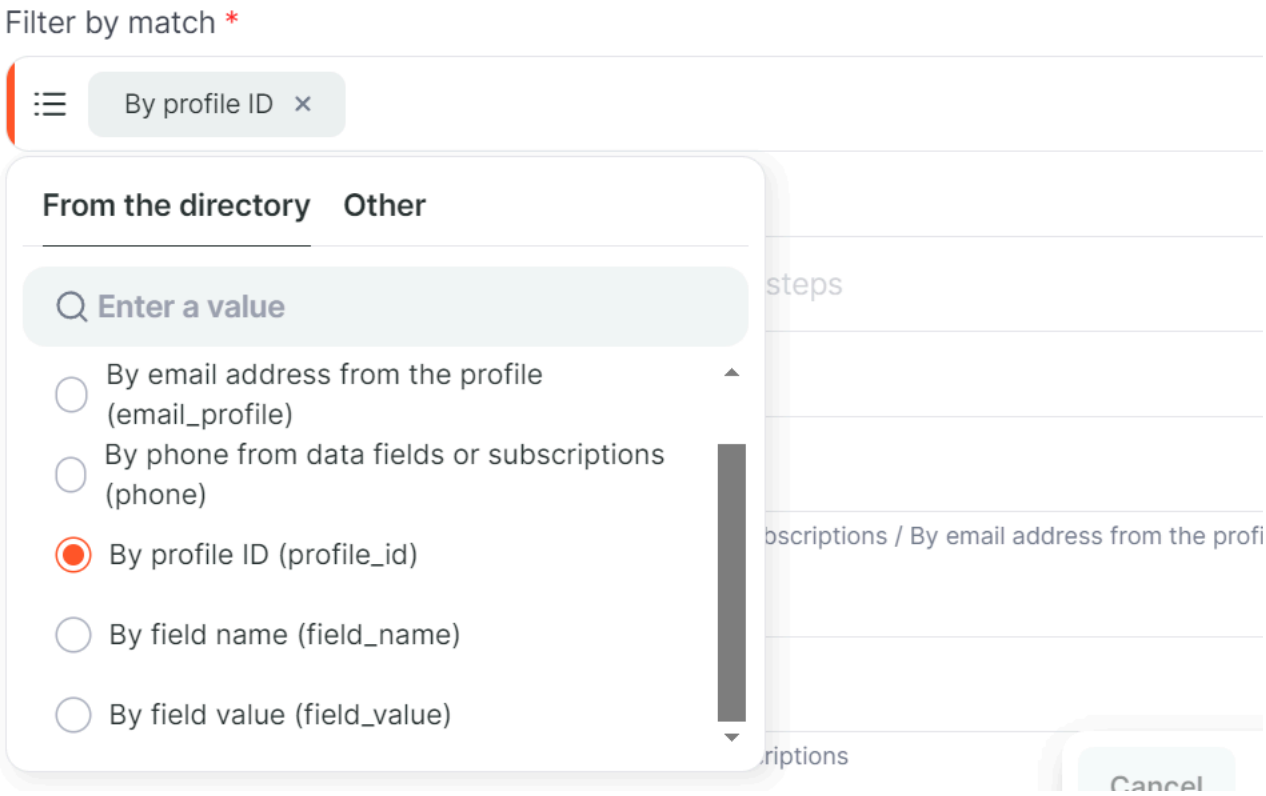
To import or update client data and transfer the profile to the scenario, select Simultaneous import and



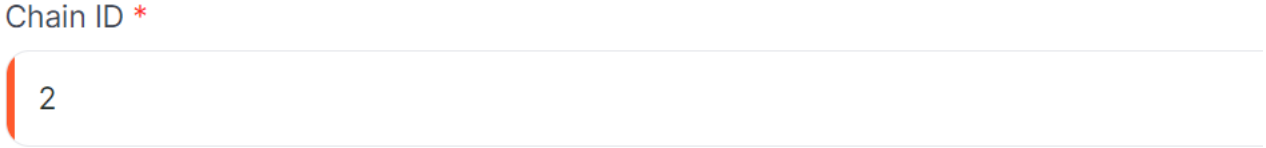
Connect an Altcraft account and specify the database for interaction. Click Add an action.

Configure the fields

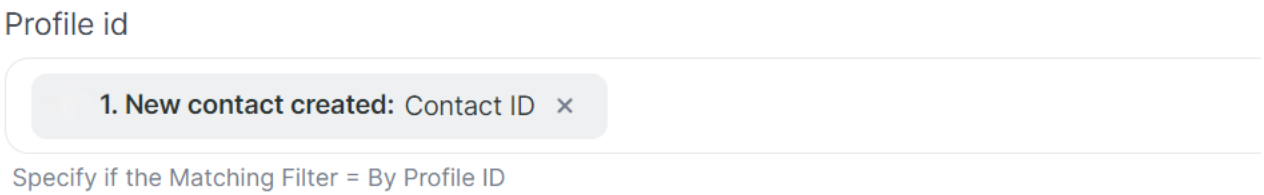
Specify the filter by which the profile will be searched in the database.



Specify the ID of the scenario to which the profile will be sent.



Depending on the selected filter for matching, specify the field by which the search will be performed.



Optionally, specify additional fields to match. After setting up, you can test or run the automations.

Working with the scenario on the platform

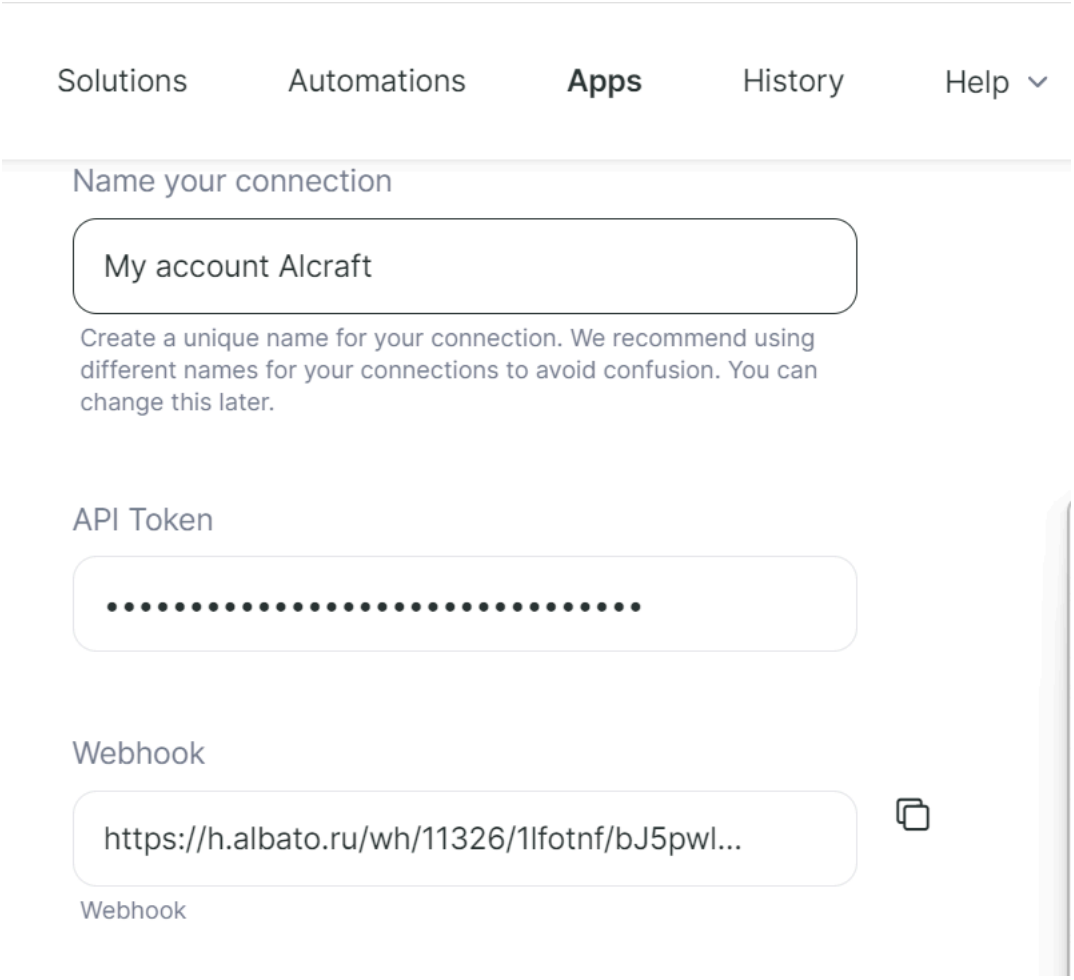
After creating a automations, customer data will be transferred via Albato to the platform when a trigg will be included in the previously prepared welcome scenario.
You can read more about how events are registered on the platform to return data to Albato in the nex

Transmitting event data

An event is an action related to the delivery of messages or the import of subscriber profiles, as well a Altcraft platform pixels on your website or on the app.
Event data can be transferred to various services using action hooks, including Albato.

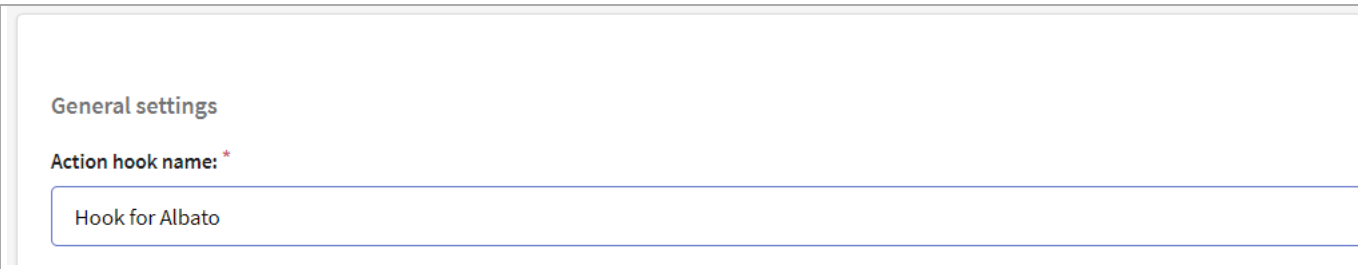
Setting up action hook on the platform

The main steps for configuring action hook are described in this article. On the URL field, specify the a —> Altcraft —> My Altcraft account



List of available events:

- **Mailing launched**
- **An SMS message has been sent for delivery to the subscriber.**
- **Not a unique click on a link in an Email message**
- **The SMS message has been delivered**
- **Click on the subscription confirmation link in the double opt-in message**
- **SMS message not delivered**
- **The subscriber followed the unsubscribe link in the email and was unsubscribed**
- **Push message sent for delivery to subscriber**
- **Not a unique opening of a Push message**
- **Push message delivered**
- **Not a unique click on a link in an SMS message.**
- **Not a unique click on a link in a Push message**
- **The promo code is activated**
- **Not a unique opening of an Email message**
- **Email message sent for delivery to subscriber**
- **A promo code is assigned to the subscriber**
- **The sender delivered the Email message to the subscriber**
- **AK-Pixel is open on the website**
- **The sender could not deliver the Email message to the subscriber**
- **You can activate event capture using the switch at the top of the screen:**

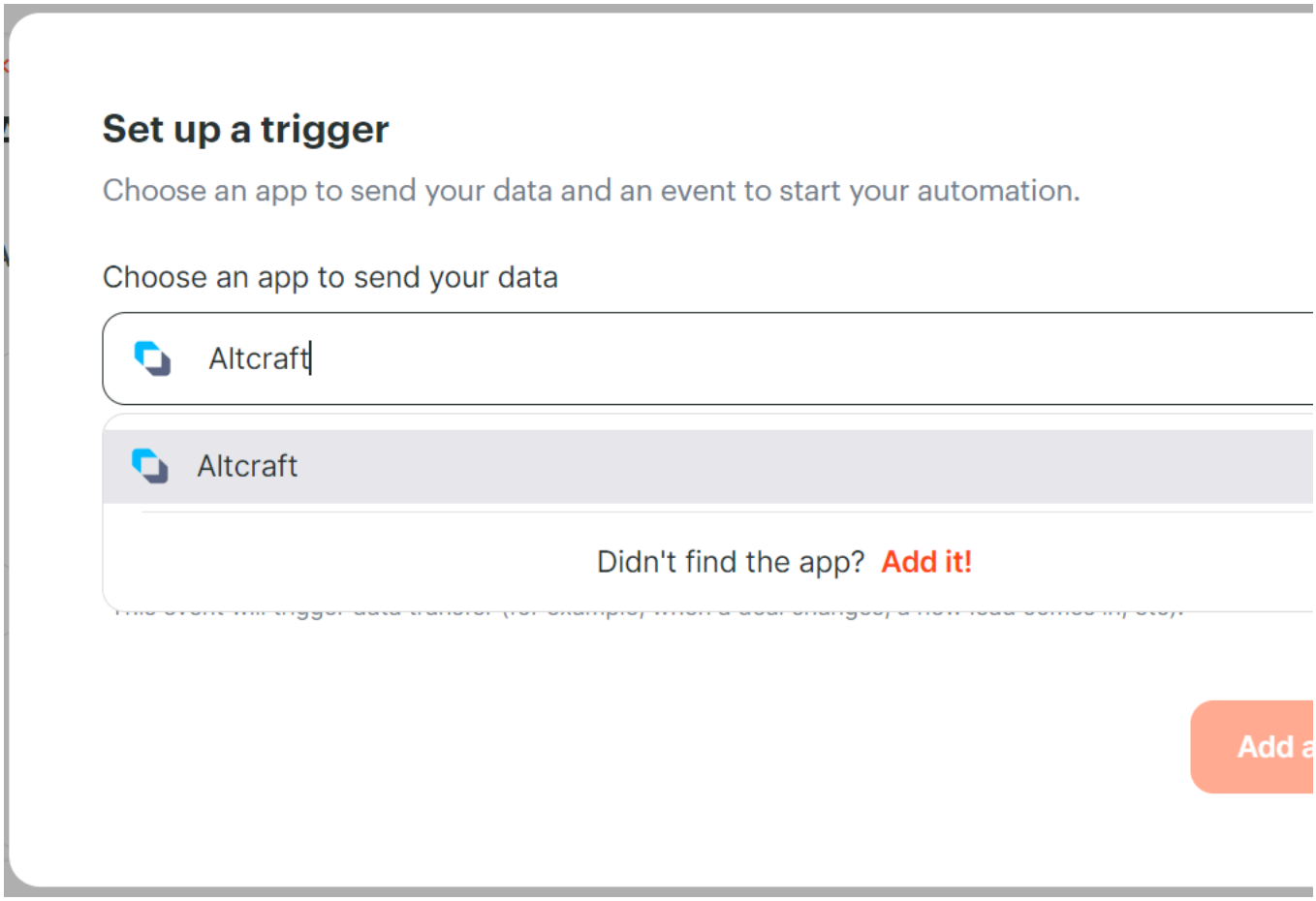


Setting up a automation in Albato

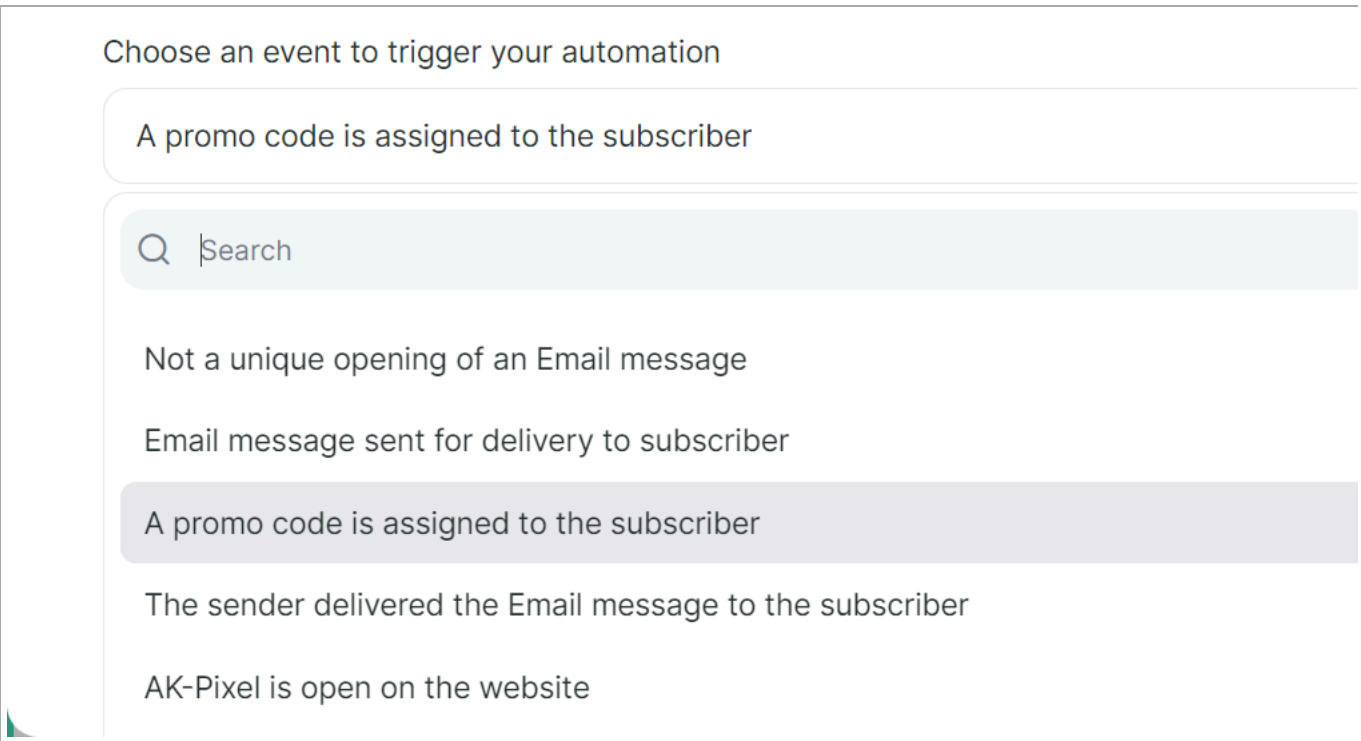
To set up a automations to transfer data from Altcraft to an external service, click Create new automa

Set up a trigger

Select Altcraft as the service that will transfer data to Albato



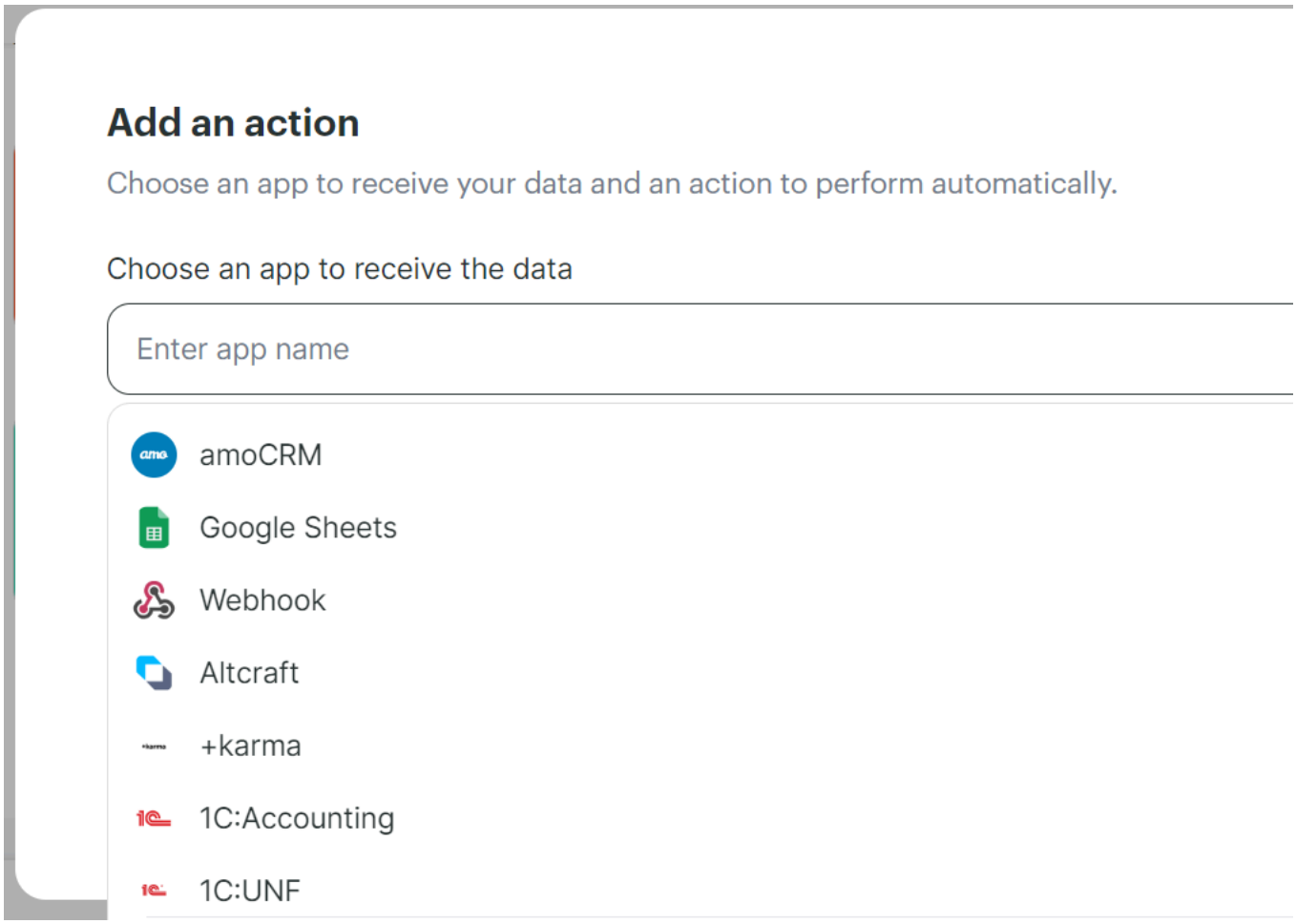
Select the event that will trigger the transfer of data to Albato:



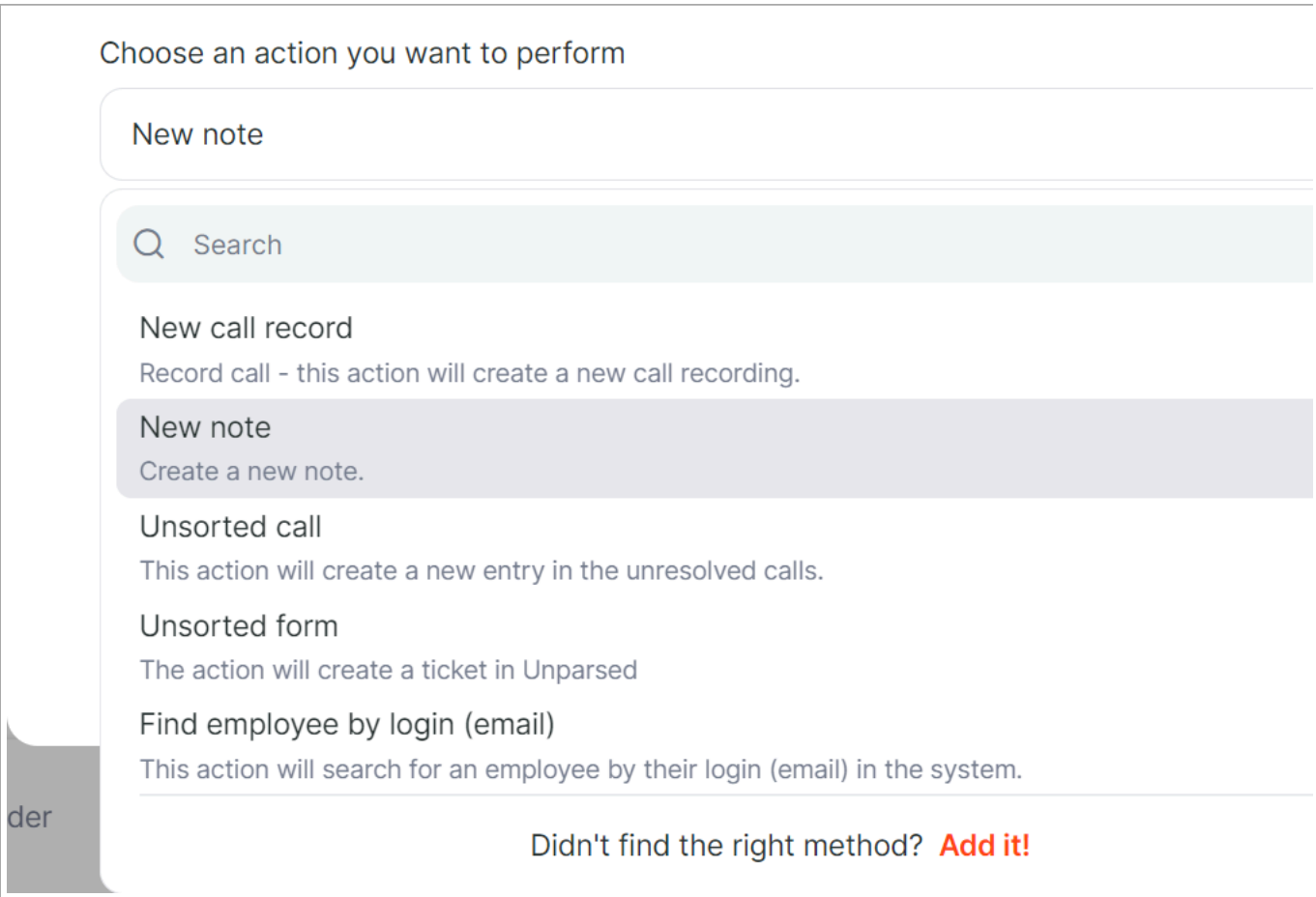
Connect your account.

Add the action that will occur after the trigger is started

Select the service where the data will be sent:



Assign an event to be performed:



Connect your account to Albato.

Click save and then configure the fields: add a “Contact”, the text “add a promo”. You can activate the necessary information.

Setting up a trigger mailing

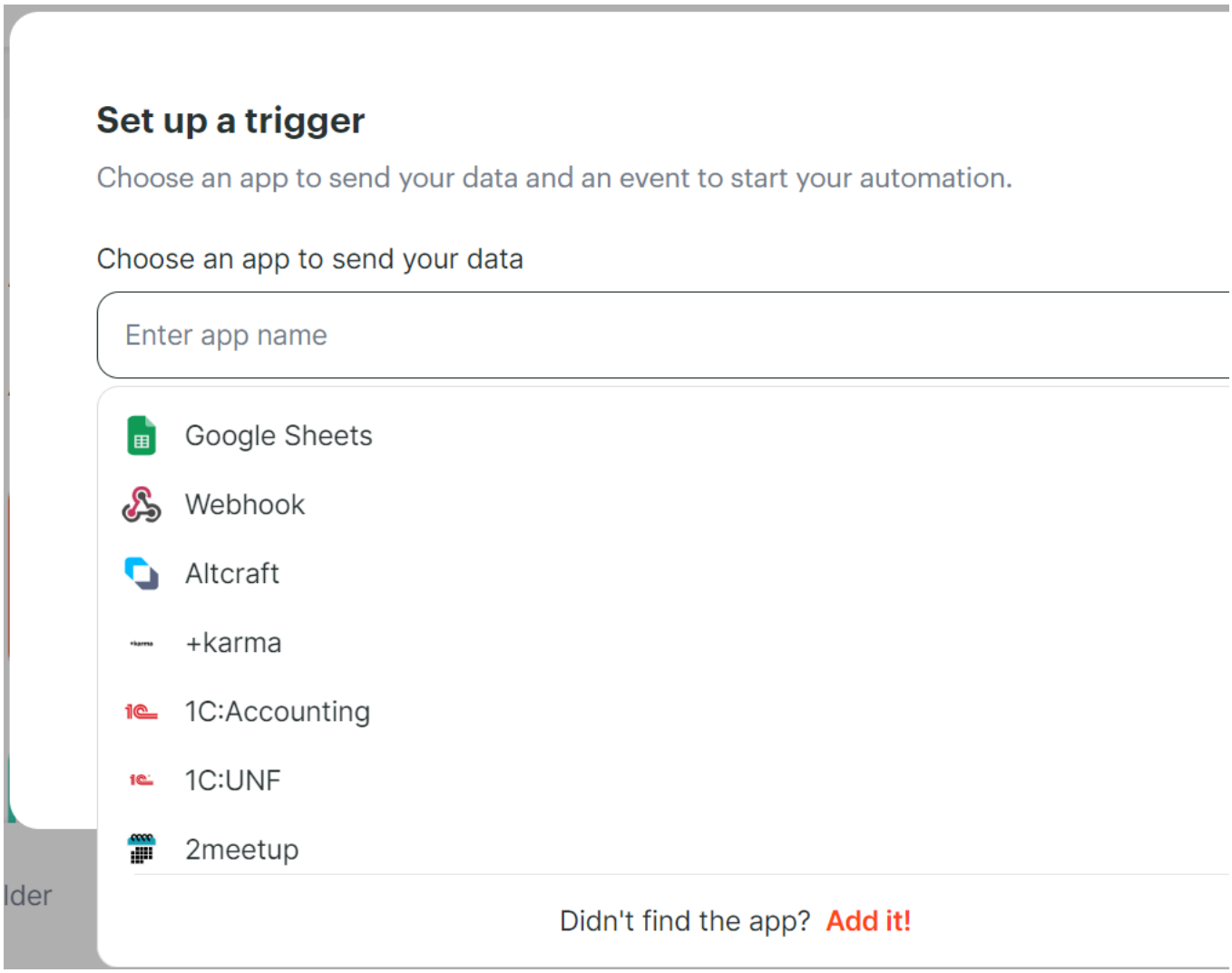
Using Albato, you can create a automation that will launch a trigger mailing on the platform.
Let's launche this type of mailing using the example of sending an invoice to a client.

Setting up automation in Albato

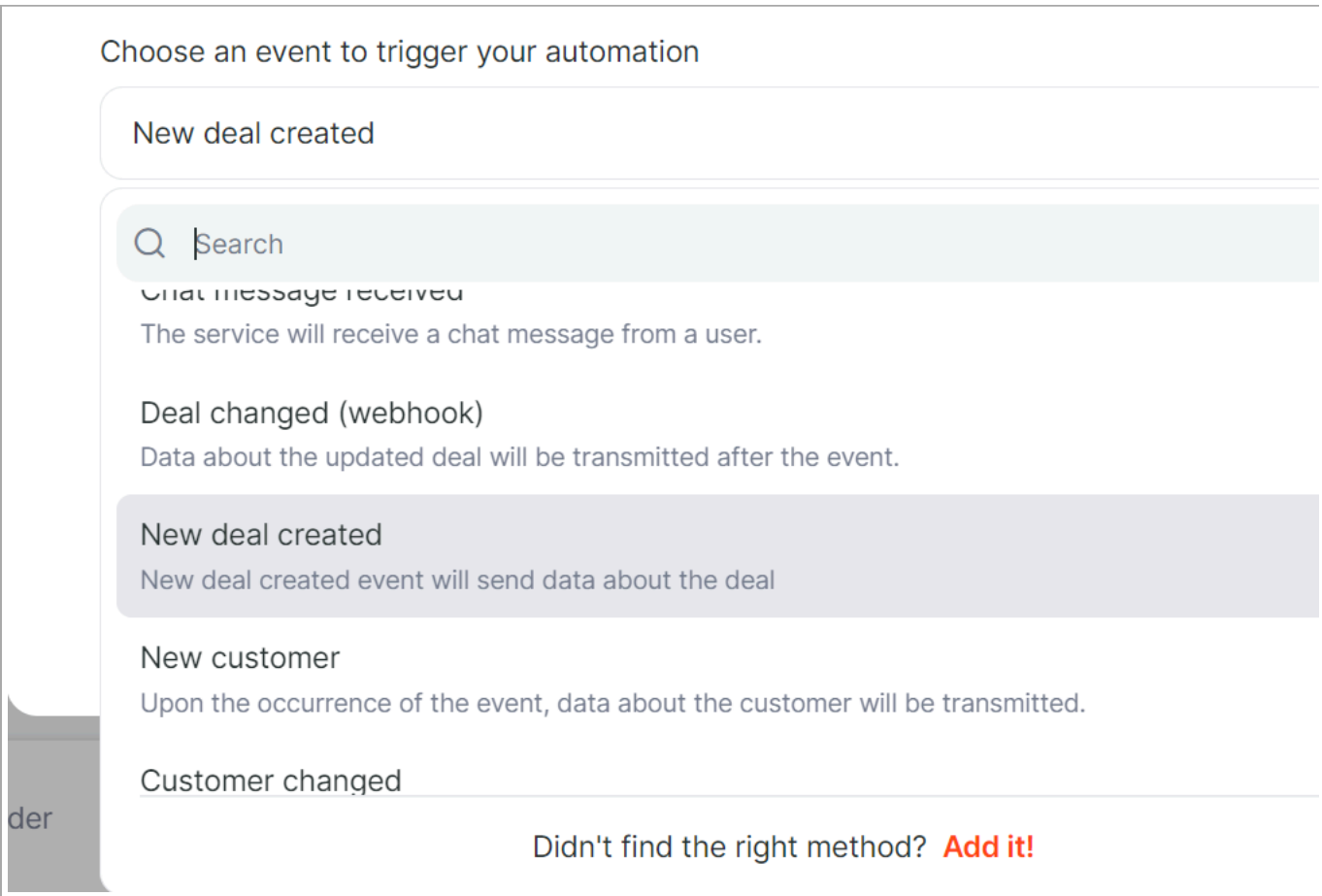
Set up a automation using this way External Service -> Albato -> Altcraft.

Add a trigger

Select the external service from which the data will be received:

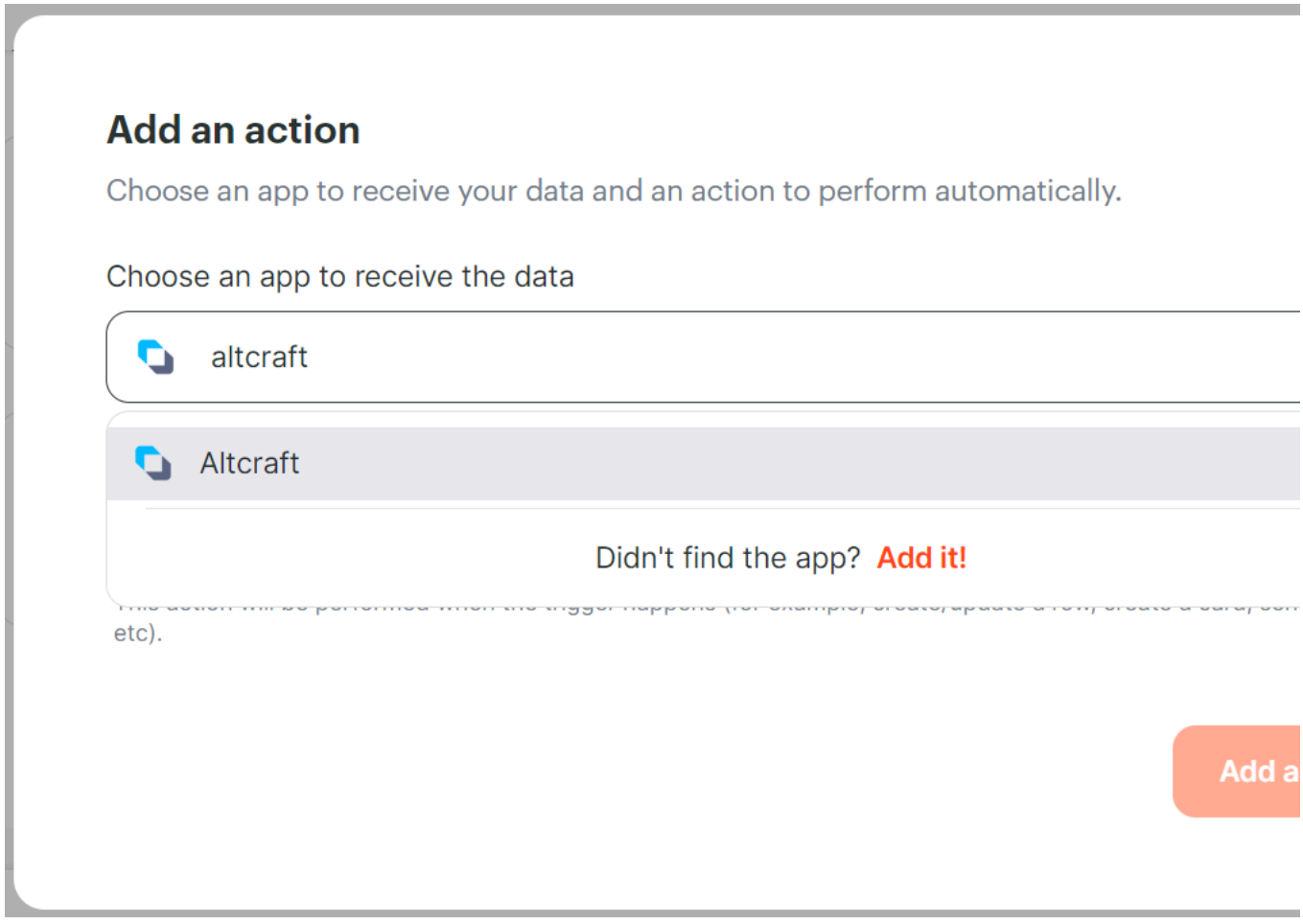


Select the event that will trigger the data transfer:

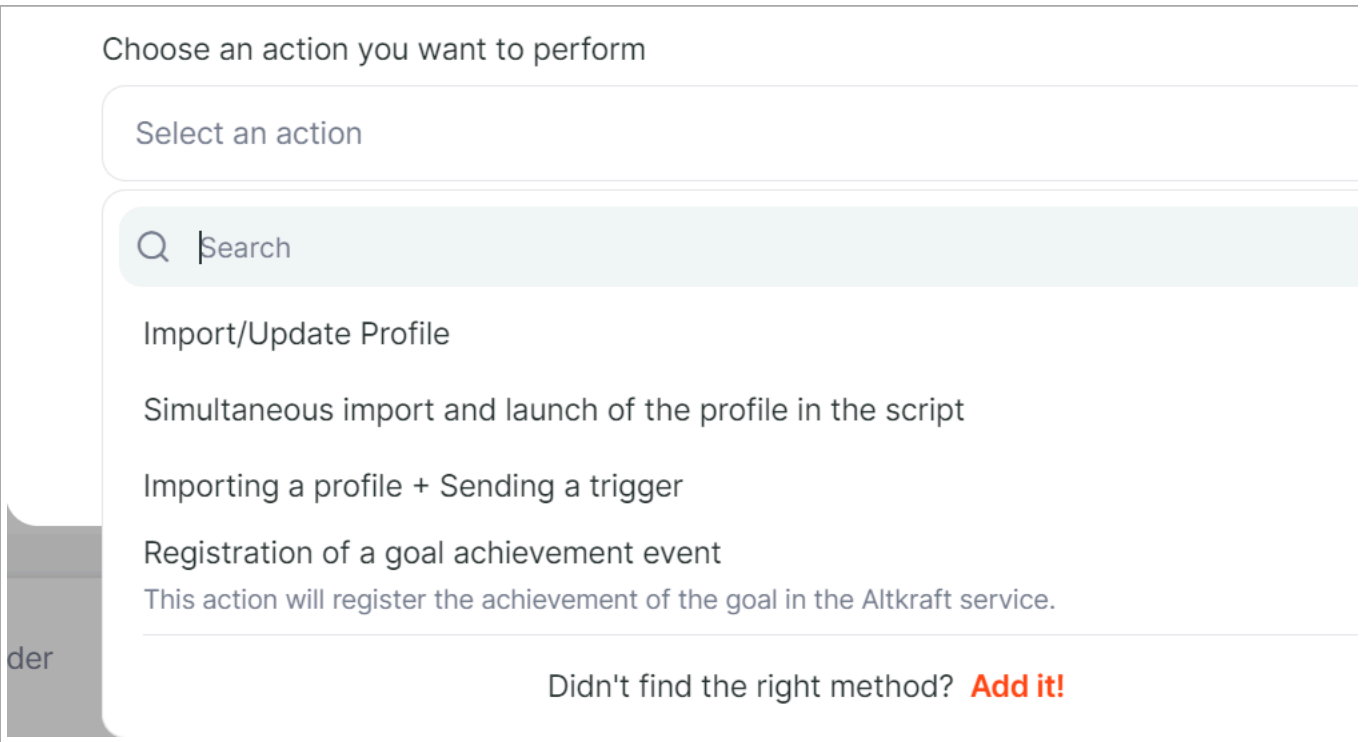


Add an action

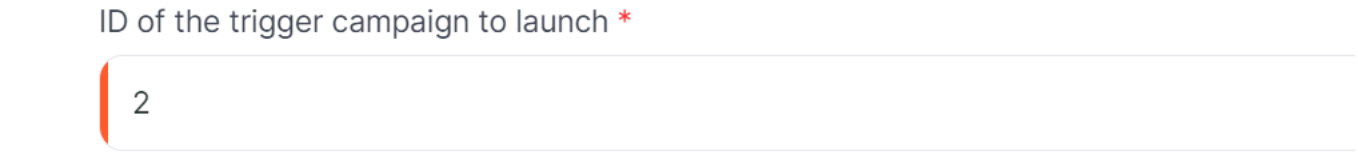
Specify Altcraft as a service for receiving data:



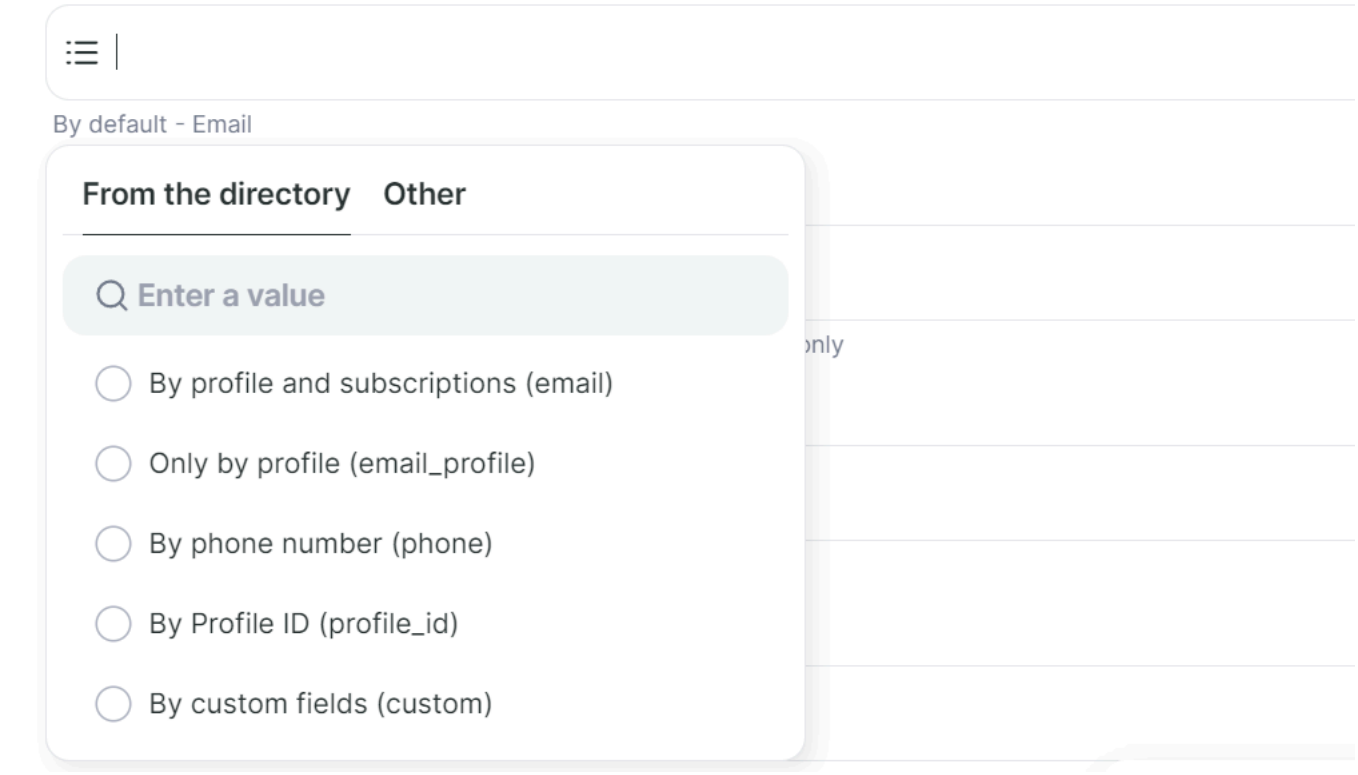
Assign an action that will take place after the event occurs. In our case, we need Importing a profile +



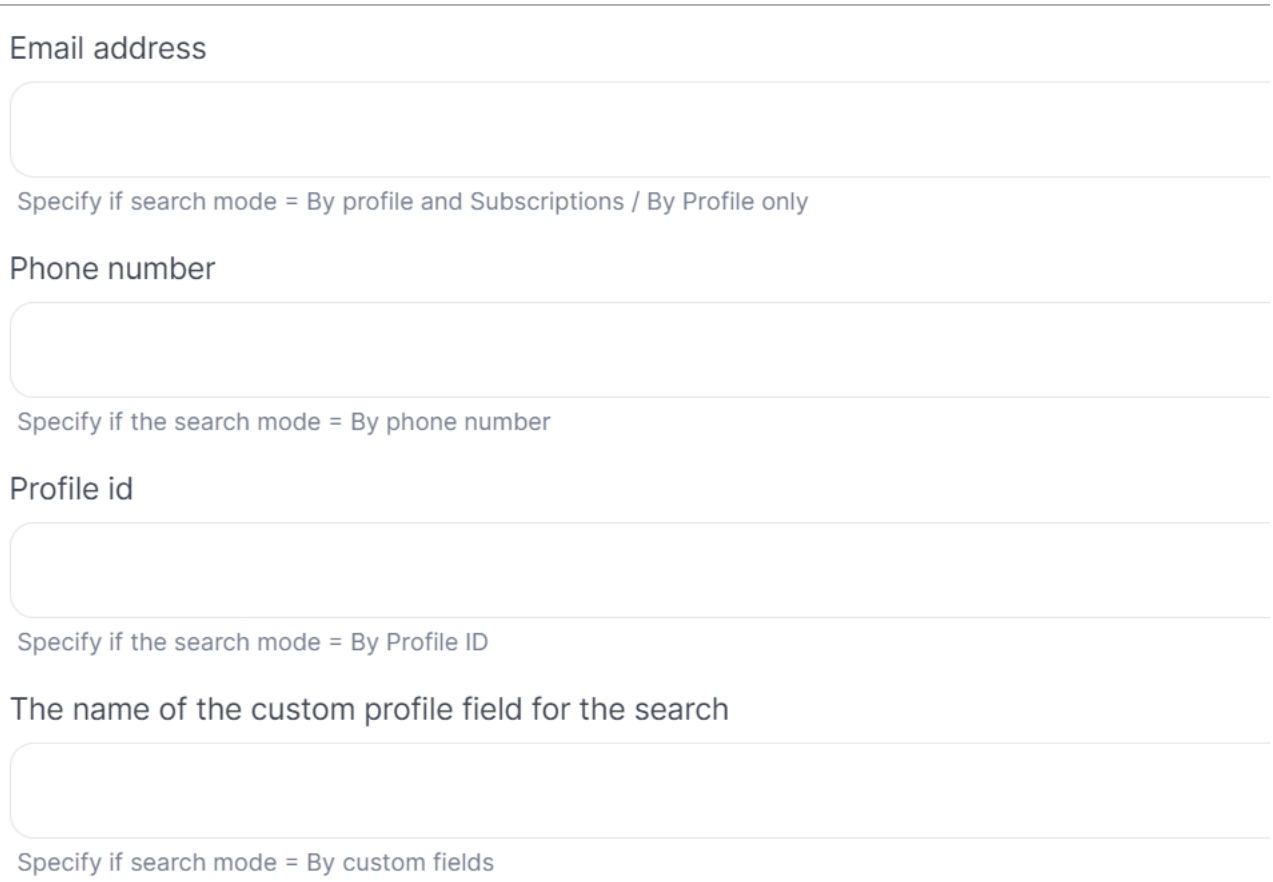
Connect your account and select the profile database.
Next, specify the ID of the trigger mailing that you would like to launch:



Select the parameter by which the subscriber will be located in the database:



Depending on the selected parameter, specify the search field:

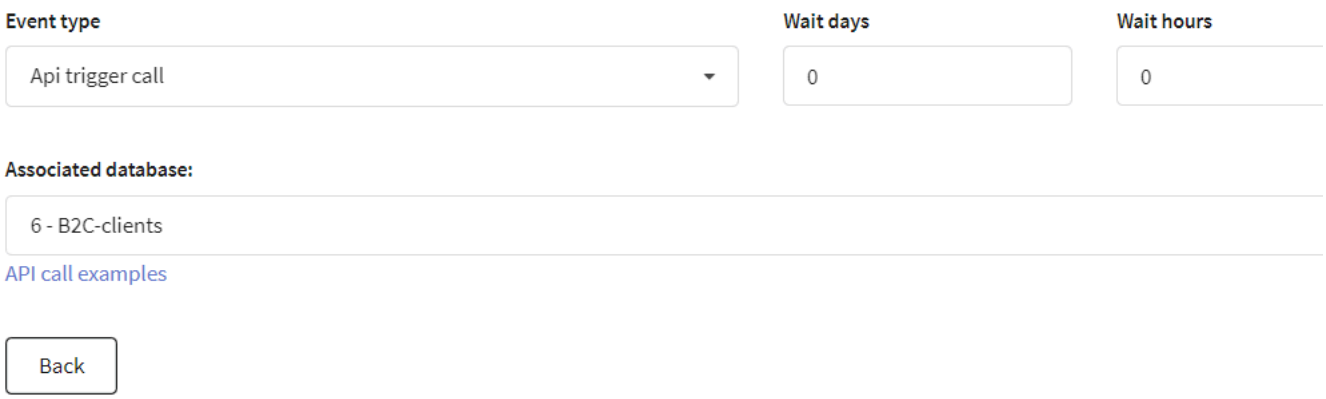


Optionally, specify the fields for matching.

Setting up a trigger mailing on the platform

First, you need to set up a [trigger mailing.]

In the Trigger section, specify Event type: API trigger call (accepts API calls from external services). W time is optional.



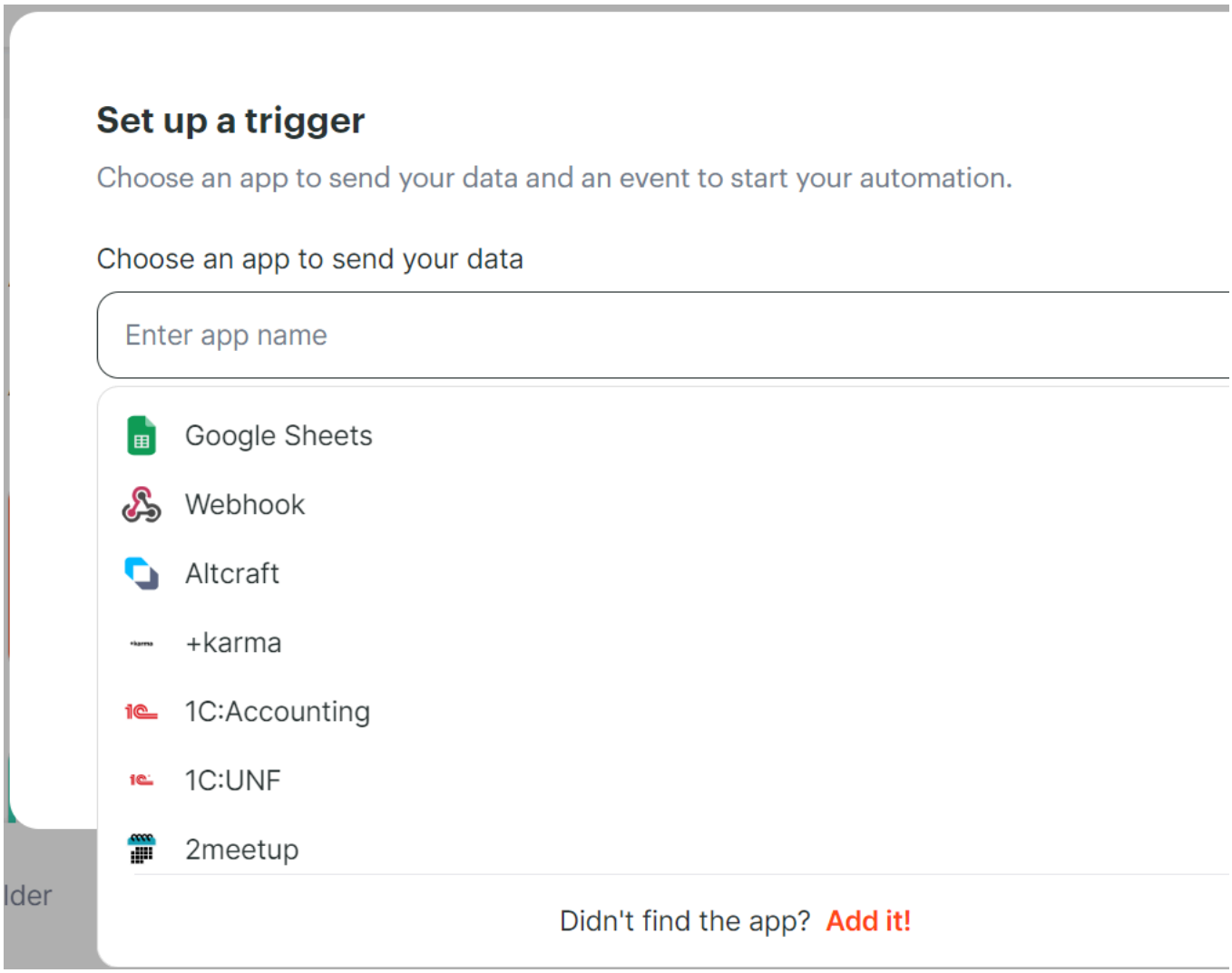
Next we have to save and launch the mailing.

Event registration

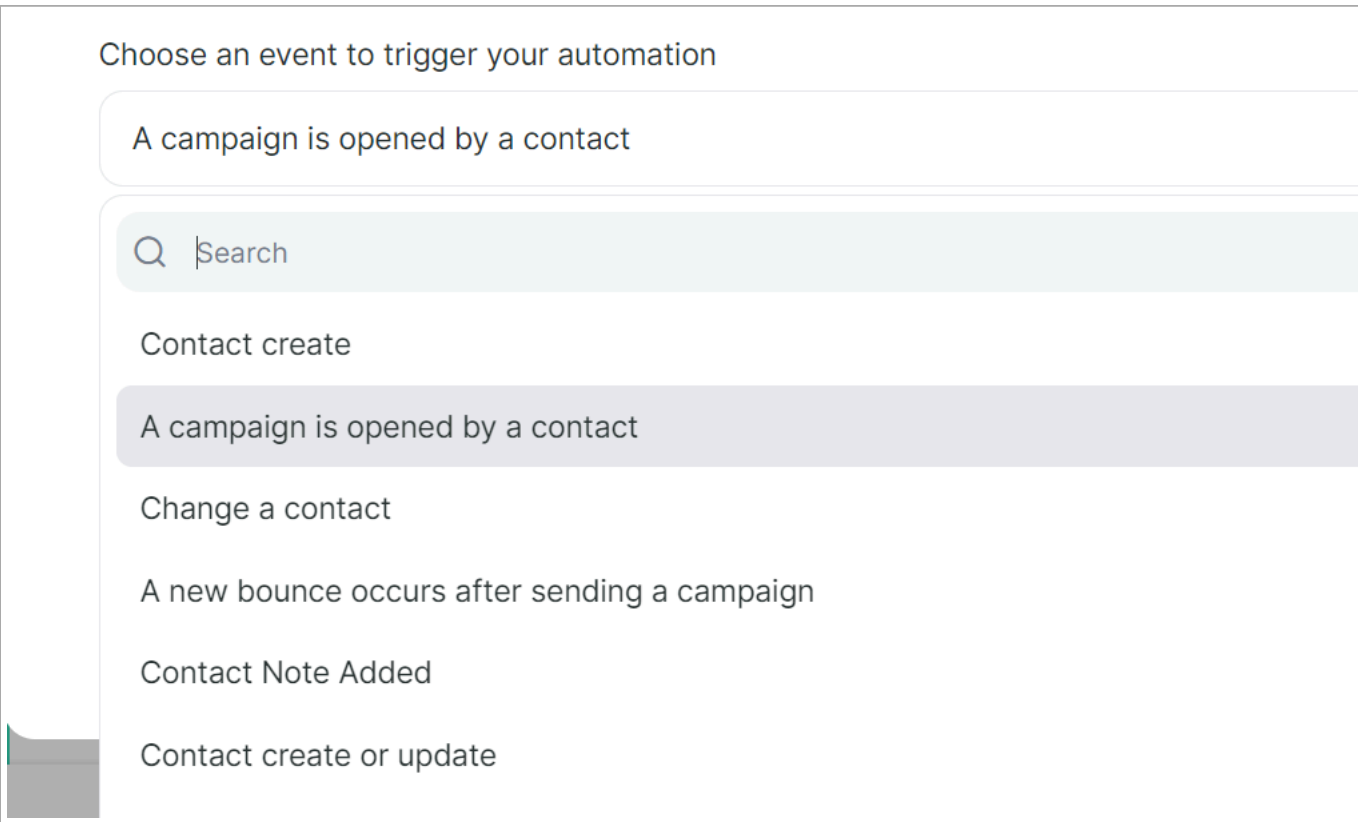
Albato allows you to transfer customer data to subscriber profile on the platform.
To do this, you need to set up a way External service —> Albato —> Altcraft

Set up the trigger

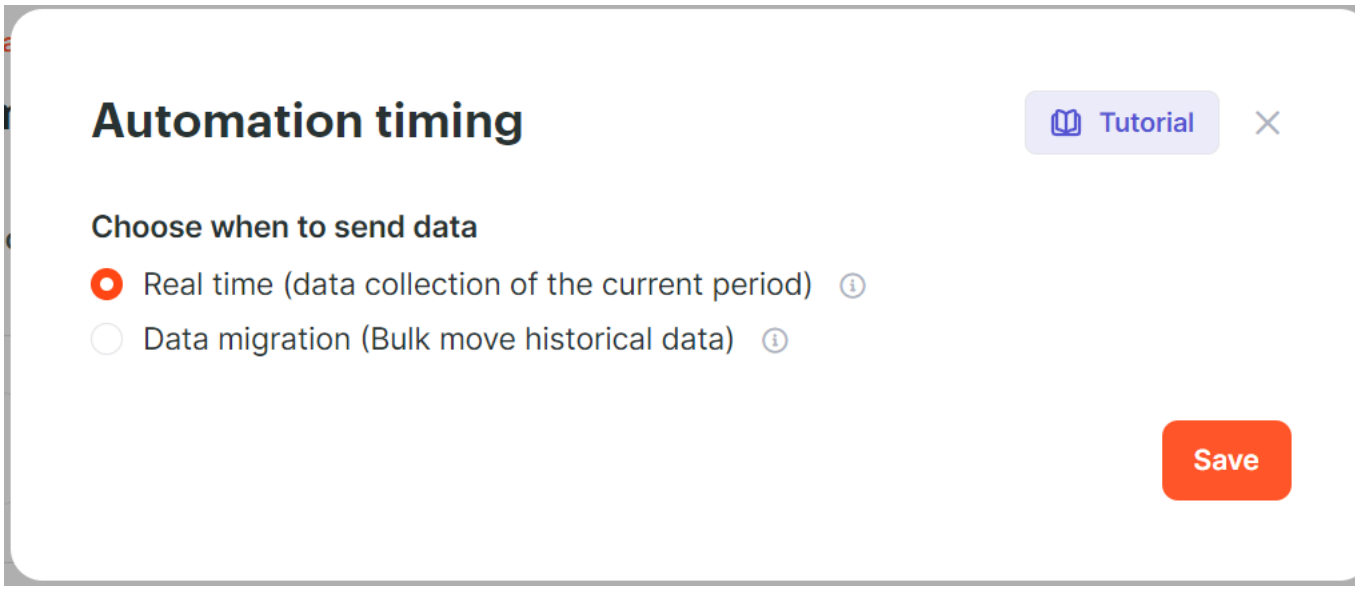
Specify the external service that will transfer data to the platform:



Select the event that will trigger the data transfer:

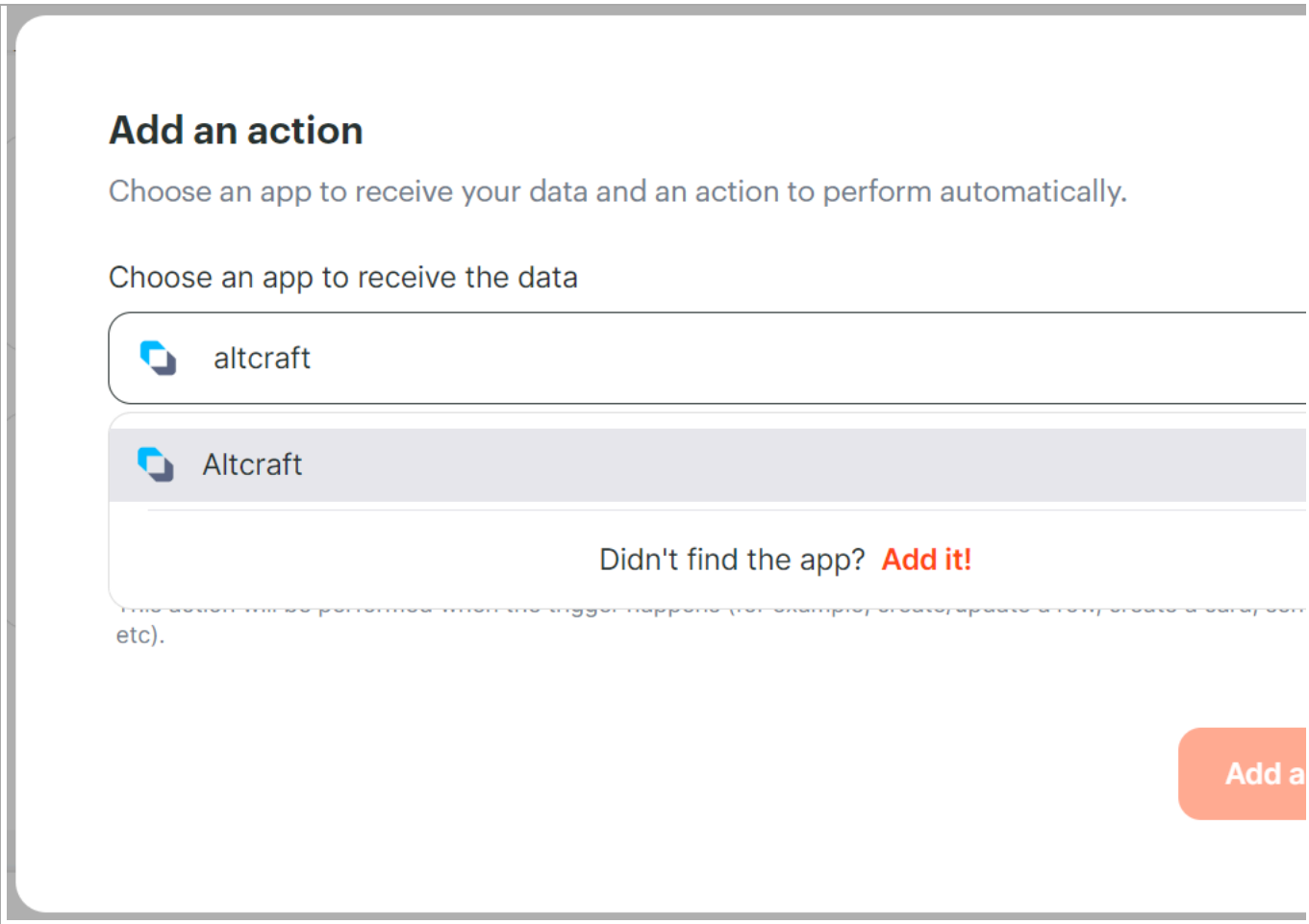


Select the mode of operation of the bundle: data can be sent to Altcraft in real time or specify for whic

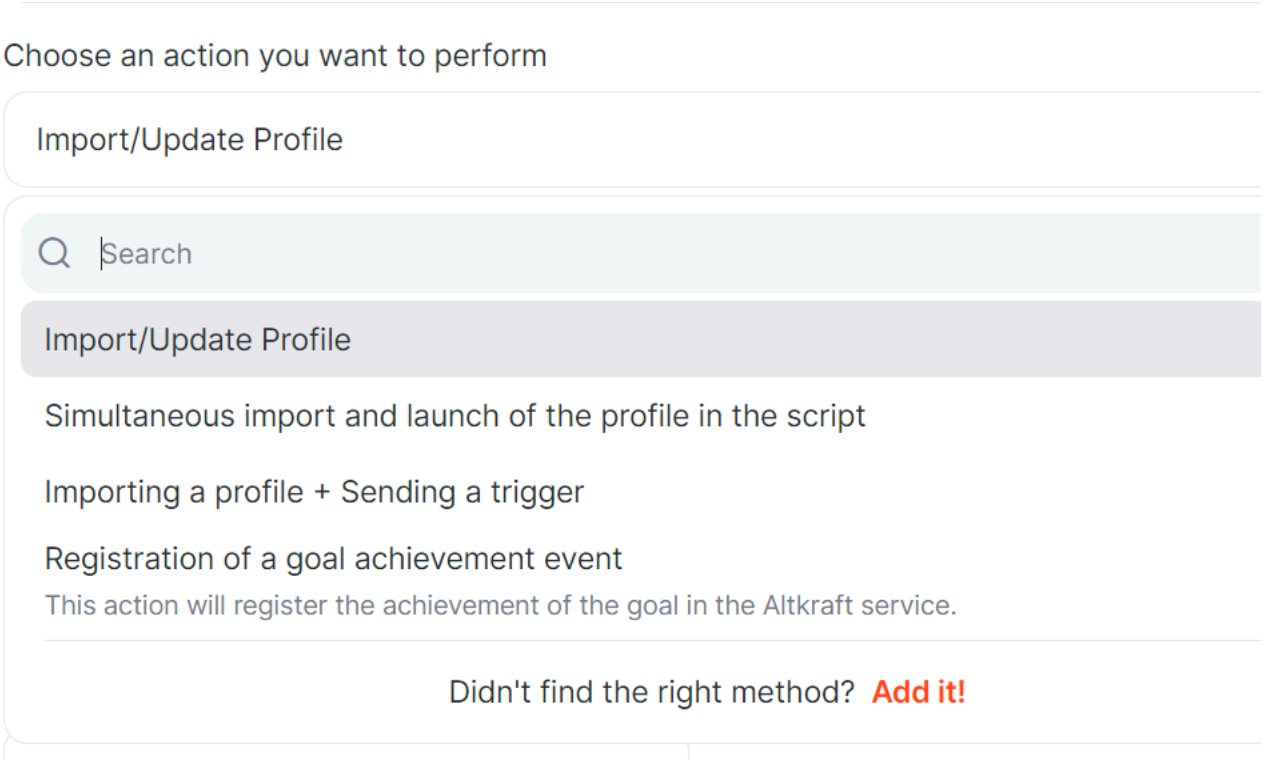


Set up the action

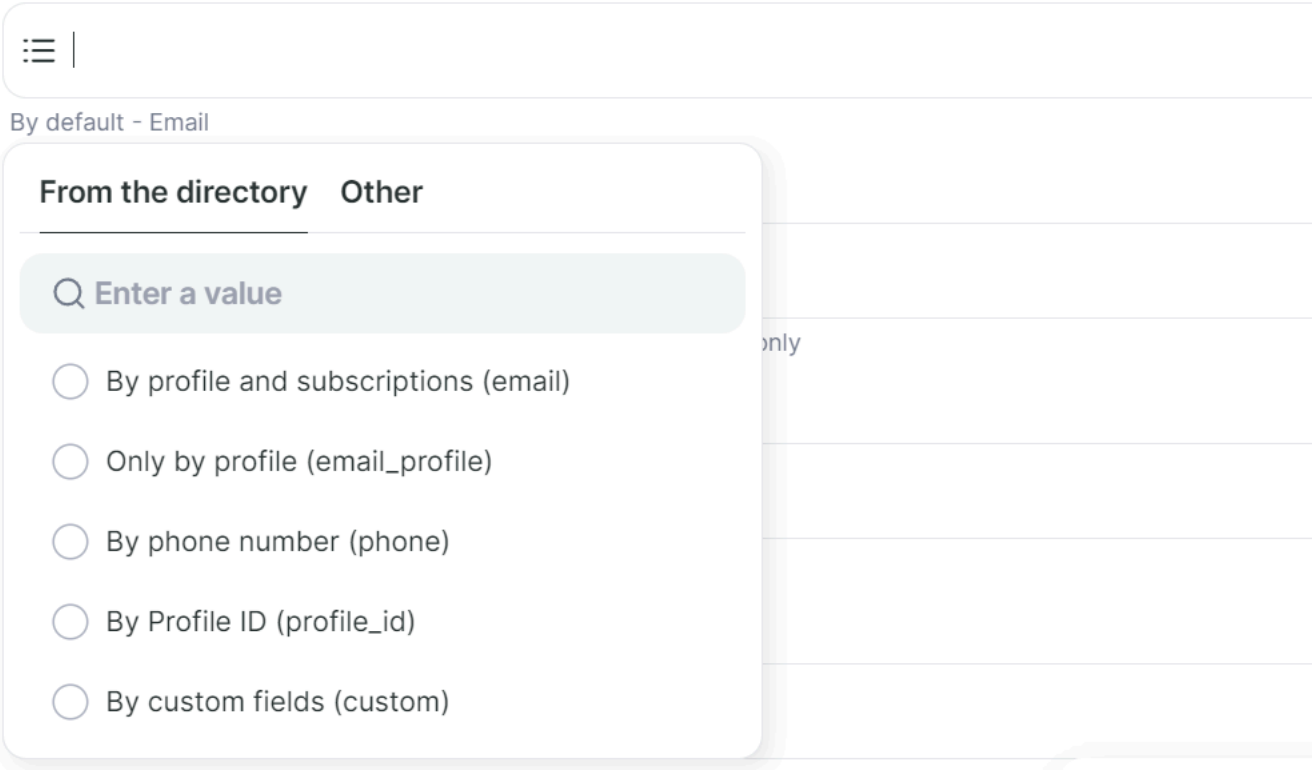
Specify Altcraft as a service for receiving data:



Assign an action that will take place after the event occurs. In order for the data to be imported into th Import/Update profile:



Adjust the filter accordingly:

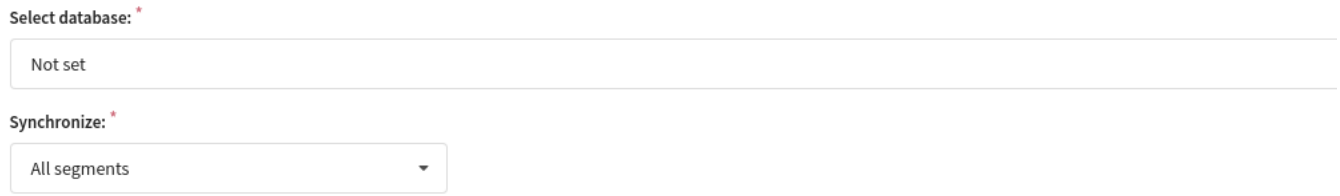


Optionally, set up additional fields.

After saving the bundle, when the trigger is started, the data will be transferred to the client's profile c

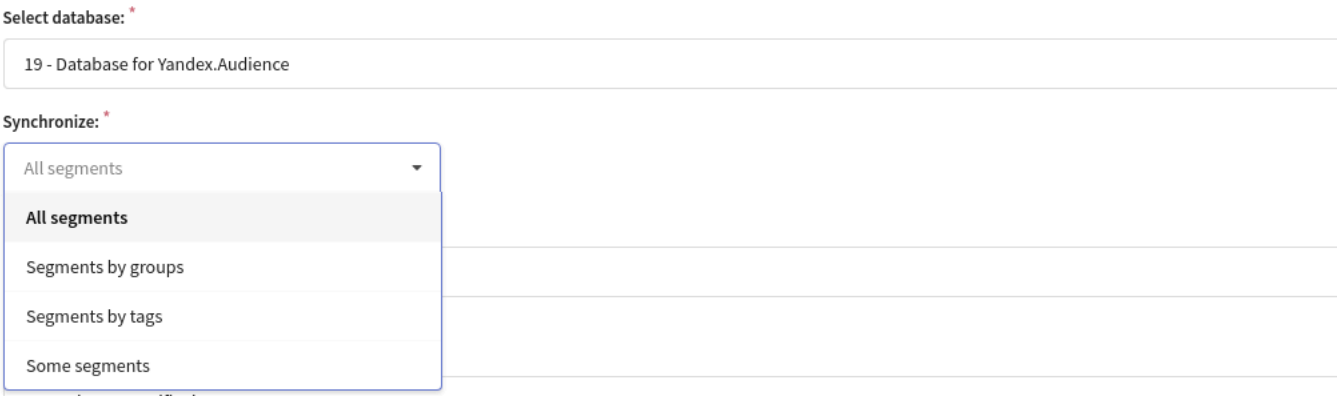
Integration scope

Integration scope is the scope within which the profiles whose data you want to export to external sys
the database of profiles and static segments.
Setting up the scope is required for all integrations that synchronize with static segments of the Altcra



Segments can be grouped as follows:

- **All segments — the data of all static segments associated with the selected database will be expo**
- **Segments by groups — the data of the segments for which a certain access group is set will be ex| from the menu on the right.**
- **Segments by tags — the data of the segments for which certain custom labels (tags) have been ad added in the main segment settings. Select the desired labels from the menu on the right.**
- **Some segments — in the menu on the right, manually select the required segments.**



Data Transmitted During Synchronization

To synchronize segments based on audiences from external sources (Google Audience, Yandex Audien data is transmitted to the corresponding systems.

This article provides a list of data transmitted during audience synchronization. This information may b request from their information security department.

Yandex.Audiences™

Authorization	
Application ID	**Client ID from Yandex.OAuth**
Application Secret	**Client Secret from Yandex.OAuth**
Transmitted Data	
email	**Email address. Transmitted in hashed form (MD5)**
phone	**Phone number. Transmitted in hashed form (MD5)**
externalIDValue	**Segment ID in Yandex.Audiences™. Transmitted in hashed form (MD5)**
PushData.SubscriptionID	**Push subscription ID. Transmitted in hashed form (MD5)**

TIP: Data Hashing

Data hashing can be disabled by setting the configuration parameter `YAAUDIENCES_SYNC_DO_HASHING` **to** `false`

Google Audience

Authorization	
Application ID	**Client ID**
Secret Key	**Client secret**
Token	**Developer token**
Transmitted Data	
email	**Email. Transmitted in hashed form (SHA256)**
phone	**Phone. Transmitted in hashed form (SHA256)**
Advertising Identifiers	**IDFA / Google Advertising ID and other advertising IDs**
Segment Data	**Segment ID and segment name**

VK Ads™

Authorization	
Client Identifier	**client_id**
Secret Key	**client_secret**
Transmitted Data	
vk	**VK user identifier**
email	**Email address. Transmitted in hashed form (MD5)**
phone	**Phone number. Transmitted in hashed form (MD5)**
ok	**Odnoklassniki user identifier**
advertising_id	**Google Advertising ID**
idfa	**Apple mobile device identifier**

Facebook Ads Manager*™

Authorization	
Application ID	**App ID**
Application Secret Key	**App Secret**
Transmitted Data	
em	**Email. Transmitted in hashed form (SHA256) after normalization**
ph	**Phone. Transmitted in hashed form (SHA256) after normalization**
ct	**City. Transmitted in hashed form (SHA256) after normalization**
country	**Country. Transmitted in hashed form (SHA256) (ISO 3166-1 alpha-2)**
fn	**First name. Transmitted in hashed form (SHA256) after normalization**
ln	**Last name. Transmitted in hashed form (SHA256) after normalization**
ge	**Gender. Transmitted in hashed form (SHA256), `m`/`f`**
db	**Date of birth. Transmitted in hashed form (SHA256) in YYYYMMDD format**
external_id	**Transmitted without hashing (recommended to send in original format)**

Meta, the organization that owns Instagram, Facebook and WhatsApp, is recognized as extremist and

API-requests

Most users of the Altcraft platform interact with it through a web interface in a browser. However, you customization and automation of processes.

API (Application Programming Interface) is a set of rules that allows programs to communicate with ea automate many tasks and integrate Altcraft platform features into your website or other systems.

As a rule, developers are in charge of writing API requests. However, understanding the principles of A greatly simplify interaction with developers: the marketer will be able to set tasks for them more clear be used to implement it.

In this section, you will find guides on how to create basic API requests such as importing a profile, lau

Basic API requests:

Import or update a profile

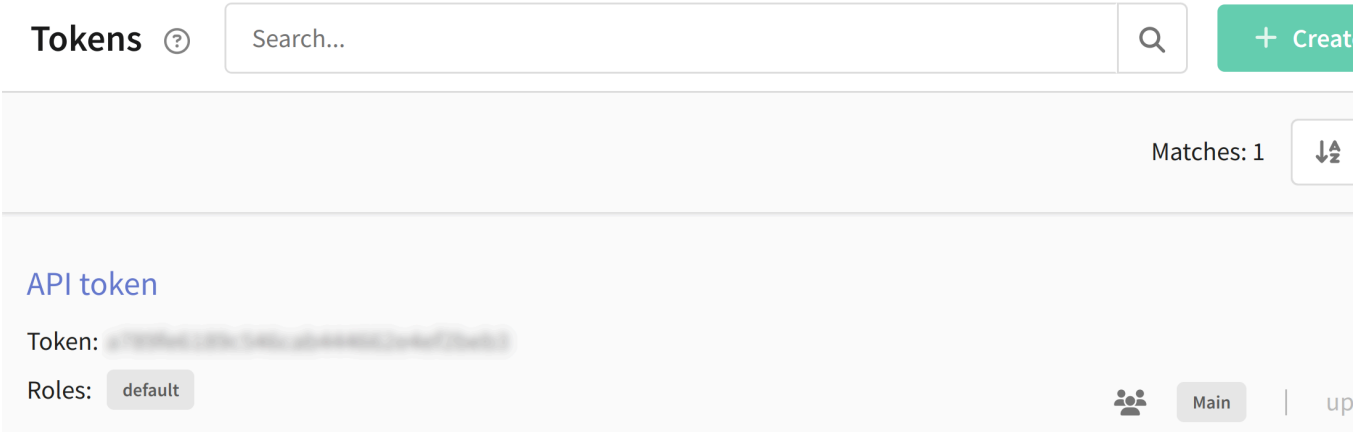
This article will show you how to write and submit API requests to perform the following tasks:

- **Adding a profile to the database**
- **Updating profile data**
- **Subscribing a profile to different channels**

Getting an API token

API token is a key to access the platform via API. It is used in any API request.

You can find the token in Altcraft platform settings, in the Tokens section. If you don't have a token yet corresponding button:



CAUTION

Only master users can create an API token. If you do not have rights to create a token, contact a user

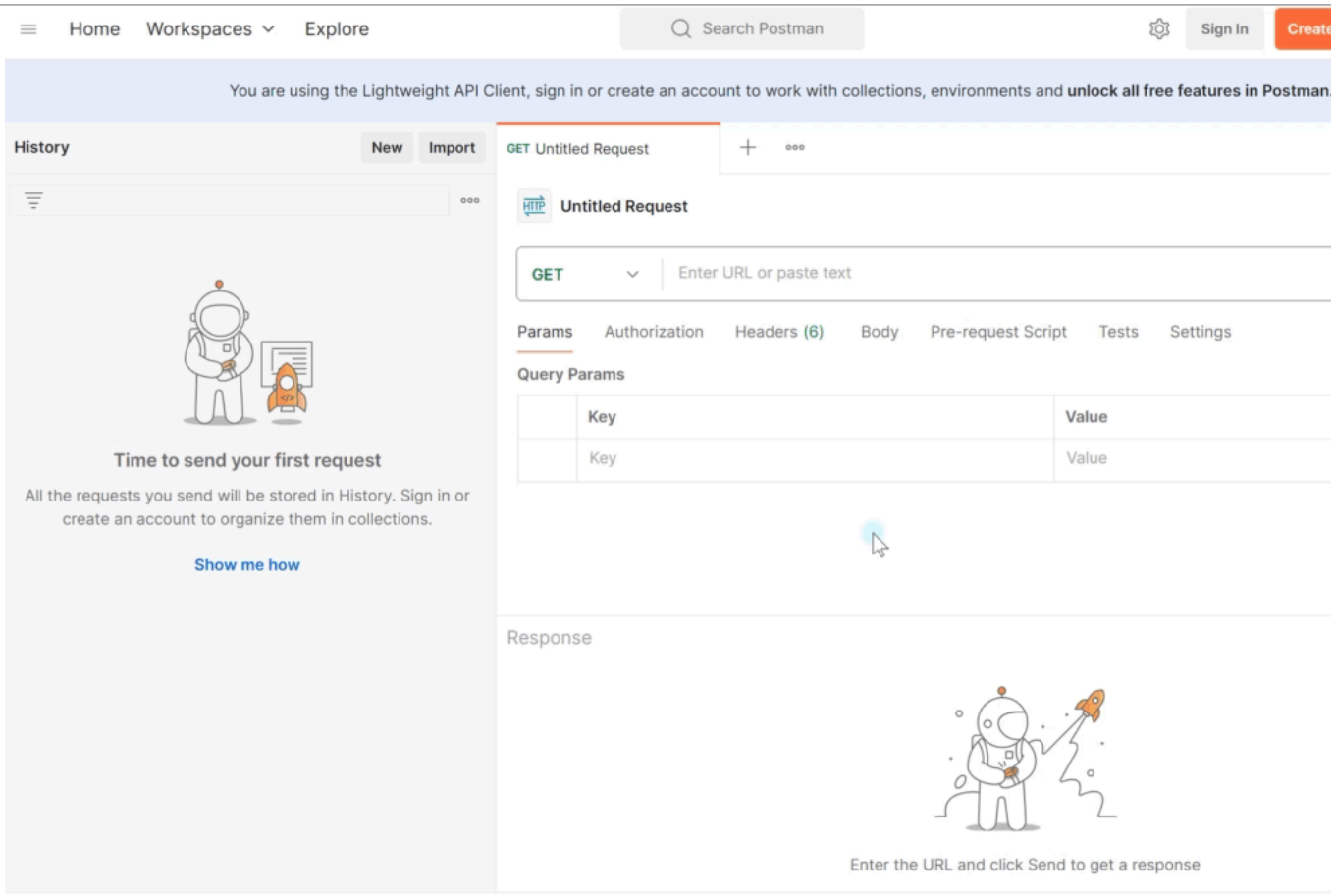
Installing and running Postman

To work with API requests, you need an application that allows you to send and receive them.

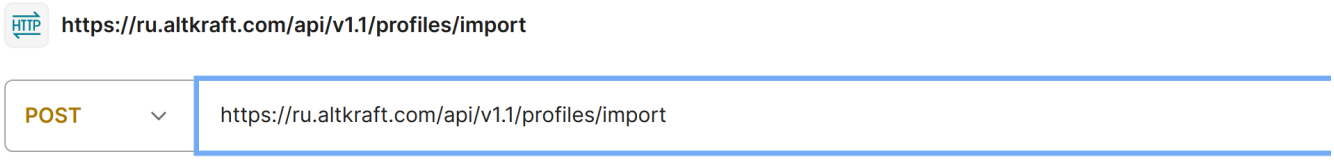
1. Download and install the Postman application or open its web version.

2. Select the POST method for your request.

3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



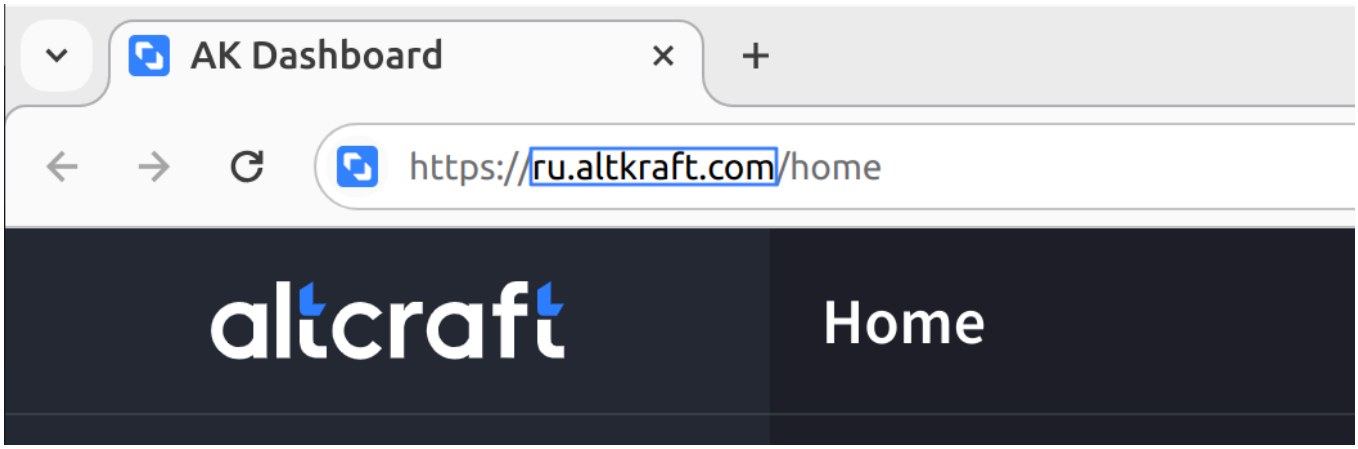
Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example URL would be:

`https://example.com/api/v1.1/profiles/import`.

Instead of `example.com` you need to specify the address where the platform is located. You can get it dir



Creating a request

Adding a new profile

Let's move on to writing the query. Suppose we want to manually add a customer profile with e-mail `i.`
Let's make a query:

```
{
     "token": "abcdefghklmnopqrstzxc",
     "db_id": 5,
     "matching": "email",
     "email": "i.petrov@example.com",
     "data": {
        "email": "i.petrov@example.com"
    }
}
```

- ****token ** - API token. You can read more about where to find it at the beginning of the article.**
- ****db_id ** - number of the database to which the profile is added. This number is indicated next to the right of the # sign.**
- ****data ** - the object that contains the profile data. In the example above, we only pass the client's**
- ****matching ** and **email ** - parameters by which the query will determine whether this profile e the email address is used to search for the profile. If the profile with this address already exists in its data, if not - it will create a new profile. The "matching" parameter can take different values. Ho is not specified, the profile will be searched by email.**

More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several sea platform, for example, the platform can search for a profile by email address, phone number, system ID additional field in the database.

The main thing is that the field by which you launch a search in the database is unique for the profile, i be the same for different profiles in the same database.

If you want to find a profile by its email, then `matching` will look like this:

```
{
     "matching": "email",
     "email": "example@example.com"
}
```

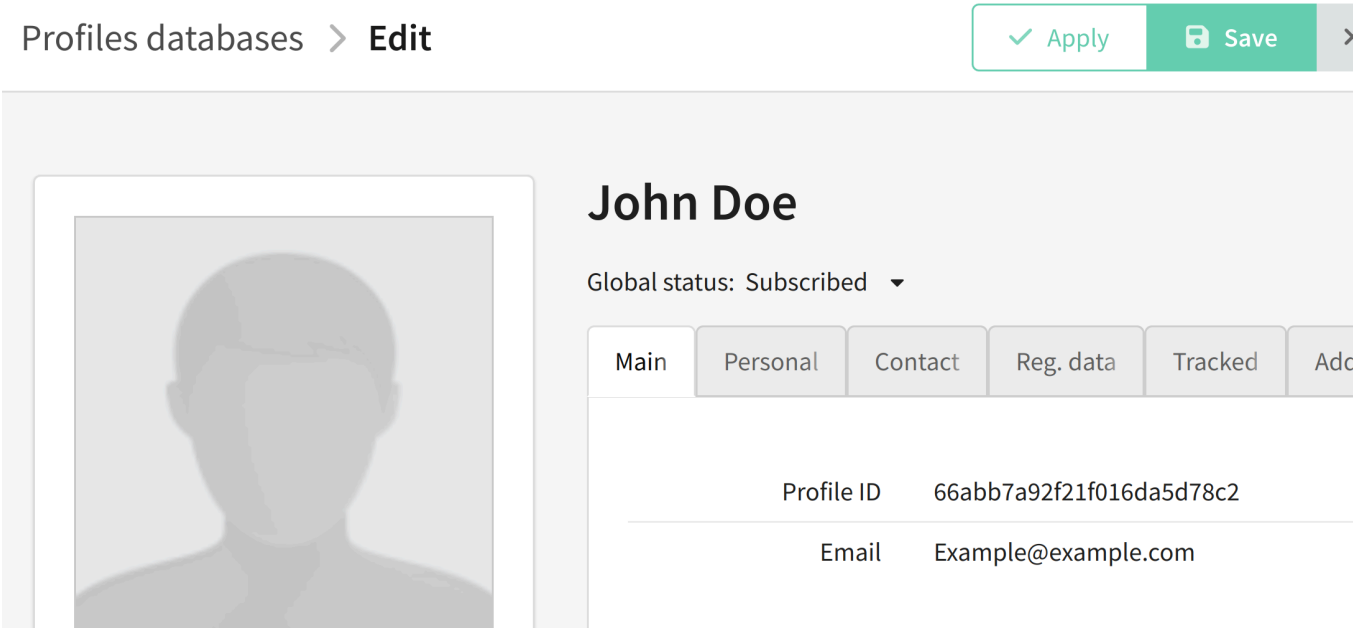
Search for a profile by phone::

```
{
     "matching": "phone",
     "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
     "matching": "profile_id",
     "profile_id": "012452gf62fcd71fjh8ec0"
}
```

The profile ID can be found here:



If you use a custom field to search for a profile, then your query will look like this:

```
{
     "matching": "custom",
     "field_name": "CRM_ID",
     "field_value": "12345"
}
```

where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

Updating profile data

Let’s assume that we have new data about the client: his name is Ivan Petrov, born 15.01.1990, lives in +79000000000. Now we need to update the profile and add this information. The query will look like thi

```
{
     "token": "abcdefghklmnopqrstzxc",
     "db_id": 5,
     "matching": "email",
     "email": "i.petrov@example.com",
     "data": {
        "email": "i.petrov@example.com",
        "_fname": "Ivan",
        "_lname": "Petrov",
        "_sex": "male",
        "_bdate": "1990-01-15T12:00:00Z",
        "_city": "Moscow",
        "phones": ["+79000000000"]
    }
}
```

We have extended the `"data"` parameter with the following data:

- **_fname - first name**
- **_lname - last name**
- **_bdate — date of birth in the format “1990-01-15T12:00:00Z”**
- **_sex - gender, “male” for men, “female” for women**
- **_city - city**
- **phones — phone number.**

Adding subscriptions to a profile

We have updated the profile and now we need to subscribe it to email, sms and push channels in resou the `subsriptions` array to `data`:

```
{
    "token": "abcdefghklmnopqrstzxc",
    "db_id": 5,
    "matching": "email",
    "email": "i.petrov@example.com",
    "data": {
      "email": "i.petrov@example.com",
      "_fname": "Ivan",
      "_lname": "Petrov",
      "_sex": "male",
      "_bdate": "1990-01-15T12:00:00Z",
      "_city": "Moscow",
      "phones": ["+79000000000"],
      "subscriptions": [
            {
                "channel": "email",
                "email": "i.petrov@example.com",
                "resource_id": 1
            },
            {
                "channel": "sms",
                "phone": "+79000000000",
                "resource_id": 1
            },
            {
                "channel": "push",
                "provider": "Firefox",
                "resource_id": 1,
                "subscription_id":"abcdefghijklmnqrstuvwxyz"
            }
        ]
    }
}
```

For each communication channel, the following parameters must be specified:

- **channel — type of channel, for example, email, sms, push**

- **resource_id is the resource ID that you must configure before. Also, depending on the channel you provide certain contact information. For example, for an email channel it is an email address, for a and for a push channel you need to specify the provider (`provider`) and the push token (`subscription_`**

TIP

Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period a request. Likewise, the query will not work if you do not close any quotes or parentheses.

Sending a request

To send a request, click Send in Postman. If the request is successful, the following message will appe

Pretty Raw Preview Visualize JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "profile_id": "66267f05ecc3993fba680df7"
}
```

If something went wrong, the error code and its description will be indicated in the reply message:

Pretty Raw Preview Visualize JSON

```
{
    "error": 400,
    "error_text": "Invalid request"
}
```

If you are unable to resolve the error yourself, please forward the error code and description to Altcraf

TIP

This article describes only some of the possible parameters for this request. The remaining paramete

Trigger mailing launch

In this article, we will show you how to run a trigger mailing through the API. Let's consider two varian

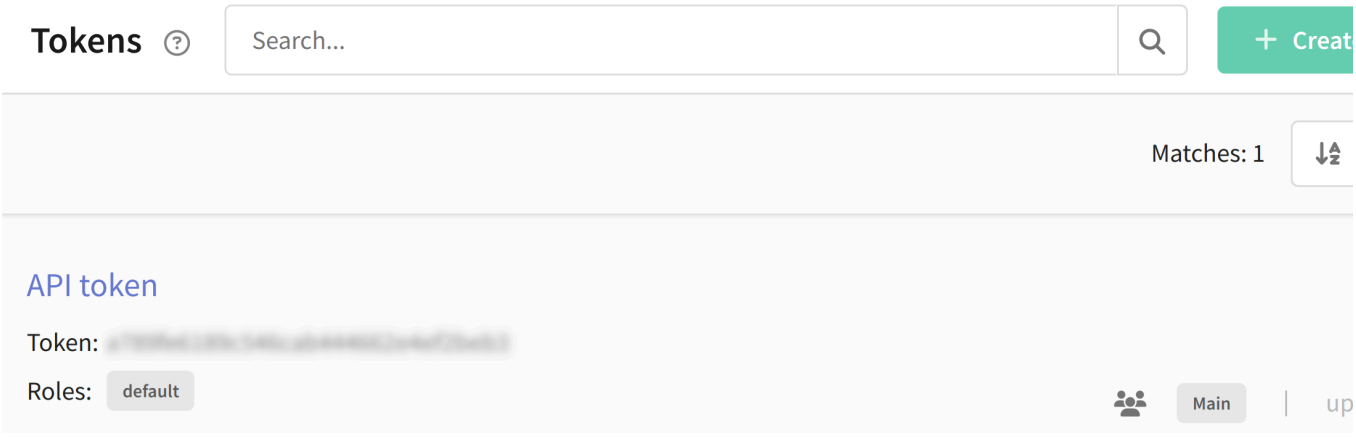
- **Sending a message to a single customer**
- **Launching a mailing for multiple subscribers**

> TIP
>
> **Only active trigger mailings set to the "When using a special API call" event can be run via the API. In how to create a trigger mailing.**

Getting an API token

API token is a key to access the platform via API. It is used in any API request.

You can find the token in Altcraft platform settings, in the Tokens section. If you don't have a token yet corresponding button:



> CAUTION
>
> **Only master users can create an API token. If you do not have rights to create a token, contact a user**

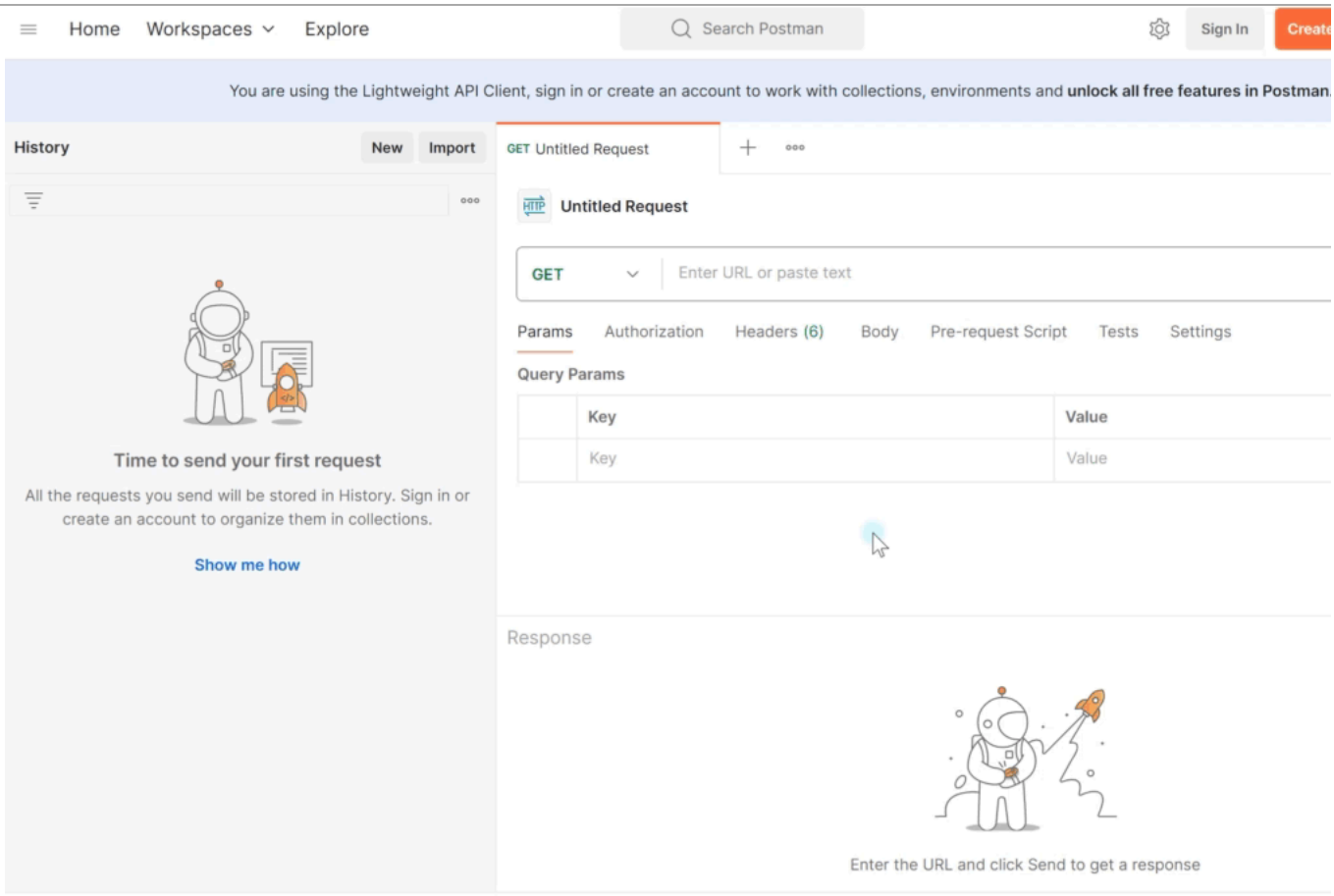
Installing and running Postman

To work with API requests, you need an application that allows you to send and receive them.

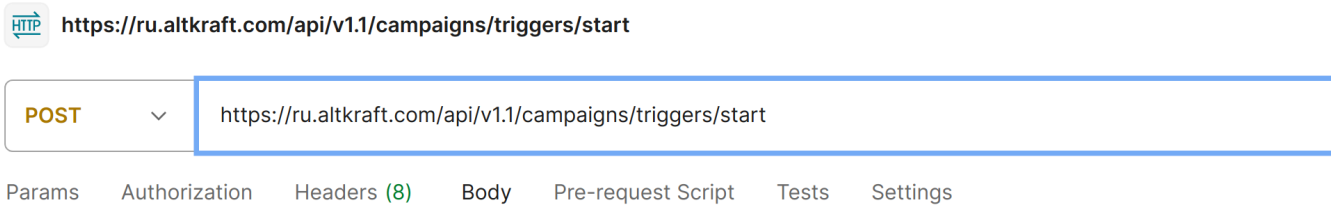
1. Download and install the Postman application or open its web version.

2. Select the POST method for your request.

3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



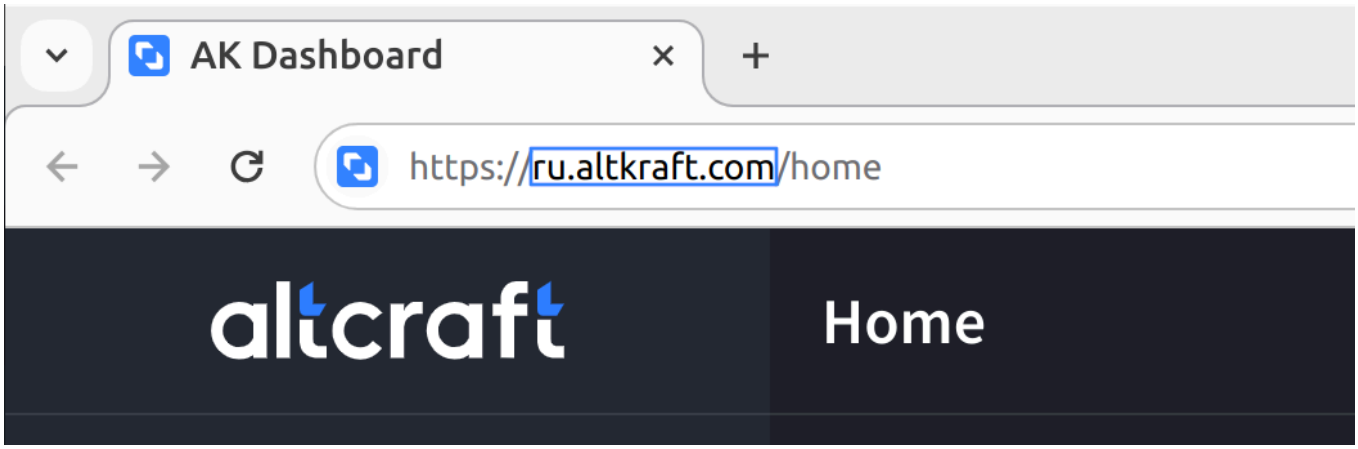
Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example the URL would be as follows:

`https://example.com/api/v1.1/campaigns/triggers/start`.

Instead of `example.com` you need to specify the address where the platform is located. You can get it dir



Creating a request

Launching a mailing for one user

Let's move on to writing the query. Suppose there is a subscriber with the email address `zaycev@example.` send a message to his email by launching trigger mailing #1. Our query might look like this:

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 1,
  "matching": "email",
  "email": "zaycev@example.com"
}
```

- **token — API token. More details about where you can find it are provided at the beginning of the ar**
- **id is the number of the trigger mailing. It is shown next to the mailing name to the right of the # sig always a number, do not confuse it with the name.**
- **matching and email - parameters by which the query will find the required profile in the database. I address of the profile is used.**

More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several sea platform, for example, the platform can search for a profile by email address, phone number, system ID additional field in the database.

The main thing is that the field by which you launch a search in the database is unique for the profile, i be the same for different profiles in the same database.

If you want to find a profile by its email, then `matching` will look like this:

```
{
    "matching": "email",
    "email": "example@example.com"
}
```

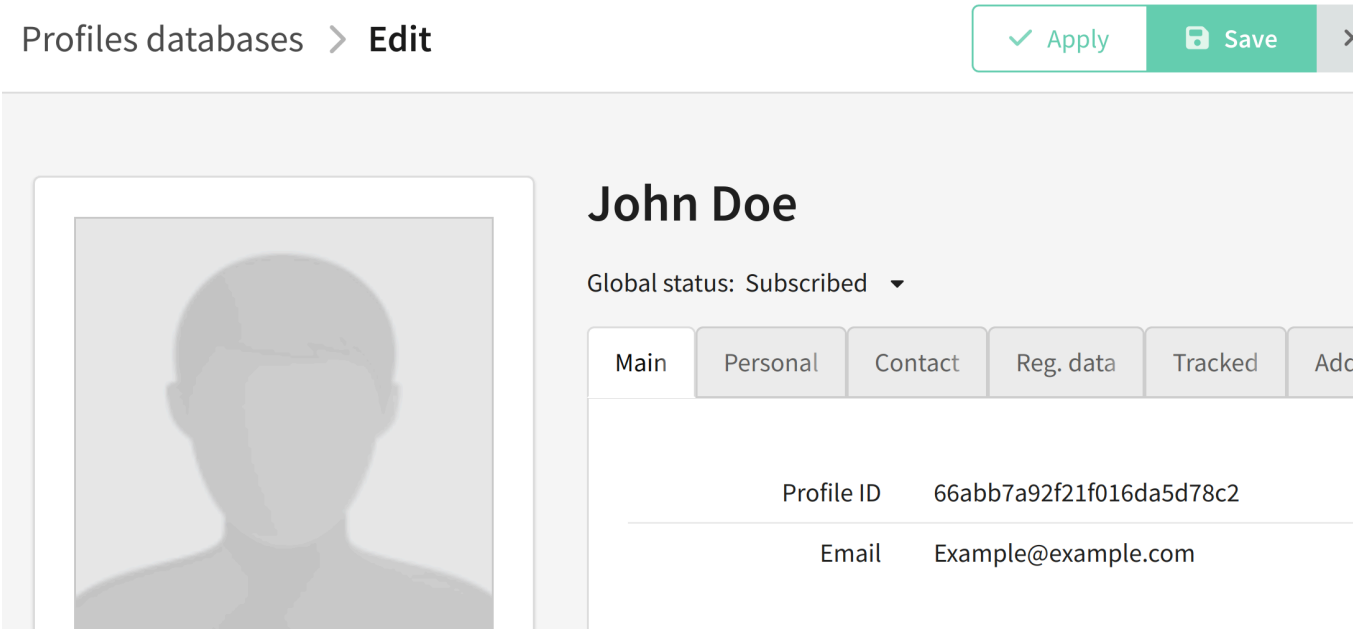
Search for a profile by phone:

```
{
    "matching": "phone",
    "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
    "matching": "profile_id",
    "profile_id": "012452gf62fcd71fjh8ec0"
}
```

The profile ID can be found here:



If you use an additional field to search for a profile in the database, your query will look like this:

```
{
    "matching": "custom",
    "field_name": "CRM_ID",
    "field_value": "12345"
}
```

where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

Launching a mailing for subscriber with custom field

Let's say we want to send a message with a new product catalog to the person who liked a VKontakte about this client except its VK ID, which is recorded in our database in the corresponding field. In this

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 5,
  "matching": "custom",
  "field_name": "vk_id",
  "field_value": "123456"
}
```

The search for profiles will search for an additional field, since we specified `custom` in `matching`. The na specified in the `field_name` and `field_value` parameters. The field value must be unique, otherwise the re indicated the VK ID of the client we are interested in, so the message will be sent only to him.

> **TIP**
>
> **Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period a request. Likewise, the query will not work if you do not close any quotes or parentheses.**

Sending a request

To send a request, click Send. If the request is successful, the following message will appear:

Pretty Raw Preview Visualize JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "profile_id": "66267f05ecc3993fba680df7"
}
```

If something went wrong, the response message will indicate the code of error and its description:

Pretty Raw Preview Visualize JSON

```
{
    "error": 400,
    "error_text": "Invalid request"
}
```

If you are unable to resolve the error yourself, please forward the code of error and its description to A

> **TIP**
>
> **This article describes only some of the possible parameters for this request. The remaining paramete**

Engage profile in scenario

In this article we will explain how to use an API request to send to the script a profile that already exis

TIP

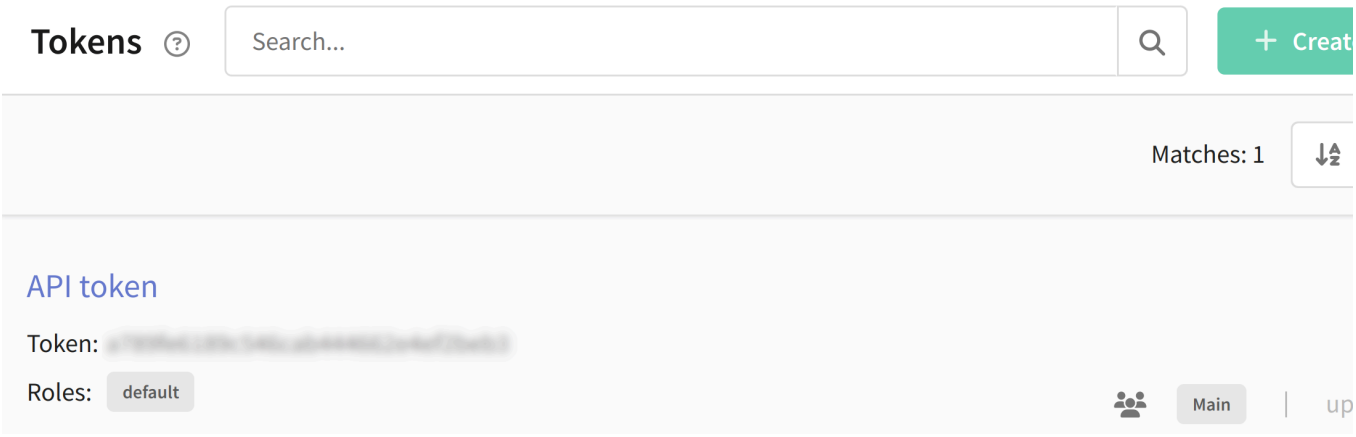
In order to run a profile in a scenario, the scenario must be created and activated. In this article you c automation scripts.

Note that the profiles in this run are included in the script via the Default **node.

Getting an API token

API token is a key to access the platform via API. It is used in any API request.

You can find the token in Altcraft platform settings, in the "Tokens" section. If you don't have a token y corresponding button:



CAUTION

Only master users can create an API token. If you do not have rights to create a token, contact a user

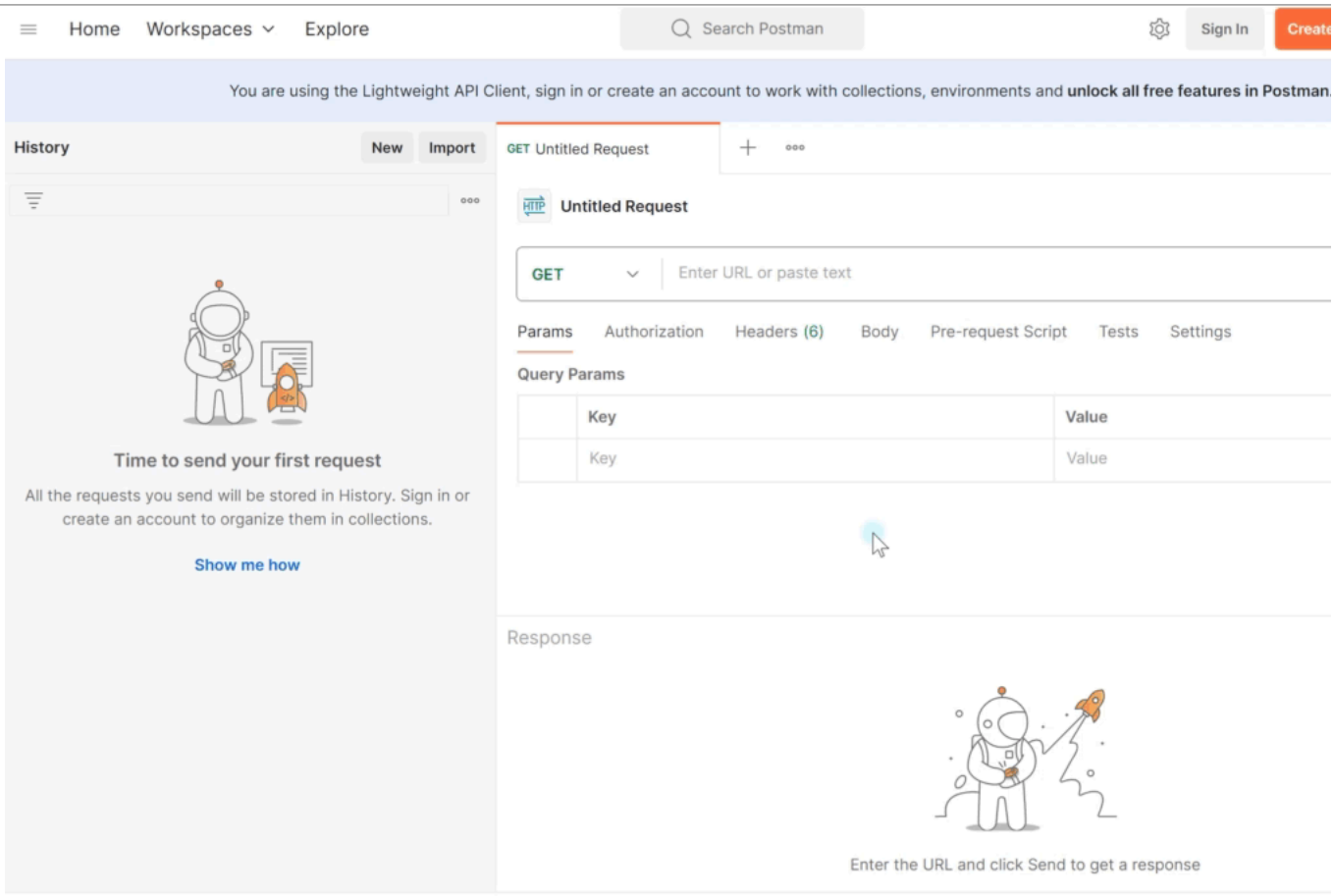
Installing and running Postman

To work with API requests, you need an application that allows you to send and receive them.

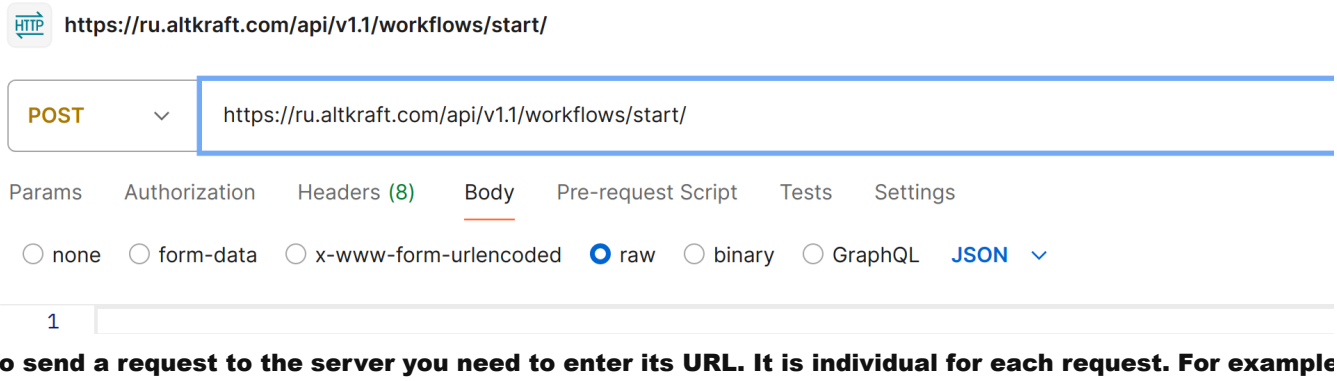
1. Download and install the Postman application or open its web version.

2. Select the POST method for your request.

3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



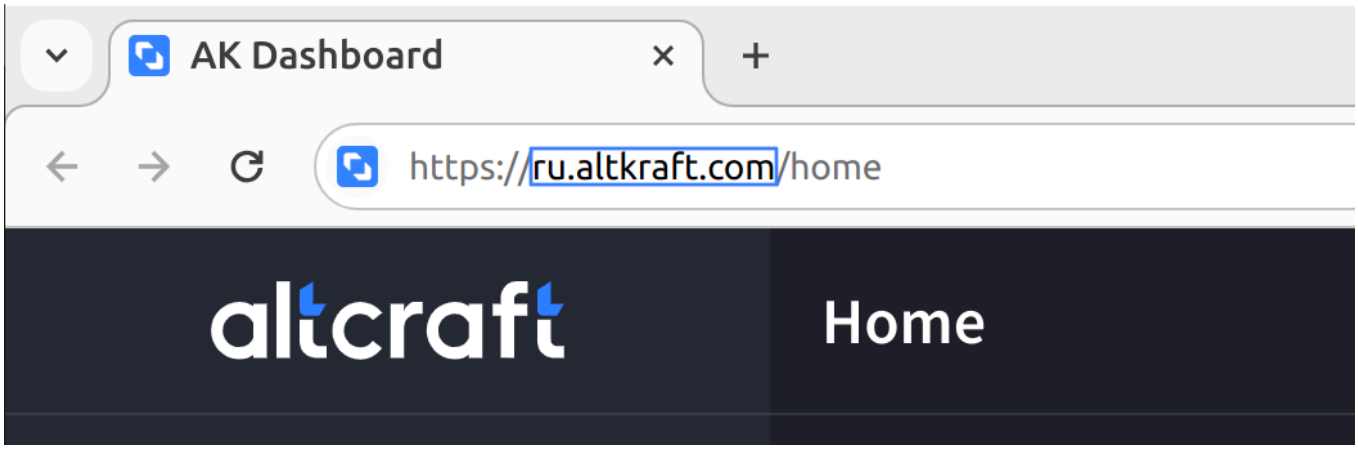
Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example URL would be:

`https://example.com/api/v1.1/workflows/start/`.

Instead of `example.com` **you need to specify the address where the platform is located. You can get it dir**



Creating a request

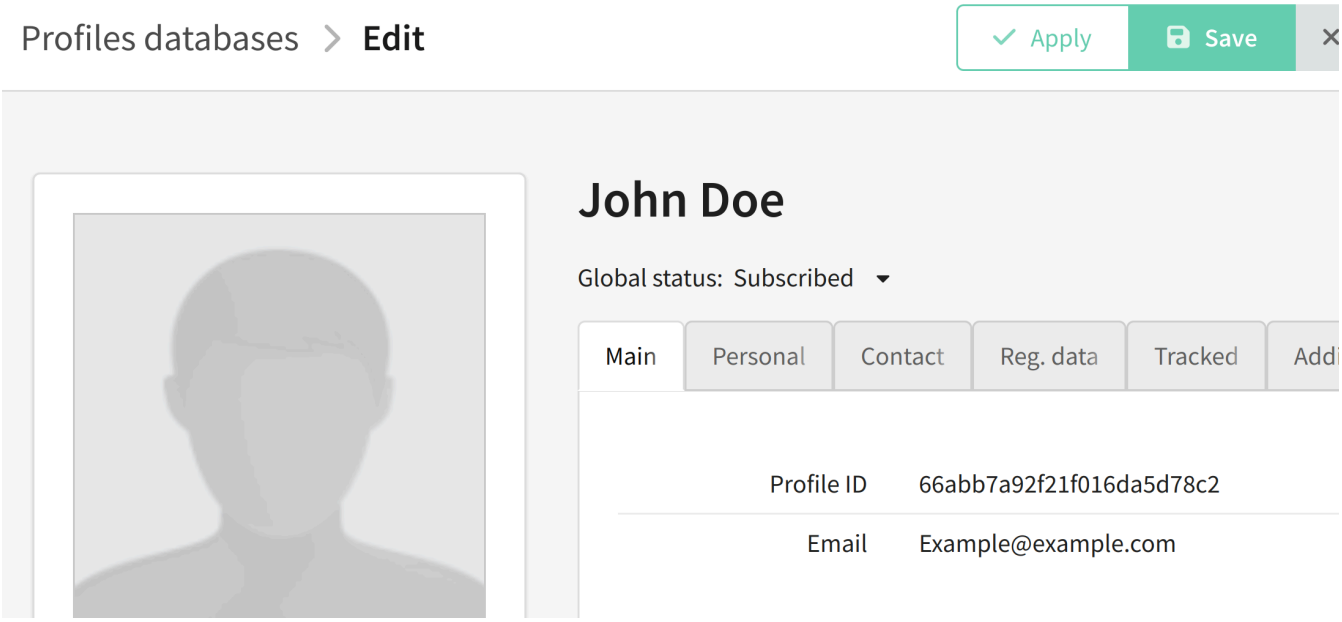
Let's move on to writing the query. Suppose we want to send a subscriber with ID `66acb337240d5213037986d` **look like this:**

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 3,
  "matching": "profile_id",
  "profile_id": "66acb337240d5213037986d8"
}
```

- **token — API token. More details about where you can find it are provided at the beginning of the ar**
- **id — scenario number. It is listed with the # sign next to the scenario name. The `id` of a scenario i it with the name.**
- **matching and profile_id - parameters by which the query will find the required profile in the databa profile ID is used.**

INFO

The profile ID can be found here:



More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several sea platform, for example, the platform can search for a profile by email address, phone number, system ID additional field in the database.

The main thing is that the field by which you launch a search in the database is unique for the profile, i be the same for different profiles in the same database.

If you want to find a profile by its email, then `matching` will look like this:

```
{
    "matching": "email",
    "email": "example@example.com"
}
```

Search for a profile by phone:

```
{
    "matching": "phone",
    "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
    "matching": "profile_id",
    "profile_id": "012452gf62fcd71fjh8ec0"
}
```

If you use an additional field to search for a profile in the database, your query will look like this:

```
{
    "matching": "custom",
    "field_name": "CRM_ID",
    "field_value": "12345"
}
```

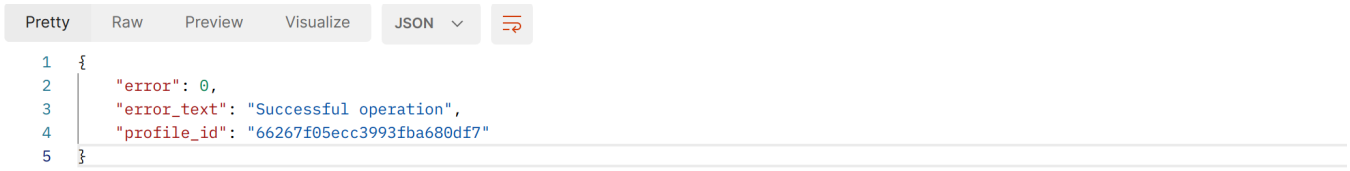
where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

TIP

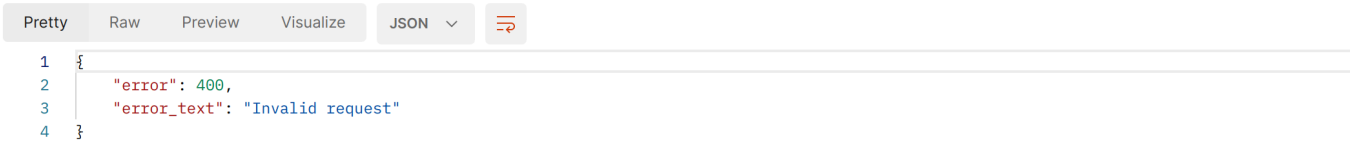
Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period a request. Likewise, the query will not work if you do not close any quotes or parentheses.

Sending a request

To send a request, click Send. If the request is successful, the following message will appear:



If something went wrong, the response message will indicate the number and description of the error:



If you are unable to resolve the error yourself, please forward the error number and description to Altc

TIP

This article describes only some of the possible parameters for this request. The remaining paramete

Changelog

INFO: **Platform update policy**

**Updates are distributed to users based on their requests. First of all, they become available to those
At this stage, the platform is evaluated and feedback is collected. After that, the update becomes ava**

**After receiving a new build, it is recommended to perform the update within 14 calendar days. Update
but also important security fixes and bug fixes.**

**Fixes and updates are only released for the current version of the platform. Outdated versions are not
that are more than two major releases behind are not addressed or fixed. To ensure stable and secure
update in time.**

TIP

**The platform version number indicates the release date and scope of the changes. For example, versi
update was released in 3rd quarter of 2024, it is the 69th version of the platform, and it includes 490 I**

v2026.3.78

CAUTION: Reconfiguration required when upgrading to v2026.3.78

As of version v2026.3.78, deferred custom channel tasks are processed only by the master procpiper. reconfiguration of `main.json` is required after the upgrade: add the `CLUSTER.PROC_PIPER` section and set th `true`) on only one node — the same way `procloyalty` is configured. Until this configuration is set, defer be processed.

Example section:

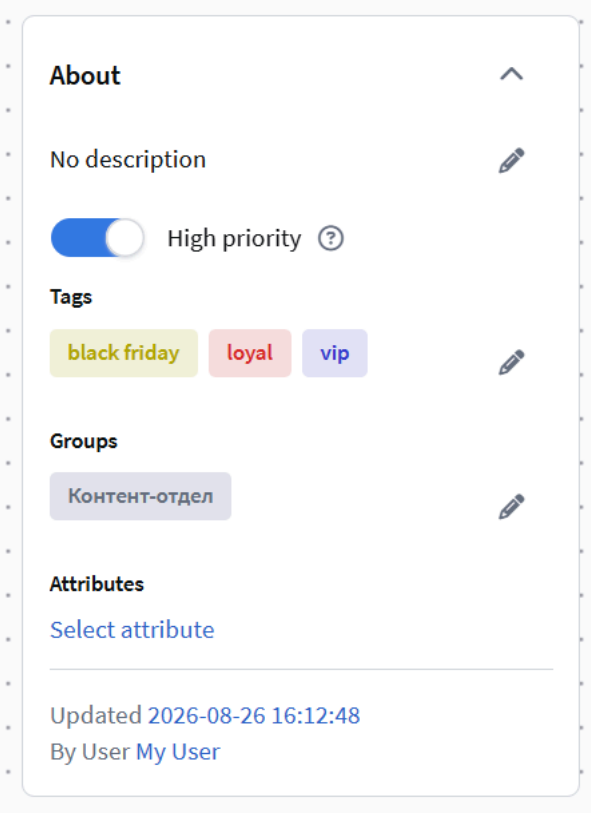
```
"PROC_PIPER": [
  {
    "HOST_ID": "app1",
    "RPC_HOST": "0.0.0.0",
    "RPC_HOST_CLI": "app1",
    "RPC_PORT": 8062,
    "FORCE_MASTER_ROLE": true
  },
  {
    "HOST_ID": "app2",
    "RPC_HOST": "0.0.0.0",
    "RPC_HOST_CLI": "app2",
    "RPC_PORT": 8062,
    "FORCE_MASTER_ROLE": false
  }
]
```

v2026.3.78.507 — Scenario priorities, unified segment condition editor, and fixes

This update introduces scenario priorities, rolls out the unified segment condition editor across the res profile history API more informative. Platform stability and security have been improved, the loyalty pro attribute handling has been improved.

Scenario priorities

You can now mark the most important scenarios — for example, business-critical ones. Their tasks are a queue behind others.



- **A High priority checkbox has been added to the scenario settings. Tasks of such scenarios are pro count is set by the administrator in the cluster configuration (the `HIGH_PRIORITY_WORKERS` parameter). T parameter is missing on some cluster nodes, it will be rejected.**
- **Mailings inside a high-priority scenario are also sent on a priority basis — no extra setup is needed automatically.**

Unified segment condition editor

The unified segment condition editor is now used in all places where profile selection is configured — in every module: funnels, profile relations, filtering mailings by channel fields, and profile capture in tri

Profile history API enrichment

The profile history API (`subscribers/history_get` and `subscribers/history_get_all`) has become more informat about related objects. For a goal (pixel), this includes its name and ID, captured data, referrer, utm tag complete picture of a profile's activity can now be obtained with a single request. Existing response fi is added as additional fields, so current integrations will keep working without changes.

Read more: Get profile action history.

Added

- **The `campaigns/list` API now supports a filter by template ID (`message_id`) — you can find all mailings t response now returns complete information about each template: subject, sender name and email, name.**
- **Push channel events now include the Read indicator — analytics on push notification reads is now**
- **Added complaint handling from the Altcraft Cloud SMTP mail server — complaints about emails are into account.**
- **In the custom channel editor, an option has been added for external object (Entity) fields: "Do not u mailing". If enabled, the field data is not included in the sent message — this reduces the load whe values.**

Improved

- **Improved stability: fixed crashes of internal processes that could make the platform unresponsive, import API. The loyalty program no longer stops due to lack of memory.**
- **Platform updates now complete correctly even with non-standard configuration paths, background database hangs after Market queries have been fixed.**
- **Fixed a severe slowdown when sending mailings to many different domains — delivery speed no lo**
- **Fixed traffic distribution in a virtual sender: mailings are now sent strictly according to the configu**
- **Lists of participants and transactions in the loyalty program now load faster — they open quickly e**
- **Fixed points calculation in the loyalty program: the balance no longer goes negative when blocked matches the interface, and the points expiration date is set correctly.**
- **Improved attribute handling: when creating an attribute, the name and API name are auto-filled (e.g from entities" function is enabled by default, and attribute conflicts produce a clear error message**
- **Fixed segment calculation errors: static segments correctly handle large data volumes, updatable fails, and numeric lists in dynamic segments and the "Relative to current date" condition display a**
- **Optimized task execution in scenarios: fixed an issue where tasks could take several tens of hours mailing launch is now distributed across the cluster and does not overload the system.**
- **Fixed deactivation of a regular mailing with the "Active (Running)" status — without the "interrupt mailing now correctly switches to the "Inactive" status.**
- **Fixed an issue where users with view-only permissions could not view platform objects.**

Other

- **Fixed duplication of delivery events when sending mailings through the Sendsay provider.**
- **Fixed an error in the subscription manager: a subscription to a resource is no longer deleted if the subscription manager.**
- **Fixed form fill export: exporting for specific dates now creates one task instead of two; the data o**
- **When exporting data and history from a profile page, all profile subscriptions are now exported.**
- **Fixed UTM tag analytics in popups: tags are now correctly taken into account.**
- **Fixed a login page error that could prevent the account from opening, and an endless reload loop v**
- **When activating a broadcast, the "Active" checkbox no longer resets before the mailing is complet**
- **The segment condition selection menu in scenarios is no longer cut off.**
- **In goal capture, the "in range" condition now supports a range of values (for example, a purchase only a single value was available.**
- **The mailing testing menu no longer shows invalid subscriptions.**
- **Fixed custom channel operation: sending mailings with external objects, the `params` and `out` servic protection against duplicate tasks when running on schedule and queue settings in the listener.**
- **Fixed encoding of event data values when inserted into links — Cyrillic and other special characte**
- **Fixed event capture: data from one hook no longer leaks into the data of other hooks.**
- **Fixed an error when creating cache tables for external SQL query results.**
- **Fixed referrer registration in profile events.**
- **Improved error diagnostics in the Soglasie SMS provider.**

- **Fixed an error creating indexes in MongoDB that could cause data operations to hang.**
- **Data sorting processes now terminate correctly and no longer accumulate on the server.**

References

v2026.3.78.507: PL-22978 PL-22314 PL-22965 PL-22925 PL-22926 PL-22949 PL-22927 PL-22943 PL-2293 22819 PL-22723 PL-22864 PL-22905 PL-22740 PL-22862 PL-22824 PL-22790 PL-22741 PL-22742 PL-2269 22185 PL-22802 PL-22579 PL-22434 PL-19404 PL-22883 PL-22806 PL-22056 PL-22089 PL-21664 PL-2242 22644 PL-22748 PL-22734 PL-22753 PL-22766 PL-22278 PL-22737 PL-22715 PL-22580 PL-22721 PL-2267 20019 PL-21920 PL-22017 PL-14389 PL-22697 PL-22712 PL-22699 PL-22682 PL-22648 PL-22514 PL-2264 22592 PL-22565 PL-22193 PL-22476 PL-22440 PL-21681 PL-22590 PL-22585 PL-22576 PL-22528 PL-2235 22529 PL-18282 PL-18655 PL-22247 PL-22126 PL-21818

v2026.2.77

v2026.2.77.189 — i-Digital Direct API integration and improvements

This update adds a new SMS provider — i-Digital Direct API, accelerates static segment recalculation, operations and the scenario editor.

i-Digital Direct API SMS provider integration

This update adds support for a new SMS provider — i-Digital Direct API.
When configuring a sender for this provider, you need to specify a token for API authorization.

Supported:

- **Request format — JSON array (batch sending up to 1000 messages)**
- **Callback statuses:** `delivered`, `undelivered`, `unsent`
- **Retry logic for HTTP codes 429, 500, 502, 503 (exponential backoff)**

If needed, the sending URL can be overridden via the `SMS_IDIGITAL_CUSTOM_SENDING_URL` **configuration param**

Improved

- **The recalculation of static segments was accelerated.**
- **Improved error messages when calculating a global control group with stratification.**
- **Optimized the PostgreSQL database process, simplifying compliance with information security req platform.**
- **Added the ability to automatically clean up old files in the** `download_taskreport` **directory.**
- **Added localized messages for new error types in the mailing log.**
- **The scenario editor performance has been improved.**
- **Improved interface:**
  - **the sender queue view button is now only displayed for Email channels;**
  - **the list of product segments now correctly shows the user who last updated the segment.**

Other

- **Fixed the passing of** `custom_data` **fields to event capture when launching a scenario via the** `workflows`
- **In the mailing testing section, a test profile is now created only if there is at least one subscriptior**
- **When deactivating a global control group, the ability to remove profiles from the group has been ac an exit event.**
- **Fixed the execution of the** `point_trc_migrate` **points transaction migration for ClickHouse in replicate**
- **Importing templates via archive no longer results in** `http` **being substituted into image links, reduc spam.**
- **In scenarios, when configuring a "Scheduled segment", the selected launch month is now correctly prohibited.**
- **Fixed error display when no profiles match the stratification conditions in the global control groups**
- **Fixed the default value substitution for numeric fields using a predefined list of values in custom cl**
- **Fixed an issue where profiles with a large number of additional fields were not added to the loyalty**
- **In the "Update static segment" scenario node, only segments from the current profile database are**
- **In analytics reports, deleted loyalty program group levels no longer appear as empty lines — their s**
- **Fixed an issue where profiles were not captured into a scenario when switching the trigger conditi "Any click in Email".**
- **When creating indexes in the Admin panel, the** `_xxh` **field can now be manually selected from the g automatically added via a separate checkbox.**
- **When changing the schedule type in broadcast mailings, incompatible options are no longer saved.**
- **In segments, user fields are no longer replaced by system fields when creating selection condition**
- **Fixed the display of** `procmarket` **and** `webmarket` **process versions.**
- **Added new conditions to segments for selecting profiles based on exclusion from the global contro**
- **Fixed the issue with selecting resources and segments in segment conditions for users with limite**
- **Fixed an issue where only numeric values were written to a field of type "List" when importing pro**
- **Fixed profile capture into a scenario when activating a pixel with the "contains" or "does not conta**
- **Improved the event capture interface for custom channels: eliminated the ability to select an event**
- **Fixed the issue with campaign activation when one of the MongoDB replica members is unavailabl**
- **Attribute fields with editing restrictions in active objects are now correctly disabled after activatio**
- **Fixed the "for the last X days" condition in segments: value 1 now correctly selects events only for**
- **Fixed slow processing of the** `hook_events` **queue when capturing events with message content.**
- **Fixed an issue where synchronization of static segments with Facebook Ads Manager did not work identifier" data type.**

- When sending a broadcast mailing by profile time zone, if no matching profiles are found, the mess selected time zones" is now displayed.
- Fixed the display of field headers in segment conditions — the headers defined in the segmentatio

References

v2026.2.77: PL-18253 PL-19527 PL-20943 PL-20994 PL-21310 PL-21505 PL-21515 PL-21579 PL-21610 PL 21769 PL-21783 PL-21786 PL-21794 PL-21806 PL-21819 PL-21821 PL-21826 PL-21851 PL-21857 PL-2185 21880 PL-21885 PL-21899 PL-21909 PL-21922 PL-21929 PL-21931 PL-21934 PL-21937 PL-21949 PL-2196 21198 PL-18441 PL-18398 PL-8898 PL-18381 PL-21870 PL-21596 PL-21990 PL-12550 PL-17566 PL-21192 21442 PL-20763 PL-20475 PL-21745 PL-21834 PL-21965 PL-21953 PL-21713 PL-21847 PL-20466 PL-1994

---

v2026.2.77.941 — Optimization, escape function

This update brings extensive optimization of internal platform processes: segmentation, template hand sending workflows have all been accelerated. The `escape` function for variable escaping and new confi segment calculation intervals have been added. The platform is now more stable and faster.

Added

- Added the `escape` function for escaping variables in templates and scenarios. It allows passing stri (e.g., `{lead.email}`) without substitution — the content inside `{escape "..."}` remains as-is. Learn mo
- Added the `SCHEDULE_RUN_INTERVAL_MIN` and `WFCAPT_SCHEDULE_RUN_INTERVAL_MIN` parameters for optimizing segr set the minimum allowed interval for launching segment calculations in scenarios.

Improved

- The `subscribers/history_get` API endpoint now supports pagination: the `limit` and `from_id` parameters page by page, and the `next_from_id` field in the response indicates which event to continue from. By are returned (limit — 1000 events). If only `date_to` is specified, the left-side date restriction is remo of a single profile.
- The launch log in the "Scheduled profile segment capture" scenario node has been improved: a col calculation status and the calculation completion time has been added.
- The block template editor performance has been optimized.
- Some visual elements of the platform have been improved, and some tooltip phrasings have been u
- Message sending via the Notify channel has been improved.
- External segmentation queries have been optimized: when using SQL queries to external database segment formation time has been reduced from several hours to several minutes.
- The `templates/add` and `templates/get` API methods have been optimized.
- Subscription field handling in custom channels has been fixed: optional fields can now be empty, w creating a subscription via the interface. Unsubscription via API works correctly for subscriptions

Other

- Fixed the import of profile files exported from the platform when spaces are present in the data.
- Fixed the display of empty pages in a segment when it contains no profiles.
- Loyalty program transactions in the profile card are now sorted correctly.
- Fixed the schedule in a scheduled segment — correct values can now be set.
- Added the `List-Unsubscribe-Post` header and DKIM signatures to emails sent via AKMTA.
- Fixed the handling of a number passed to an enum field via API — it is now interpreted as a value,
- Column names from Hive queries are now correctly parsed in templates.
- Fixed an error when using a "Month" type variable in segments.
- Profiles can now be sent to a scenario from a dynamic segment without saving it first.
- Fixed an error when duplicating values in a capture condition based on a "Tag" type field change i
- UTM tags are now correctly saved in the profile when clicking a link from an email.
- Fixed the layout of the weekly report.
- Inherited attributes are now correctly passed through a custom channel.
- Fixed an error in the `/tasks` list.
- Entity search by attributes in segmentation has been optimized.
- Fixed the activity toggle when deactivating scheduled profile imports.
- Eliminated infinite loading when configuring order segmentation in the loyalty program.
- Fixed the segmentation condition fields "hit from [n] to [m] times within [x] days/period".
- Fixed the issue of returning to the beginning of the mailing list when switching between pages.
- Fixed incorrect display of banners in the campaign editor.
- Placement campaigns now pass validation when saved.
- Fixed the calculation of an updated segment with the condition "No N orders meeting all condition
- Fixed button descriptions in the interface.

- Updated the error message when activating a loyalty program if an active program with the same
- Fixed the appearance of the parameter value panel when creating a segment with a segmentation
- Fixed pagination in the table component.
- In the segment builder, a single day can now be specified using the “Date/age relative to current da
- Fixed an error when changing the segmentation query in a segment: when replacing a query with p parameters, the old parameters are no longer saved along with the new query.
- In forms, when opening a “List” element, the actions page is no longer blocked.
- Error logs for “Condition” nodes in scenarios now display the scenario and node IDs.
- Fixed a memory leak in the `procpiper` process.
- Buttons in Telegram Bot and Group now contain correct links.
- The CallerID is now pulled into the SMS template preview.
- In multivariate tests, the tested audience limit has been fixed.
- Attribute values can now be selected in Viber, WhatsApp, Notify, and MAX templates.
- `proctask` now correctly sends scheduled broadcast mailings.
- Eliminated the non-clickable “Parts” tab in visual template editor variables when no parts have bee
- Fixed the display of profile counters for accounts in the admin panel.
- Fixed the display of long names in the breadcrumb top header.
- Fixed errors in the campaign editor.
- The `format` variable is no longer automatically replaced with `formatdate` when saving a template.
- Added profile database validation for connected static and updated segments.
- After a platform update, field mappings in scheduled import settings are no longer reset.
- Fixed the default value for the “Week” type parameter when creating a segment with a segmentati
- For scenarios, the user who updated them is now correctly recorded.
- Links with the `no-track` attribute containing the `}` character at the end are now handled correctly.
- Fixed the campaign launch date and time setting when creating from the calendar.
- Long field names in order segments no longer extend beyond the form boundaries.
- Eliminated an empty popup when creating a loyalty program level with an additional identifier.
- The language switch script on the unsubscription page now works correctly.
- Fixed documentation links in the platform.
- Eliminated a JS error when saving templates via the visual beta editor.
- The trigger mailing launch delay time is now displayed correctly.
- Fixed the narrowing of the update log table when re-expanding in updated and static segments.
- Segments with segmentation queries are now saved correctly.
- In the “Date in range” condition, a “to” date earlier than today can now be selected.
- Line breaks are now displayed in the MAX template preview.
- Eliminated a JS error when clicking the automatic UTM tag insertion button in a template.
- Fixed the renaming of segment groups.
- Fixed an error when opening a segment after deleting the associated profile database.
- Product segment recalculation can now be interrupted even when `proctask` crashes.
- Updated the loyalty condition texts inside the segment builder.
- Eliminated the overlap of page blocks in the form logic editor.
- Fixed link creation on triple click.
- Fixed errors related to the loyalty program.
- The template editor no longer attempts to load pixels from non-existent URLs.
- Fixed the display of the control group membership tag inside scenarios.
- Fixed the validation of repeated “Action Execution” nodes in scenarios.
- The campaign popup in calendars now opens correctly.
- Long names are now displayed correctly in the campaign log.
- Fixed the operators in the complex User-agent pixel condition in the segment builder.
- The segment is now saved when the script contains valid values.
- Fixed an error when creating a test profile when the tariff is exhausted.
- Added confirmation popups when removing entities from lists.
- The `action_hook_fill.log` log is now correctly rotated.
- In the API node of scenarios, the `Authorization` header can now be set.
- The ability to edit the category selection in filters of an updated product segment has been added.
- Eliminated uncontrolled generation of `data-placeholders` when creating templates for segmentation
- Fixed the date format in the segmentation query and in the segment condition.
- The selected launch method for a placement campaign is now displayed correctly when opened.
- Links in breadcrumbs for nested Market entities now work.
- Fixed the error text in the popup editor when platform variables are used incorrectly.

- The “Attribute Value” variable is now substituted into URLs.
- When adding a link to a text fragment, a checkbox for the confirmation link has been added.
- Added a selection option in goal segmentation by complex condition.
- `webcontrol` performance has been optimized.
- Fixed the validation and editing of required attributes in the attribute editor.
- Added Firebase JSON file validation when uploading to a resource.
- Fixed user password validation in the interface and in the admin panel.
- Updated the list of browsers supported by the platform:
  - Google Chrome, version 101 and above
  - Mozilla Firefox, version 102 and above
  - Microsoft Edge, version 101 and above
  - Opera, version 91 and above
  - Chromium, version 101 and above
  - Safari, version 15.0 and above
  - Yandex Browser, version 23.2 and above

References

v2026.2.77.941: PL-17761 PL-20895 PL-21643 PL-22018 PL-22052 PL-22039 PL-21855 PL-18603 PL-2208 21689 PL-21512 PL-22021 PL-22053 PL-22035 PL-21178 PL-22028 PL-21947 PL-21576 PL-21573 PL-2201 21539 PL-22051 PL-19111 PL-22080 PL-22071 PL-21799 PL-20995 PL-19748 PL-21055 PL-17177 PL-2206 21150 PL-22066 PL-21540 PL-16016 PL-21587 PL-21932 PL-22006 PL-22033 PL-21985 PL-18477 PL-2204 18275 PL-20561 PL-21944 PL-20996 PL-21467 PL-21674 PL-17466 PL-20676 PL-21179 PL-17297 PL-1745 20898 PL-22061 PL-21843 PL-22057 PL-21096 PL-22055 PL-21859 PL-21854 PL-21525 PL-22041 PL-2203 13622 PL-19090 PL-17409 PL-17995 PL-21778 PL-22029 PL-22008 PL-18958 PL-18423 PL-21946 PL-2177 21878 PL-22020 PL-21984 PL-16252 PL-20375 PL-16594 PL-11413 PL-21584 PL-18197 PL-21580 PL-1439 21973 PL-21948 PL-21976 PL-17169 PL-21963 PL-21930 PL-21900 PL-21044 PL-19611 PL-20721 PL-1882

v2026.2.77.1060 — Bug fixes

Improved

- The date picker fields in segment conditions now support manual date entry and paste.
- Improved reporting for the points transaction migrator.
- Optimized memory consumption of the `prochook` process.

Other

- Fixed the display of the cyclic segment dependency error.
- Fixed segmentation by MAX channel events — profiles with events are now correctly selected.
- Notifications are now sent every time the tariff limit for profile import is reached.
- Fixed tooltip offset in the displayed fields editor in Safari and Chrome.
- Fixed an interface bug that prevented removing a value in the segment limit field.
- Fixed segment statistics population in ClickHouse.
- Fixed availability of custom channel custom events in segments and hooks.

References

v2026.2.77.1060: PL-22151 PL-21447 PL-20788 PL-22188 PL-22177 PL-22156 PL-22160 PL-22154 PL-221 PL-21974 PL-21507 PL-21897 PL-21926 PL-18636 PL-20913 PL-21896 PL-21043

---

v2026.2.77.1610 — Segmentation improvements and bug fixes

This update revamps the segment builder interface for improved condition readability, and fixes issues segmentation, global control groups, the loyalty program, and the audit log.

Improved

- The segment builder interface has been revamped for improved condition readability:
  - color indicators have been added for filters and operators;
  - a collapsed/expanded state mode for conditions has been added;
  - the display of conditions with a large number of selected entities has been redesigned — sepa hover, and the ability to collapse and copy the list have been added;
  - the order of fields in a condition has been changed: date, range, and resource selection has be
  - the “Search” placeholder display in the condition selection menu has been fixed.
- Global control group validation has been improved: minimum profile count check has been added, t
- The error message when exceeding the maximum number of profiles in a global control group is n

- **Segment calculation with profile history conditions has been optimized: multiple conditions are co query, significantly speeding up segment formation.**
- **Event capture content storage has been optimized: content is saved only for events matching the c**
- **Template size limits have been moved from constants to configuration, allowing limits to be chang**
- **Metrics have been added for the `procpiper` process: pipe processing time, HTTP request time, even processing time.**
- **A timeout has been added for Firebase push notification sending, preventing process blocking whe**
- **The code editor has been improved: shared button components have been added for use in HTML, S**

Other

- **Fixed an issue where the "Account ID" and "User ID" columns had the same name in the audit log e**
- **Fixed an issue where it was impossible to build a segment by relations between different profile da for reverse relations.**
- **Fixed an issue where adding multiple conditions with attributes "never in the last X days" resulted**
- **Fixed an issue where age selection relative to the current date did not work in segments.**
- **Fixed segment builder validation for the "day/age relative to current date" condition.**
- **Fixed an issue where resources and segments were unavailable for selection in segment condition permissions.**
- **Fixed an issue where the `sort` process consumed all available memory during updated segment re**
- **Fixed an issue where inactive subscription highlighting was missing in the test mailing for push res**
- **Fixed an issue where moving active benefits in the loyalty program tree reset the active benefit co**
- **Fixed an issue where mailings were not sent via a custom channel.**
- **Fixed an issue where a custom channel template did not enforce configured size and format rules f**
- **Fixed an issue where the `MTA_ATTACH_MAX_SIZE` parameter did not account for local files in custom cha**
- **Fixed an issue where a custom channel template with an external database query in the "Serialize saved. Frontend validation has been disabled for JSON type fields in custom channel templates.**
- **Fixed an issue where mini-preview was not displayed in the template list for custom channel templ**
- **Fixed an issue where local image paths were not replaced during block editor template import in t**
- **Fixed template import from other systems.**
- **Fixed control group issues that occurred when stratification settings were modified.**
- **Fixed an issue where some newly imported profiles were not captured into a scenario due to false**
- **Fixed an issue where single activation/deactivation of a campaign or scenario from the list did not the interface.**
- **Fixed an issue where the platform continuously required a password change after setting the "Use login" flag in the admin panel.**
- **Fixed a panic that occurred when calculating a global control group with the `FILEDB_ENABLE: false` pa**
- **Fixed double URL escaping in webhooks when receiving statuses from the EDNA provider (WhatsA**
- **Fixed an issue where the `HSETEX` command, not supported in Kvrocks, was used.**
- **Fixed an issue where one out of three SMS messages was not delivered when sending via the Rapp**
- **Fixed an issue where the "Dropdown" form element was disabled when adding a custom option.**
- **Fixed an issue where PostgreSQL connections hung in the `ClientWrite` state after market queries.**

References

v2026.2.77.1610: PL-20821 PL-22237 PL-22278 PL-21873 PL-22086 PL-22334 PL-22273 PL-22322 PL-222 PL-22243 PL-22190 PL-21862 PL-16725 PL-19825 PL-22056 PL-16289 PL-20731 PL-20584 PL-15635 PL-2 PL-22281 PL-22240 PL-22259 PL-19892 PL-21422 PL-22027 PL-22258 PL-22173 PL-22103 PL-22183 PL-2 PL-22168 PL-22137 PL-22083 PL-21349 PL-21922 PL-21751 PL-21356

---

v2026.2.77.1885 — Flutter SDK, form API, and bug fixes

This update adds Flutter support to the Altcraft mSDK, introduces a set of form-related API handlers, a exchange type in the custom channel publisher pipe. It also fixes issues related to segmentation, scen operations.

Added

- **The ability to specify the exchange type (direct, topic, fanout, headers) in the custom channel publ**
- **Flutter support has been added to the Altcraft mSDK. The `altcraft_sdk` package provides push notif RuStore), mobile events, subscriptions, profile management, and push token management via the D**
- **API handlers for working with forms have been added:**
  - **`/api/v1.1/forms/list` — retrieve a list of forms with pagination and filtering;**
  - **`/api/v1.1/forms/get` — retrieve information about a single form;**
  - **`/api/v1.1/forms/fills` — retrieve form submissions for a specified period;**

- `/api/v1.1/forms/activate` — activate (publish) a form and return a link;
- `/api/v1.1/forms/deactivate` — deactivate a form;
- `/api/v1.1/forms/delete` — delete a form and its submissions;
- `/api/v1.1/forms/fill` — export form submissions of a specific user with matching.

For more details, see the Forms section of the developer documentation.

Other

- **Fixed an issue where API tokens remained blocked after account unfreeze.**
- **Fixed referrer display in profile events when registering a goal.**

> CAUTION
>
> **The `referer` parameter in the goal registration API now accepts a full URL or a relative path (in the br you previously passed only a domain without a path, update your integration — otherwise the referrer events. For details: Goals and goal values registration.**

- **Fixed an error when working with a virtual sender containing multiple sender types.**
- **Fixed tag display in profile custom fields: data is now correctly separated on initial display.**
- **Fixed attachment validation in custom channel templates: file size and format limits are now enfor**
- **Fixed segment condition processing for time-limited events: "never in the last N days" conditions n**
- **Fixed a `prochook` process panic when configuring event capture in Kafka broker with SASL+SSL and**
- **Fixed account metadata updates when changing tariff: change time and user are now recorded co**
- **Fixed profile export from segments: files download correctly even when the profile database name**
- **Fixed profile capture in scenarios via the "Field value change" trigger for Tag-type fields when addi**
- **Fixed profile freezing on condition nodes in scenarios under high load.**
- **Fixed campaign and scenario state display in lists: the interface now updates without page reload**
- **Fixed the "cannot get campaign" error when launching mailings in scenarios.**
- **Fixed a custom channel pipeline error when using the `toObject` function with new template fields.**
- **Fixed an issue with increasing the profile limit: after reaching the limit, the system correctly proce restriction flag is reset when the tariff is changed.**

References

v2026.2.77.1885: PL-21607 PL-22422 PL-22278 PL-22416 PL-22300 PL-20726 PL-22397 PL-17877 PL-213 PL-22386 PL-22362 PL-22359 PL-22044 PL-22133 PL-19310 PL-22272 PL-20545 PL-21590 PL-22354 PL-1 PL-22333 PL-19892 PL-22139 PL-22146 PL-22116 PL-21658 PL-22105 PL-20853 PL-21661 PL-21981 PL-2

---

v2026.2.77.2234 — PlatformRoles mode for LDAP/Keycloak, segment improvements, and bug fixes

This update adds the PlatformRoles mode for LDAP/Keycloak authentication, makes scheduling optiona the ability to view segmentation conditions from the update log in static segments. It also fixes issues segmentation, mailings, custom channels, the audit log, and process stability.

> NOTE: MAX Integration Change
>
> **The MAX messenger API has been updated. Starting July 19, 2026, the MinTsifra certificate must be i integration to work.**

Added

- **The PlatformRoles mode for LDAP/Keycloak has been added: when enabled, user roles and groups interface and are not overwritten on each login. LDAP/Keycloak is used only for authentication and (email, name, status).**
- **Updated segments now support disabling the recalculation schedule.**
- **Static segments now have a button in the update log to view segmentation conditions in read-only the conditions.**

Improved

- **Memory consumption of the `procworkflow` process has been optimized.**

Other

- **Fixed button display when viewing completed tasks in lists: the "Close" button is now shown.**
- **Fixed mailing testing: the tracking domain in the test email is now taken from the mailing resource**
- **Fixed the variable selection menu in the scenario API call node: the menu now closes after selecti appended to the existing text in the URL field.**
- **Fixed error display when deleting a mailing source: the error is now shown at the source level.**

- **Fixed the color of gray labels in the filter component.**
- **Improved stability of the `proctask`, `proctrigger`, and `procloyalty` processes.**
- **The user JWT token is no longer readable on service subdomains (pixel/tracking).**
- **Campaign events have been added to the audit log.**
- **Fixed phone number formatting in the Viber channel when creating a subscription via the interface now used.**
- **Fixed the “not equal” and “not in list” segment conditions for email subscriptions: profiles without selected in both cases.**
- **Fixed table view settings when search is active.**
- **Fixed date range validation in segment conditions: the same date can now be selected in the “Fron**
- **Restored the ability in the segment builder to create a “opened at least N times in the last X days” without specifying a specific mailing or template.**
- **Fixed date and time display in the scheduled segment: the date is displayed in the interface langua and end times are shown in the account time zone.**
- **Fixed an issue where deleting a custom field in one channel of a resource also removed fields with subscriptions on other channels of that resource.**
- **Fixed event condition grouping optimization in segments: queries are now correctly combined whe “within period” conditions.**
- **Fixed an issue where invalid subscriptions on one channel blocked message sending on another ch**
- **Fixed benefit application when using order segments in the loyalty program: discounts are now ap segmentation conditions.**
- **Updated the MAX channel integration: the API domain has been changed to `platform-api2.max.ru` and been added. Note: the chat ID must now be entered manually — it can be found in the [web.max.ru](web.max.ru) l**
- **Profiles now correctly leave the local control group when stratification settings are modified.**
- **Fixed new and old password identity validation in the admin panel.**
- **Fixed ClickHouse query error in undelivery analytics with IP address grouping.**
- **Fixed wildcard usage in allowed SMS sender numbers in the Admin panel.**
- **Improved archive upload security mechanism.**
- **Added proper MongoDB error handling when deleting a profile.**
- **Fixed an issue where users with limited permissions could see inaccessible scenarios.**
- **Fixed traffic distribution between senders in a virtual sender: traffic is now distributed strictly acc**
- **Fixed local control group calculation upon mailing activation: the number of profiles in the LCG no percentage.**
- **Fixed an issue where mailings could not be created when the “Enable smart settings” option was c**
- **Fixed a segmentator bug where the selection returned 1% more profiles than configured in the inte**
- **Fixed a Market API error when importing orders with custom fields via `api/v1.1/market/orders/import`.**
- **Fixed an error that occurred when creating ClickHouse tables for certain SQL queries.**
- **Fixed data collection for unique events in form analytics.**

References

v2026.2.77.2234: PL-19566 PL-21942 PL-22036 PL-20727 PL-21621 PL-22489 PL-22514 PL-22349 PL-196 PL-22542 PL-22291 PL-22435 PL-22500 PL-22473 PL-22450 PL-22342 PL-18064 PL-22445 PL-22475 PL-2 PL-21650 PL-20096 PL-22183 PL-19604 PL-22198 PL-22172 PL-21541 PL-19535 PL-22335 PL-22419 PL-2 PL-22285 PL-21433 PL-21945 PL-22214 PL-22593 PL-22505 PL-22314 PL-22472 PL-22013

v2026.1.76

v2026.1.76 — MAX integration, Loyalty Programs & Segments optimization

Migration

> DANGER
> **Migration is required for the platform to function correctly.**

To perform the migration, follow these steps:

1. **Replace the platform files with the files from the new build.**
2. **Set the `CH_READ_ACTIONS: true` parameter in the configuration file on ALL hosts.**
3. **Run `./akd upgrade` and confirm the migration.**
4. **Set the `CH_READ_ACTIONS: false` parameter in the configuration file on ALL hosts.**
5. **Fully restart the platform using the `./akd restart` command.**

The migration is resource-intensive and requires updating all profiles across all account databases. Mo are involved in the process.

Additionally, in this update, all active mailings that do not use a resource will stop working.

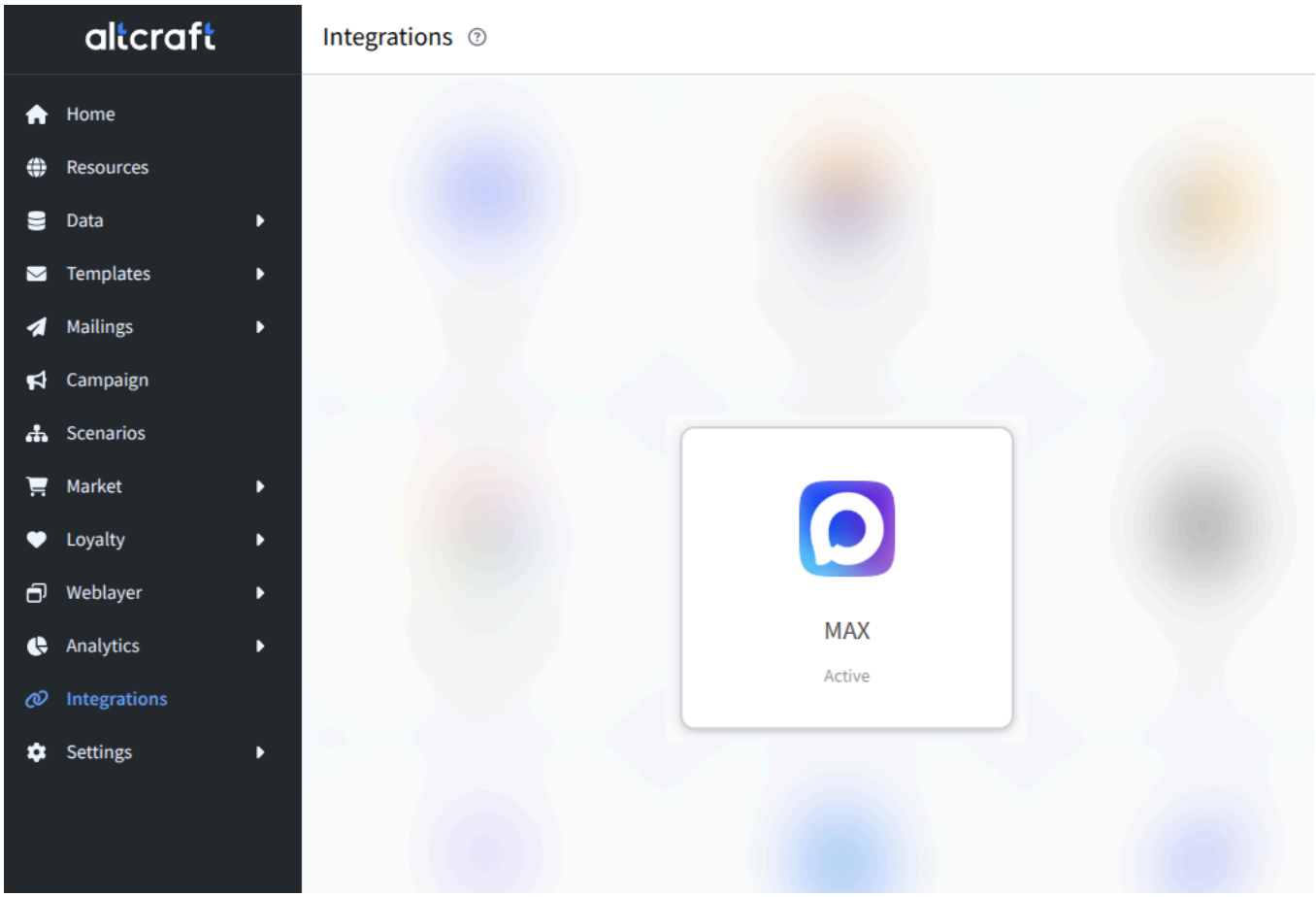
To ensure mailings continue to work correctly, create a new or select an existing resource and assign

A list of such mailings can be obtained using the command `./aktool mailings upgrade-blocker-list` or `./akto account-id 12345` (for a specific account).

MAX messenger integration

This update introduces support for a new communication channel — integration with the MAX messeng available:

- **MAX Bot — for sending personalized messages to users**
- **MAX Group — for sending messages to group chats**



You can create a message template for the MAX messenger using the built-in template editor. A templa elements:

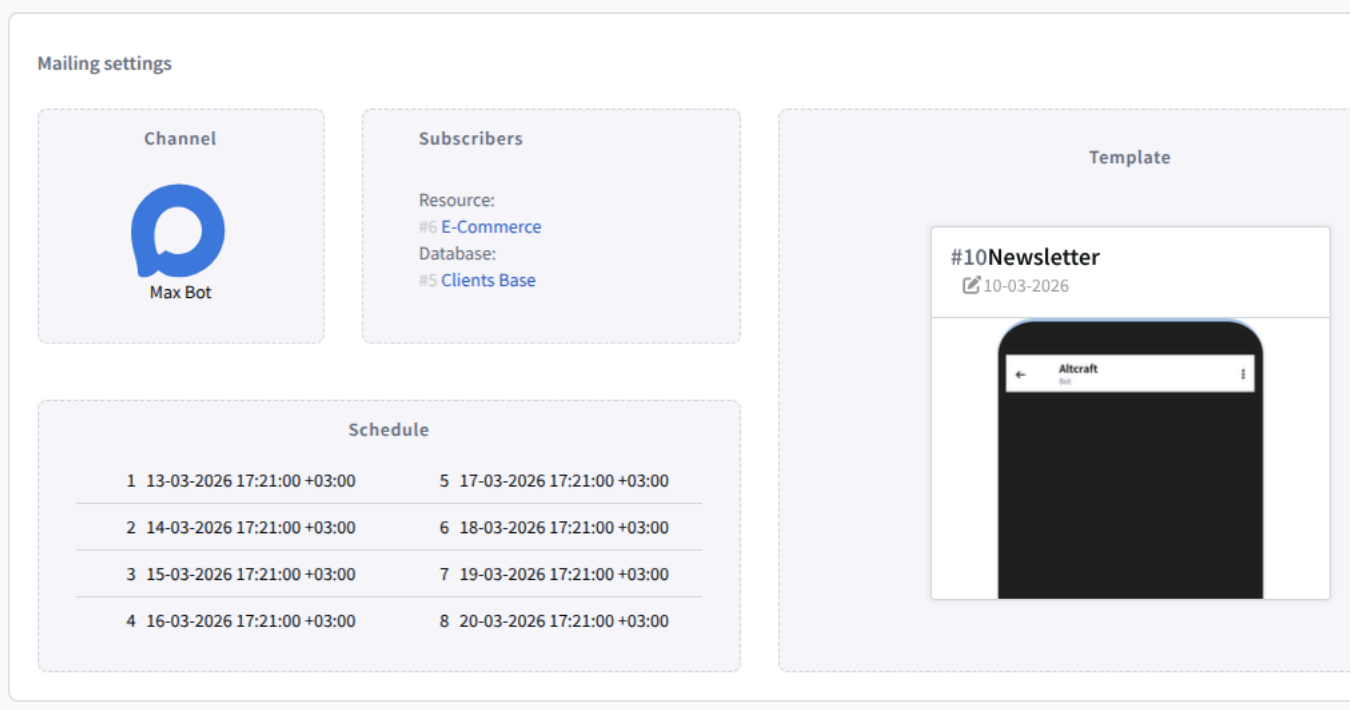
- **Text.**
- **Image.**
- **Video.**

- **Audio.**
- **File.**

You can also immediately add interaction buttons for recipients:

- **Text button — displays a button with text.**
- **Link button — opens a specified link.**
- **Callback button — returns a specified text or command to the bot after clicking.**
- **Open application — opens a mini-application.**
- **Request location — requests the user's location.**
- **Request contact data — requests the user's contact details and phone number.**
- **Button group — a combination of buttons of several types.**

The channel can be used in mailings, scenarios, and campaigns, and analytics is also available:



Segmentation Optimization

Processing of exists / does not exist conditions for channel events (send, open, click, etc.) has been op segmentation:

- **up to 90x faster for simple conditions.**
- **up to 8x faster for complex conditions with multiple filters.**

The algorithm for recalculating static segments based on dynamic conditions has also been optimized, speeding up recalculation of large segments.

To use the accelerated segmentation mechanism, complete the migration steps described at the begin

Other

- **When importing into the stop list, a report with rejection reasons is now displayed (similar to datal**
- **Added the ability to update object attributes (for example, completed Mailings).**
- **Fixed an issue where Mailings inside a campaign were not sent when using dynamic segments.**
- **Fixed an issue with `procloyalty` startup on dedicated servers.**
- **Product segments can now be used in order filters.**
- **Added logging for Web Push notifications in Safari.**
- **Fixed an issue where links did not work in preview for Telegram bot templates and Custom channe**
- **Improved loyalty program analytics interface.**
- **Fixed an issue where it was impossible to add a Mailing via a Custom channel after cloning a Scen**
- **Optimized performance of loyalty program APIs.**
- **Fixed event duplication in ClickHouse. Added deduplication and reversal mechanisms for duplicate pixel goals.**
- **Fixed an issue with profile entry into a Scenario when using scheduled segments with the "in dyna**
- **Fixed an issue where profiles returned to a previous node if the "Condition" node was unavailable.**
- **Fixed an issue where static segments with conditions were not recognized as updatable.**
- **Profiles now correctly exit the "Wait for event" node in Scenarios.**
- **Fixed incorrect display of Scenario IDs in condition nodes.**

- **Attributes can now be selected in segment or RTG setup even if no value is assigned.**
- **Deactivated and hierarchically assigned attributes are now available for viewing in entity objects.**
- **Fixed issues related to form creation and configuration.**
- **Fixed an issue where content in the event capture storage was saved when it should not have bee**
- **Fixed an issue when creating a transaction with points accrual and a 5-year expiration.**
- **Added message lifetime (TTL) configuration. You can now define message validity in Smart Mailing triggered Campaigns. Messages with expired TTL are guaranteed not to be delivered.**
- **Segmentation by Custom channel events now works correctly.**
- **Fixed issues with parallel execution of identical tasks on different cluster hosts.**
- **Fixed an issue where scheduled segments did not start in Scenarios linked to Campaigns.**
- **Unsubscribe events via API are now correctly captured in event tracking.**
- **Link shortening in Telegram communications works correctly again.**
- **Reduced memory consumption of certain platform processes.**

References

v2026.1.76.0: PL-21468 PL-16462 PL-21464 PL-21432 PL-21456 PL-21443 PL-20569 PL-21441 PL-21439 I 15793 PL-21420 PL-21418 PL-21397 PL-21413 PL-21360 PL-21369 PL-21377 PL-21402 PL-18631 PL-2137 19366 PL-21229 PL-21364 PL-21357 PL-20988 PL-21193 PL-21294 PL-21295 PL-17213 PL-21269 PL-2120 21217 PL-21214 PL-21162 PL-21190 PL-21183 PL-21003 PL-21164 PL-17598 PL-17820 PL-17675 PL-2113 21045 PL-19209 PL-21004 PL-20969 PL-20775

v2026.1.76.951 — Fixes and improvements

> **CAUTION: Important**
>
> **All platform processes now write logs exclusively in JSON format. If you use external log collection s into account when configuring.**

> **INFO**
>
> **This update changes the event registration logic in the `eventer` pipe for custom channels.**
> - **The `event_type` field is no longer used for system events (`read`, `open`, `click`, etc.). If the pipeline co channel will not pass validation. Remove this field, leaving only `action_type`.**
> - **The exception is the `stop`, `moderate_pass`, and `moderate_error` events, for which `event_type` remains re**

Improved

- **Base Docker image has been updated to Ubuntu 24.04.**
- **Optimized the performance of the webcontrol process when calculating dynamic segments.**
- **Optimized the operation of API, segments, mailings, and scenarios — read requests are now distrib**
- **Logs from all processes are now correctly written in JSON format.**
- **Fixed an issue where a disabled loyalty module continued to create queues.**
- **Fixed an issue where support chat history was lost after page reload.**
- **Added performance monitoring metrics for mailings.**
- **Added API handlers `loyalty/register_member_batch` and `loyalty/register_member_batch_task` for bulk profile i**

Other

- **Fixed profile looping in the “Condition” node when no filters were present.**
- **Improved loading speed of the static and updatable segment editing page.**
- **Fixed an error with date substitution via variables in segmentation queries to external databases.**
- **Fixed an issue where synchronization of updatable segments with SQL databases was not working**
- **Fixed recalculation of segments with segmentation queries containing “array” type parameters.**
- **Fixed an issue where segment search was not working when importing conditions.**
- **Calculations with “OR” conditions inside nested segments now work correctly.**
- **Fixed an issue where custom channel mailings were not sent when the template contained require**
- **Links in the preview of custom channel templates now open in a new window.**
- **Added link validation for buttons in Telegram templates — a template with an invalid link can no lo**
- **Fixed JSON validation when saving templates — templates with dynamic JSON containing variable**
- **Variables in custom channel mailing fields are now correctly substituted.**
- **Fixed the resource filter in profile subscriptions — the filter no longer resets when deleting a subs**
- **Fixed an issue where a profile with a custom channel subscription could not be saved.**
- **Fixed an error in the “Condition” node when checking the total purchase amount in the Market mo**
- **Conditions for selecting updatable product segments are now available in scenarios.**

- **Fixed column display in analytics.**
- **Users without “Master” permissions can now save promo codes.**
- **Fixed matching by `profile_id` in scheduled imports.**
- **Fixed list filtering by attribute with the “Not set” value.**
- **Fixed minor UI issues related to the block editor and browser navigation.**

References

v2026.1.76.951: PL-21645 PL-21633 PL-21630 PL-21613 PL-21595 PL-21599 PL-21615 PL-21606 PL-2159 21583 PL-21565 PL-21498 PL-18083 PL-20863 PL-21562 PL-21477 PL-21548 PL-13978 PL-21029 PL-2141 21338 PL-21187 PL-21191 PL-21184 PL-20897 PL-20634 PL-20940 PL-19867 PL-15846 PL-21484 PL-2153 21499 PL-21386 PL-21435 PL-21385 PL-21169 PL-21424 PL-21475 PL-21457 PL-21454 PL-21384 PL-2066 21416 PL-20179 PL-21400 PL-21346 PL-21354 PL-21322 PL-21277 PL-21316 PL-21270 PL-21138 PL-2112 21016 PL-20713 PL-20983 PL-20985 PL-17974 PL-20964 PL-20667 PL-18947 PL-19639 PL-18914 PL-1897

---

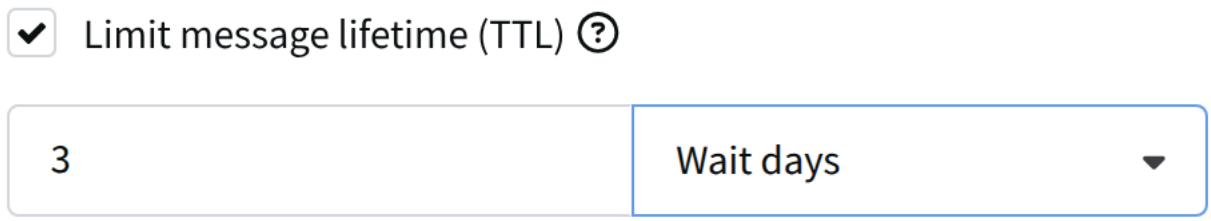
v2026.1.76.1578 — TTL for messages

Added

TTL for sent messages

Message time to live (TTL) is the period during which the system tries to send a message from the que delivered during this time, sending stops and the message is deleted from the queue.

TTL should be used for campaigns that are only relevant for a limited time. For example, when a messa reminder, one-time code, or offer that quickly loses relevance.



To configure TTL, enable the Limit message time to live (TTL) parameter in the campaign Smart settin days, hours, minutes, or seconds.

After the specified time expires, the system stops trying to deliver the unsent message and registers a

TIP: Details of using TTL for messages

- **The TTL value must be within the limits set by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration param**
- **If TTL is set in several places, the value with the highest priority is applied: API parameter `expirat` configuration parameter `MTA_TTL_DEFAULT`.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefo different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts, even if the channel can still technically de**
- **TTL does not work for channels that do not return message sending statuses: Telegram, MAX, an**

Improved

- **Added the “Segment participation” filter for product and SKU segments.**
- **Removed the limit of 30 values for a single attribute.**
- **Improved UI/UX components.**
- **Added deduplication for web events and mSDK events.**

Other

- **Connections to the MTA server no longer accumulate in the cache or cause performance issues.**
- **Fixed an error where the delete button for an active user was available in that user’s settings.**
- **Improved stability of the `procworkflow`, `api`, `webcontrol`, and `procloyalty` processes.**
- **Fixed some errors that occurred when calculating the GCG.**
- **Variables are now correctly substituted into URLs inside the “API call” scenario node.**
- **Fixed an error where the Play Mobile SMS gateway did not return campaign statuses.**
- **The “at least [n] times within a period” condition in segments now works correctly again.**
- **Fixed errors in the handler for updating profile fields.**

References

v2026.1.76.1578: PL-21638 PL-21815 PL-21755 PL-21800 PL-21803 PL-21784 PL-21772 PL-21765 PL-217 PL-21740 PL-21738 PL-21718 PL-21714 PL-21659 PL-21710 PL-21702 PL-21693 PL-21697 PL-21365 PL-2 PL-21537 PL-21502 PL-21390 PL-21389 PL-10664 PL-21329 PL-21290 PL-21293 PL-21273 PL-18968 PL-2 PL-20870 PL-20652 PL-20650 PL-20620 PL-20339

Other

- **Fixed an issue where a segment could not be opened in the interface.**
- **Fixed an issue where messages were not sent via custom channels after changing the event regist compatibility has been restored for channels that had the `event_type: custom` field in their configurati**
- **Fixed an issue where clicks on link buttons in MAX messages were not registered in analytics.**
- **Changed the logic for registering system events in custom channels — the `event_type: custom` field is `click`, and similar events.**
- **Added a configuration parameter for setting a proxy server specific to the Telegram channel, allow routed through a dedicated proxy without affecting other channels and services.**
- **Optimized API process memory consumption, preventing crashes due to out-of-memory conditions.**
- **Fixed field mismatch in Action Hook — event data structure now matches the documentation.**
- **Removed the limit on the number of attribute values. Added search in the default value selection li**
- **The delete button is no longer displayed for active users in settings.**
- **Fixed an error that occurred when saving promotions in the loyalty module.**
- **Added a periodic SSH connection cache cleanup mechanism with `SSH_CACHE_CONNECTION_TIMEOUT` (conne `SSH_CACHE_CONNECTION_CLEANUP` (cleanup interval) parameters.**
- **Fixed variable substitution in URLs for the "API call" element in scenarios.**
- **Fixed delivery status retrieval when sending SMS via the PLAYMOBILE gateway.**
- **Fixed the "at least [n] times within a period" condition in dynamic segments — search now covers**
- **Fixed a JS error in the template editor that occurred when inserting a variable.**
- **Optimized procleadsaver and cookie_saver processes.**
- **Fixed control group (GCG) calculation when a segment is selected as the audience source.**
- **Optimized PostgreSQL locks in the loyalty module.**
- **Added the ability to specify message TTL via API when launching triggered campaigns.**
- **Fixed an issue where updatable segments were not displayed for selection in mailing settings.**
- **Fixed attribute inheritance in child objects.**
- **Added stratification condition validation in campaigns — empty conditions can no longer be saved.**
- **Fixed segment operations when the user lacks permission to read and export the database.**
- **Fixed encoding of form fill files sent by email — files now open correctly in Excel.**
- **Fixed profile movement from control group to test group when stratification conditions change.**
- **Fixed delivery status retrieval when sending via relay.**

References

v2026.1.76.1642: PL-21366 PL-21864 PL-21829 PL-21736 PL-21802 PL-21409 PL-20634 PL-21449 PL-210 PL-21880 PL-21791 PL-21868 PL-21500 PL-21826 PL-21387 PL-21646 PL-21636 PL-21692 PL-21684 PL-2 PL-20843 PL-21629 PL-20150 PL-21601 PL-21554 PL-21199 PL-21550 PL-21345 PL-18234 PL-20786 PL-2 PL-20862 PL-20308 PL-20735

v2025.4.75

v2025.4.75.0 – Loyalty Programs, New Segmentation, UI Update

CAUTION

To ensure correct platform operation, a migration is required. To do this:

1. **Replace the platform files with files from the new build;**
2. **Run `./akd upgrade` and confirm the migration;**
3. **Fully restart the platform using `./akd restart`.**

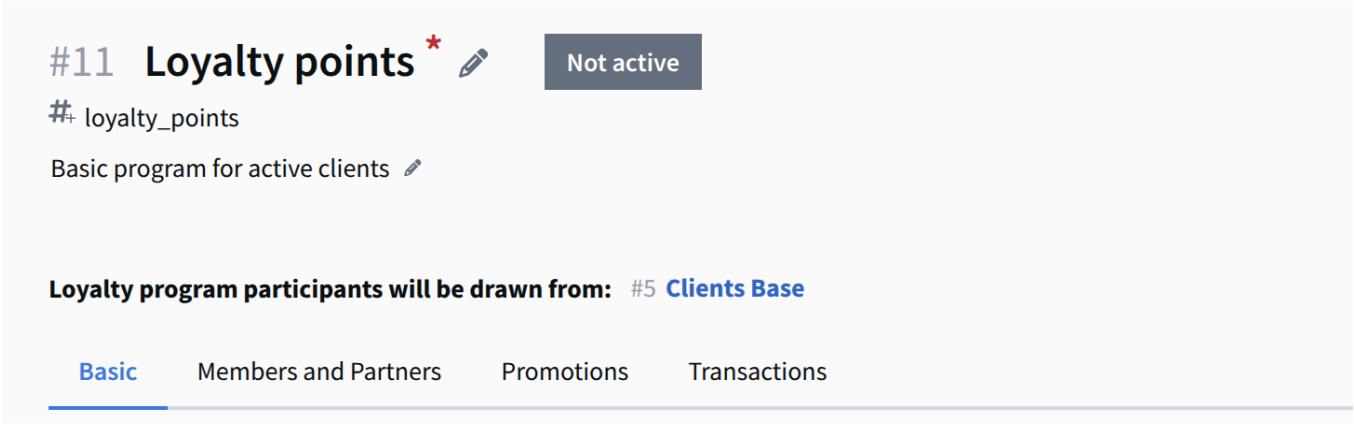
You also need to add the following required objects to the `main.json` configuration file:

- **`SYSTEM_REDIS` — main Redis storage configuration**
- **`LOYALTY_CACHE` — Redis storage configuration used for loyalty program operations**
- **`PROCWORKFLOW_DISTRIBUTED_CACHE_DB` — parameters of the distributed cache storage used by the `procwork`**
- **`CAMP_DUPLICATESDB` — Redis configuration for the duplicates database**

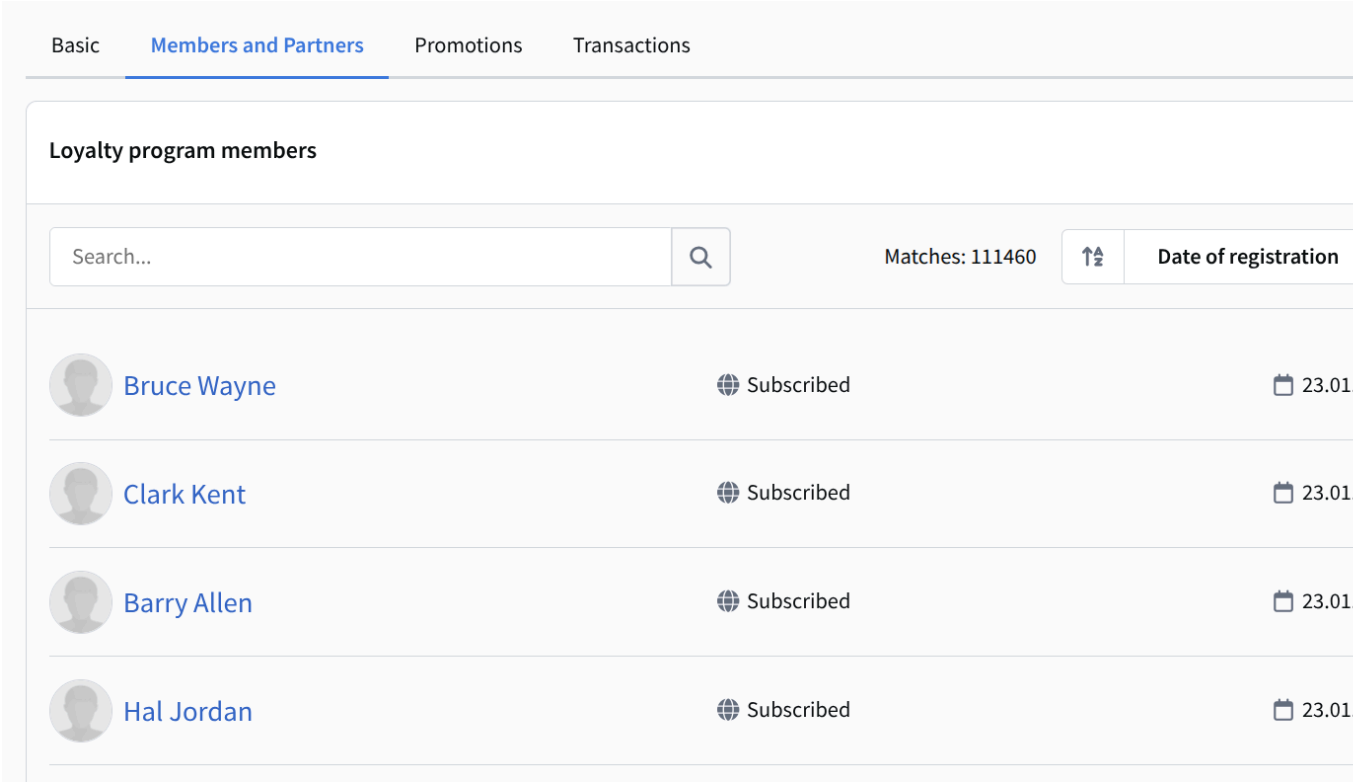
More details about these configuration parameters are available in the administrator documentation.

Loyalty Programs

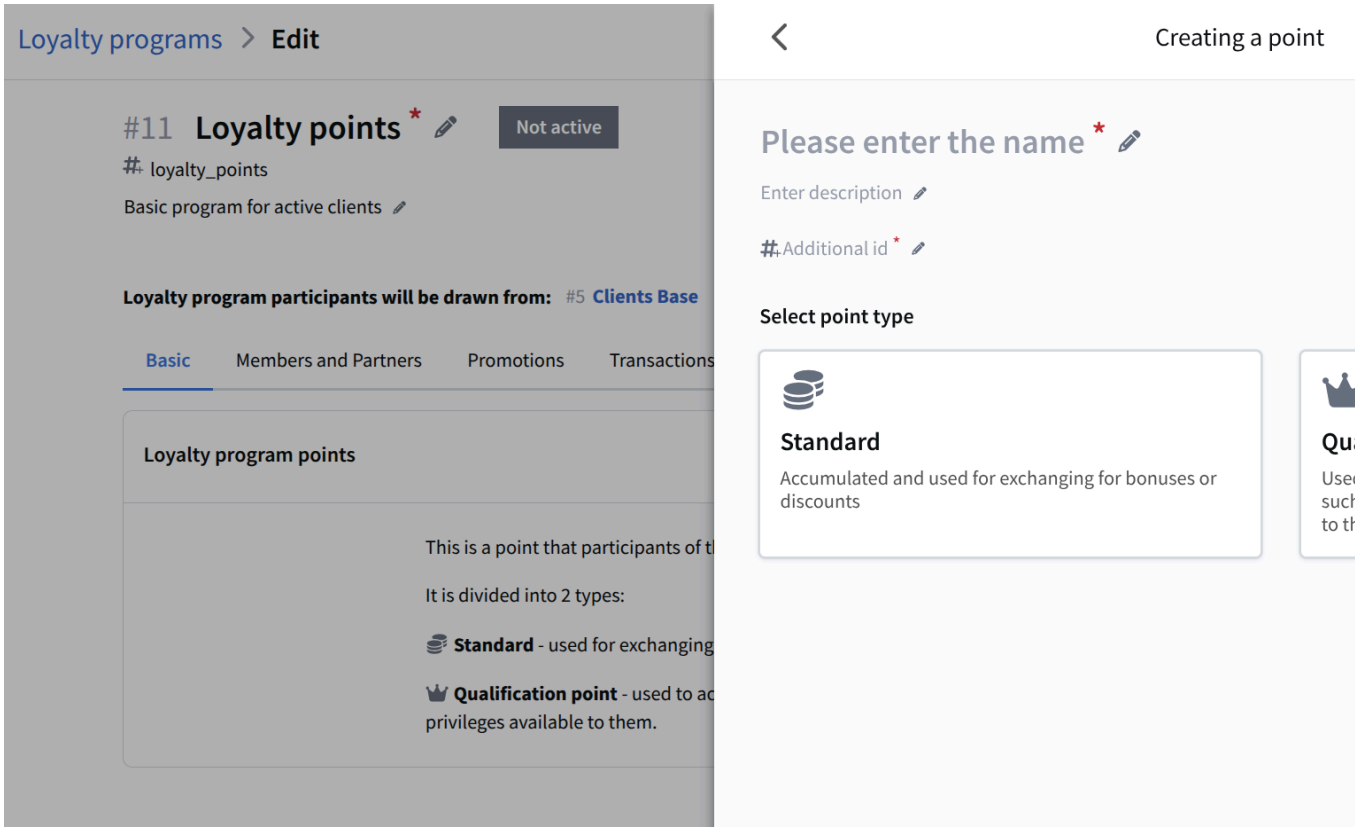
In this update, the Loyalty Programs module has been added to the Altcraft platform. A loyalty program target actions such as purchases, registration, content views, and other activities. Customers earn po discounts, movies, plans, and services.



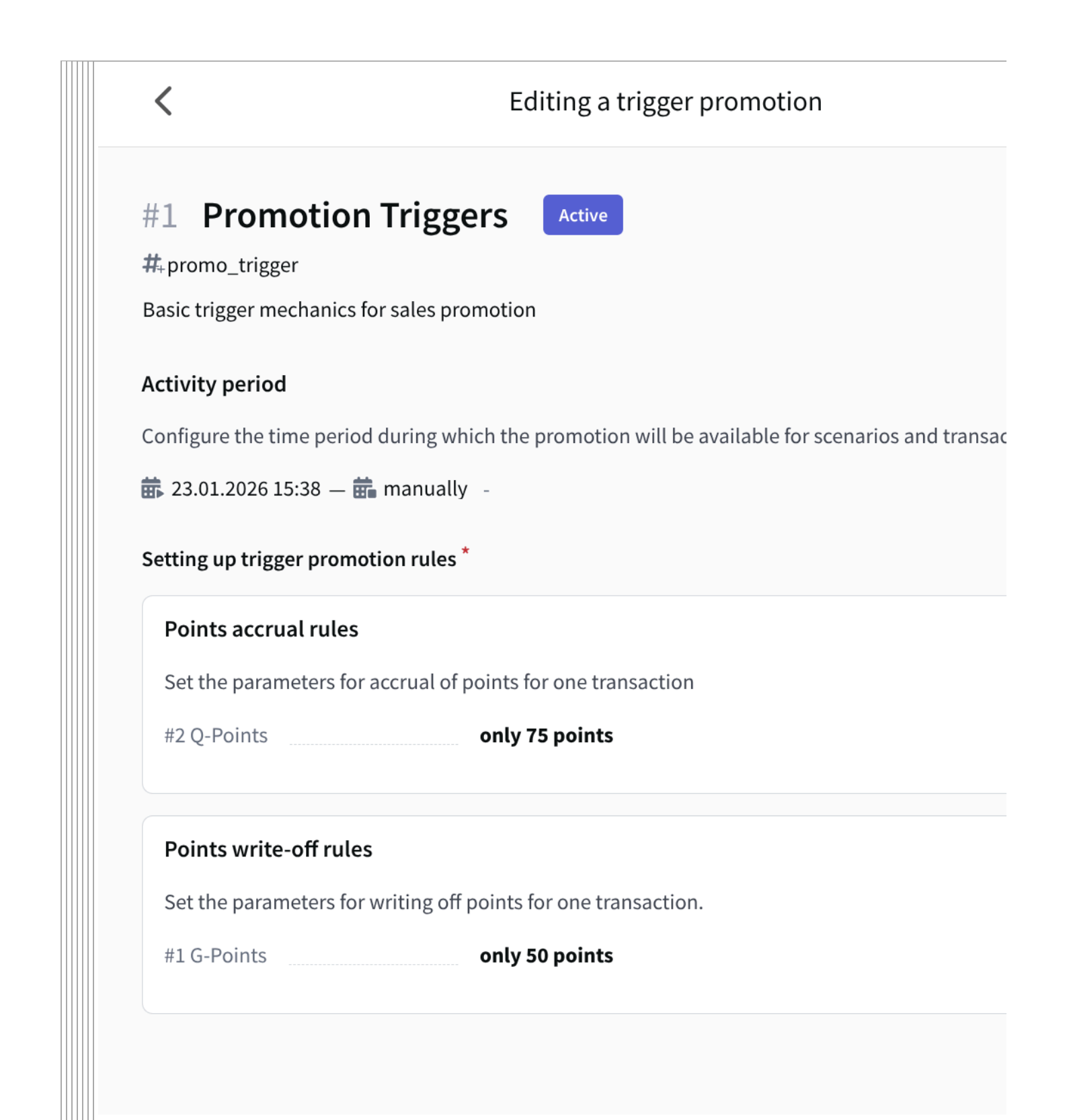
All profiles in the database or a specific segment can participate in a loyalty program. Participants car requests:



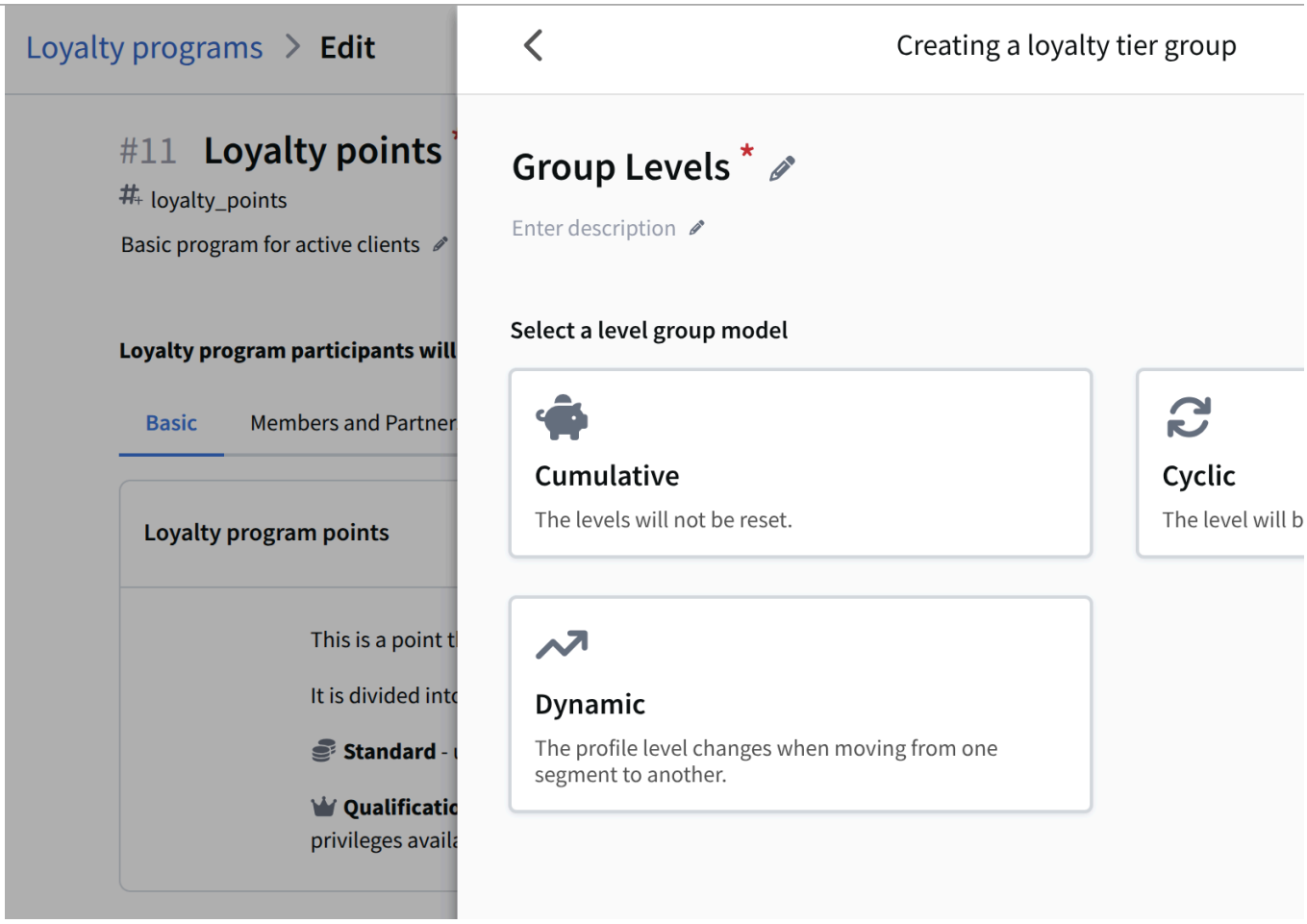
Loyalty program participants can earn points. Points are divided into standard and qualification points. rewards, while qualification points determine the participant's level. Each point type can be assigned a



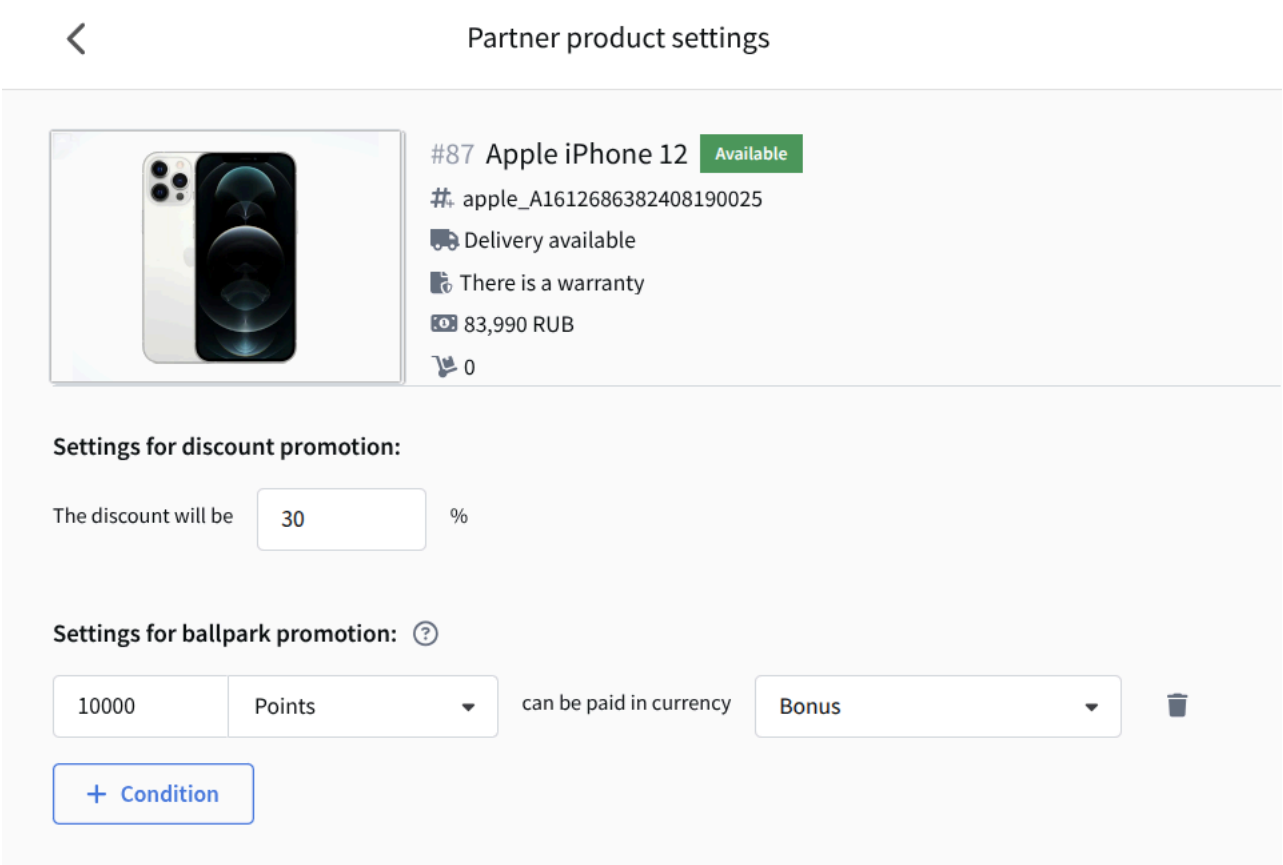
Point accrual and redemption are managed using trigger-based promotional campaigns:



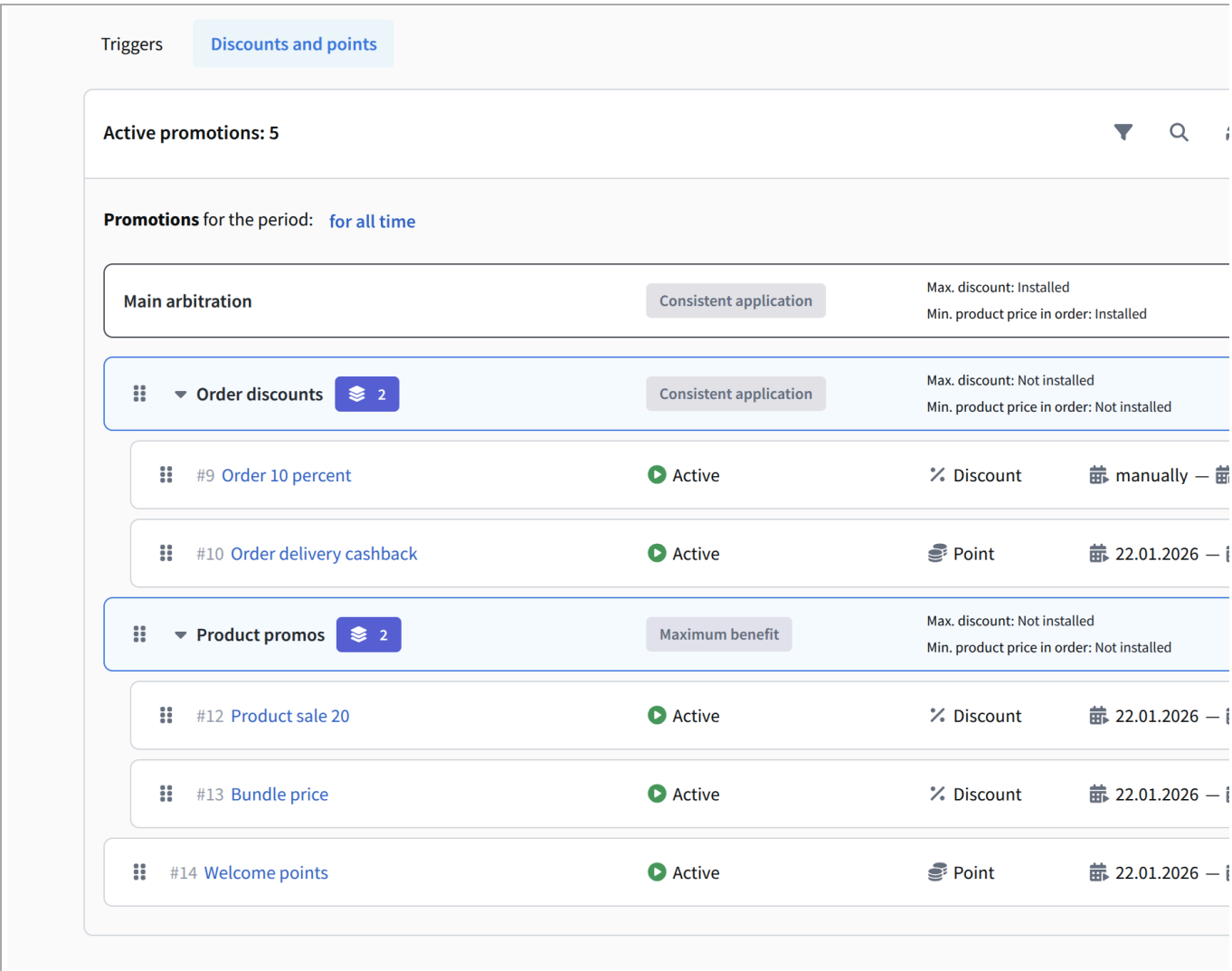
For loyalty levels, groups are configured with their own rules for status review and point reset. Differen mechanics can be set up for participants at each level:



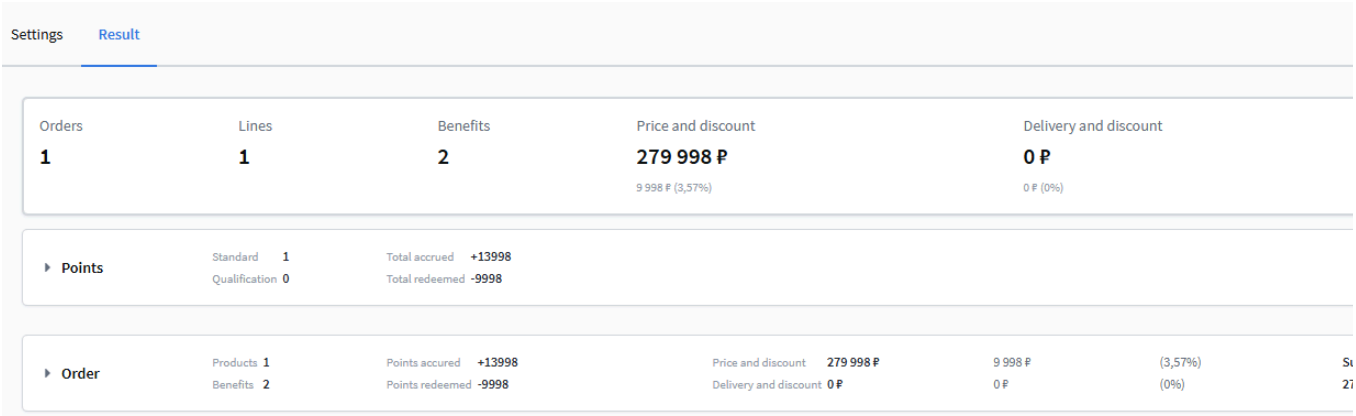
You can configure partners within a loyalty program. Special offers are available for partner products f



Discount and points-based promotional campaigns handle rewards for ordering products from the Mark campaigns and fine-tune their interaction using arbitration:



After configuring points and promotional campaigns, you can verify that everything works correctly usi
You can perform a preliminary order calculation for both an existing profile in the database and a test p
context:



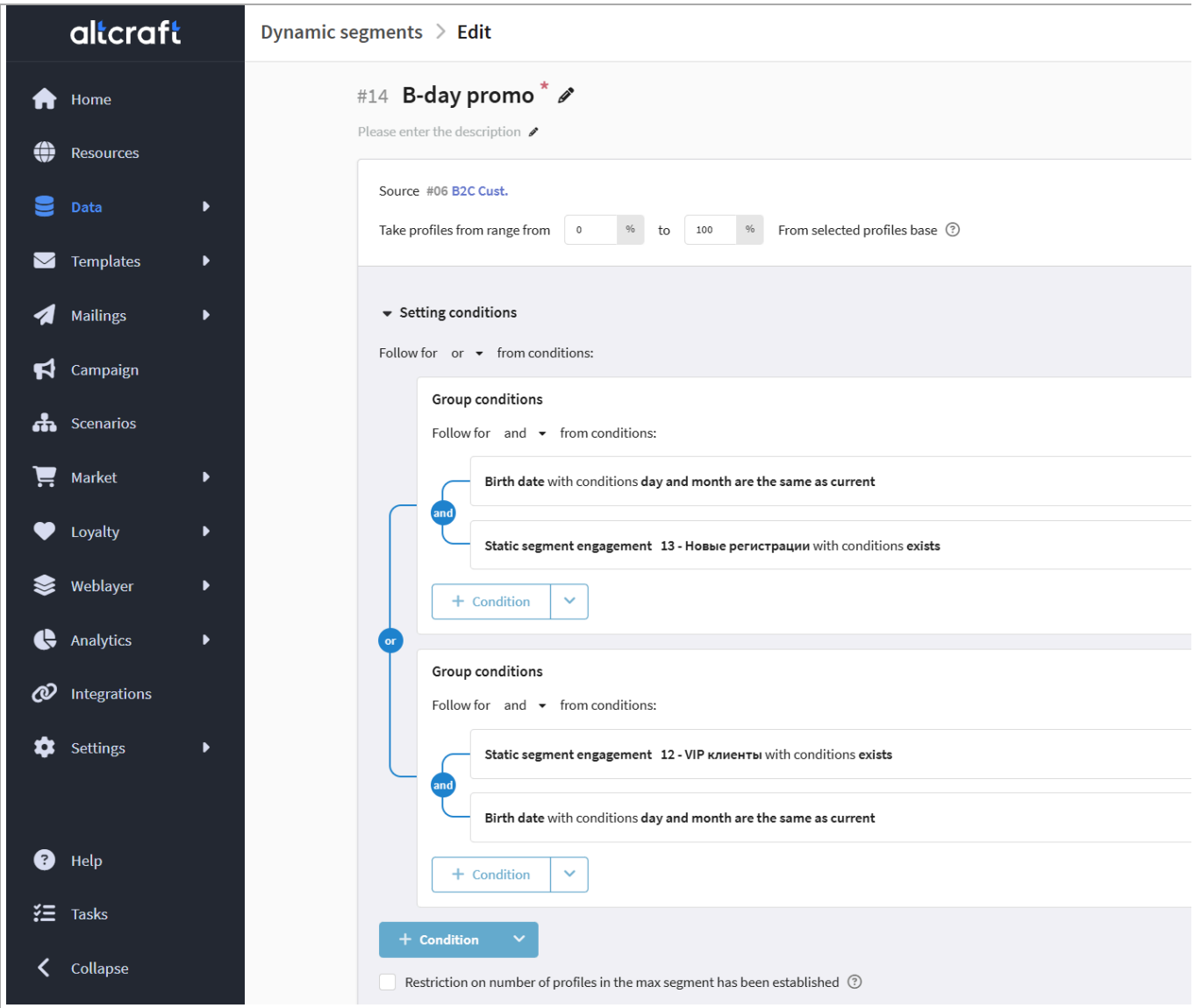
Testing makes it clear how and in what order benefits are applied for loyalty program participants.
Learn more about creating and configuring loyalty programs here.

Updated Segmentation

Major changes have been introduced to the profile segmentation mechanism. The key update is a redes

- **Static segment — an entity for storing profiles, filled manually or via API. The segment is not recal automatically. It can be saved empty; only a name and database selection are required.**
- **Dynamic segment — essentially a set of conditions rather than a list of profiles. The segment is re display selection results without saving the segment.**
- **Updatable segment — cannot be filled manually or via API. Profiles are added based on defined cor via manual recalculation.**
- **Quick segments have been removed. Use a dynamic segment instead and click Calculate when cor**

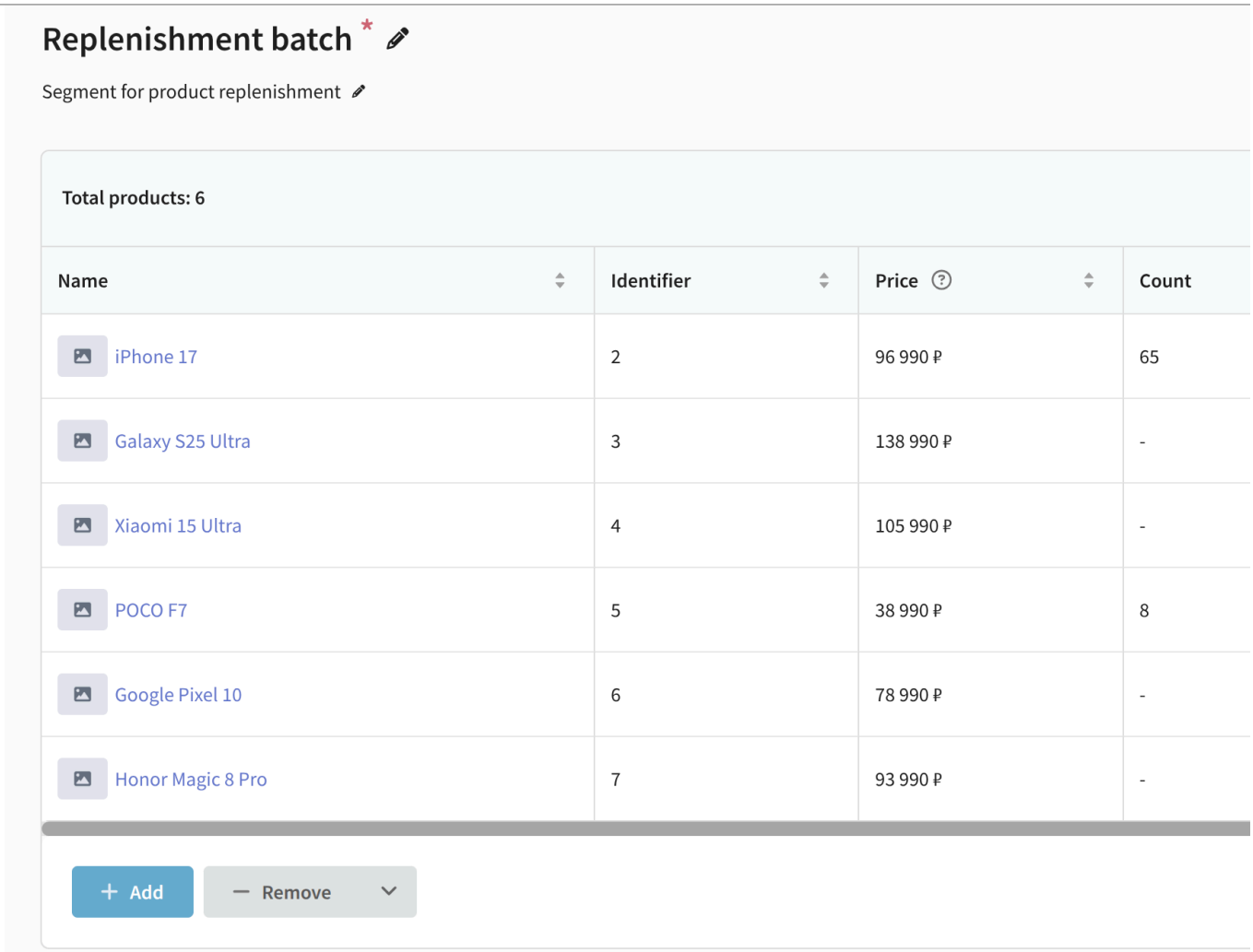
The segment configuration menu UI has also been significantly redesigned:



The interface is now more modern and intuitive. More details about segmentation are available in [this](#)

Product Segments

Within the Market, you can now group products into product segments. These are sets of products and specific attribute:



There are two types of segments: static (grouped manually or via API using selection conditions) and u recalculated on a schedule). A completed segment can be exported and used for sales data analysis.

Instructions for configuring product segments are available in [this article] of the Market documentation

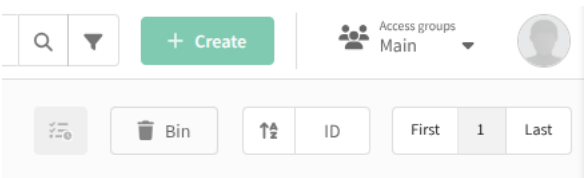
UI Updates

Along with the update, the interface has been improved:

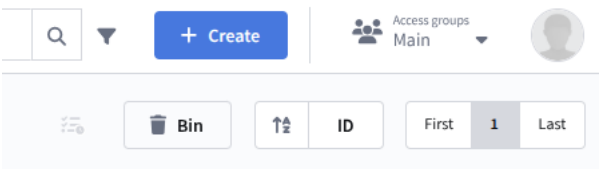
- **Updated the platform's color palette;**
- **Changed the appearance of some components, such as the date range picker;**
- **Improved the display of auxiliary side menus (for example, stratification in control groups);**
- **Tags and attributes in lists now look more consistent with the overall design;**
- **Fixed several table display issues.**

Color Palette

Before:

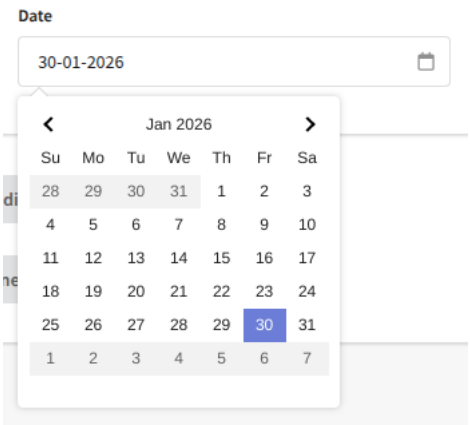


After:

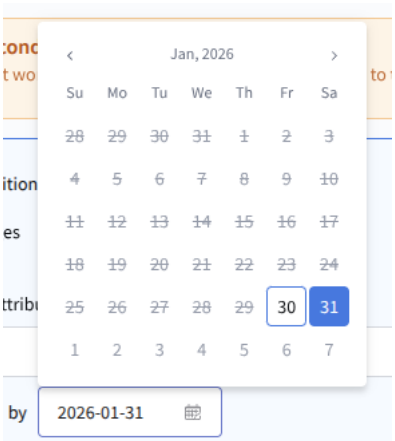


Date Ranges

Before:

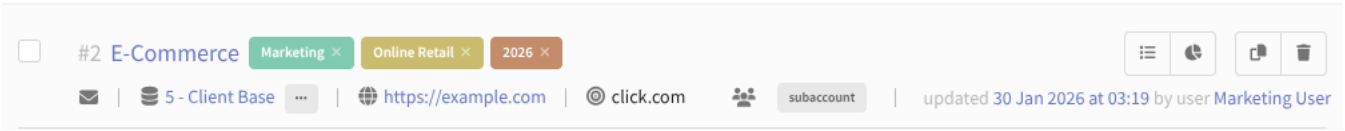


After:

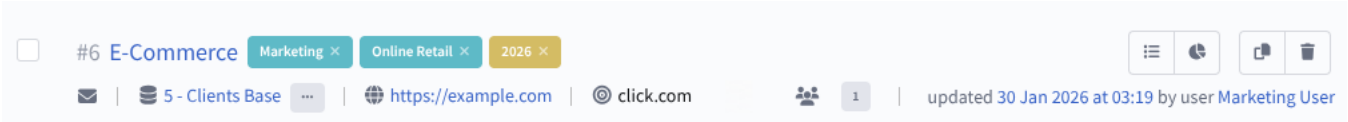


List Display

Before:



After:



These small improvements enhance usability, make the interface more cohesive and modern, and redu

Keycloak Integration

This update adds support for integrating the platform with Keycloak. Administrators can configure acc authentication and assign groups and roles according to Keycloak settings.

Connector creation and access configuration are performed via the Admin Panel.

Other

- **Changes in virtual sender settings are now saved correctly again.**
- **Fixed an issue where the `CLICKHOUSE_SYSTEM.CONN_TIMEOUT` parameter affected the overall host connecti connection attempt.**
- **Control groups no longer cause errors after being recreated.**
- **Fixed an issue where a required custom channel field could become optional.**
- **Email addresses are now correctly converted to lowercase when creating LDAP users.**
- **Fixed an issue where the regular expression for arrays only validated the first array item and failed**
- **Telegram template previews in scenarios now display correctly.**
- **Improved display of errors and warnings across the platform.**
- **Duplicate emails warning about profile import limits are no longer generated.**
- **Fixed an issue where some scenarios could not be saved.**

- **Profile import speed has been optimized.**
- **Deleted custom events no longer appear in the event capture list and no longer trigger events.**

References

v2025.4.75.0: PL-20429 PL-20573 PL-20313 PL-20551 PL-20498 PL-20474 PL-20485 PL-20095 PL-19769 I
20344 PL-20141 PL-20057 PL-19157 PL-19942 PL-20372 PL-20306 PL-17956 PL-20270 PL-20273 PL-1910
19087 PL-18576 PL-20170 PL-19912 PL-19289 PL-19935 PL-19938 PL-19762 PL-20032 PL-20060 PL-1978
19455 PL-18995 PL-19781 PL-19639 PL-17787 PL-18173 PL-19851 PL-19548 PL-17393 PL-18637 PL-1910

---

v2025.4.75.1005 – Loyalty fixes, minor improvements

Improved

- **Added the ability to wait until a specific date in the "Event Waiting" node.**
- **Added validation to ensure a template is selected when launching a mailing, as well as a restrictio mailing.**

Other

This release includes fixes in the loyalty program module, as well as a number of other issues.

- **Creating a mailing by a non-master user no longer causes errors.**
- **Fixed an issue where some scenarios were not attached to campaigns.**
- **The restriction on repeated form submission works correctly again.**
- **Added validation and inheritance of configuration parameters related to Redis storage.**
- **Fixed issues in collecting and displaying loyalty program analytics.**
- **Events in the campaign log are now displayed correctly.**
- **Improved stability of the `webcontrol` process.**
- **Added filtering by statuses in campaign lists.**
- **Fixed an issue that occurred when attempting to save a campaign.**
- **The platform now displays correct error information in case of connection issues with Yandex.Aud**
- **Navigation to a template from the preview in the template list now works correctly.**
- **Fixed an issue where the profile ID was not passed when exporting form data.**
- **The time in the "Pause" scenario node now updates correctly.**
- **Fixed an issue that prevented entering a value less than 100% in virtual sender rules.**
- **Filtering by groups in the loyalty program list works correctly again.**
- **Fixed issues that occurred when opening an email template.**
- **Deactivating a mailing during sending now works correctly.**
- **Fixed several issues in the visual editor.**
- **The preview of the AMP version of an email template now opens and displays correctly.**
- **Fixed an issue where the provider was not specified in complaint events.**
- **The date in event capture and API is now passed correctly.**
- **Fixed an issue where the preview in custom channels was always displayed as an HTML message.**
- **The `subscription_fields` variable is now correctly passed in test sendings.**
- **Fast segment calculation with a query to an external database now works correctly.**
- **Added validation to prevent linking to a non-existent group in the platform.**

References

v2025.4.75.1005: PL-20912 PL-20916 PL-20899 PL-20826 PL-20881 PL-20624 PL-20847 PL-20791 PL-207
PL-20767 PL-19819 PL-20760 PL-20205 PL-20757 PL-20744 PL-20754 PL-20749 PL-20718 PL-20644 PL-2
PL-20041 PL-20469 PL-20705 PL-20632 PL-20704 PL-20552 PL-20702 PL-20635 PL-20683 PL-20693 PL-2
PL-20380 PL-20674 PL-20099 PL-20669 PL-20651 PL-20671 PL-20648 PL-20668 PL-20666 PL-20620 PL-2
PL-20069 PL-20653 PL-20649 PL-20647 PL-20637 PL-19347 PL-20233 PL-20148 PL-20625 PL-20606 PL-2
PL-20534 PL-19602 PL-20500 PL-18808 PL-20376 PL-20169 PL-20377 PL-20073 PL-19950 PL-19940 PL-1
PL-19802 PL-19688 PL-18742 PL-19370 PL-19619 PL-16670 PL-18771 PL-19338 PL-19174 PL-18741 PL-1

---

v2025.4.75.2408 – React Native mSDK, product segment improvements

DANGER: Migration

If you have already upgraded to version 75, you need to run a migration to ensure correct operation o

1. **Stop the `webcontrol`, `api`, and `proctask` processes.**
2. **Run the migration using the command `./aktool segments run-separation-migration --config=path-to-configu`**
3. **Start the stopped processes.**

Altcraft mSDK for React Native

Support for the cross-platform framework React Native has been added to the Altcraft mobile SDK. Thi for both iOS and Android applications simultaneously.

You can learn more about integrating and using Altcraft mSDK for React Native here. The NPM packag

Improved

- **Added a filter by participation in segments for product segments, as well as the ability to import c**
- **When creating and editing a template or template fragment, the interface now displays a list of ava**
- **API requests that transmit mailing data now include information about the associated campaign.**
- **Added the ability to update profile fields in the mobile SDK.**
- **Segment audience updates are now blocked if the segment is used as the main segment of a camp**
- **Added a notification about the need to complete a campaign before deleting it.**
- **Added a Redis storage connection check at platform startup.**

Other

- **Attributes are now correctly displayed in the mailing template preview.**
- **JSON from the Object and Array of objects fields in the API request node is now substituted correc**
- **Fixed Viber sending issues via SMSTraffic caused by UTC time in the `start_date` parameter.**
- **Improved collection and display of loyalty program reports.**
- **The event type filter in the profile card works correctly again.**
- **Emails about form submissions no longer contain invalid special characters.**
- **Fixed several issues related to attributes.**
- **The limit in the `http_request` pipe of custom channels now works correctly.**
- **Fixed issues when cloning custom channel templates.**
- **The “exists” condition for string fields in fast segments works correctly again.**
- **Improved stability of the `webadmin`, `proctask`, and `api` processes.**
- **Increased the processing speed of the `trk_pixel_def` queue.**
- **Links in push templates in the beta editor now work correctly.**
- **Fixed issues with connecting to the platform via LDAP.**
- **Fixed an issue where template previews were not saved in FileDB.**
- **Processing of a custom field with the name of a system field now works correctly.**
- **Custom channels can be used in scenarios again.**
- **Fixed an issue where the audit log export contained a time not matching the account time zone.**
- **The `utm_medium` UTM parameter is now updated in the profile card when registering a goal via API.**
- **Fixed an issue where hard bounces from Telegram bot mailings were not registered.**
- **Sending policies are now applied during массовой trigger-based sending of profiles to a scenario.**
- **The character limit is no longer applied when using dynamic variables with long logical conditions.**
- **Fixed an issue where the `/profiles/list` API method did not return all profiles from the database.**
- **Fixed issues with displaying the External object value in custom channel templates.**

References

v2025.4.75.2408: PL-20956 PL-21090 PL-21117 PL-21116 PL-21118 PL-21115 PL-19263 PL-21087 PL-210 PL-17906 PL-20943 PL-21038 PL-20987 PL-21023 PL-20499 PL-20999 PL-20641 PL-21008 PL-20971 PL-2 PL-18467 PL-20802 PL-20846 PL-20780 PL-18407 PL-17935 PL-19747 PL-20682 PL-20719 PL-20652 PL-2 PL-21027 PL-21009 PL-20990 PL-18111 PL-20914 PL-20980 PL-21025 PL-20968 PL-20970 PL-21002 PL-2 PL-20966 PL-20931 PL-20953 PL-17662 PL-20896 PL-20875 PL-20862 PL-20541 PL-20764 PL-20738 PL-2 PL-20201 PL-17754 PL-18981 PL-21018 PL-20992 PL-20815 PL-20967 PL-20963 PL-20456 PL-20962 PL-2 PL-20588 PL-20236 PL-18822 PL-20901 PL-20525 PL-19672 PL-20751 PL-17954 PL-20980 PL-20965 PL-2 PL-20908 PL-20955 PL-20927 PL-20935 PL-20922 PL-20297 PL-2087 PL-20879 PL-20886 PL-20866 PL-20 PL-20814 PL-20789 PL-20720 PL-19344 PL-20808 PL-20714 PL-20803 PL-20707 PL-20790 PL-20786 PL-2 PL-20709 PL-20685 PL-18299 PL-20712 PL-20623 PL-20408 PL-20164 PL-20677 PL-20160 PL-20594 PL-2 PL-16893

---

v2025.4.75.2774 – Custom Authentication, block editor improvements

CAUTION: Sending without a resource will be disabled

In upcoming updates, all active mailings that do not use a resource will stop working.

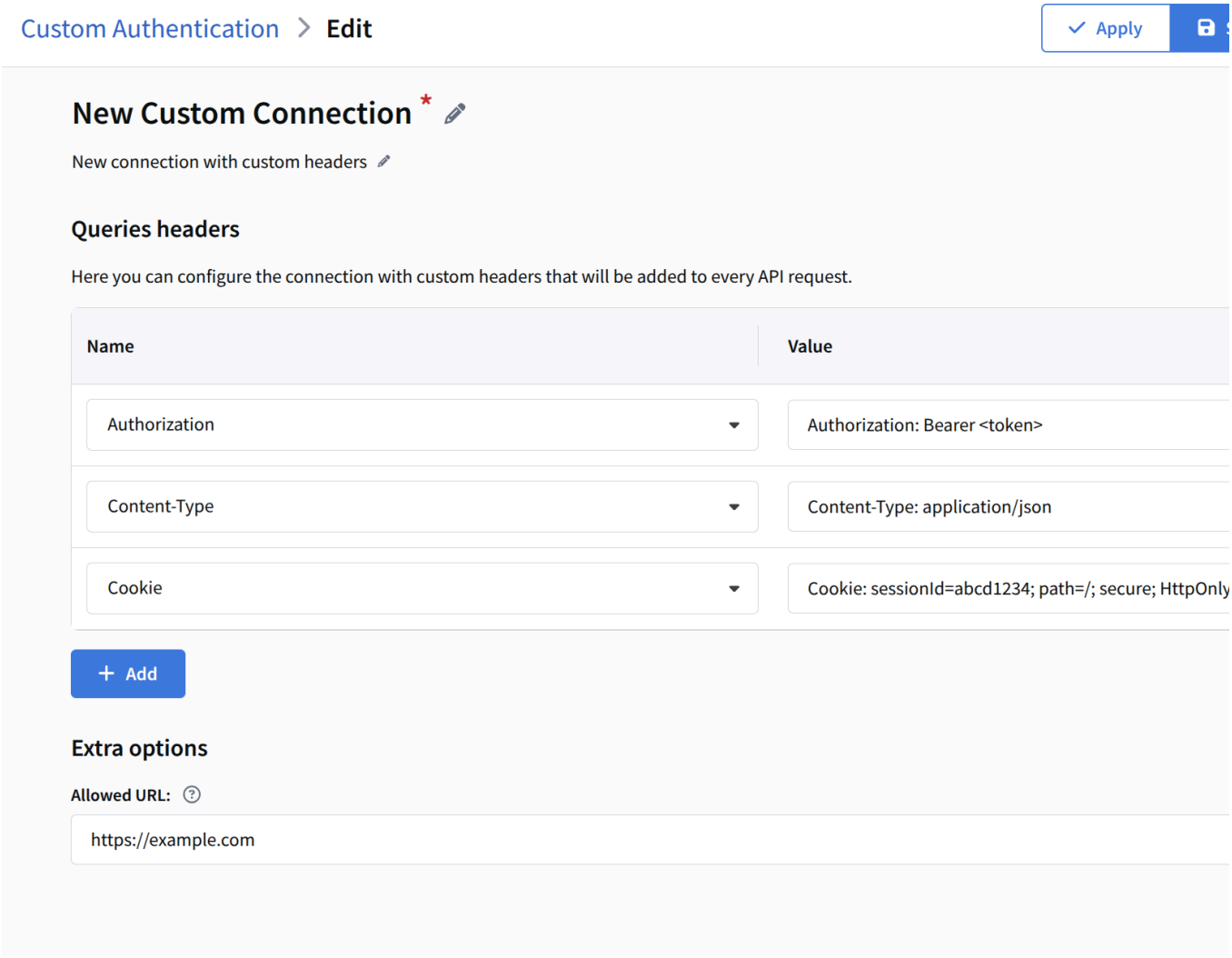
To avoid issues, create a new resource or use an existing one and add it in the mailing settings.

You can retrieve the list of such mailings using the `./aktool campaigns upgrade-blocker-list` or `./aktool camp 12345` command.

Added

A new type of connector for external connections has been introduced — Custom Authentication. This configuration of custom HTTP headers for API requests, including the predefined Authorization header. configuration and sending of API requests and supports validation and error handling.

With this connection type, you can integrate the Altcraft platform with any external service that requir authentication or other purposes, including API requests that use a key in the header.



Learn more about configuring this connection type here.

Improved

The email template block editor has been improved:

- **ability to insert links and emoji into tables added;**
- **save and export UI updated;**
- **proxying for empty images fixed;**
- **visual and UI improvements applied to several interface elements.**

Read more about the block editor in this article.

Other

- **The Altcraft team continues fixing issues in the Loyalty module and improving its stability.**
- **Fixed an issue where the `data/segment` directory was not cleared automatically.**
- **PostgreSQL connection passwords containing special characters now work correctly.**
- **Fixed an issue that prevented creating transactions with a long expiration period for points.**
- **Editing of mailings, scenarios, and campaigns is now restricted if a required attribute is assigned v**
- **Improved platform stability when executing multiple trigger API calls.**
- **Fixed issues with analytics and report display in the platform.**
- **Values in templates for `external object` fields are now saved correctly.**
- **Fixed several issues related to attributes and their value processing.**
- **Campaigns now deactivate correctly even if the deactivation moment coincides with the audience**

References

v2025.4.75.2774: PL-20989 PL-21186 PL-20844 PL-21107 PL-21228 PL-21163 PL-19018 PL-19865 PL-212 PL-21051 PL-21145 PL-21154 PL-16968 PL-21074 PL-21130 PL-21111 PL-20834 PL-20870 PL-21106 PL-2 PL-21102 PL-21099 PL-18812 PL-21064 PL-20938 PL-21000 PL-20071 PL-20932 PL-20066 PL-20973 PL-2 PL-20774 PL-20858 PL-20809 PL-20703 PL-21205 PL-18625 PL-20952

v2025.4.74

v2025.4.74.0 – Campaigns

CAUTION

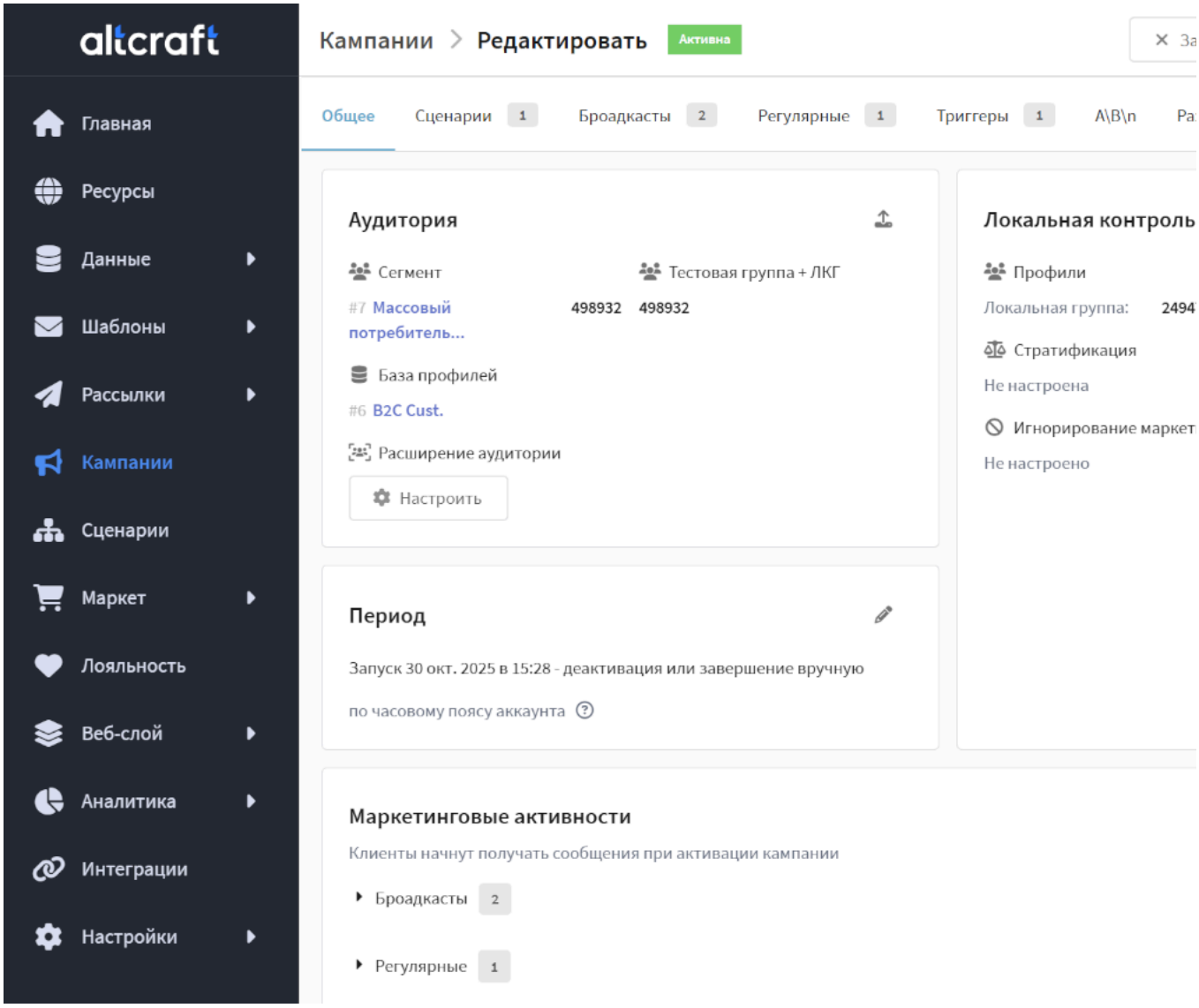
To ensure correct platform operation, a migration is required. Follow these steps:

1. **Replace the platform files with files from the new build;**
2. **Run `./akd upgrade` and confirm the migration;**
3. **Fully restart the platform using `./akd restart`.**

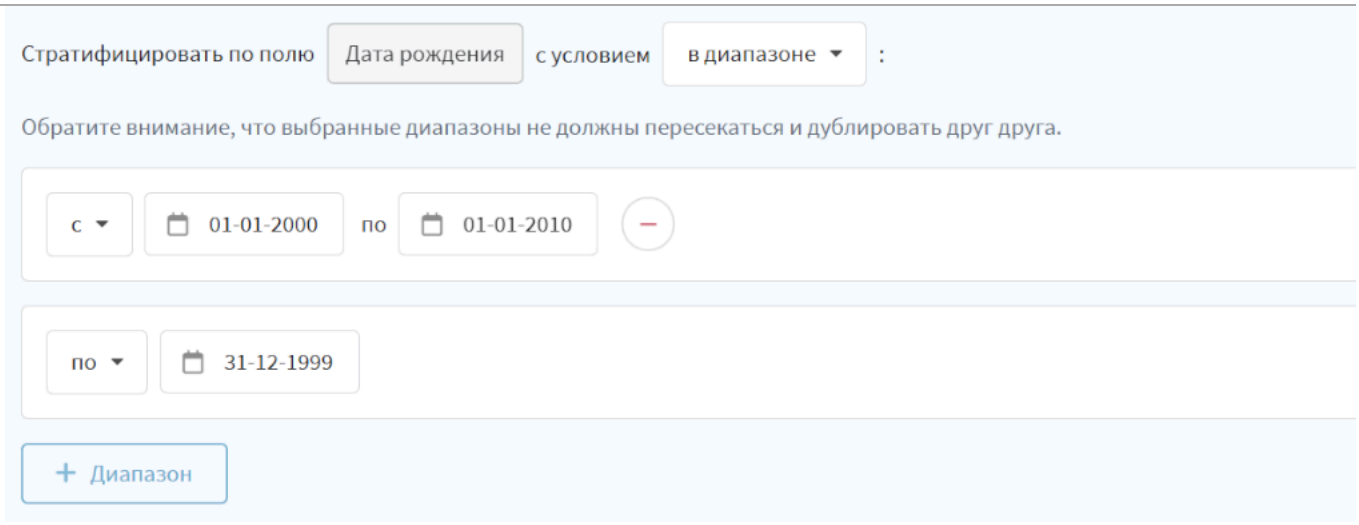
Campaigns: a comprehensive marketing tool

Previously, all types of marketing messages in the platform were called Campaigns. In the new platfor “Mailings”, while Campaigns now represent a tool for comprehensive management of marketing activit

Within a campaign, you can configure audiences, segment databases, manage user profiles, control gr support both creating new and adding existing marketing activities: scenarios, broadcasts, recurring, t placements.



Campaigns also support local control groups. A local control group allocates part of the audience that within the campaign. Later, by comparing the main and control groups, you can measure campaign effe support stratification — splitting into strata based on a selected attribute. Stratification creates a cont target audience.



Analytics for all activities within a campaign are available in a unified report with multiple metrics:

New campaigns centralize management of different activities, track communication effectiveness, and
More details about campaigns are available here.

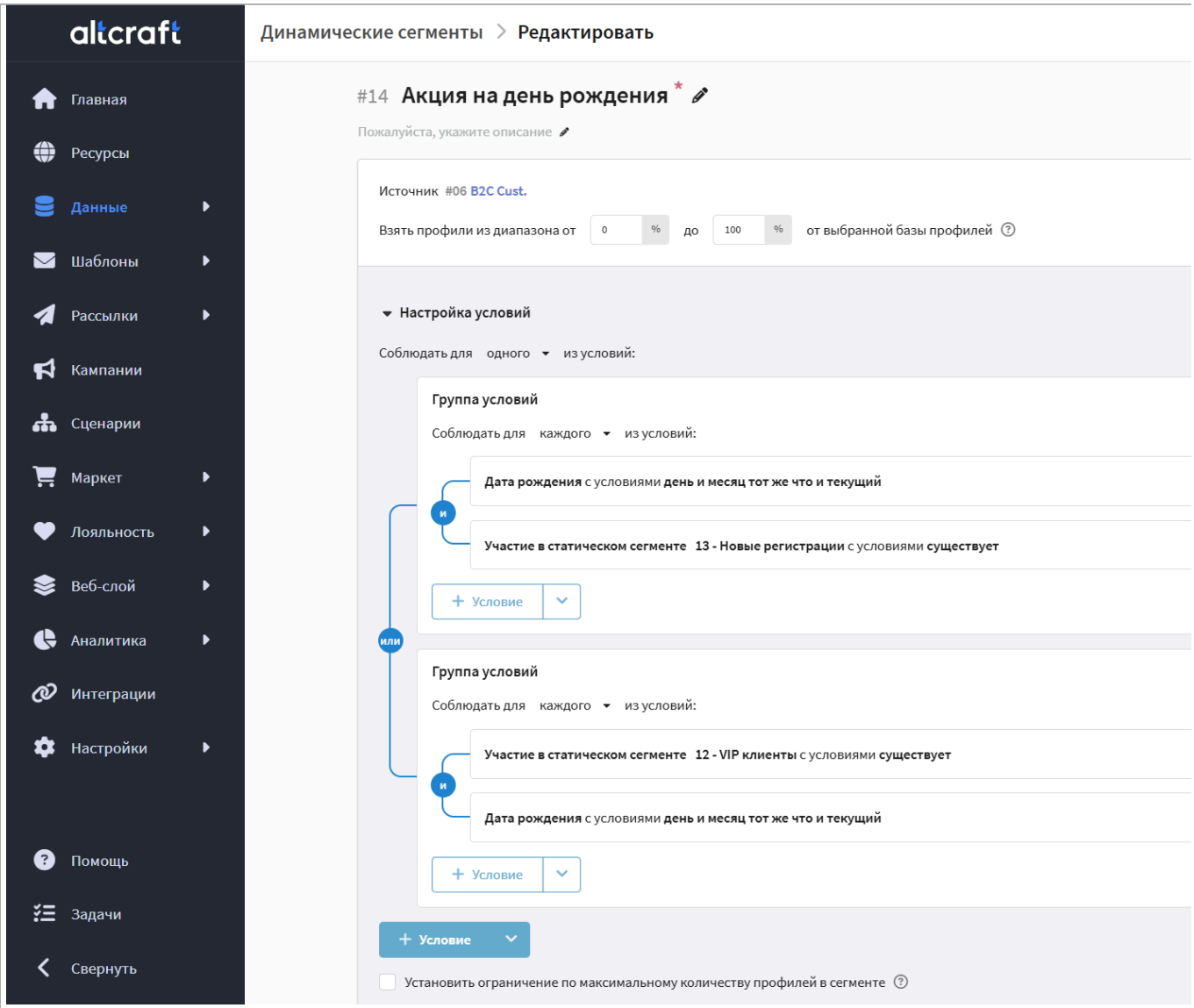
Improved

Updated segmentation

Major changes were introduced to the profile segmentation mechanism. The key update is a redesign o

- **Static segment — an entity for storing profiles filled manually or via API. It is not recalculated on u automatically. The segment can be saved empty; only a name and database selection are required.**
- **Dynamic segment — a set of conditions rather than a stored profile list. It is calculated before each results without saving the segment.**
- **Updatable segment — cannot be filled manually or via API. It is populated based on defined conditi manual recalculation.**
- **Quick segments were removed. Dynamic segments replace them; use the “Calculate” button when**

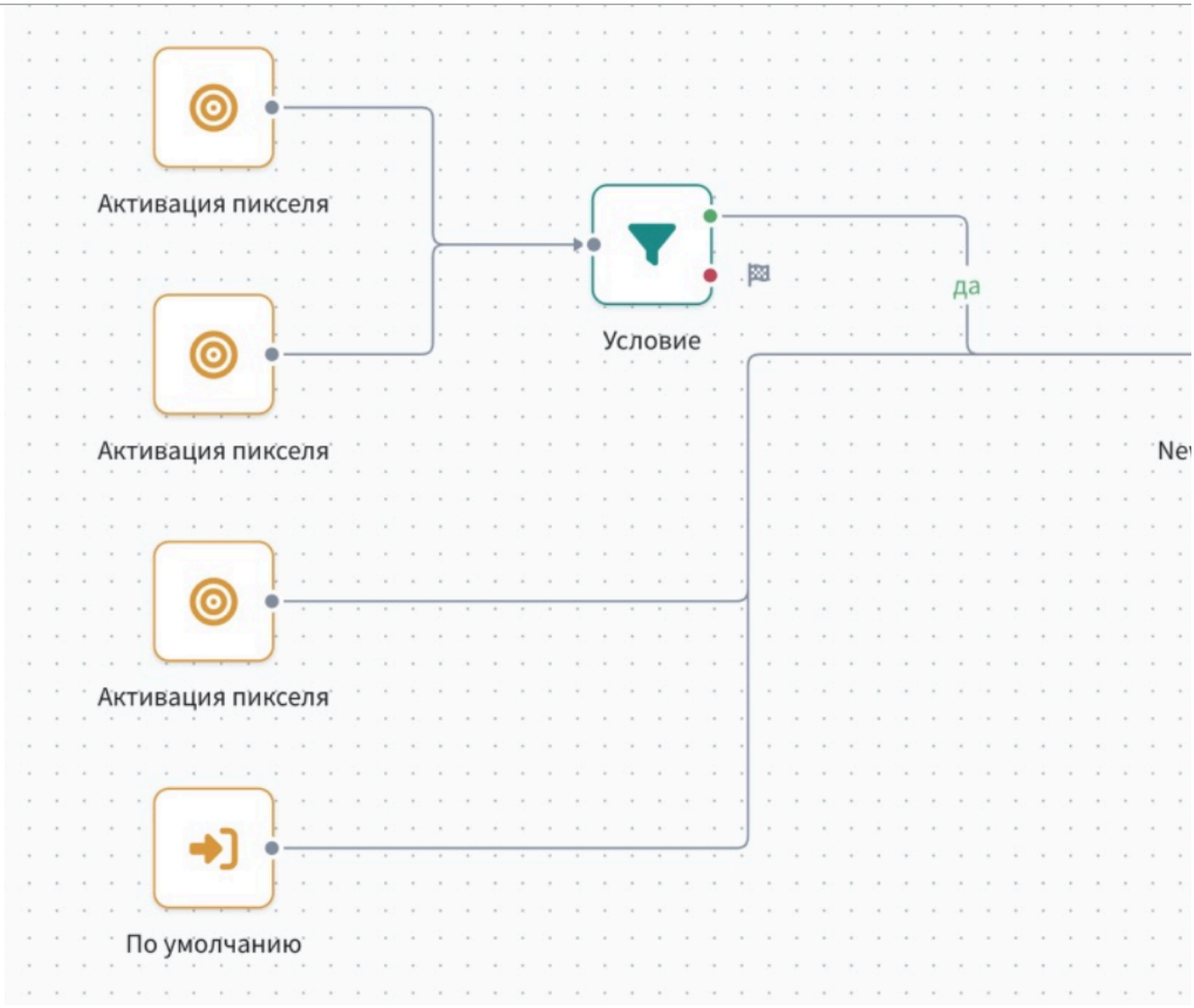
The segment configuration UI was also significantly redesigned:



The interface is now more modern and intuitive. More details on segmentation are available [in this sec]

Combined profile capture in scenarios

Previously, a scenario supported only one entry trigger per type. Now multiple triggers of the same typ activation of different pixels or profile field updates:



The maximum number of start nodes is limited to 10. This value is configurable via `WORKFLOW_MAX_START_NOD`

Other

- **Several UI/UX components were improved.**
- **Fixed issues in some Market APIs.**
- **Improved stability of** `proctask` **and** `procpixel` **processes.**
- **Fixed issues with campaigns using custom channels.**
- **Message delivery in the Intellin SMS provider integration is now correctly registered.**
- **Fixed cases where segments failed to calculate or calculated extremely slowly.**
- **Goals with SMID from a custom channel are registered correctly again.**
- **When starting the platform with the** `--config` **flag, the custom configuration file path is now passed process.**

References

v2025.4.74.0: PL-16253 PL-20320 PL-20303 PL-20304 PL-19139 PL-20225 PL-20226 PL-20194 PL-19904 I 20114 PL-20092 PL-20082 PL-19212 PL-19729 PL-19606 PL-19775 PL-19709 PL-19870 PL-19734 PL-1991 19818 PL-19671 PL-19661 PL-19633 PL-19328 PL-18624 PL-19600 PL-19567 PL-1785 PL-19053 PL-17465 19257 PL-19254 PL-19227 PL-18933 PL-19082 PL-1875 PL-17774 PL-18922 PL-8756 PL-18527 PL-18857 18788 PL-17775 PL-18232 PL-17740 PL-17008 PL-19207

v2025.3.73

CAUTION

Starting from v2025.3.74, the following API changes will take effect:

- **The old API version 1.0 will be disabled. Please switch to the current version in advance.**
- **The API token can no longer be sent in GET parameters or in the JSON body. Use the header inste**
- **The `format` parameter will no longer be supported. Set the desired format using the Accept heade `application/json`).**

We recommend making the necessary changes in advance to avoid API issues.

Also, campaigns without a linked resource will no longer run. Please make sure that all your campaig

v2025.3.73.1 – Attributes, GCG stratification, Scenario improvements

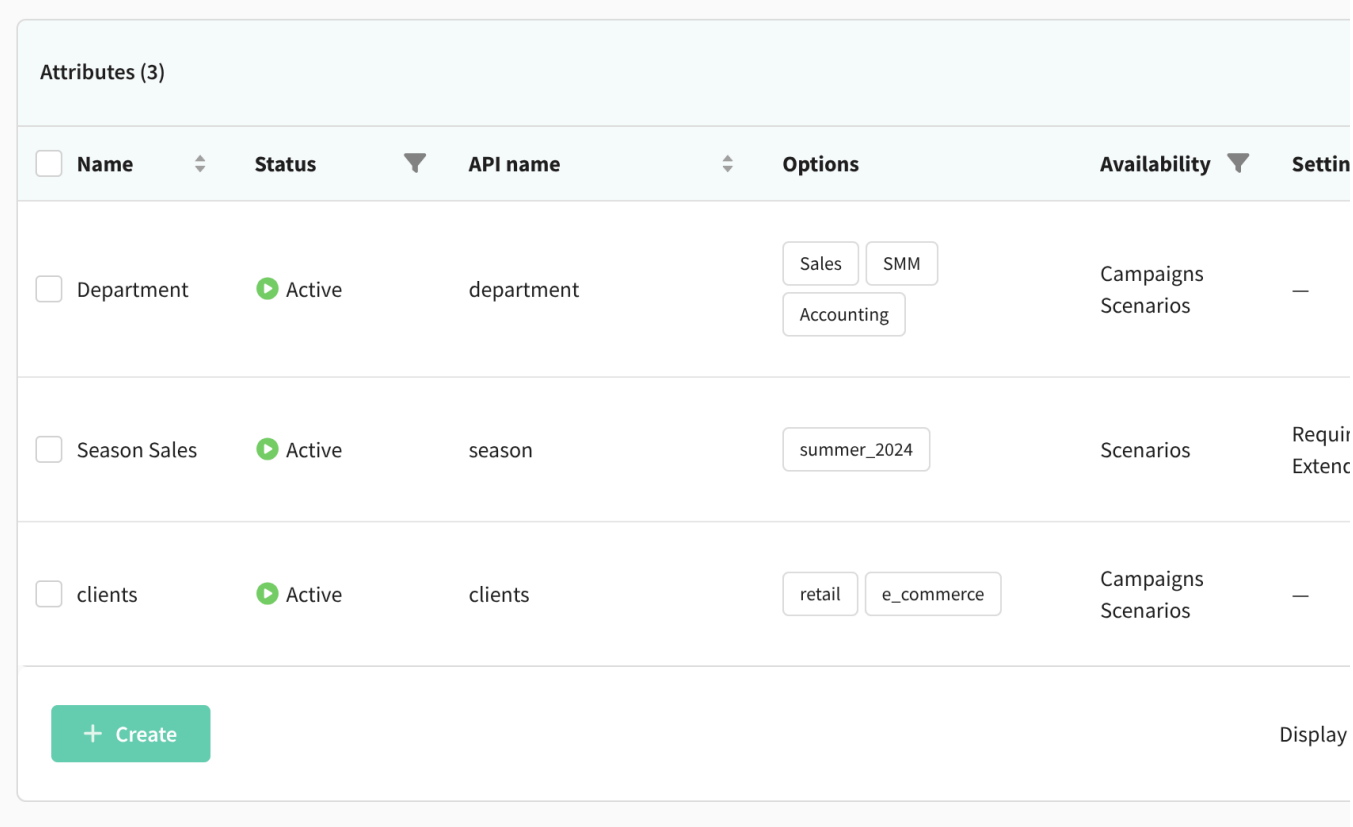
Added

Attributes

Attributes have been added for platform entities. Attributes are special custom fields added to platforn parameter with a set of possible values.

With attributes you can:

- **search and organize objects;**
- **select campaigns and scenarios for segmentation;**
- **insert values into content and API requests.**



Learn more about attributes in this section.

Stratification in GCG

Now you can configure stratification for the global control group. Stratification is a method of randoml division of the audience into subgroups (strata) based on specified characteristics. The audience is sp reflects the composition of the whole population.

Stratification

Stratification is a method of random selection of profiles with preliminary division of th
(strata) according to specified characteristics. The audience is divided into groups so th
composition of the whole aggregate.

+ Add

From each group, a portion of profiles is randomly selected proportional to the size of that group. This the same distribution of characteristics as the test group, making campaign results more accurate. St relevant article of the documentation.

New Scenario element "Event Waiting"

A new scenario element has been added — the Event Waiting operator. It stops the profile flow until a s soon as any of the specified events are registered by the platform, the profile moves to the next eleme If the event does not occur within the specified time, the profile follows another branch.

Event waiting

Node name: *

Event waiting

Wait for a specific time period

0 d. 1 h. 0

Rules for captured events

Event type:

Action execution

This event will be registered when the profile performs a specific action within a channel.

Action: Any link clicked in email

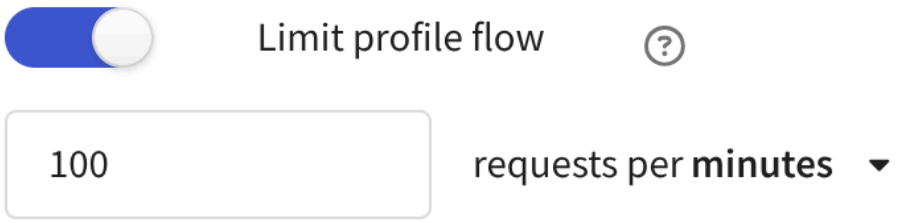
Template: 15 - Black Friday

+ Add

With this element, you can configure reactivation chains or, for example, an abandoned cart scenario. I

Profile limiter in the “API Call” element

For the “API Call” scenario element, it is now possible to limit the number of profiles passing through tl time. This is necessary to avoid overloading the external services that we call from the scenario. The l leaky bucket algorithm. Learn more in the corresponding section of the article “Scenario elements”.



Improved

- **Optimized the performance of `proctask` and `procintegras` processes.**
- **Updated the CSS inliner in message templates and campaigns.**

Other

- **Various UI/UX issues in the platform have been fixed.**
- **Retry counters in senders now correctly display the queue.**
- **Fixed errors that occurred when trying to create and save a multivariate test.**
- **Display and filtering of senders in the Admin Panel now works correctly.**
- **Fixed an error that prevented searching for profiles in a specific scenario element. JSON content li now opens without errors.**
- **SQL export to segments works correctly again.**
- **Fixed errors related to generating and displaying template previews.**
- **Added the ability to save a connector to an external database with SSL connection without specify**
- **Analytics for goals with slashes in the name is displayed correctly again.**

References

v2025.3.73.1: PL-18644 PL-18219 PL-18646 PL-18645 PL-18715 PL-18218 PL-18220 PL-18217 PL-18452 I 17534 PL-18216 PL-18802 PL-18149 PL-18309 PL-18133 PL-18184 PL-18082 PL-17559 PL-18772 PL-1659 18867 PL-18899 PL-18885 PL-18537 PL-18072 PL-18864 PL-18861 PL-18875 PL-18825 PL-12690 PL-1824 18873 PL-18722 PL-18832 PL-18881 PL-15383 PL-18351

---

v2025.3.73.1140 – Altcraft Mobile SDK v2

Added

Altcraft mobile SDK v2

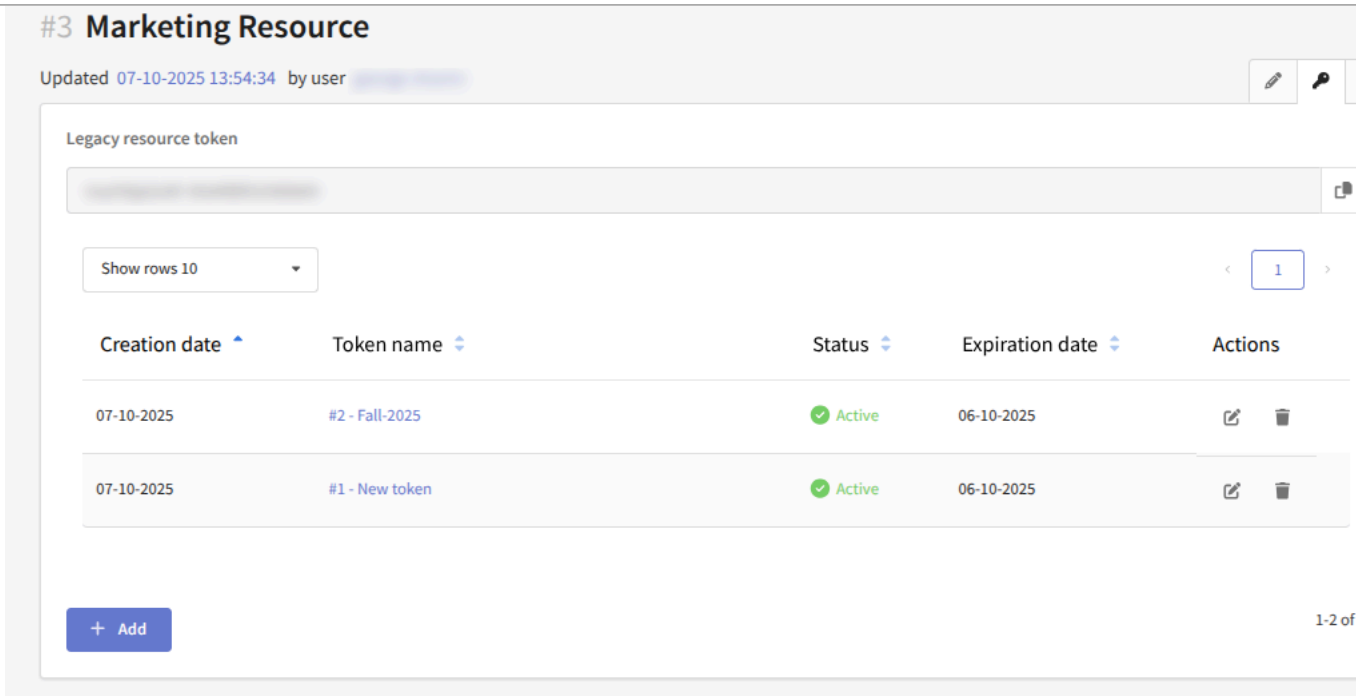
We have improved our mobile SDK for handling push notifications in applications. Key updates include:

- **Support for multiple authorization types: JWT authorization and Altcraft role token authorization;**
- **Ability to display push notifications with context configured in the platform;**
- **Automatic token refresh when it changes;**
- **Automatic delivery and open event tracking;**
- **New functions for working with anonymous and registered users, as well as with multiple users on**

Learn more in the developer documentation section. Altcraft mSDK is available on GitHub.

Other

- **Added variables for using pixel event data in Scenarios and various types of content. More about th**
- **Added a new tab for managing resource tokens. On this tab, you can create a token for a resource, roles. You can access this tab after creating and saving the resource (located in the upper-right co**



- **Added API handlers for deleting products and SKUs from the Market.**
- **Added the option to create an integration with Yandex.Appmetrica without creating a pixel. This c profiles without using goals.**
- **Added validation to ensure that dynamic JSON provided by the user contains an object (arrays are**
- **Added the ability to reference the current day in selection conditions and segmentation.**
- **Added a Redis connection check during platform startup.**

Improved

- **Message history in Jivo chat is now linked to the user's login — this helps support staff resolve cu**
- **Platform process stability has been improved.**
- **Various UI/UX component improvements and fixes.**
- **The short link generator no longer creates links ending with a dot.**
- **The `procintegras` process logs now include profile-related information and push delivery details for**

Other

- **Default value for the `FIREBASE_PUSH_BATCH_WORKERS` parameter is now 200.**
- **Fixed an issue where a dot or comma could be entered into an integer campaign field.**
- **SMS sending status handling in the SMS Traffic integration now works correctly again.**
- **Fixed an issue that caused segments with the "Participation in segment" condition to be calculate**
- **Reserved tasks are now correctly set to the `registered` status.**
- **Fixed an issue where a boolean profile field did not update after a form submission.**
- **Attributes now correctly substitute values when used in Custom channels.**
- **Fixed analytics issues in NPS forms.**
- **The integration with EDNA now works properly again.**
- **Fixed errors related to data migration during platform updates.**
- **Rustore push event registration now works correctly.**
- **The `formatdate` function in the "API Call" scenario element now correctly processes profile fields.**
- **Fixed an issue where web version collections were deleted too early.**
- **The subscription manager now correctly displays recipients of messages.**
- **When updating a template via API, the modification date and time are now updated correctly.**
- **Occupied IP addresses are no longer displayed when creating a sender.**
- **Fixed an issue that prevented sending a profile to a scenario containing an attribute ignored by the**
- **Fixed several frequent issues in the visual email editor:**
  - **multiple font display now works correctly;**
  - **Transition, Perspective, and Transform functions are functioning properly again;**
  - **background setup and saving issues have been resolved;**
  - **images in style blocks now load correctly.**
- **Link redirection in emails now works correctly again.**
- **Default values for parameters `SEGSTATIC_PROFILE_CHANGES_BATCH_SIZE` and `API_GET_PROFILES_BATCH_SIZE` have b respectively.**

- **Fixed an issue where the `templates/list` API request did not return the `next_from_id` field.**
- **Password change via the `aktool` utility now functions correctly.**
- **Fixed an issue where the stratification limit from configuration parameters was not displayed in th**

References

v2025.3.73.1140: PL-19333 PL-19309 PL-19326 PL-18399 PL-19176 PL-19273 PL-18437 PL-17717 PL-183 PL-19137 PL-18870 PL-18665 PL-17683 PL-17885 PL-17853 PL-17154 PL-17100 PL-19285 PL-18320 PL-1 PL-19254 PL-19229 PL-18327 PL-19218 PL-18836 PL-15418 PL-19233 PL-19204 PL-17933 PL-19214 PL-1 PL-18328 PL-19199 PL-18432 PL-18891 PL-19116 PL-19176 PL-19068 PL-18334 PL-10830 PL-19131 PL-1 PL-19117 PL-17977 PL-18933 PL-17274 PL-18912 PL-18611 PL-17809 PL-17768 PL-18308 PL-13780 PL-1 PL-19115 PL-18155 PL-18471 PL-18594 PL-18547 PL-16869 PL-19079 PL-19086 PL-19075 PL-19083 PL-1 PL-19010 PL-18607 PL-18223 PL-18620 PL-16448 PL-17580 PL-19026 PL-19267 PL-19314 PL-19262 PL-1 PL-19277 PL-18994 PL-19238 PL-18363 PL-19105 PL-18343 PL-19143 PL-18643 PL-18538

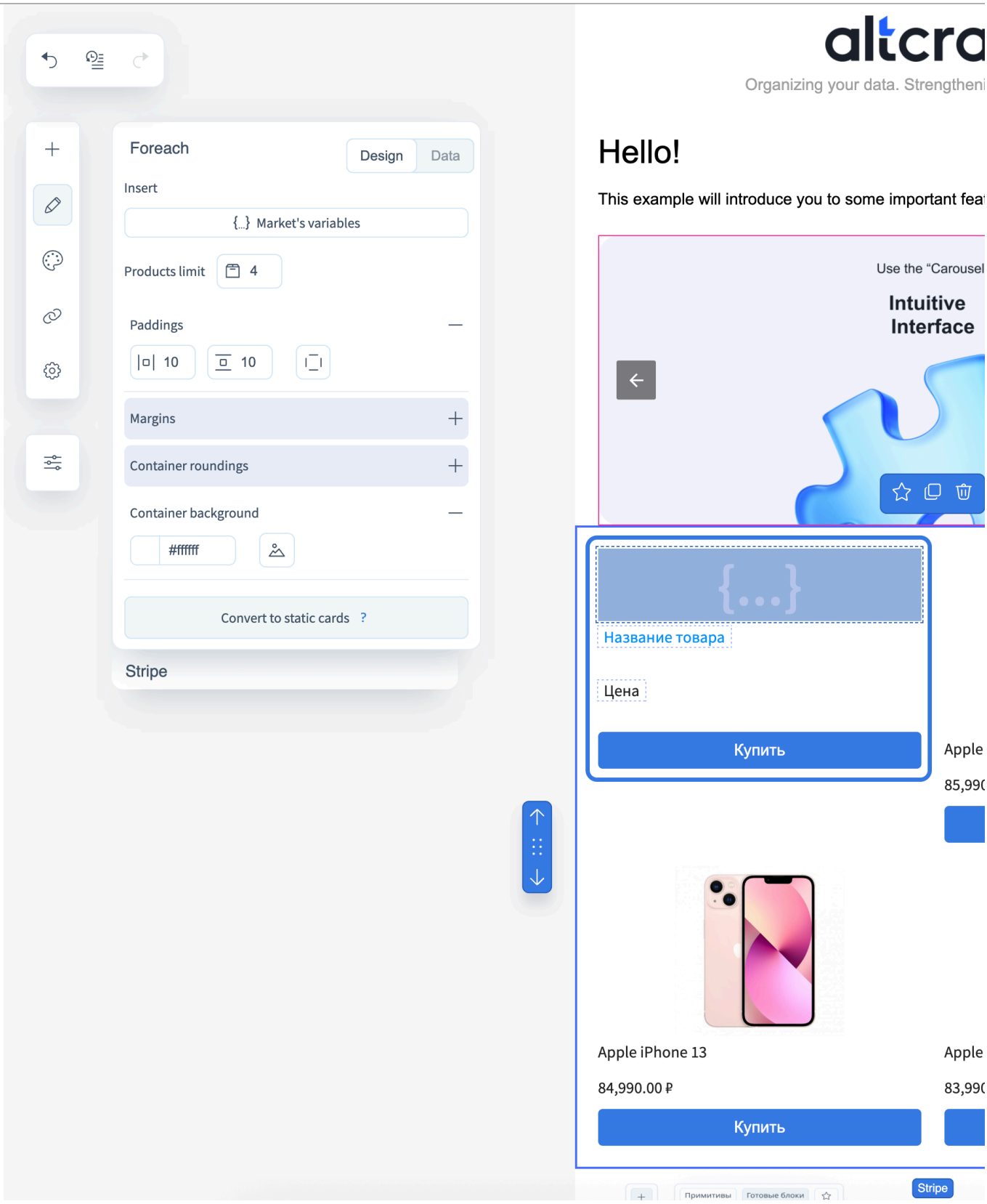
---

v2025.3.73.1942 – Updated email block editor, Faster Imports and Campaigns

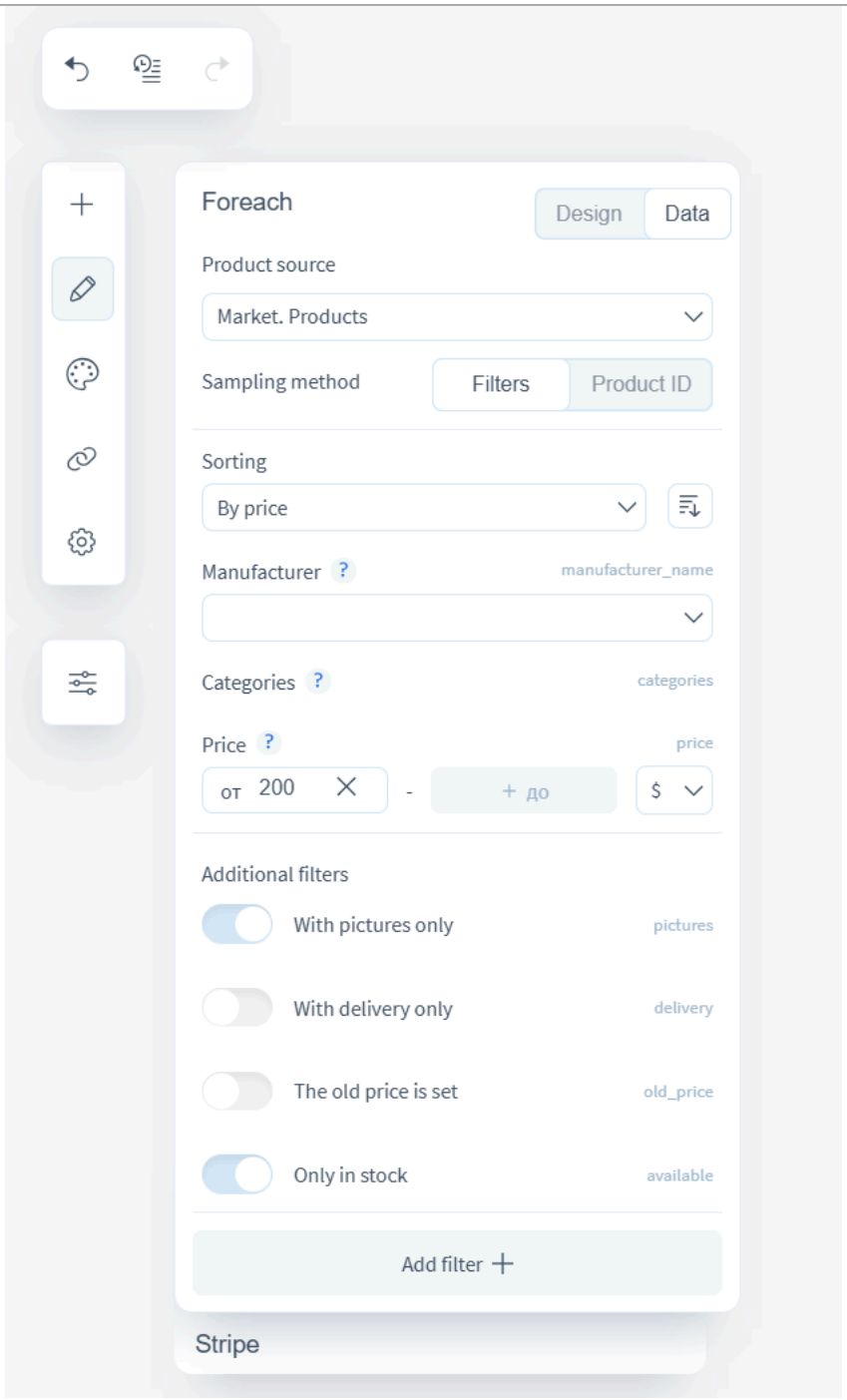
Added

Updated Block Email Builder

The Block Email Builder is now more functional and user-friendly. It includes a new “Products” element and SKUs from the Altcraft Market.



You can customize the design of product cards and apply filters directly in the editor using an intuitive dynamically when sending emails, and for one-time mailings, you can lock products in the card to turn



The block editor gallery also now lets you edit images — crop, resize, rotate, and flip them directly in t

Other

- **The SDK now supports registering custom user events.**
- **The method *`/api/v1.1/templates/get`* is extended: parameters are added to filter templates by chann used in the template.**
- **If a campaign runs inside a scenario, the methods *`/api/v1.1/campaigns/triggers/get`* and *`/api/v1.1/ca*` **workflow ID (`workflow_id`). The *`/triggers/list`* method also allows filtering campaigns by `workflow_id`.**
- **Custom subscription fields can now be displayed in templates using the `subscription_field` function.**
- **A `len` function was added to compare the length of an array with a given number. For example, to i that have images, you can use the expression `products("!len(product.pictures, 0)" 10 "manufacturer_name"`**

Improved

- **When exporting collected form data, the profile ID is now included in the report, making it possible form.**
- **Improved the audit log event recording process: events are now processed through a separate `ak_a` batches.**

- **Significantly optimized the process of checking and updating sending policies. Previously, during c (subscription's) policy was checked and updated individually; in the new version, policies are proce cases can increase campaign execution speed up to 10 times.**
- **Optimized the profile import process through batch data processing. In some cases, this can doubl and SQL tables. To enable the new mode, set the `IMPORT_BATCH_OPTIMIZER_ENABLE` parameter to `true`.**

Other

- **In the "Dependent objects" section of segments, all related objects are now shown correctly, inclu campaigns.**
- **API content is now displayed correctly when viewing a profile through a scenario on wait nodes.**
- **Global control group calculation now works correctly for large profile databases.**
- **Some security vulnerabilities in the platform have been fixed.**
- **Fixed an issue where after merging profiles, a goal was saved in history but the profile did not app**
- **Sending policy now works correctly, and profiles that do not match campaign conditions are no lon**
- **When using SQL queries in templates, the template now renders correctly even if the profile is mis are used instead.**
- **Fixed goal registration via API for custom channels using `send_message_id`.**
- **Spaces at the beginning and end of goal names are now automatically trimmed.**

References

v2025.3.73.1942: PL-19442, PL-19540, PL-19064, PL-17853, PL-19472, PL-19467, PL-19453, PL-18128, PI 16031, PL-19142, PL-19030, PL-17284, PL-17393, PL-19439, PL-18582, PL-18428, PL-17607, PL-19434, P 17769, PL-19347, PL-19255, PL-17817, PL-18222, PL-19195, PL-18533, PL-19098, PL-18954, PL-19055, P 18221, PL-17392, PL-18109, PL-19360, PL-19219, PL-19345, PL-19259, PL-19222, PL-19237, PL-17829, P 18842

---

v2025.3.73.2276 – Platform Improvements, Fixed Event Context

This update fixes frequent issues and improves the platform's functionality and stability.

Improved

Event Context

Fixed errors in the handling of variables related to the event context:

- **The `event` object now contains the number of events specified in the `WORKFLOW_EVENT_CONTEXT_LENGTH` pa**
- **The event iterator within a template (such as `{for $index $event = event.all}`) now works correctly ag**

Learn more about these variables and how to use them [in this article](#).

Trigger Re-fire Blocking

The pixel option Block trigger re-firing now limits specific goals of a pixel rather than the entire pixel. I workflows triggered by the same pixel but with different goals, they will all work correctly. This enable marketing tasks.

Other

- **Fixed a logging issue related to the incorrect behavior of the `LOG_ROTATE_MAX_FILES` parameter. It now log files excluding the current one, while the `LOG_ROTATE_ZIP_FILES` parameter specifies how many of**
- **Added configuration parameter `PROC_TASK_RESERVED_TASK_IDLE_TIME_LIMIT_MIN`. It defines the maximum idle status. If a task remains in this status longer than the defined limit, it returns to the `registered` stat during connection interruptions.**
- **Static segments are no longer recalculated automatically when sent to a Scenario — only profiles at the moment of sending are included.**
- **Scenarios using SKU additional field filters now work correctly again.**
- **Special characters now display properly in push messages sent via AppMetrica.**
- **Fixed an issue where link addresses were truncated in web versions of emails.**
- **Export of emails created in the [Visual Editor](#) now works correctly again.**
- **Fixed an issue where some push templates failed to insert Promo Code variables.**
- **The URL decoder in the Admin Panel can now process tracking links from Custom channel messag**
- **Fixed a message delivery issue that occurred after changing Custom channel settings. Templates i correctly without re-saving.**
- **Fixed an authorization issue that occurred when using the `PASSWORDS_MAX_AGE` parameter.**
- **Message templates containing the `{resource_token}` variable now render correctly.**

- **Optimized workflow statistics logging, reducing RabbitMQ load.**
- **Improved process stability.**
- **Fixed several UI/UX issues.**
- **Added Redis connection check at platform startup.**
- **Connection to the platform via LDAP (both SSL and TLS) now works correctly again.**

References

v2025.3.73.2276: PL-19691 PL-18213 PL-19277 PL-11085 PL-19553 PL-19022 PL-19574 PL-16848 PL-192 PL-19596 PL-19432 PL-19318 PL-19618 PL-19608 PL-17716 PL-19473 PL-19554 PL-19247 PL-19270 PL-1 PL-19600 PL-19564 PL-19555 PL-18362 PL-18568 PL-19155 PL-11262

v2025.3.73.2565 – Action Hook for Past Events

Added

Historical Events Capture

You can now transfer past events using the historical action hooks. When configuring the capture, use specify the export date range. The set of transferred fields is limited, but all key event data will be pre Learn more about historical action hooks here.

Improved

- **The platform now registers a hard bounce event and updates the subscription status when sending existent ID.**
- **The block editor has been updated: template rendering is more stable, the default email template h with sending invalid AMP versions has been fixed.**
- **A new version of the 360Dialog integration for WhatsApp messaging is now available.**
- **Interface improvements have been made for tables and counters.**
- **Events related to global control groups have been added to event capture.**

Other

- **Validation for wait time and pixel value on the pixel code page has been added.**
- **Stability of the `procsmsev` and `proctask` processes has been improved.**
- **An issue occurring when connecting a database without tables has been fixed.**
- **Markdown support for custom channel previews has been added.**
- **A misleading modal window during scenario activation has been fixed.**
- **The `/cookiesaver/readiness` handler works correctly again.**
- **Issues with the SMS Traffic sender integration have been resolved.**
- **A user with an expired password is now required to change it upon the next login.**
- **A parameter can once again be inserted and saved in the "Click-through link (SSL only)" field when**
- **Logging of web push deliveries to Safari has been improved.**
- **The `format` function no longer strips the minus sign from negative fractional numbers beginning wi**

References

v2025.3.73.2460: PL-19898 PL-19216 PL-19876 PL-19680 PL-18358 PL-19638 PL-19681 PL-19862 PL-195 PL-19013 PL-19547 PL-19141 PL-13097 PL-19572 PL-19497 PL-19501 PL-19746 PL-17495 PL-17658 PL-1 PL-19670 PL-19588 PL-17528 PL-19592 PL-19594 PL-19394 PL-19469 PL-19334 PL-18736 PL-17537 PL-1

---

v2025.3.73.2888 – Tag Manager variables in HTML tags

Added

Using Tag Manager variables in HTML tags

Previously, variables could only be used in Tag Manager triggers. Now they can be inserted directly int format `{{varName}}`. This makes it easier to reuse tags in different contexts. Learn more.

Other

- **The pixel activation trigger now checks date ranges correctly.**
- **NPS survey statistics no longer reset after updating the form.**
- **Scenarios that use external table queries now run without errors.**
- **Multi-step forms with profile field substitutions can be submitted successfully.**

- **Fixed an issue where apicontent was incorrectly handled during bulk profile imports into a scenari**
- **Numeric values from apicontent are now inserted into API node URLs in the correct format.**
- **Profile history export now includes goal data, including pixel_data.**
- **Fixed an issue that prevented connecting to a Yandex Cloud database using a CA certificate.**
- **Optimized the performance of proctask and procintegras processes.**
- **Some UI/UX issues in the interface have been fixed.**

References

v2025.3.73.2888: PL-16683, PL-17392, PL-18422, PL-18751, PL-18847, PL-18862, PL-18876, PL-19523, PL 19881, PL-19884, PL-19899, PL-19916, PL-19924, PL-19926, PL-19983, PL-20002, PL-20003, PL-20039, P 20196, PL-20200, PL-20304, PL-20320

v2025.2.72

v2025.2.72.0 – Redis Storage, Process Optimization

Added

Redis Storage

For the correct functioning of the platform, it is now required to use Redis-compatible storage. Such st

- **enables a gradual transition away from file-based persistent storage;**
- **increases the platform's fault tolerance and manageability;**
- **ensures more reliable and faster interaction between services.**

We recommend using Kvrocks. The platform installation documentation includes a guide for connecting
Using Redis is allowed but only with AOF data persistence mode or RDB + AOF mode.

Support for RabbitMQ Quorum Queues and Unified Virtual Host

> **CAUTION**
>
> **Before installing the update, you need to create a new virtual host (by default, altcraft) and ensure tha permissions on it.**

> **INFO**
>
> **If there are messages left in old virtual hosts, you can use the Shovel plugin to transfer them. Below i from the `trk_read` queue in the `/` virtual host to the `altcraft` virtual host:**

```
rabbitmqctl set_parameter shovel trk_read '{
  "src-protocol": "amqp091",
  "src-uri": "amqp://altcraft:abcdefghijklmnopqrstuvwxyzABCDEF@localhost:5672",
  "src-queue": "trk_read",
  "dest-protocol": "amqp091",
  "dest-uri": "amqp://altcraft:abcdefghijklmnopqrstuvwxyzABCDEF@localhost:5672/altcraft",
  "dest-queue": "trk_read"
}'
```

Other Changes

- **Users can now create a new report based on an existing one.**
- **Added support for configuring a handler pool for the RMQ Event Sender.**
- **The user session duration is now configured via the `PROCAUTH_SESSION_TIME_SEC` parameter, and the abil the IP address changes is controlled by `PROCAUTH_SESSION_ALLOW_CHNET`.**

Improved

- **Platform performance has been improved.**
- **Stability of the `procintegras` process has been improved.**
- **Task processing in the `proctask` process has been optimized.**

Other

- **Platform performance improved.**
- **Fixed several UI/UX issues.**
- **Increased stability of the `procintegras` process.**
- **Fixed authorization-related errors.**
- **Optimized task processing logic in `proctask`.**
- **Fixed incorrect display of analytics metrics.**
- **Confirmation link validation in templates works properly again.**
- **Automation Scenario elements now operate more reliably — previously occurring errors have been**

References

v2025.2.72.0: PL-18318 PL-18343 PL-18519 PL-18367 PL-18349 PL-18323 PL-18192 PL-18321 PL-18322 I 18445 PL-18482 PL-18276 PL-18458 PL-18427 PL-18389 PL-18372 PL-18347 PL-18319 PL-18332 PL-1840 18379 PL-18302 PL-18307 PL-18295 PL-18255 PL-17490 PL-17886 PL-18134 PL-17904 PL-16949 PL-1694 17223 PL-15798 PL-17545 PL-17068 PL-17474 PL-17453 PL-17131 PL-17451 PL-17442 PL-17399 PL-1691 17192 PL-17243 PL-17023 PL-16948 PL-17101 PL-17087 PL-17091 PL-11284 PL-16885 PL-17030 PL-1702 16541 PL-16356 PL-16702 PL-3437 PL-16530

---

v2025.2.72.707 – Custom channel improvements, link shorter optimization

> **DANGER**
>
> **Pay attention to the important changes in the link shortener behavior:**
>
> - **The content of the web version of the message is now transmitted without modifications. This me appear in the profile card as content actions. Links to web versions are still transmitted with the**
> - **The link format has changed from `^(/s/[A-Za-z0-9_-]{5,20})$` to `^(/s[A-Za-z0-9_\-!~\.]{10,12})$`.**

Added

- **Added dynamic display of process status when starting and stopping the platform using `akd`.**
- **The number of profiles entered and exited is now displayed above scenario elements.**
- **The variable insertion menu now includes scenario name and scenario ID, as well as element ID an**
- **Users can now override form templates using `custom` files.**
- **Added a launch log for the trigger scenario element "Scheduled Capture".**
- **The profile card now displays information about scenario errors if the profile was in the GKG.**
- **API requests for the scenario list and template info now support `updated_user` and `update_time` param**

Improved

Custom Channel Pipeline Improvements

The operation of custom channel pipelines has been significantly improved. The following changes wer

- **Fixed errors occurring during pipeline execution.**
- **Added access to object names.**
- **Pipelines can now accept event dates in the EventObject.**
- **Added a unique message identifier in the pipeline, matching the ID in ClickHouse.**

Other Improvements

- **It is now possible to override the Firebase URL for each platform account in the configuration file.**
- **The search and filtering interface across the platform has been updated.**
- **Refactored link shortener operation. Duplicate link IDs were eliminated, and the maximum link leng**
- **The `maxAmountOfChanges` constant was moved from `segstat_recalc` to a separate configuration paramet `SEGSTATIC_PROFILE_CHANGES_BATCH_SIZE`.**
- **Optimized the Best Send Time ML service. Related configuration parameters were also updated.**
- **Improved platform performance.**
- **Improved stability of `procworkflow`, `webcontrol`, and `proctask` processes.**

Other

- **Static segments created via API are now recalculated correctly.**
- **Fixed incorrect calculation of segments with identical file names.**
- **The last update date is now correctly displayed in profile databases.**
- **The subscription capture trigger in scenarios now considers the selected channel and resource.**
- **Fixed a JavaScript error on the campaign type selection page when creating a campaign from the**
- **The session in the web interface no longer expires too quickly.**
- **The `adddate` function now works correctly in the "API Call" scenario element.**
- **The JSON filter now supports fields with underscores in their names.**
- **Push templates imported via API can now be opened in the beta editor.**
- **The `not_strict` and `subscription_filter` parameters in scenarios now work correctly.**
- **AKMTA now correctly processes DSN when using the `From` field address in `MAIL FROM`.**
- **Fixed 500 error when requesting the web version via API for profiles in the stop list.**
- **Fixed an error that occurred when placing a pixel via the Tag Manager.**
- **Fixed parsing of status responses from the SMSTraffic sender.**
- **Phone numbers in .xls files are now parsed correctly.**
- **The `send_message_id` value in custom JSON for Push now matches the `send_message_id` from the event**

References

v2025.2.72.707: PL-18764 PL-18548 PL-18710 PL-18763 PL-18671 PL-18659 PL-17043 PL-16495 PL-1789 18674 PL-18761 PL-18095 PL-18634 PL-18661 PL-18548 PL-18531 PL-17937 PL-18383 PL-18264 PL-1844 13124 PL-18737 PL-18070 PL-17894 PL-17772 PL-18132 PL-18672 PL-18703 PL-18712 PL-18669 PL-1861 13124 PL-18573 PL-15659 PL-18734 PL-18705 PL-18693 PL-18668 PL-17784 PL-18662 PL-18608 PL-1741

**18685 PL-18683 PL-18676 PL-18656 PL-18605 PL-18524 PL-18660 PL-18658 PL-18447 PL-18494 PL-1863
17120 PL-18534 PL-18446 PL-18304 PL-18261 PL-17705 PL-18417 PL-15337 PL-18164**

---

v2025.2.72.1475 – password recovery and billing info

> CAUTION
>
> **As part of this update, the structure of stored data has been changed, so for the platform to work cor
> migration. To do this, stop the platform and run the following command:**
>
> `./aktool migrate add-updatefeed --config=/opt/MarketingPlatform/config/main.json`
>
> **Do not run the migration multiple times! The migration is performed only once. If any errors occur, ple**

Added

- **Metrics of profile processing time have been added for each campaign stage.**
- **A numeric postfix has been added to the log file upon rotation.**
- **For content, variables `{segid}` and `{segname}` have been added — ID and name of the segment.**
- **New parameter `PROCAUTH_WELCOME_LINK_TTL_SEC` added — lifetime of the registration completion and pas**
- **Additional logs have been added to the Reply processing in AKMTA.**
- **New indexes have been added in MongoDB for the most frequent queries to improve platform stabil**
- **Integration with SMS provider 7tech has been added.**
- **Logs for Notify requests have been added to obtain integration status.**

Improved

- **Profile sorting removed when filling a static segment.**
- **Extended information in the freeze and account deactivation notification.**
- **Password change page updated, along with the success notification email.**
- **Improved stability of the `proctask` process.**
- **Additional Market product fields are now imported with correct data types.**

Other

- **Fixed several UI/UX issues in the interface.**
- **The `--faster` flag of the `./akd restart` command works correctly again.**
- **Fixed a bug where conditions with array references were not displayed in the “JSON filter” elemen**
- **Images in Viber templates are once again correctly inserted through the gallery.**
- **Eliminated false triggers related to profile relations.**
- **Campaign schedule start time is now calculated correctly, even when using a time zone different f**
- **Fixed errors occurring when recalculating profile segments.**
- **Scenarios are now saved correctly when using the “Capture profiles on subscription add/change” n**
- **Fixed JS error when opening the “Segment by schedule” node.**
- **AKMTA now saves emails received from the postmaster mailbox.**
- **Added `X-Letter-Id` header for Sendsay sender.**
- **Profile card now displays unsubscribe events for the Viber™ channel again.**
- **The `uppercasefirst` function now works correctly with profile fields.**
- **Fixed errors related to sending emails after form submissions.**
- **Telegram template preview works correctly again.**
- **Fixed a bug where incorrect data was displayed in GCG.**
- **Fixed issues related to user roles and access rights.**

References

**v2025.2.72.1475: PL-18764 PL-18548 PL-18710 PL-18763 PL-18671 PL-18659 PL-17043 PL-16495 PL-178
PL-18674 PL-18761 PL-18095 PL-18634 PL-18661 PL-18548 PL-18531 PL-17937 PL-18383 PL-18264 PL-1
PL-13124 PL-18737 PL-18070 PL-17894 PL-17772 PL-18132 PL-18672 PL-18703 PL-18712 PL-18669 PL-1
PL-13124 PL-18573 PL-15659 PL-18734 PL-18705 PL-18693 PL-18668 PL-17784 PL-18662 PL-18608 PL-1
PL-18685 PL-18683 PL-18676 PL-18656 PL-18605 PL-18524 PL-18660 PL-18658 PL-18447 PL-18494 PL-1
PL-17120 PL-18534 PL-18446 PL-18304 PL-18261 PL-17705 PL-18417 PL-15337 PL-18164**

---

v2025.2.72.1664 – interface improvements

> TIP
>
> **As part of the update, the structure of stored data has been changed, so to ensure the platform works
> performed. To do this, stop the platform and run the following command:**

./aktool migrate add-updatefeed --config=/opt/MarketingPlatform/config/main.json

If you have already performed the migration, do not run it again! Migration is performed only once. If y
contact [the technical support team](#).

Improved

Interface fixes

Several UI/UX issues have been resolved:

- **Localization has been added instead of system names in certain places**
- **Improved display of some warnings and error messages**
- **Added a documentation link for Basic Authentication and Token Authentication connections**
- **Updated fonts and styles of some texts in the platform**
- **Fixed overlapping of certain Scenario elements**

Other

- **WhatsApp sending with a “PHONE” button in EDNA now works correctly**
- **Fixed an issue where templates with images could not be imported from an archive**
- **Field values are now saved correctly when creating promo codes**
- **Optimized performance of the `procintegras` process**
- **Fixed issues with sender queue counters**
- **Improved stability of the “Change field value” node**
- **Fixed several issues related to template previews**
- **Added the ability to save a connector to an external DB with SSL connection without specifying ce**
- **Fixed an error when trying to view analytics for goals containing slashes in their name**
- **Viewing API content through a Scenario element now works correctly**
- **Fixed `empty query` error in SQL segmentation requests**

***Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a**
banned in the Russian Federation.
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

v2025.1.71

v2025.1.71.0 - Global control groups, administrative panel rework, NPS testing

CAUTION

For the platform to work correctly after the upgrade you need to perform a migration, to do this:

1. Before running `procevent`, perform the data migration for ClickHouse:

```
/opt/MarketingPlatform/ak/bin/aktool profile_history add_gcg_id_column --config /opt/MarketingPlatform/config/main.json
/opt/MarketingPlatform/ak/bin/aktool history_storage add_gcg_id_column --config /opt/MarketingPlatform/config/main.json
/opt/MarketingPlatform/ak/bin/aktool history_storage add_profile_marks --config /opt/MarketingPlatform/config/main.json
```

The platform can only be started after the ClickHouse migrations are completed.

2. After completing the ClickHouse migrations, run `aktool` from the new build to migrate MongoDB:

```
./aktool --config config/main.json profiles migrate-gcg-flags
```

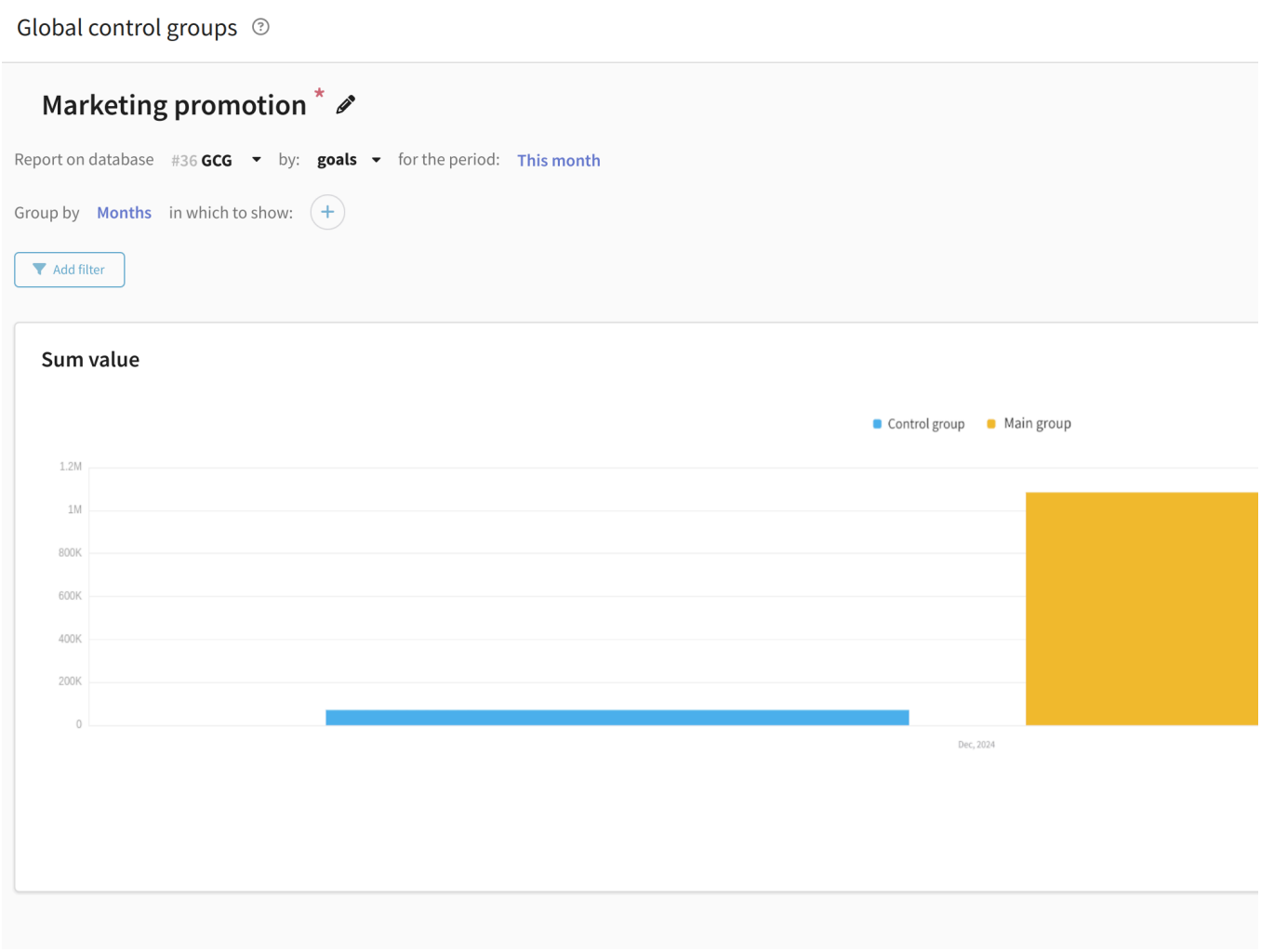
Note that the MongoDB migration may take from one to several hours. The global control groups featu MongoDB migration is completed.

3. Start `procevent`.

Major changes

Global control groups

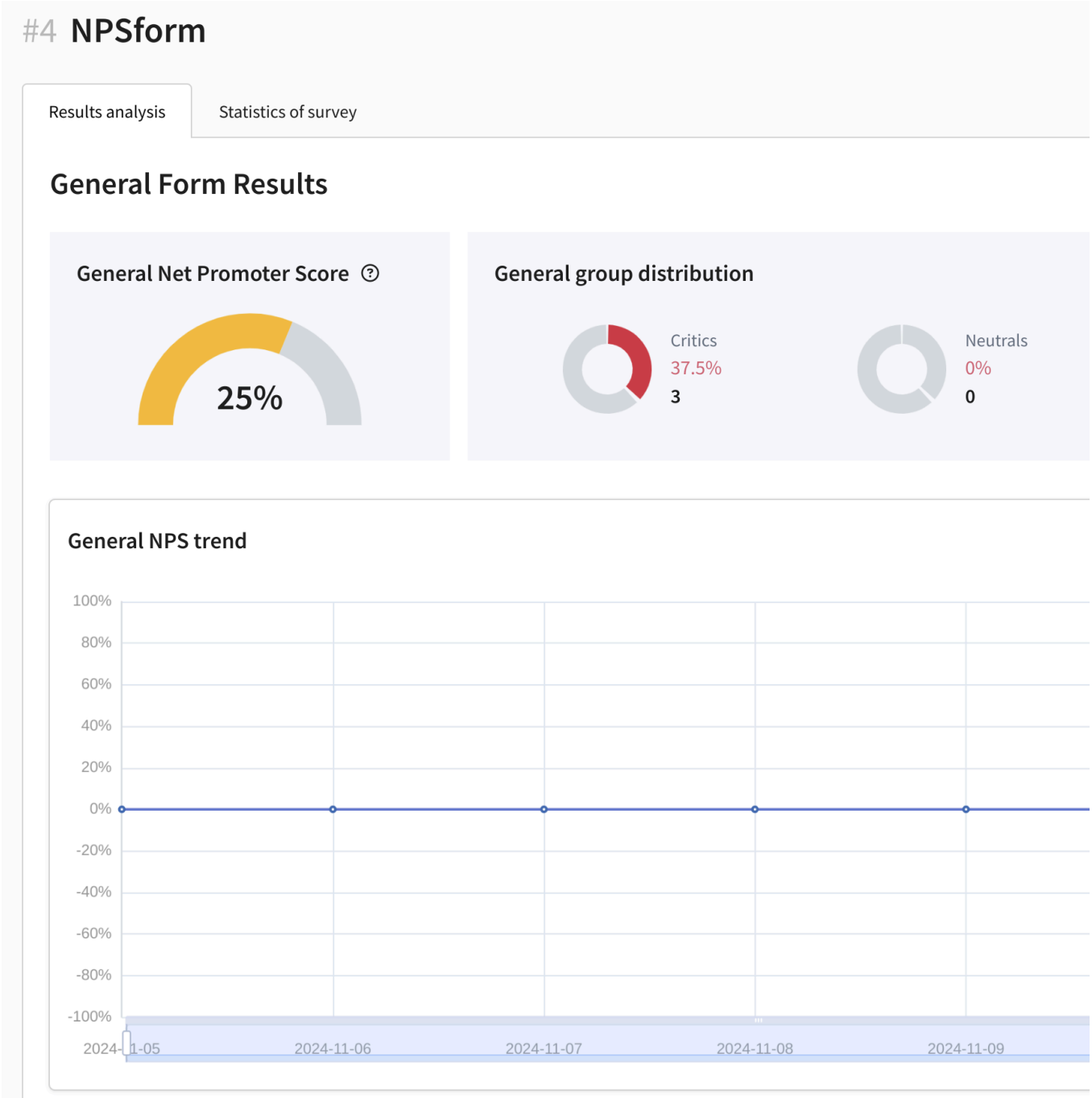
The platform now includes a feature to select global control groups. Global control groups are parts of temporarily does not receive marketing communications in order to analyze the effectiveness of marke the user can compare key metrics of the main part of the clients and the control group. In case the gro the marketer can conclude that his campaigns are ineffective:



You can read more about global control groups here.

NPS testing in Forms

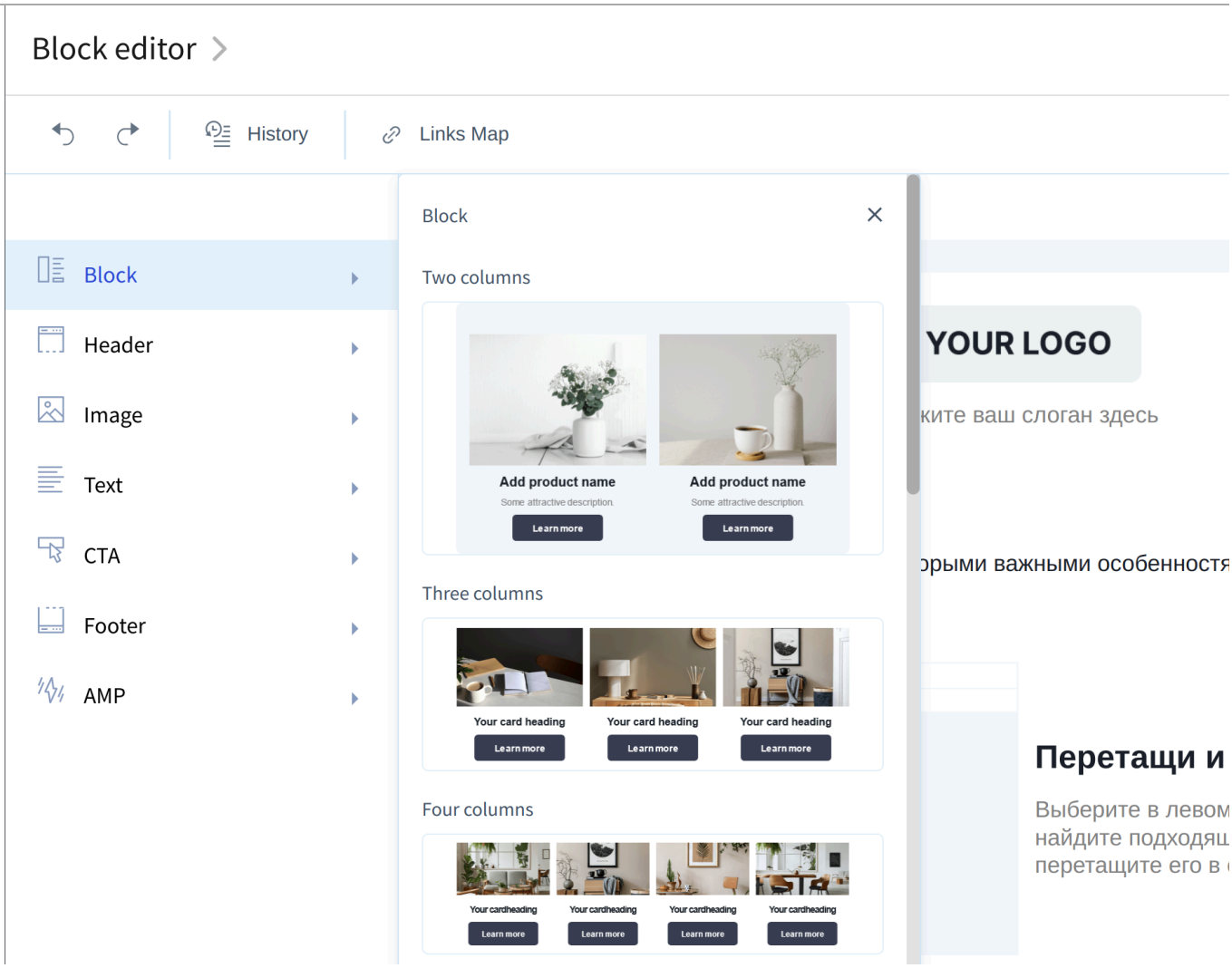
You can now add a NPS survey to web forms created in the Altcraft platform. These surveys help you c (customer loyalty index), a metric that shows customer commitment to a company or product. This ind measuring customer loyalty. NPS allows you to analyze the strengths and weaknesses of a product and



To learn how to set up NPS surveys in the platform, read the article or use this video guide.

Block email editor

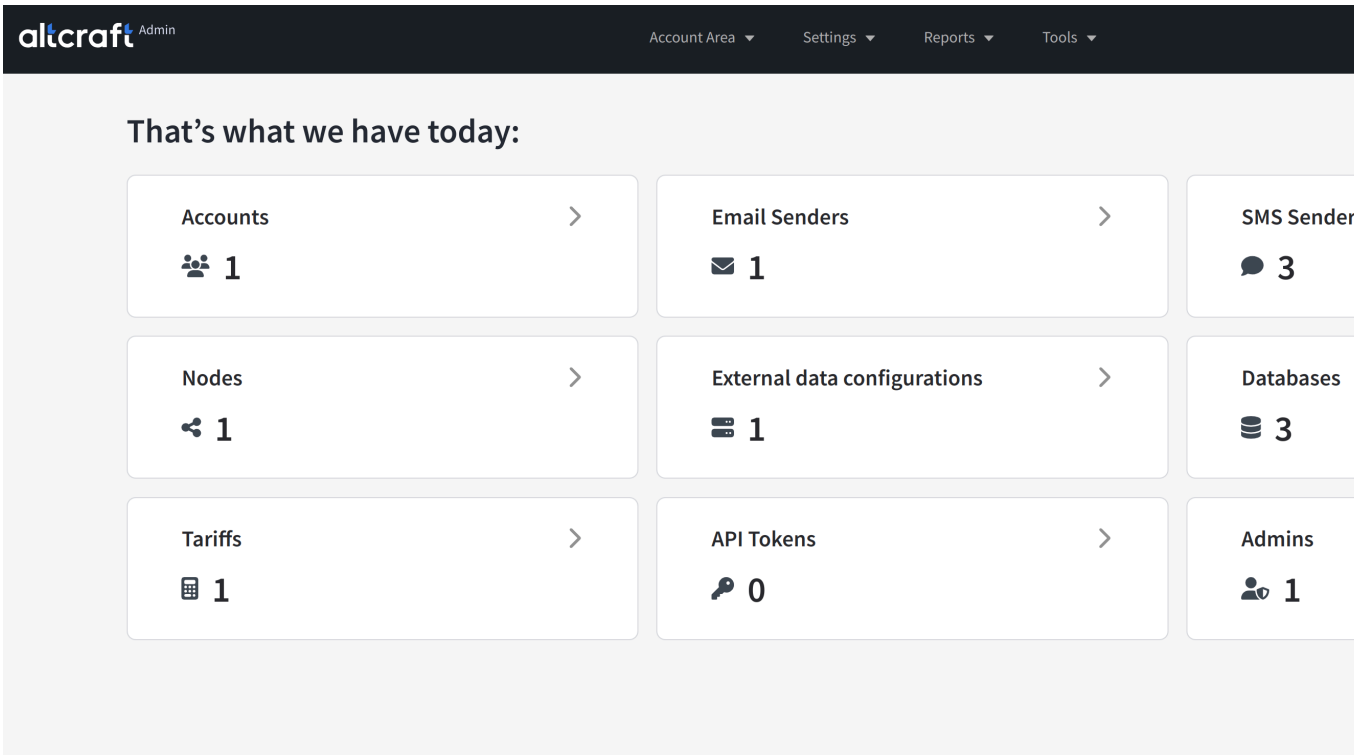
The block editor Pixcraft has been added to the email template editor. It allows you to quickly and easi will display correctly on all types of devices. The editor also supports AMP content for your emails:



You can read more about block editor here.

Redesigned administrative panel

The administrative panel has been significantly redesigned: the interface has been changed, new funct panel engine has been completely updated:



WARNING

Starting with version 72, the platform will require the use of Redis-compatible storage, we suggest us documentation is available here.

Integration with the SMS provider Stream Telecom

Added an integration with the Stream Telecom SMS provider. To set up the integration, configure the c Administrative Panel.

Minor changes

- **Mass operations in the platform (activation, deactivation and deletion) for such elements as script improved.**
- **Added support for Request-ID and Token header passing for API 1.1. requests, also improved reque**
- **The connection interface has been updated.**
- **Added a warning when attempting to delete profiles that have recently participated in campaigns.**
- **Updated the scheduling settings interface for some platform sections.**
- **Clicking the "Save" button no longer closes the editor or redirects the user to the section menu.**

Fixes

- **Fixed a bug where the status of machine learning settings was not saved in the platform.**
- **Now the user cannot delete entities from the cart if the account is frozen.**
- **Fixed a bug that caused clicks in SMS and push messages to register unnecessary delivery events**
- **Fixed errors in Tag Manager variables.**
- **Scenarios configured to capture when a pixel is activated now work correctly.**
- **Made some changes to the Rich Push editor.**
- **Fixed a bug where while importing products with custom fields, an attempt was made to re-create**
- **Screening a link with parameters inside a pixel now works correctly.**
- **Improved stability of the `procworkflow`, `procintegras`, `proctask`, `proctrigger`, `procevent` and `akmtad` proce**
- **Fixed an issue where a Viber campaign with SMS Traffic sent a message to only one subscriber.**
- **Information about a profile's participation in a scenario is now updated correctly.**
- **Resolved an issue where login credentials were not displayed in the welcome email.**
- **Messages in RabbitMQ queues are now processed correctly again.**
- **Added validation for file extensions when creating a segment based on the "In file column" conditi**
- **Special characters in file names no longer cause errors when querying the data table.**
- **The SMS template is now can be saved even if the "SMS Limit" field is empty.**
- **Resolved issues with test sending for Telegram channels.**
- **Form data is no longer duplicated when transferred to the platform.**
- **Fixed an issue where clicking links in push notifications did not work in Mozilla Firefox.**
- **Telegram campaigns now use less memory when launched.**
- **Improved stability of the Analytics module: fixed various issues related to report generation and da**
- **The `$dateNowFunc` function in JSON requests now works correctly again.**
- **Fixed an issue where tags were not added to multiple objects.**
- **Template preview images are now exported correctly.**
- **Fixed errors in the `market/sku/import` and `profiles/segstatic/list` API requests.**
- **Added validation for deleting external databases.**
- **Deep Links no longer disappear when saving a template in the push message editor.**
- **Tags in scenarios are now displayed correctly.**
- **Fixed an issue where a campaign would not open when navigating from a report or profile history.**
- **Profiles frozen in a scenario are no longer deleted when the scenario is modified.**
- **Fixed a bug where AKMTA was sending empty messages.**
- **Day of the week waiting condition works correctly in scripts again.**
- **Improved validation of email addresses in the list of recipients of a hidden copy of an email.**
- **Push message receive and open events are logged correctly.**
- **Optimized the process of generating and sending Push messages.**
- **Improved logging for certain processes.**
- **Interface improvements: fixed display issues for some elements and resolved errors and typos in p**

References

v2025.1.71.0: PL-17608 PL-17599 PL-17574 PL-17572 PL-17570 PL-17564 PL-17558 PL-17556 PL-17549 I 17518 PL-17516 PL-17508 PL-17497 PL-17494 PL-17487 PL-17486 PL-17470 PL-17459 PL-17455 PL-1744 17410 PL-17402 PL-17398 PL-17374 PL-17373 PL-17372 PL-17369 PL-17327 PL-17322 PL-17305 PL-1730 17282 PL-17278 PL-17277 PL-17259 PL-17258 PL-17247 PL-17240 PL-17220 PL-17208 PL-17187 PL-1717

17084 PL-17078 PL-17074 PL-16987 PL-16979 PL-16954 PL-16953 PL-16930 PL-16907 PL-16854 PL-1670 16640 PL-16607 PL-16600 PL-16579 PL-16558 PL-16542 PL-16539 PL-16537 PL-16523 PL-16515 PL-1651 16477 PL-16468 PL-16460 PL-16430 PL-16414 PL-16410 PL-16406 PL-16404 PL-16403 PL-16386 PL-1635 16328 PL-16296 PL-16247 PL-16201 PL-16144 PL-16069 PL-16024 PL-16003 PL-15998 PL-15997 PL-1599 15829 PL-15825 PL-15777 PL-15647 PL-15639 PL-15557 PL-15498 PL-15404 PL-15379 PL-15378 PL-1537 14968 PL-14230 PL-13369 PL-12949 PL-12732 PL-12297 PL-17562 PL-17634 PL-17579 PL-17613 PL-1703 17135 PL-16695

---

v2025.1.71.255 - Mandatory use of the resource in campaigns

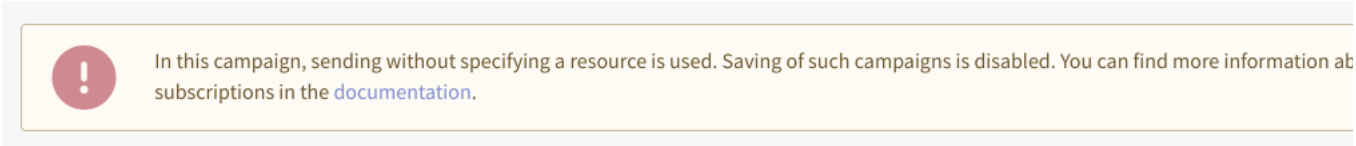
CAUTION

Starting with the next update, sending campaigns without an assigned resource will not be possible! If you have been using campaigns without a resource, please assign a resource to them.

Major changes

Mandatory use of the resource in campaigns

From the next update, all campaigns in the platform will use a resource. Therefore, it is now impossibl specifying a resource:



Minor changes

- **Improved logging of Reply processing in AKMTA.**
- **Added configuration parameter responsible for the maximum size of unloaded data from a SQL tabl**

Fixes

- **Fixed display problems with some interface elements.**
- **Improved operation of scenario elements “Capture profiles on pixel activation” and “Profile update from the scenarios.**
- **Fixed errors occurring when registering Viber events in SMS Traffic integration.**
- **Viber integration with Rapporto now works correctly on all accounts.**
- **Optimized `procevent` queue parsing process.**
- **Fixed a bug where deleting profiles was available to users without proper permissions.**
- **Profile counter in a segment now works correctly when sending via Telegram bot.**
- **Fixed an issue where data received from push messages was not fully transferred to the platform.**
- **Improved stability of `procworkflow` and `proctask` processes.**

References

v2025.1.71.255: PL-17644 PL-17561 PL-17618 PL-17648 PL-17643 PL-17640 PL-16953 PL-17552 PL-1755 17270 PL-17039 PL-17031 PL-16369 PL-15722 PL-17164 PL-17650 PL-17597 PL-17360 PL-17680 PL-1576

---

v2025.1.71.507 - Restructuring the policy counter repository

CAUTION

You will need to migrate for the platform to work correctly after the upgrade. Before launching the ne

1. **Stop the required platform services:**

```
./akd stop {proctrigger,procworkflow,proctask}
```

2. **Run `aktool` from a new build with the command:**

```
./aktool migrate policystore --config /opt/MarketingPlatform/config/main.json
```

3. **Restart the platform services:**

```
./akd restart
```

Help for the migrator:

```
Migrate policystore db to accounts dbs collections policy_contact, policy_profile

Usage:
  aktool migrate policystore [flags]
  aktool migrate policystore [command]

Available Commands:
  ensure_policystore_indexes Ensures policystore migrate indexes created

Flags:
      --account int          account id
      --batch_interval int   batch interval in seconds (default 1)
      --batch_size int       batch size (default 10000)
      --batch_workers int    number of batch workers (default 8)
  -h, --help                 help for policystore
      --workers int          number of workers for accounts (default 8)

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output

Use "aktool migrate policystore [command] --help" for more information about a command.
```

Changes

- **Optimization of the repository structure of sending policy counters has been performed.**
- **Metrics on API calls to the scenarios are now available in analytics.**
- **Special characters can now be used in entity names.**
- **Fixed some bugs in the functionality and interface of the Administrative Panel.**
- **Logging of some platform processes has been improved.**

Fixes

- **Fixed an error where push campaigns generated a large number of non-existent undeliveries.**
- **Statuses in SMS integration with Devino Telecom are now processed correctly.**
- **Fixed errors occurring when testing custom channel campaigns.**
- **Entity analytics with HTML tag in the name works correctly again.**
- **Fixed errors occurring when registering events in Viber integration with SMS Traffic.**
- **Improved stability of Web Push SDK methods.**
- **Fixed some interface bugs.**
- **Improved stability of `proctask` process.**

References

v2025.1.71.507: PL-17756 PL-17753 PL-17446 PL-17727 PL-17731 PL-17728 PL-17715 PL-17720 PL-1758 17704 PL-17648 PL-17698 PL-17677 PL-17670 PL-17656 PL-17642 PL-17626 PL-17641 PL-17249 PL-1727 17211 PL-17111 PL-16993 PL-16354

---

v2025.1.71.627 - RMQ buffer base mode

Changes

- **Added buffer base mode for RMQ in `prochook`.**
- **Metrics on API calls to the scenarios are now available in analytics.**
- **Special characters can now be used in entity names.**
- **Added the option to configure `global qos` for RMQ.**
- **Fixed some bugs in the functionality and interface of the Administrative Panel.**
- **Logging of some platform processes has been improved.**

Fixes

- **Fixed an error where push campaigns generated a large number of non-existent undeliveries.**
- **Statuses in SMS integration with Devino Telecom are now processed correctly.**
- **Fixed errors occurring when testing custom channel campaigns.**
- **Fixed a bug related to incorrect display of inactive campaign status.**
- **Entity analytics with HTML tag in the name works correctly again.**
- **Fixed errors occurring when registering events in Viber integration with SMS Traffic.**
- **Improved stability of Web Push SDK methods.**
- **Fixed some interface bugs.**
- **Improved stability of `proctask` process.**

References

v2025.1.71.627: PL-17756 PL-17753 PL-17446 PL-17727 PL-17731 PL-17728 PL-17715 PL-17720 PL-1758 17704 PL-17648 PL-17698 PL-17677 PL-17670 PL-17656 PL-17642 PL-17626 PL-17641 PL-17249 PL-1727 17211 PL-17111 PL-16993 PL-16354

---

v2025.1.71.873 - SSL support for CH

Major changes

SSL support for ClickHouse

Added support for SSL-secured connections to ClickHouse DB. To enable this, activate the correspondi

Minor changes

- **The global control group now retains groups assigned to the original profile database.**

Fixes

- **Custom channel events are now displayed correctly in profile history.**
- **Improved message sending via Telegram bots in scenarios.**
- **Campaign testing within scenarios is functioning correctly again.**
- **Fixed an issue where subscription fields for user channels were not saved.**
- **Relations now work correctly in message templates.**
- **Fixed an issue causing the `send_message_id` parameter to take different values for the same message**
- **Phone number filtering in scenarios is functioning correctly again.**
- **Fixed an issue causing sending policy parameters to change randomly.**
- **Several improvements and fixes in the Admin Panel.**
- **Fixed issues related to formatting floating-point numbers.**
- **Improved profile history and certain logs.**
- **Fixed an unintended subscription priority change when modified via API.**

References

v2025.1.71.873: PL-17067 PL-17351 PL-17361 PL-17888 PL-17876 PL-17814 PL-17801 PL-17536 PL-1774 17433 PL-17824 PL-17719 PL-17846 PL-17555 PL-17810 PL-17774 PL-17757 PL-17755 PL-17587 PL-1744 17627 PL-17703 PL-17796 PL-17707 PL-17729 PL-17472 PL-17708

---

v2025.1.71.1059 - Scenario API improvements

Major changes

Scenario-related API improvements

- **Added activation and deactivation requests for a scenario: `/workflows/activate` and `/workflows/deactiva`**
- **Scenario API responses now include campaign and template identifiers, as well as information abo**

Minor changes

- **The maximum length of scenario titles has been increased to 156 characters.**
- **Improved logging of certain platform processes.**

Fixes

- **Improved layout for several UI elements.**
- **Fixed an issue where test launches of campaigns using a dynamic segment as a subscriber source**
- **LDAP authentication now works correctly again.**
- **The campaign filter now finds campaigns that were created within a scenario.**
- **Fixed errors occurring during Yandex.Audience segment resynchronization attempts.**
- **Fixed an issue where form publish and hide times were incorrectly set in the settings.**
- **Tracking services again correctly listen to the address and port passed through parameters.**

References

v2025.1.71.1059: PL-17896 PL-17938 PL-16077 PL-17925 PL-17832 PL-17831 PL-17823 PL-16045 PL-168 17930 PL-17482 PL-17899 PL-17538 PL-17291 PL-17762 PL-17668 PL-17664 PL-17636 PL-17928 PL-1788

---

v2025.1.71.1487 - API calls optimization, improved logging

Changes

- Extended data type support in ClickHouse table integrations.
- Added the option to specify the launch time of global control groups in the platform configuration f
- The platform now logs the processing time for each profile at every campaign stage.
- Added a button to stop segment recounting.

Fixes

- Fixed an issue where data from a deleted profile field was still returned via the API.
- Duplicate relations no longer appear when created via API.
- Optimized the `/profiles/import_batch` API method.
- The formatted date variable in the “API call” scenario element works correctly again.
- Fixed font rendering issues in popups.
- Links to the web versions of emails now work correctly in test sends.
- Fixed several UI-related bugs.
- Render errors are now properly displayed in the campaign log.
- Fixed an issue where templates and campaigns were not moved to the trash.
- The “Profile update” scenario element can now set a null value.
- The `blockquote` tag renders correctly in Telegram templates.
- Fixed several bugs related to custom channel.

References

**v2025.1.71.1487: PL-17984 PL-18003 PL-17901 PL-17947 PL-15365 PL-13140 PL-17926 PL-17900 PL-179
PL-17052 PL-17356 PL-15570 PL-17949 PL-17254 PL-17786 PL-17663 PL-17701 PL-17553 PL-18018 PL-1
PL-17931 PL-17525 PL-17968 PL-17938 PL-16892 PL-17965 PL-17097 PL-17280 PL-17808 PL-17841 PL-1
PL-11838 PL-17241 PL-17074 PL-17738 PL-17689 PL-17603 PL-15518 PL-8650 PL-16616**

---

v2025.1.71.2447 - SberMobile and Intellin integrations

Major changes

New integrations: SberMobile and Intellin

The following integrations have been added:

- integration with the Intellin SMS provider;
- integration with the SberMobile SMS gateway.

Improved data logging for sends and scenarios

- Logs for Notify requests have been added to retrieve integration statuses. These logs help you und have become inactive.
- The USMID has been added to AKMTA headers. USMID is a unique message identifier that contains send. Knowing it can help you analyze possible delivery issues more easily.

Multiple links in “Link click” condition

You can now list multiple URLs in the segmentation condition Link click.

Minor changes

> CAUTION
>
> **If you specified an address in the RPC_HOST_CLI parameter, all clients (even from other hosts) will co installing the update, be sure to set a public or “gray” address for this parameter.**

- The `procworkflow` RPC client now connects using the address set in the dedicated configuration par
- Added the ability to batch-send web versions of messages to S3 storage.

Fixes

- Fixed issues related to two-factor authentication in the Admin Panel.
- Improved the stability of the Tilda integration.
- Fixed a bug where scheduled segments were not recalculated when a scenario was launched.

- **Revised the logic for sending form submission reports, improving data accuracy.**
- **The `format` function now works correctly in loops and when used with `apicontent`.**
- **Improved click registration in Telegram bots.**
- **Delivery statuses and duplicate messages are now correctly registered for SMS campaigns.**
- **Resolved issues where users without appropriate roles could access platform entities.**
- **Test campaigns with dynamic subscriber segments now launch correctly.**
- **Fixed several UI/UX issues in the interface.**
- **Resolved errors related to using conditions in segments and scenarios.**
- **Custom bounce rules for AKMTA now function correctly again.**
- **Improved stability of the following processes: `webcontrol`, `procintegras`, `proctask`, `prochook`, and `procw`**

References

v2025.1.71.2447: PL-18148 PL-18022 PL-18189 PL-18131 PL-18102 PL-17352 PL-17981 PL-17931 PL-173 PL-18194 PL-17998 PL-18130 PL-15386 PL-18079 PL-18077 PL-18178 PL-18165 PL-18123 PL-18105 PL-1 PL-17695 PL-18048 PL-18047 PL-15625 PL-17771 PL-17735 PL-17770 PL-17689 PL-17653 PL-15344 PL-1 PL-18098 PL-17804 PL-18060 PL-18084 PL-17799 PL-18050 PL-17976 PL-17983 PL-17721 PL-17395 PL-1 PL-17275 PL-17803 PL-18097 PL-17463 PL-18074 PL-17179 PL-15009 PL-17861 PL-17977 PL-17874 PL-1 PL-16022 PL-17105 PL-17489 PL-17972 PL-17294 PL-17942 PL-17836 PL-17488 PL-17747 PL-17943 PL-1 PL-17815 PL-17950 PL-15815 PL-17286 PL-17896 PL-17887 PL-8823 PL-16693

---

v2025.1.71.2873 - Hive connector support

CAUTION

You will need to migrate for the platform to work correctly after the upgrade. Before launching the ne

1. Stop the required platform services:

```
./akd stop {proctrigger,procworkflow,proctask}
```

2. Run `aktool` from a new build with the command:

```
./aktool migrate policystore --config /opt/MarketingPlatform/config/main.json
```

3. Restart the platform services:

```
./akd restart
```

Help for the migrator:

```
Migrate policystore db to accounts dbs collections policy_contact, policy_profile

Usage:
  aktool migrate policystore [flags]
  aktool migrate policystore [command]

Available Commands:
  ensure_policystore_indexes Ensures policystore migrate indexes created

Flags:
      --account int          account id
      --batch_interval int   batch interval in seconds (default 1)
      --batch_size int       batch size (default 10000)
      --batch_workers int    number of batch workers (default 8)
  -h, --help                 help for policystore
      --workers int          number of workers for accounts (default 8)

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output

Use "aktool migrate policystore [command] --help" for more information about a command.
```

Major changes

SQL Connector Hive

Added support for the SQL Hive/Hadoop connector as an external database. Learn more about configur

Minor changes

- **Updated APIs for the new versions of Google Ads and Facebook™* Ads Manager.**
- **Added support for Rabbit version 4+.**
- **The Altcraft platform has been certified for compatibility and proper operation with the domestic o**

- **The `MTA_DEFAULT_TTL` parameter, which defines the delivery waiting time for SMS and Email, now defa**
- **The `"not\\savailable"` blocking rule has been removed from the MTA sender rules because it caused operation.**

Fixes

- **Enhanced platform security mechanisms.**
- **Fixed various UI/UX issues.**
- **Improved stability of `akd`, `akmta`, `aktool`, `procpiper`, `proctrigger`, `procintegras`, `proctask`, and `procworkfl`**
- **Optimized `oxy_triggers` queue parsing.**
- **Custom bounce rules in AKMTA now work correctly.**
- **Optimized log rotation mechanism.**
- **Web versions of emails sent through the platform are now saved in the correct format.**
- **Improved 2FA stability.**
- **Fixed issues related to editing message templates within the platform.**
- **JSON filter now works correctly for nested objects.**
- **Fixed an issue where form data was mistakenly submitted twice.**
- **Added validation for certain fields in the Admin Panel.**
- **Fixed some issues related to access roles.**
- **Additional profile fields are now exported correctly in personal data exports.**
- **Fixed an issue where `apicontent` was incorrectly handled during bulk profile imports into a Scenario**

References

v2025.1.71.2873: PL-17550 PL-18489 PL-18393 PL-18384 PL-18106 PL-18136 PL-15177 PL-18457 PL-182 PL-18104 PL-15824 PL-18424 PL-18401 PL-18312 PL-18238 PL-18244 PL-18139 PL-18415 PL-18238 PL-1 PL-18331 PL-18339 PL-18287 PL-15410 PL-18294 PL-18289 PL-18135 PL-18235 PL-18314 PL-18290 PL-1 PL-18281 PL-18154 PL-18229 PL-18150 PL-17878 PL-17807 PL-18283 PL-18268 PL-17604 PL-18038 PL-1 PL-18265 PL-18255 PL-17816 PL-18053 PL-18161 PL-17546 PL-18196

---

v2025.1.71.3092 - Event backward compatibility

Changes

- **Added backward compatibility for events from custom channels of legacy platform versions.**
- **Security mechanisms optimized.**

Fixes

- **Test sends to Telegram now work correctly.**
- **The value `0` in the “in data table” condition is now displayed when passed via query parameters.**
- **Fixed an issue in the form editor where incorrect access rules were applied.**
- **Added a timeout for hanging queries in PostgreSQL.**
- **The ability to specify a custom configuration file path for platform services now works properly.**
- **Improved readability of API logs.**
- **Tags and additional fields are now correctly processed when creating objects via API.**
- **The visual editor now correctly updates custom block settings.**
- **Fields of type “Integer” now work properly when forming a control group from a file.**
- **Reduced memory consumption by the `campaign` process when sending Telegram messages with att**
- **Confirmation link check now works correctly in templates.**
- **Fixed the “Prioritize new subscriptions” option in resource settings.**
- **AKMTA now correctly switches to backup MX records when the primary is unavailable.**
- **The profile date is now correctly inserted in templates when using the `formatdate` function.**
- **Fixed issues related to handling events from Amazon SNS sender.**

References

v2025.1.71.3092: PL-10119 PL-18521 PL-18315 PL-17401 PL-18125 PL-18000 PL-17857 PL-17818 PL-182 PL-18478 PL-18269 PL-18193 PL-18384 PL-18323 PL-17622 PL-18527 PL-17548 PL-18041 PL-18425 PL-1

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a banned in the Russian Federation.
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

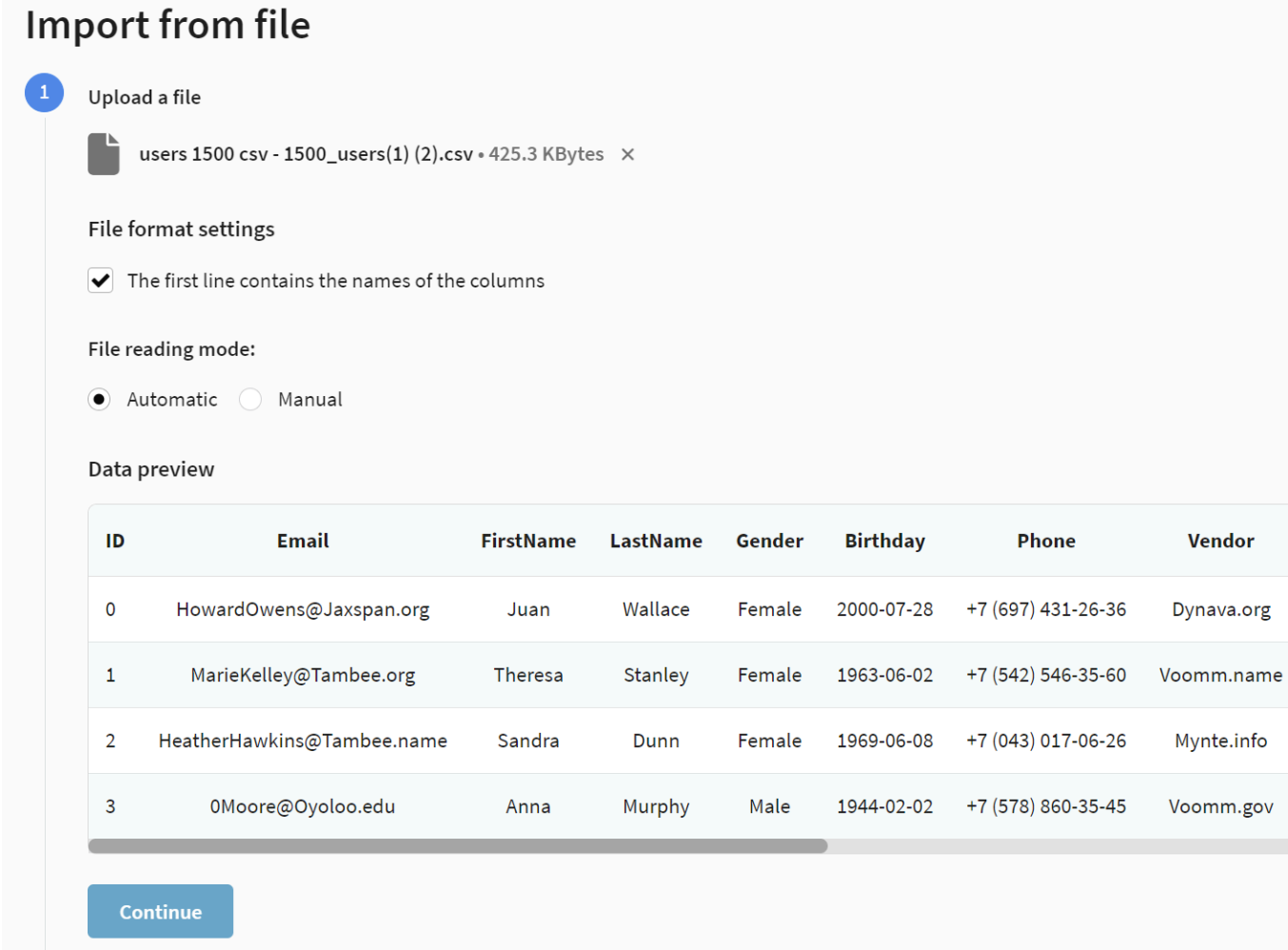
v2024.4.70

v2024.4.70.0 - New import menu, Rapporto integration, License info

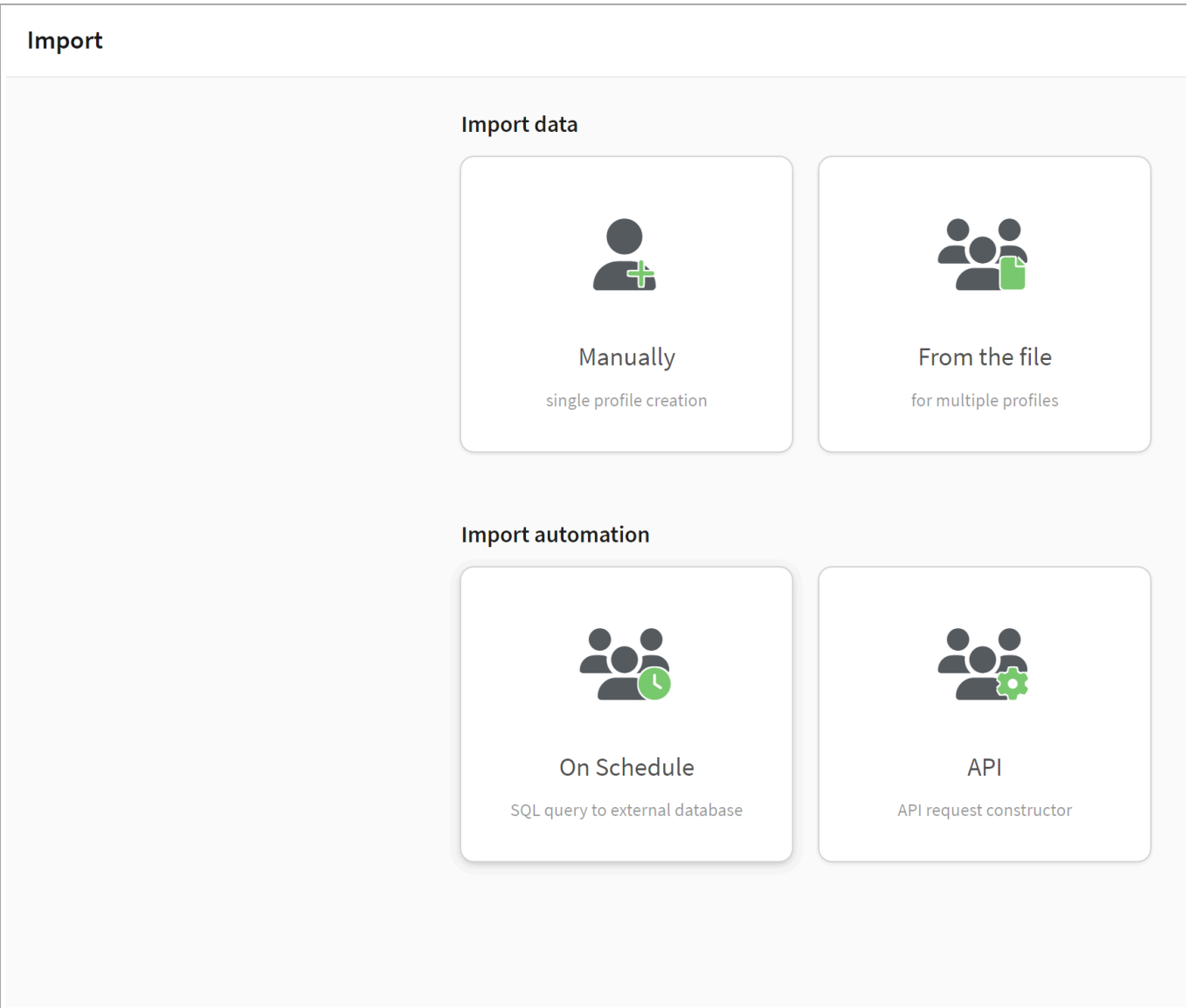
Major changes

Improved import of profiles

Completely redesigned importing profiles from a file. Now when you select this item, a detailed step-by description of all import steps and settings will appear in front of you:

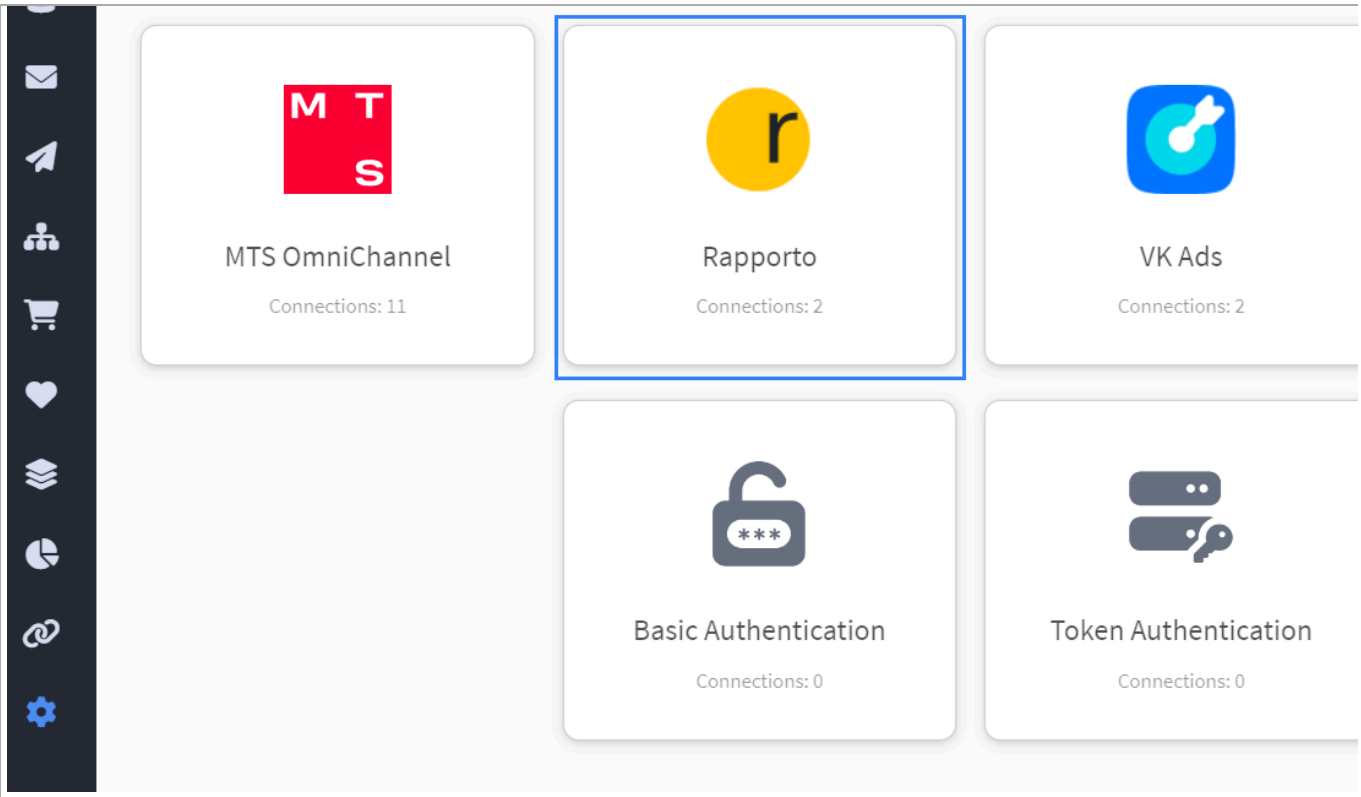


You now have the possibility to select file encoding, delimiter, select date format and edit the format o Also the import menu to the profile database has been updated:



Rapporto integration

Added support for integration with Rapporto. Now you can use this platform as a gateway to send mes



Before setting up the integration, you need to connect your Rapporto account with Altcraft Platform vi
about setting up the integration here.

License information

Added License item in the Tools section of the Administrative Panel. In this item you can see detailed i
version, license and its validity period.
In the Notify about license changes to field, you can specify the email address to which you want to re
status changes or expiration. Notifications will be sent 2 weeks before the license expires and when th

Minor changes

- **New operators have been added to the "JSON filter" scenario element:** `contains`, `in`, `notin` **and** `!~=`
  here.
- **It is now possible to add custom headers to the "API request" scenario element.**
- **You can now send emails without the text version of the message.**
- **The** `genders` **directory has been updated.**

INFO

If you need to keep the old `genders` **directory configuration or make some changes to it, create a new**
"custom.genders.json" and make the necessary edits there. You can read more about it here.

- **Optimized operation of the "Segment" scenario element.**
- **The mechanism of interrupting cycles in scenarios has been redesigned, now it works correctly in**

Fixes

- **Fixed a bug where the exported template did not contain all source data.**
- **Fixed errors occurring during resynchronization with VK Ads service.**
- **Conditions of waiting for a certain day of the week now work correctly in scenarios.**
- **Fixed an issue where some channels could not be selected in trigger campaigns.**
- **Fixed an issue when exporting profiles with subscriptions.**
- **Fixed an issue where segmentation with negative conditions on Push deliveries was incorrectly se**
- **Segmentation condition "Found in list" now correctly selects values containing a slash.**
- **Elements in scenarios now connect graphically to each other correctly.**
- **Fixed a bug where the delivery event for web push in Safari was logged before the actual delivery.**
- **Fixed false triggers of the "Timer" trigger in the Tag Manager.**
- **Fixed errors occurring when setting up connection to Facebook Ads.**
- **Improved analytics on test sends.**

- **Campaign filter now correctly detects campaigns created from scenario.**
- **Fixed a bug where moving campaigns and templates to trash did not work correctly.**
- **Improved stability of `akmtad`, `procworkflow`, `proctrigger` processes.**

References

PL-16980 PL-17004 PL-16990 PL-16934 PL-15925 PL-16939 PL-16170 PL-16565 PL-16649 PL-16306 PL-1
PL-16459 PL-16180 PL-16357 PL-15777 PL-16326 PL-16323 PL-16312 PL-16317 PL-16226 PL-16023 PL-1
PL-12949 PL-16857 PL-17055 PL-17050 PL-17041 PL-17036 PL-17022 PL-17001 PL-16978 PL-16983 PL-1
PL-16875 PL-16856 PL-16790 PL-16545 PL-16707 PL-16299 PL-16310 PL-16244 PL-16045 PL-15518 PL-1

---

v2024.4.70.963 - Action Hook SMID, push suspend method

Major changes

Changes in the identifiers of the action hook messages

The structure of the event capture message has been changed. Now such messages contain fields:

- `send_message_id` **- unique identifier (USMID);**
- `full_message_id` **- FSMID for the event (if any);**
- `email_message_id` **- ESMID for an email channel.**

Introduction of new identifiers allows you to find necessary events and thus avoid difficulties in error a unique message ID, but now it looks much shorter.

New method to suspend push subscription

Implemented a new method to suspend push subscription of users to a resource via web push SDK. Yo configuration here.

Minor changes

- **Added “Text + Image” combination for Viber templates.**
- **Added support for selecting authentication fields for MFMS EDNA SMS gateway.**
- **Expired web version collections are now deleted along with their contents.**
- **Added custom header editor for “API request” scenario element.**

Fixes

- **The counter of imported profiles in the Administrative panel is now automatically recalculated wh**
- **Attaching a file in the Segment section now works correctly.**
- **Fixed a bug where the platform was mistakenly registering more send and open events for some cl**
- **Images are now loaded correctly when importing an HTML template.**
- **Improved stability of the Analytics section, fixed some server errors related to analytics.**
- **Winner selection by clicks in multivariant tests now works correctly.**
- **Fixed a bug causing duplicate subscriptions for a profile when clicking the save button.**
- **Statuses on sends via WhatsApp* channel now arrive correctly.**
- **Improved validation when filling out fields in some sections of the platform.**
- **Import via file works correctly again for clients of the cloud version of the platform.**
- **Fixed an error occurring when connecting PostgreSQL connector to database on Yandex Cloud.**
- **Import via zip archive works correctly again.**
- **Fixed bug related to reaching the maximum limit of active sessions in MongoDB.**
- **Profiles caught in a scenarios are now parsed evenly by handlers without creating delays.**
- **Fixed a bug where a part of a message was cut off when previewing an SMS template.**
- **The Segment Participation condition now does not display the last created segment if the segmen**
- **Improved full screen mode of template editor on Safari.**
- **Email events in Sendsay integration are now logged correctly.**
- **Fractional number selection is now available again in the segment by numeric target data.**
- **Removed PostgreSQL queries to Market in scenarios where they don’t need to be initiated.**
- **Template with custom structure is now correctly imported into the platform.**
- **Fixed a bug where the platform would not start when one node in a ClickHouse cluster was unavai**
- **Day of week wait condition now works correctly in scenarios again.**
- **Push notifications with special Unicode character sequences are now sent correctly on Firebase i(**
- **Fixed auto-alignment algorithm in scenarios.**
- **When sending an empty SMS template, an error entry now appears in the campaign log.**

- **Event type in Notify is now correctly displayed when exporting profile history.**
- **Fixed a bug where it was possible to create a product without filling in mandatory fields.**
- **Fixed errors related to saving or deleting a subscription to a user channel resource.**
- **Disabling notifications in Safari browser now works correctly.**
- **Supported file formats are now listed for importing from a file in Market.**
- **Fixed an issue where when copying a segment, the segment structure would retain incorrect cond**
- **The irrelevant error “Config server db posgres is missing for account” has been removed from the**
- **Fixed errors related to saving Telegram templates.**
- **Improved stability of `procworkflow`, `webcontrol` processes.**
- **Fixed some bugs in the platform interface.**

References

PL-17403 PL-17338 PL-17301 PL-16970 PL-17313 PL-17190 PL-17167 PL-17116 PL-13523 PL-17152 PL-1
PL-17069 PL-16689 PL-16988 PL-16645 PL-16785 PL-16811 PL-16729 PL-17290 PL-16126 PL-17333 PL-1
PL-17261 PL-17129 PL-17202 PL-17238 PL-17226 PL-16615 PL-17217 PL-16840 PL-17148 PL-17201 PL-1
PL-17171 PL-15854 PL-17151 PL-16967 PL-17117 PL-17124 PL-17130 PL-17142 PL-15860 PL-17093 PL-1
PL-16803 PL-16802 PL-17034 PL-15434 PL-16789 PL-16440 PL-16821 PL-16819 PL-17196 PL-17081 PL-1
PL-13026 PL-16823 PL-17185 PL-17086 PL-14429 PL-16866 PL-16887 PL-17073 PL-15837 PL-17007 PL-1
PL-17019 PL-16859 PL-16941 PL-16992 PL-17012 PL-17011 PL-16202 PL-15451 PL-15679 PL-16627 PL-1
PL-16753 PL-2915 PL-5732 PL-11319 PL-16381 PL-16313 PL-16237 PL-17053 PL-16847 PL-17020 PL-165

---

v2024.4.70.1088 - Web services handlers, stability improvements

Major changes

API requests to web services handlers

Correct balancing of web services requires from the user a constant and accurate transfer of their sta
API requests to process status handlers have been implemented. Requests to the handlers of the follo
`webcontrol`, `webadmin`, `webapi`, `trkaction`, `trkwebctrl`, `cookiesaver`.
You can learn more about these API requests at link.

Minor changes

- **Added validation when uploading a video file in Telegram template.**
- **Profile history is now updated faster, ensuring correct operation of scenarios with complex logic.**

Fixes

- **Temporary platform files are now correctly cleared after the expiration of the lifespan period.**
- **Fixed incorrect display of growth report indicators.**
- **Fixed issues with the display of some elements in the interface.**
- **Fixed errors that caused the log of actions in the Administrative panel to record not all events.**
- **Improved stability of `procworkflow` and `proctask` processes.**

References

PL-17467 PL-17445 PL-17127 PL-17267 PL-17182 PL-17090 PL-16638 PL-16931 PL-17183 PL-17016 PL-1

***Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a**
banned in the Russian Federation.
****Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.***

v2024.3.69

v2024.3.69.0 - Altcraft mSDK, Notify

Major changes

Mobile push SDK

Altcraft mSDK is a mobile programming interface (SDK) for integrating push notifications into mobile a connect Altcraft's notification system to their Android and iOS apps via Firebase and APNs platforms.

Altcraft mSDK allows you to:

- **Automatically subscribe users to push notifications.**
- **Support sending notifications with text, images and action buttons.**
- **Update and refresh device tokens on the server for seamless notification delivery.**
- **Track notification delivery and user interaction with notifications.**
- **Automatically retry failed requests and work in the background.**

> TIP
>
> **Currently, the Altcraft mSDK is available upon request in the form of an archive.**

Migration of mobile push templates

The structure of mobile push templates has been updated in the following files:

- `json_consts/templates/AndroidFirebase.json`
- `json_consts/templates/IOSAPNS.json`
- `json_consts/templates/IOSFirebase.json`

If you are sending mobile push notifications through these providers and haven't replaced these files w to move the content of the original files before updating.**

To do this, copy the contents of each original file into the corresponding `custom.` prefixed files:

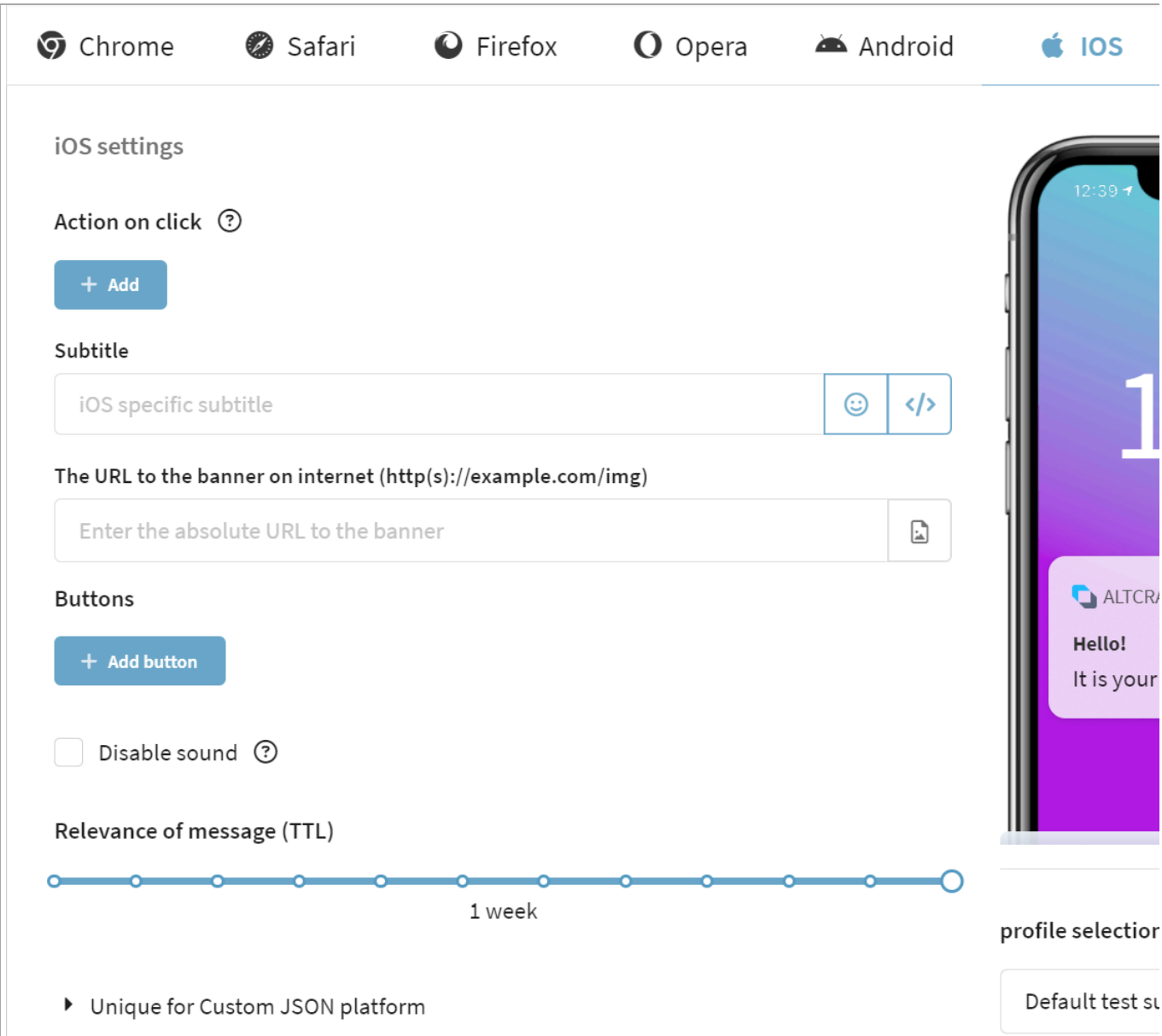
- `json_consts/templates/custom.AndroidFirebase.json`
- `json_consts/templates/custom.IOSAPNS.json`
- `json_consts/templates/custom.IOSFirebase.json`

Or copy them into files linked to your account:

- `json_consts/templates/custom.<account_number>.AndroidFirebase.json`
- `json_consts/templates/custom.<account_number>.IOSAPNS.json`
- `json_consts/templates/custom.<account_number>.IOSFirebase.json`

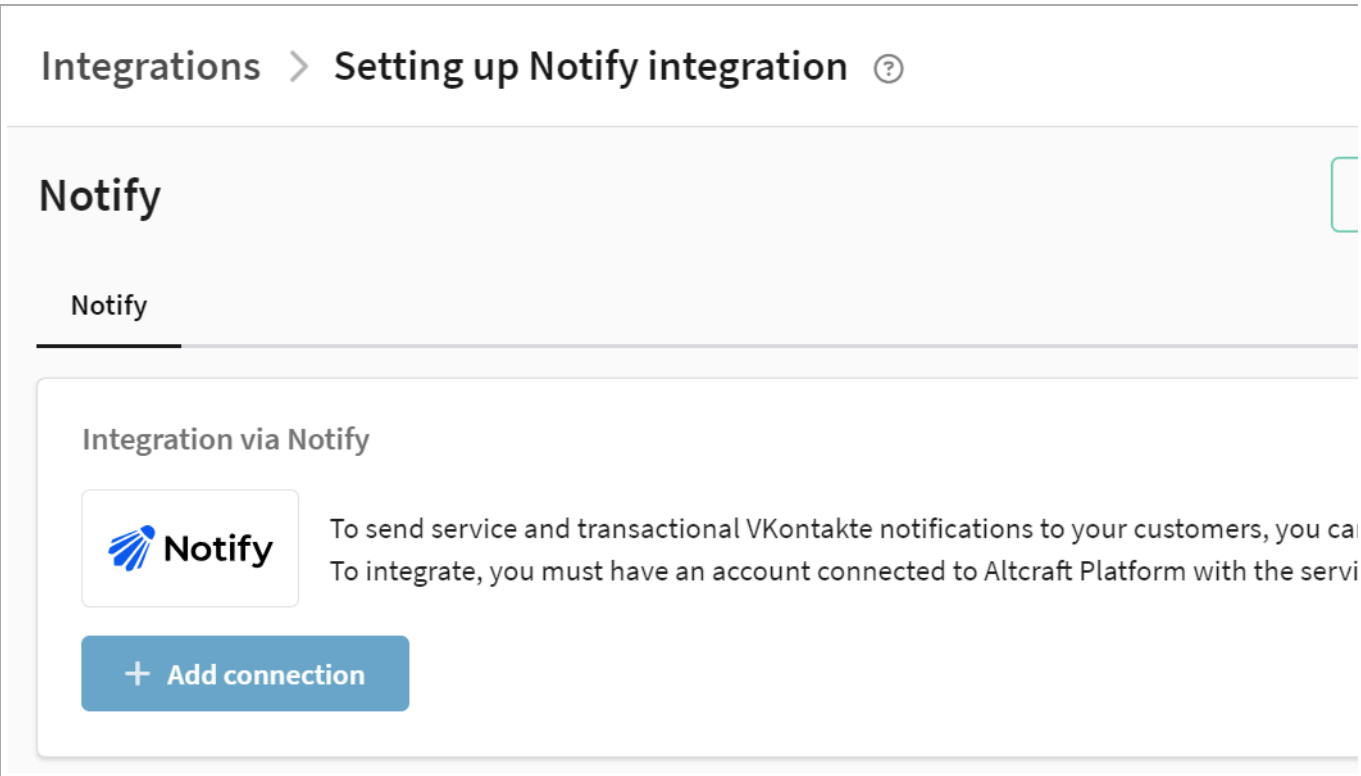
Mobile Push Editor update

Mobile Push Editor has been updated (editor β). Removed redundant customization fields, added new b action, implemented banner loading from the platform gallery, added the ability to mute. The "Critical I been removed for iOS.



Notify integration

The platform now features a direct integration with [Notify](), allowing you to send transactional and serv Odnoklassniki, and SMS. Implemented the ability to configure the resource channeland create custom



Minor changes

- **Added correct XML support to API 1.1 version of the platform. Learn more.**
- **Improved scenarios. Added possibility of scaling on the touchpad. Changed the system of setting li**
- **Template previews have been improved.**

Bug fixes

- **Fixed an issue causing campaigns to hang during the RMQ reconnection stage due to high memory**
- **Export of profile history now executes faster.**
- **Changing the API token in the settings of the web push channel for Safari is now successfully save**
- **The analytics for undeliverables now correctly displays statistics for web push notifications.**
- **In the profile field of type array of objects, it is now possible to manually save an empty array by s| the API by passing `"[]"` or `""` to the field. A warning has been added when selecting an expired la schedule.**
- **A warning has been added when saving a scenario if no conditions are selected for the "Scheduled segment is chosen, to prevent accidentally running it on the entire database.**
- **The HTTP(S) protocol is now displayed in the image URLs in the gallery.**
- **TypeScript has been updated for improved support and compatibility.**
- **The stability of the proctask and webcontrol processes has been improved.**
- **Minor interface issues in the platform have been fixed.**

References

PL-16878, PL-16717, PL-16767, PL-16799, PL-16860, PL-16798, PL-16843, PL-16020, PL-16831, PL-16814 PL-16800, PL-16822, PL-16796, PL-16758, PL-16832, PL-16815, PL-16768, PL-16769, PL-16757, PL-16766 PL-16751, PL-16235, PL-16251, PL-16726, PL-15887, PL-15878, PL-15677, PL-16568, PL-16658, PL-16644 PL-16292, PL-16223, PL-6765, PL-15227, PL-16554, PL-16391, PL-16544, PL-16431, PL-16419, PL-14376, 15875, PL-15840, PL-15830, PL-14796, PL-8846, PL-15522, PL-13267, PL-15429, PL-15372, PL-15367, PL

---

v2024.3.69.490 - Batch import API improvment

Major changes

Improved mass import of profiles via API

Optimized API method for mass import of profiles into the database (`/v1.1/profiles/import_batch`). It now t than before.

Minor changes

- **Updated integration with Google Ads due to the release of a new version of Google Ads API.**

- **Added validation for the “Limit” field in the Relations section.**
- **Added validation for the extension of the uploaded file in Telegram templates.**

Interface improvements

- **Fixed display of quotes in tags and groups.**
- **Corrected texts of many errors in platform logs.**
- **Fixed display of elements on the mobile version.**
- **Improved display of dropdown lists.**
- **Added a tooltip in the interface for the “Copy profiles from database” option.**
- **Improved layout in trigger activation conditions.**
- **Fixed a bug where the “Apply” and “Save” buttons were incorrectly displayed in different sections**

Fixes

- **Fixed a bug where some web push notifications were not sent to users.**
- **Request to Close method of MongoDB driver now work correctly.**
- **Fixed an error occurring when navigating between pages in profile activity history.**
- **Profile import requests now display the correct error code.**
- **Fixed a bug where it was not possible to use cyclic enumeration for Market entities and fields.**
- **SASL authorization mode in the Action Hooks setting now works correctly.**
- **Fixed a bug where it was impossible to save a resource for mobile push without an `appId` field.**
- **The sending mode selection for push notifications now displays correctly for all campaigns.**
- **Fixed a bug where some processes would not work when running the platform with SOCKETS enab**
- **Scenarios that have a Firebase API Legacy resource specified in the campaign are now saved corr**
- **Fixed errors occurring when saving funnels with tag filtering.**
- **Queues on the Sender are now removed correctly.**
- **Fixed an issue where part was missing in the html version while saving a template.**
- **Profiles participating in a static segment are now correctly included in a quick segment.**
- **Fixed a bug where the trigger for importing a profile via push did not work.**
- **Fixed an error when the “JSON-filter” scenario element did not work correctly.**
- **Improved stability of the “API Call” element in scripts.**
- **PostgreSQL connector now correctly connects to the database in Yandex Cloud.**
- **Fixed a bug where the “Email subscription: domain (not) in list” condition operator in scenarios sea instead of by domain.**
- **Quick Segment now correctly searches for profiles by specified conditions.**
- **Fixed a bug where a new item was created in an order instead of changing the data of the old one.**
- **Clearing temporary files in `data/segment` now works correctly.**
- **Fixed bugs related to WhatsApp integration.**
- **Fixed memory overload when a database query was added to the template.**
- **Template body is now not imaged when sending push notifications via IOS ANPs.**
- **Fixed errors occurring when sending push notifications with old structure.**
- **The option to send a push notification to multiple identical subscriptions now works correctly.**
- **Profiles with an incorrect phone number are now displayed correctly on export.**
- **Stability of the `akmtad` process has been improved.**

References

PL-16976 PL-16955 PL-16940 PL-16928 PL-16752 PL-16886 PL-16877 PL-15784 PL-15086 PL-16194 PL-1 PL-16929 PL-15817 PL-16922 PL-16897 PL-16905 PL-16910 PL-16196 PL-16881 PL-16840 PL-16876 PL-1 PL-14418 PL-8897 PL-16275 PL-6070 PL-15583 PL-15438 PL-16925 PL-16916 PL-16919 PL-16909 PL-162 16865 PL-15765 PL-16810 PL-16181 PL-16728 PL-16839 PL-16750 PL-16711 PL-16475 PL-14771 PL-1186

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a banned in the Russian Federation.
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

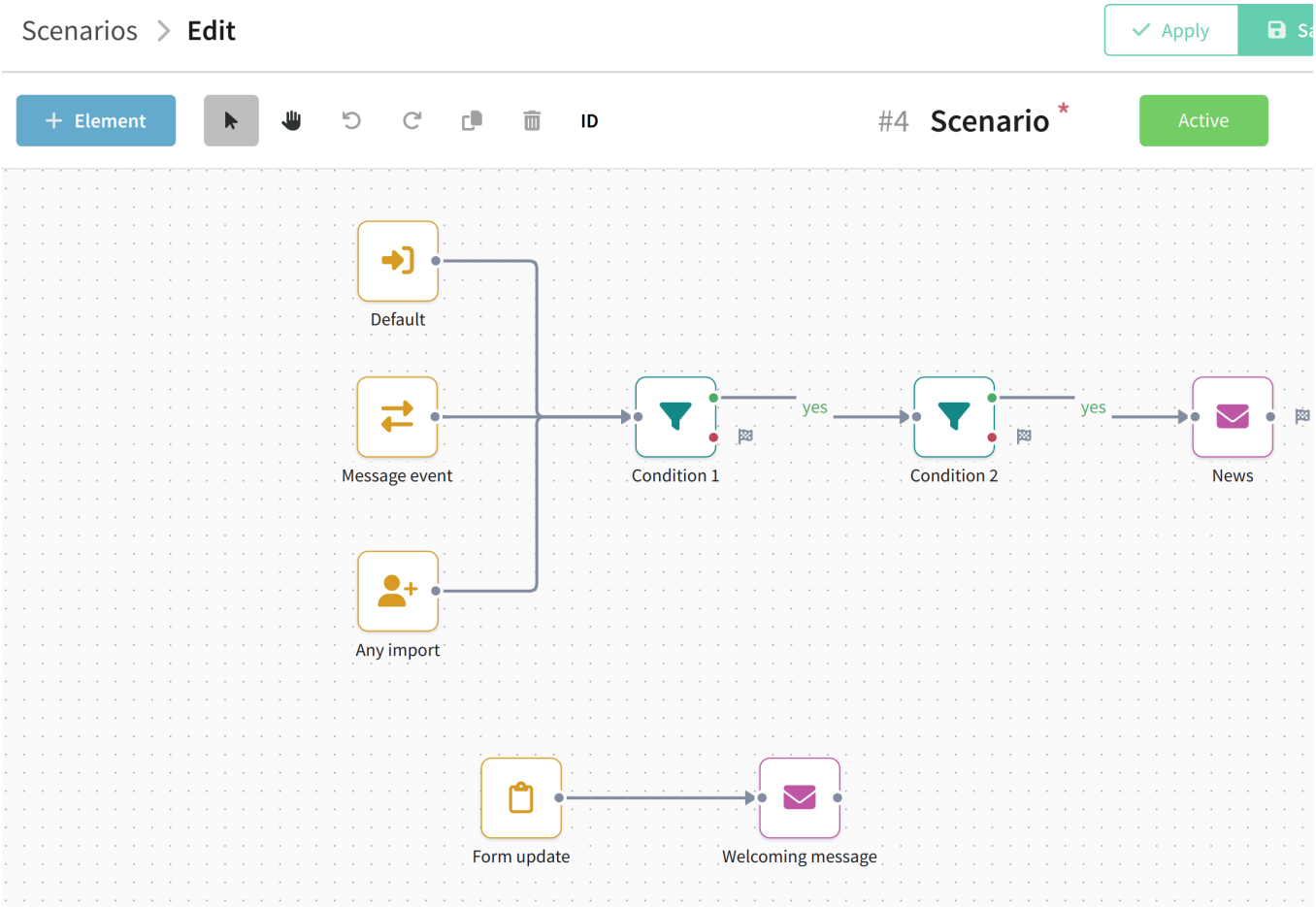
v2024.2.68.2

v2024.2.68.2.287 - Multi-capture scenario, Product feed, Pixel trigger improvements

Major changes

Combined profile capture into a scenario

Added the ability to configure capture profiles with multiple conditions within a single scenario. This c different entry options without creating multiple new scenarios. Up to 10 such conditions can be adde



Also statistics on captured profiles is available, including for each condition separately. This feature al from different business points. You can learn more about customizing the scenario elements right ther

Importing a product feed by link in the Market

CAUTION

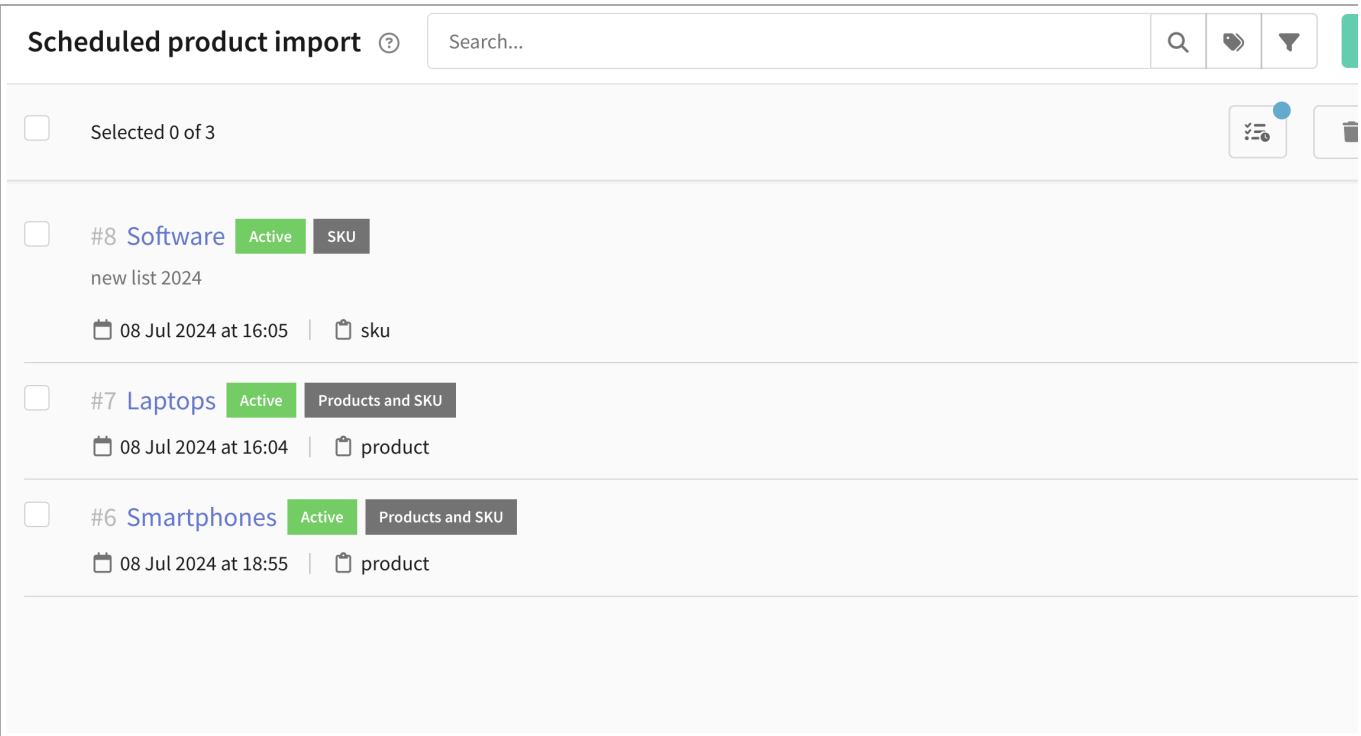
After adding a product feed, you need to migrate for the Market module to work correctly. To do this, the new version:

1. **Stop the `webcontrol` service.**
2. **Start `aktool` from new build with the command:**

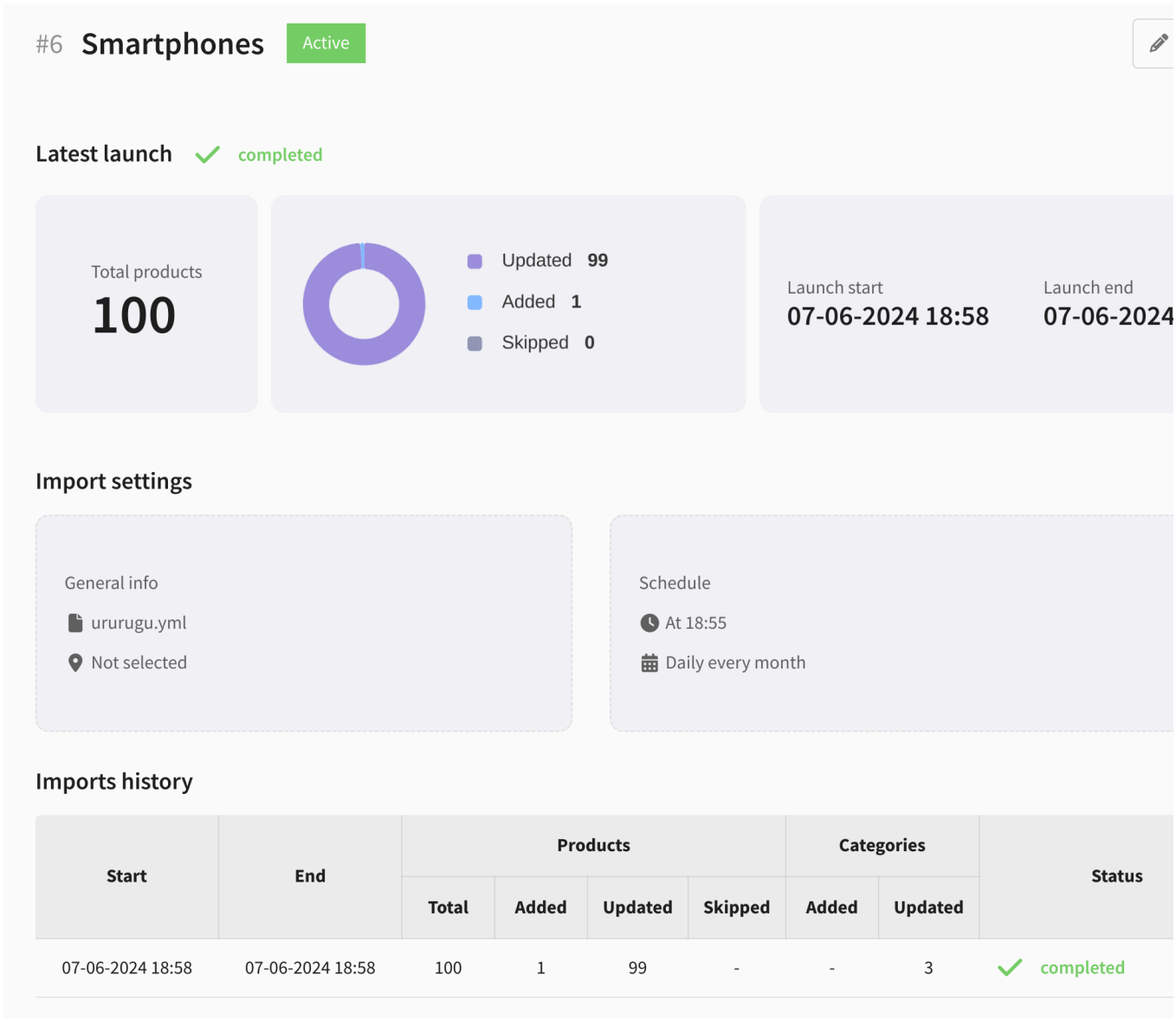
```
./aktool migrate importfeed --config /opt/akd/config/main.json
```

3. **Start the `webcontrol` service.**

Now you can set up synchronization of products and SKUs in the Market with your product feed data. T feature allows you to specify a link to the feed file and the time at which the import will take place. Yo synchronization with product feed in this article.



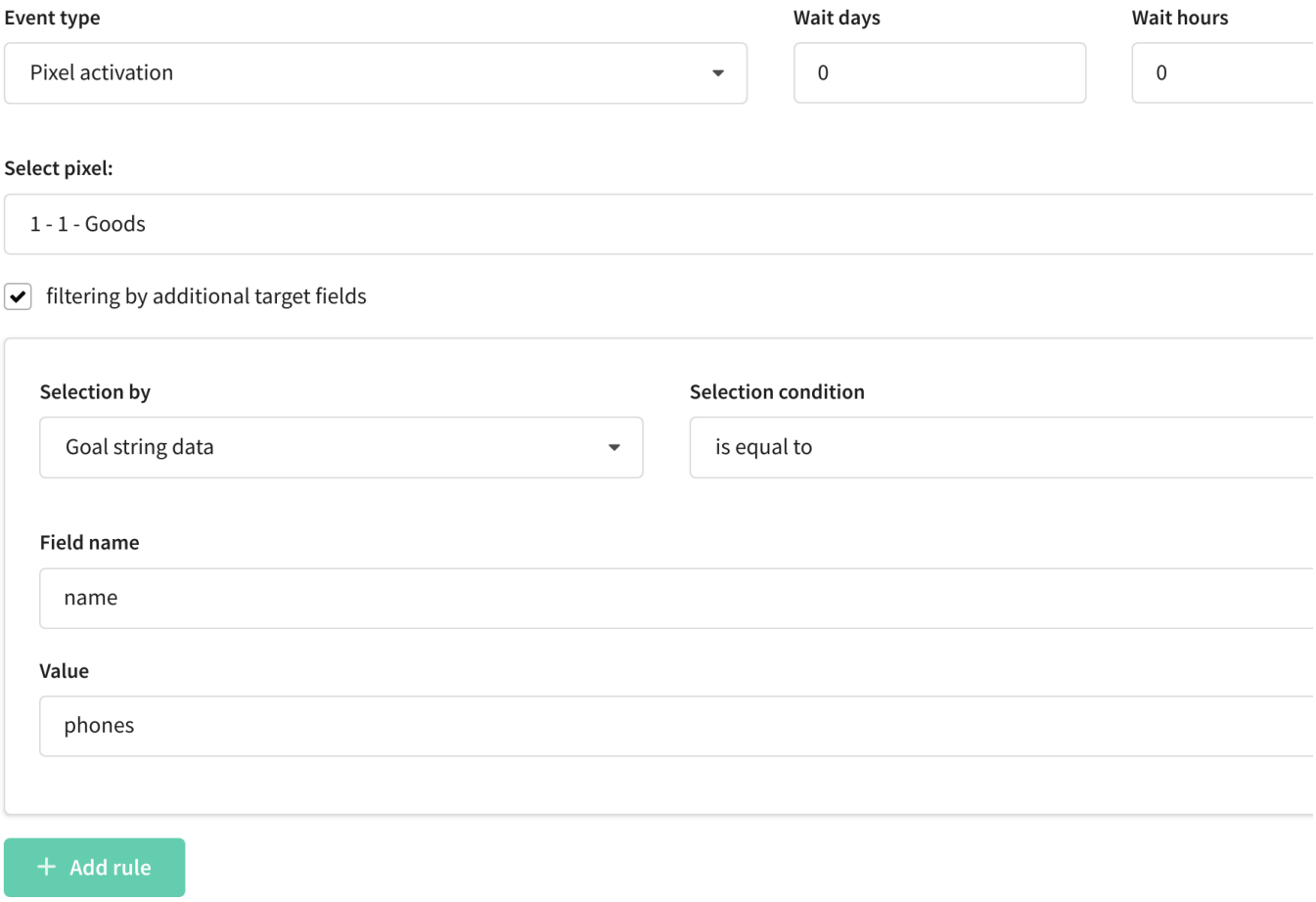
Detailed import statistics are available for each synchronization:



Start	End	Products				Categories		Status
		Total	Added	Updated	Skipped	Added	Updated	
07-06-2024 18:58	07-06-2024 18:58	100	1	99	-	-	3	completed

Filtering by additional fields for pixel activation trigger

Added the ability to specify selection rules for additional fields when setting up a trigger by activating precisely customize the campaign for a certain group of clients or specific products:



Minor changes

- **New connection types have been added: Basic Authentication and Token Authentication.**
- **ClickHouse system driver package and connector have been updated.**

Interface improvements and fixes

- **Fixed a bug in the tooltip in the profile list.**
- **Scenario exit item ID is now displayed in the profile card.**
- **Fixed role management for master users.**
- **Pixel reference is now correctly displayed in the profile history.**
- **Market when importing products now displays the correct number of products.**
- **Forms now work correctly in preview mode.**
- **The image insertion panel in the template editor now displays the correct URL.**
- **Information about the creator and the date of the last change in integrations and connections is di**
- **Outdated Firebase API has been removed from the resource settings for push notifications.**
- **List by metrics in popup analytics is now displayed correctly.**
- **Improved display of some data in the Analytics section.**
- **Improved localization.**
- **Fixed typos and incorrect links.**

Performance improvements

- **Fixed some issues in AKMTA.**
- **Redesigned mechanism of getting the name of global function `acpopup`.**
- **Fixed high memory consumption in `akmtad` and `procworkflow` processes.**
- **Fixed hang when sending email messages with some sending policies.**

Fixes

- **Fixed bug where the “Import via popup” trigger condition could not be selected.**
- **Profiles are now correctly included in the statistical segment by complex condition again.**
- **Fixed the error “Not calculating statistics with errors” when receiving a report in the Administrato**
- **Traffic analytics with the “By popups” filter is now displayed correctly.**
- **Sorting by age in Databases now works correctly.**
- **Fixed a bug that occurred when adding 18 or more characters to an additional field with the data t in Market entities.**
- **Tracking now does not pause when the encryption key is too long.**
- **Link addition events are now correctly displayed in the profile history.**
- **Additional parameters are now correctly added to `custom_data` in Action hooks.**
- **Various segmentation issues have been fixed.**
- **Statistics in scenarios now work correctly.**
- **Added handling of query parameters when loading a script to control popups.**
- **Fixed an issue where editing campaigns with send policies was only available to master users.**
- **Added validation to save templates for sending via WhatsApp*.**
- **Exporting profile history by pooches now works correctly.**
- **Fixed bugs in the Market module.**
- **Added validation for deleting external databases.**
- **Fixed a bug where it was impossible to create a segment via API.**
- **Sending policies in the trigger campaign work correctly again.**
- **Improved the operation of Observer Databases when sending a campaign.**
- **Fixed a bug that occurs when copying a deleted order or creating an order with the same name as**
- **Added ttl for links in SMS messages.**
- **Fixed bug where LDAP users change authorization type after saving.**
- **2FA connection for LDAP-user works correctly again.**
- **An empty string can now be added as a delimiter in the `format` function in templates.**
- **Save and Apply buttons are no longer available for frozen accounts.**
- **Fixed a bug where tab settings were reset while changing mandatory fields in profiles.**
- **Connection to Devino Telecom using login and password works correctly.**
- **Fixed some product search errors.**
- **Pictures with names less than 12 characters are now loaded correctly when importing HTML temp**
- **Fixed errors occurring when updating a trigger campaign.**
- **Stop list now skips domains that contain more than 6 characters after a period.**
- **Fixed panic when trying to save a deleted object in segments.**
- **Fixed panic when adding a template using API.**
- **Fixed panics in `cookie_saver`, `procintegras`, `xls.go`, `proctask` and `tariffcontroller`.**

References

PL-16064 PL-16026 PL-15957 PL-15900 PL-15899 PL-15850 PL-15809 PL-15421 PL-15414 PL-15402 PL-1 PL-15322 PL-14665 PL-13927 PL-13297 PL-13196

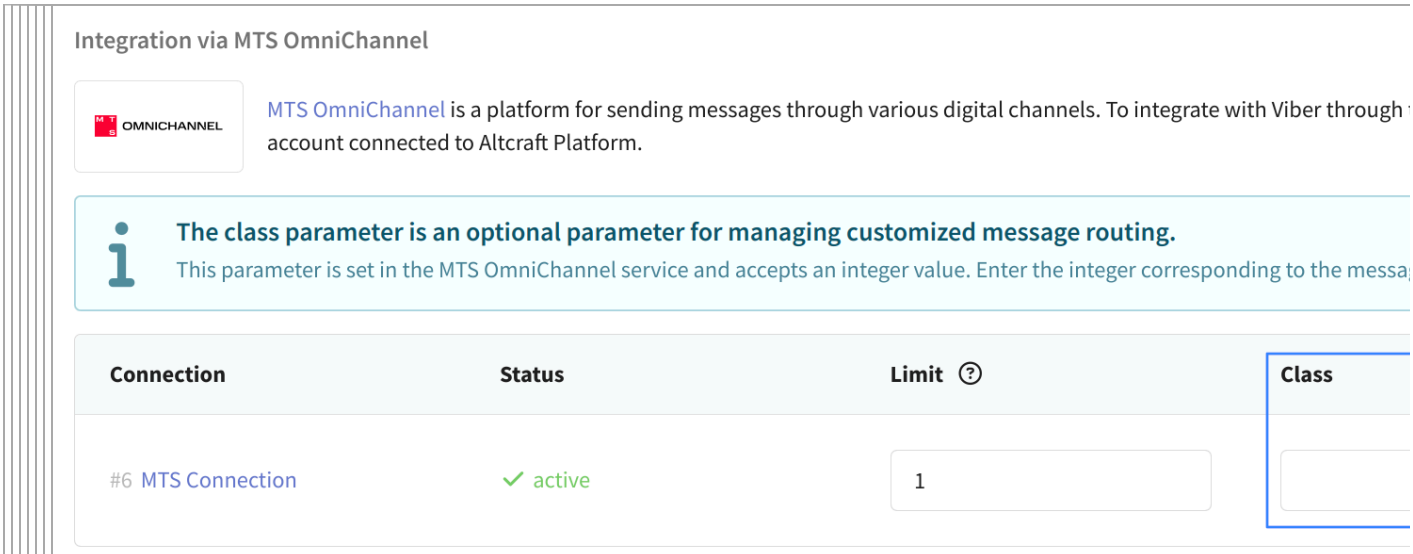
---

v2024.2.68.2.749 - Viber integration updates

Major changes

Improvement of Viber channel integration through MTS OmniChannel

In the channel integration settings using MTS OmniChannel gateway, the `class` parameter has been ad routing in the service:



Fixes

- **Adjusted texts and links in the platform interface, including in the mobile version.**
- **Improved html parsing when importing a message template.**
- **Fixed an error when it was impossible to delete email from a profile.**
- **Improved platform stability.**
- **Fixed Java Script errors in the “Segments” and “Integrations” sections.**
- **Fixed an error that occurred when opening the default template preview.**
- **Test form submission now correctly saves the result of filling out the form.**
- **Fixed a bug where MongoDB was generating multiple queries when saving scenarios.**
- **Deleting profiles by segment is now faster.**
- **Fixed errors that occurred when creating a resource.**
- **Filter by connection status in the connection list now works correctly.**
- **Fixed status of new connections to Yandex.Audience.**
- **AKPush now does not create duplicate profiles when clearing storage and cookies.**
- **Fixed errors when adding groups to the Database in bulk.**
- **Template search in the “Click Map” report now works correctly.**
- **Custom unsubscribe link is now displayed in the “Click Map” report.**
- **Fixed an issue where it was not possible to remove region and category in the Products section.**
- **AKMTA queues in RabbitMQ are now cleared after deleting a Sender in the Administrative panel.**
- **The message counter in the AKTMA queue now displays the correct values.**
- **Fixed some sending errors in AKMTA.**
- **Regular expressions in profile structure segmentation queries now work correctly.**
- **MongoDB driver has been updated to the current version.**
- **API query summary report now returns actual parameters.**
- **Improved control of privileges and user rights for some actions (e.g. exporting tasks).**
- **Fixed errors with HTTPS api in some libraries.**
- **Channel unsubscription is now correctly displayed in the profile’s action history when it enters the**
- **Fixed a bug where a goal not registered before could not be specified in trigger campaigns.**
- **Forms now correctly save data after forced emailing.**
- **Fixed panics in akmtad.**

References

PL-15737 PL-16402 PL-16405 PL-16407 PL-16174 PL-8775 PL-16439 PL-16467 PL-16461 PL-16058 PL-15 PL-16021 PL-16445 PL-16394 PL-16420 PL-16423 PL-5732 PL-16401 PL-16121 PL-16175 PL-16278 PL-15 PL-13916 PL-7600 PL-15192 PL-15466 PL-16395 PL-16187 PL-16409 PL-14836 PL-13393 PL-16269 PL-16

---

v2024.2.68.2.950 - Improved bot filtering mechanism

Major changes

Bot filtration improvment

The mechanism for filtering bot actions in the platform has been improved. The filtering settings are co
`bots_ip.json` **and** `bots_ua.json`**. With their help you can set actions that will be applied to detected bot ac**
how these files work in the article.

Minor changes

- **It is now possible to customize in the platoform configuration the timeout for connecting to the dat**
- **Improved performance of some platform processes.**

Fixes

- **Fixed an issue where the “API Request” node passed incorrect request parameter data to the platf**

CAUTION

To fix this error, the user needs to migrate. To do this, do the following before launching the new versi

1. **Stop** `webcontrol` **and** `procworkflow` **services**
2. **Start** `aktool` **from new build with the command:**

```
./aktool workflow api_nodes_convert --config /root/MarketingPlatform/config/main.json
```

3. **Start** `webcontrol` **and** `procworkflow` **services**

Help for the migrator:

```
Usage:
  aktool workflow api_nodes_convert --accs=1,2... --wfs=1,2... [flags]

Examples:
./aktool workflow api_nodes_convert --accs=1,2 --wfs=1,2

Flags:
  -a, --accs ints   Account IDs (comma-separated list, empty means all)
  -h, --help        help for api_nodes_convert
  -w, --wfs ints    Workflow IDs (comma-separated list, empty means all)

Global Flags:
      --config string                   config path (default "/opt/akd/config/main.json")
  -o, --output string                   file path to output in the result of the command (default "stdout")
```

- **Fixed a bug where a space character was incorrectly encoded by the platform in the “API Request**
- **AMP-version of the email template is now formed correctly.**
- **Fixed a bug where the scenario would not include profiles by activating pixels.**
- **Fixed a bug where the** `procpush` **and** `tariffcontroller` **processes would not terminate without a force**
- **Now data about previous popup form fills is not deleted after renaming fields of this form.**
- **Fixed errors related to incorrect deletion of entities from profiles.**
- **External variables of email templates are now correctly converted in the platform when importing**
- **Fixed a bug where location detection when importing profiles did not work correctly in some case**
- **Improved display of some interface elements in the mobile version.**
- **Corrected description of some parameters in the platform.**
- **Fixed some bugs in the platform interface.**
- **Fixed panic in** `tzdata`**.**

References

PL-16513 PL-16476 PL-16441 PL-15870 PL-16283 PL-16134 PL-16351 PL-16499 PL-16490 PL-15447 PL-1
PL-15646 PL-14839 PL-16186 PL-16133 PL-16198 PL-16243 PL-16160 PL-16028

---

v2024.2.68.2.1131 - SSL support for MongoDB

Minor changes

- **Added SSL support for connecting to MongoDB.**
- **Added “Domain not listed” segmentation condition for the “Email subscription” field.**

Interface improvements and fixes

- **Profile card now shows the id of the element from which the profile has exited the scenario.**
- **Improved design of the form editor interface in the platform.**
- **Fixed errors in the text of some tooltips.**

Fixes

- **Fixed a bug where profiles were not included in scenarios with a scheduled segment under the “Dy**
condition.

- **Information about occurring errors is now correctly displayed above scenario elements.**
- **Fixed a bug where profiles were not captured in scenarios configured to capture when a pixel is ac**
- **Fixed a bug in the `products` query that occurred when passing an identifier containing single quotes**
- **It is now possible to pass an api variable for a field with type “Number” in the “Update Profile” scri**
- **Fixed “invalid syntax” error in GET-request to get action history of several profiles.**
- **Fixed some vulnerabilities in the interface**
- **Now when saving a profile with invalid data in one of the fields, the platform will display an error a**
- **Fixed a bug where a tag with a comma in its name was defined as two different tags.**
- **Updated some libraries to avoid potential vulnerabilities.**
- **Fixed panics in `proctask`, `webcontrol` and `procworkflow`.**

References

PL-16567 PL-16563 PL-16513 PL-16557 PL-16276 PL-14369 PL-16535 PL-16526 PL-16105 PL-16544 PL-1
PL-16496 PL-16103 PL-16373 PL-15154 PL-16471 PL-16449 PL-16307 PL-16236 PL-16137 PL-16487 PL-1

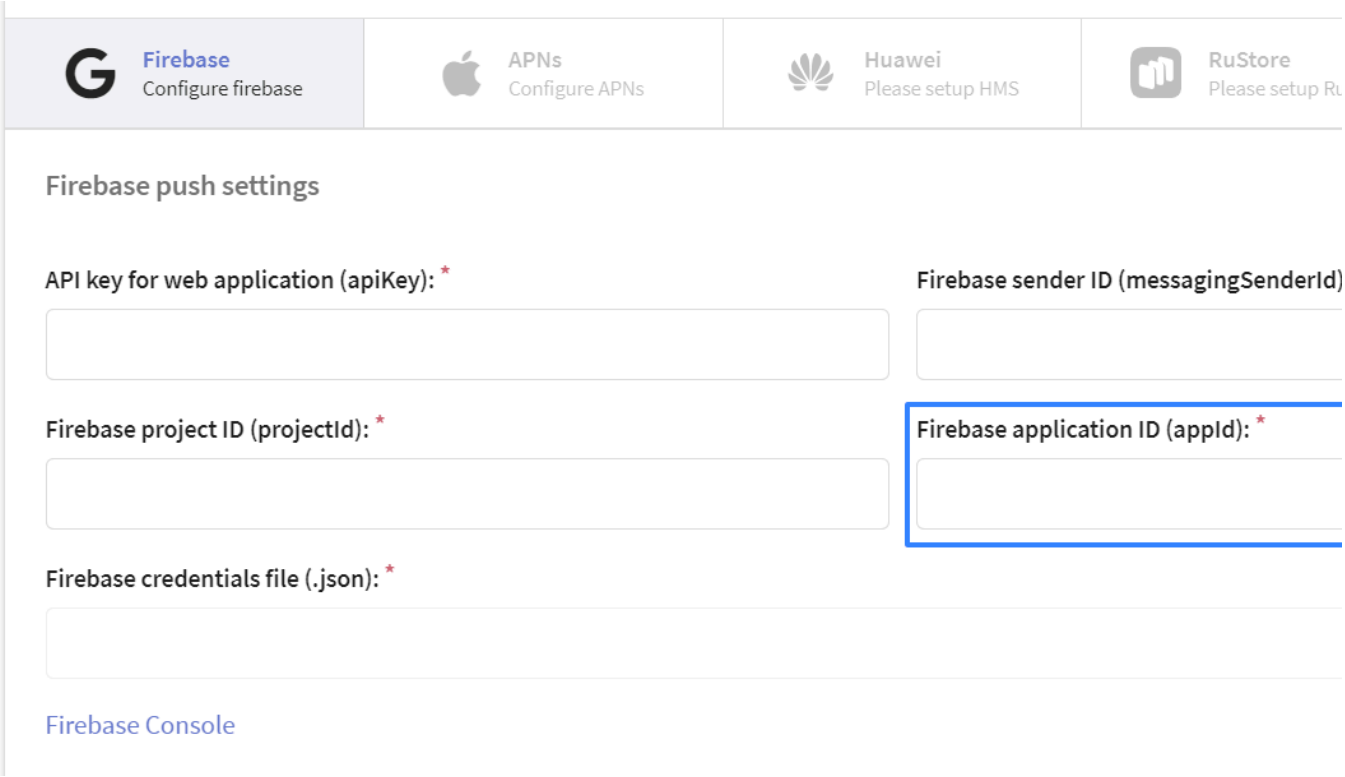
---

v2024.2.68.2.1378 - Firebase API updates

Restore the work of web push-notifications

Google Firebase service has changed some API handlers, in this connection for normal work of web pu
carry out migration:

- **In all resources, scripts of which are hosted on sites, it is necessary to fill in the Application ID fie**



The Application ID value can be found in the project settings in the Firebase console.

- **If static ak-push.js and service-worker.js are placed on the sites, then after editing and saving the these scripts to the site.**

Minor changes

- **Images in the template editor are now inserted with `https` at the beginning of the link, rather than**
- **Links between a sceanario start element and other elements are now added correctly.**
- **Fixed a bug where it was not possible to generate a Promo Code without a quantity limit.**
- **The maximum number of characters for a first-level domain in the platform has been increased to 1**
- **Fixed errors that occurred when interacting with an empty profile database.**
- **Fixed an error when data was not sent to email after a user fills out a form.**
- **It is now possible to specify a tracking domain with a single-letter subdomain in the resource setti**
- **Fixed a bug where scripts with scheduled segment did not work correctly after the first run.**
- **Faster page loading in the user interface when using LDAP authorization.**
- **Fixed a bug where changes made to the Funnel were not saved in the platform.**
- **Improved parameter validation in some API requests.**

- **Drop-down lists work correctly in the latest versions of Google Chrome.**
- **Traffic analytics report is now uploaded correctly.**
- **Fixed interface errors in some sections of the platform.**
- **The descriptions of some errors in the platform have been corrected.**

References

PL-16623 PL-16613 PL-16148 PL-16599 PL-16604 PL-7192 PL-16555 PL-16571 PL-16334 PL-16511 PL-16 PL-10157 PL-11465 PL-16556 PL-16447 PL-15479 PL-16513 PL-16499 PL-15655 PL-13207 PL-16165 PL-1

---

v2024.2.68.2.1509 - Firebase push fixes

Major changes

- **Fixed errors occurring when sending push notifications using Google Firebase.**

Minor changes

- **Added option to disable deleting link data from ClickHouse.**
- **Now the value of “Lock double trigger activation” in pixel settings cannot be lower than one.**

Interface improvements

- **Fixed display of filters on mobile version.**
- **Improved appearance of “Upload Photo” button in profiles.**
- **Improved appearance of the line break button in the template editor.**
- **Fixed layout errors in some Market elements.**

Fixes

- **Fixed a bug where links in push notifications did not work.**
- **Database IDs are now displayed correctly when launching test campaigns.**
- **Fixed some vulnerabilities in the platform.**
- **The “Capture profiles when changing an order” scenario element now works correctly.**
- **Fixed a bug where uploading 0-day history would result in uploading 1-day history.**
- **Market variables now work correctly in the push and Viber template editor.**
- **Fixed an error occurring when trying to send rich push notifications to profiles that don’t have field**
- **The `campaigns/triggers/import_and_start/` method now works with the unconditional send flag.**
- **Fixed panic in webcontrol.**

References

PL-16637 PL-16634 PL-15610 PL-16179 PL-16146 PL-16625 PL-16620 PL-16150 PL-16619 PL-14996 PL-1 PL-16586 PL-16135 PL-16483 PL-16473 PL-16030 PL-14749

---

v2024.2.68.2.1788 - Order status API

Major changes

- **Added API method to check order status in the Market. Read more.**

Minor changes

Interface improvements

- **Fixed layout errors in “Forms” section on mobile devices.**
- **Drop-down lists in scenario modal windows close correctly when focus is reset.**
- **Fixed scroll sliders not working on some platform pages.**

Fixes

- **Fixed errors related to reaching the maximum limit of active sessions in MongoDB.**
- **Profile exit record from a scenario is no longer overwritten when it is cleared of all profiles.**
- **Fixed errors occurring when running dynamic segment with selection condition on a custom field.**
- **Fixed duplication of web push notifications in browsers.**
- **Relations are now correctly overwritten when editing from profile.**
- **Fixed bug where user creation time was reset when updating its data.**
- **Now when cloning a scenario, error messages are not copied with it.**

- **Fixed a bug where an unavailable resource was displayed in the list without being labeled accordi**
- **Custom `custom_json` in push notification templates now works correctly.**
- **Fixed a bug where the last user who edited a script was incorrectly displayed.**
- **Unsubscription event when notifications are forbidden via Safari is now correctly logged by the pla**
- **Fixed a bug where a remote ISP was available to a node in the Administrative Panel.**
- **Improved stability in `webcontrol`, `akmta` processes.**

References

PL-16629 PL-16658 PL-16650 PL-16646 PL-16593 PL-16425 PL-16446 PL-16566 PL-15758 PL-16056 PL-1 PL-16639 PL-16266 PL-16615 PL-16428 PL-16596 PL-16257 PL-16470

---

v2024.2.68.2.1957 - Web version compression mode

Major changes

- **Added option to disable compression when creating a web version storage policy in the Administra**

- **Campaign log now shows the time a segment was received.**

Minor changes

Interface improvments

- **Added display of the time of the next launch of the Multivariate testing campaign.**
- **Fixed a bug where tooltip pop-ups did not disappear after some time.**

Fixes

- **Fixed a bug where the platform did not display the campaign queue on senders.**
- **Fixed a bug where the platform did not take summer and winter time into account for time zones ir**
- **The campaign log works correctly again when opened from the campaign editor.**
- **Fixed various bugs related to updating a profile in the platform.**
- **Profiles with Apple Safari Native push notification subscriptions are now saved correctly.**
- **Fixed a bug where you could create a campaign with a draft template.**
- **Web versions of emails are correctly compressed before saving.**
- **Importing to Market using a file now works correctly.**
- **Fixed a bug where the result cache in the preview of template/segmentation SQL queries was not l**

References

PL-16095 PL-16671 PL-16655 PL-16281 PL-10310 PL-16553 PL-16609 PL-11837 PL-16132 PL-15971 PL-1 PL-16617 PL-16633 PL-8356 PL-16274 PL-16292 PL-16221 PL-13312

---

v2024.2.68.2.2206 - Sendsay, Fixes

Major changes

Sendsay integration

We've added the ability to send emails through the Sendsay SMTP relay. The platform administrator ne AKMTA sender. Delivery statuses are tracked using webhooks. Learn more.

Minor changes

- **Expanded API requests for retrieving profile history: `/api/v1.1/subscribers/history_get` and `/api/v1.1/sub` response now includes additional fields that help identify unique events and obtain more detailed i**
- **Removed the `tariffcontroller` and `procnotify` processes; their functionality was migrated to `proctask`**
- **Improved import of email templates uploaded from third-party services into the platform. Images c as the HTML file or in a folder with one of the following names: `images`, `image`, `img`, `imgs`, `pictures`, email template should not include the files `text.txt`, `subject.txt`, and `template.json`, as their presenc same time, the previous structure of the archive with the template continues to work unchanged.**
- **The "Web imports" metric in the Audience growth report now includes the total number of profiles forms and pop-ups on the website.**

> CAUTION
>
> **Previously, the "Web imports" metric only included data from form imports. To update the metric to in you'll need to perform a migration.**
>
> **Migration is performed after the platform is updated to the new version and does not require stopping**
>
> **1. Drop the `engagement_report` view and table:**
>
> ```
> ./aktool stats delete engagement --config=/opt/akd/config/main.json
> ```
>
> **2. Initialize the table in the new format:**
>
> ```
> ./aktool stats init engagement --config=/opt/akd/config/main.json
> ```
>
> **3. Recalculate the records in the table:**
>
> ```
> ./aktool stats fill engagement --config=/opt/akd/config/main.json
> ```
>
> **If successful, the console will show how long it took to fill the table. Example: `complete fill stats table`**

Bug fixes

- **Fixed the issue with sending web push notifications to the Safari browser via APNS.**
- **Restored the ability to select a pool in sender settings.**
- **Fixed a problem with adding extra parameters to `custom_data` in hooks.**
- **Resolved the issue that caused hooks to not retry when the `_createtime` field was added to `custom_da`**
- **Segment filtering by "List" type fields using the "Equals" condition now works correctly**
- **The counter above the SMS campaign node in the scenario now correctly displays the number of p**
- **Resolved the issue of duplicated weekly and daily reports for platform users.**
- **The campaign log now displays the correct time for receiving a segment during regular campaign t**
- **A trigger campaign set up for a custom popup event can now be successfully saved.**
- **The next_from_id parameter in the `/api/v1.1/subscribers/history_get_al` l response works correctly.**
- **Missing categories field in the /api/v1.1/market/sku/import request no longer causes errors.**
- **The `ampcarousel` component is now connected in the demo AMP template version.**
- **Improved validation in scenarios and subscriptions for Viber and WhatsApp* channels.**
- **Stability improvements for the `procactions` process.**
- **Minor UI fixes in the platform.**

References

PL-15845 PL-16842 PL-15693 PL-16694 PL-16796 PL-16763 PL-16805 PL-15149 PL-16743 PL-16737 PL-1 PL-16685 PL-16704 PL-11654 PL-15426 PL-16122 PL-16599 PL-16663 PL-16069 PL-16631 PL-16656 PL-1 PL-16709 PL-16614 PL-16116 PL-16533 PL-16644 PL-16676 PL-14390 PL-16476 PL-16259 PL-16608 PL-1

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a banned in the Russian Federation.

Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

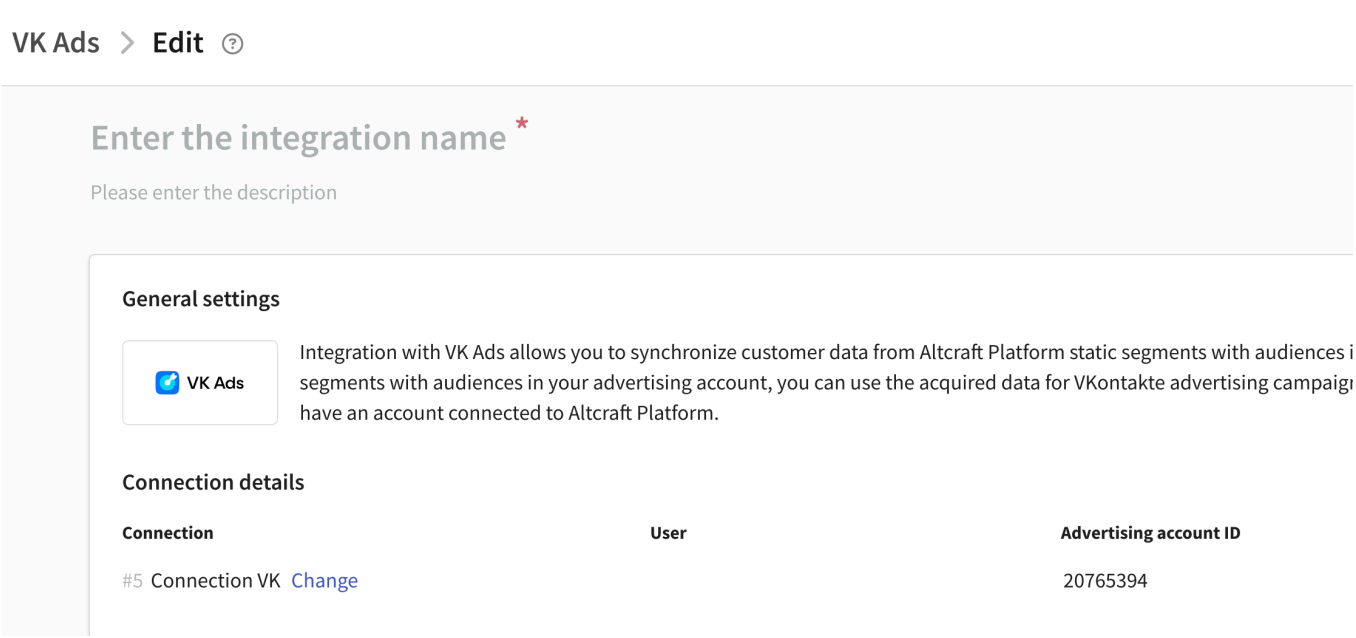
v2024.1.68

v2024.1.68.0 - VK Ads™, new AKMTA sender settings

Major changes

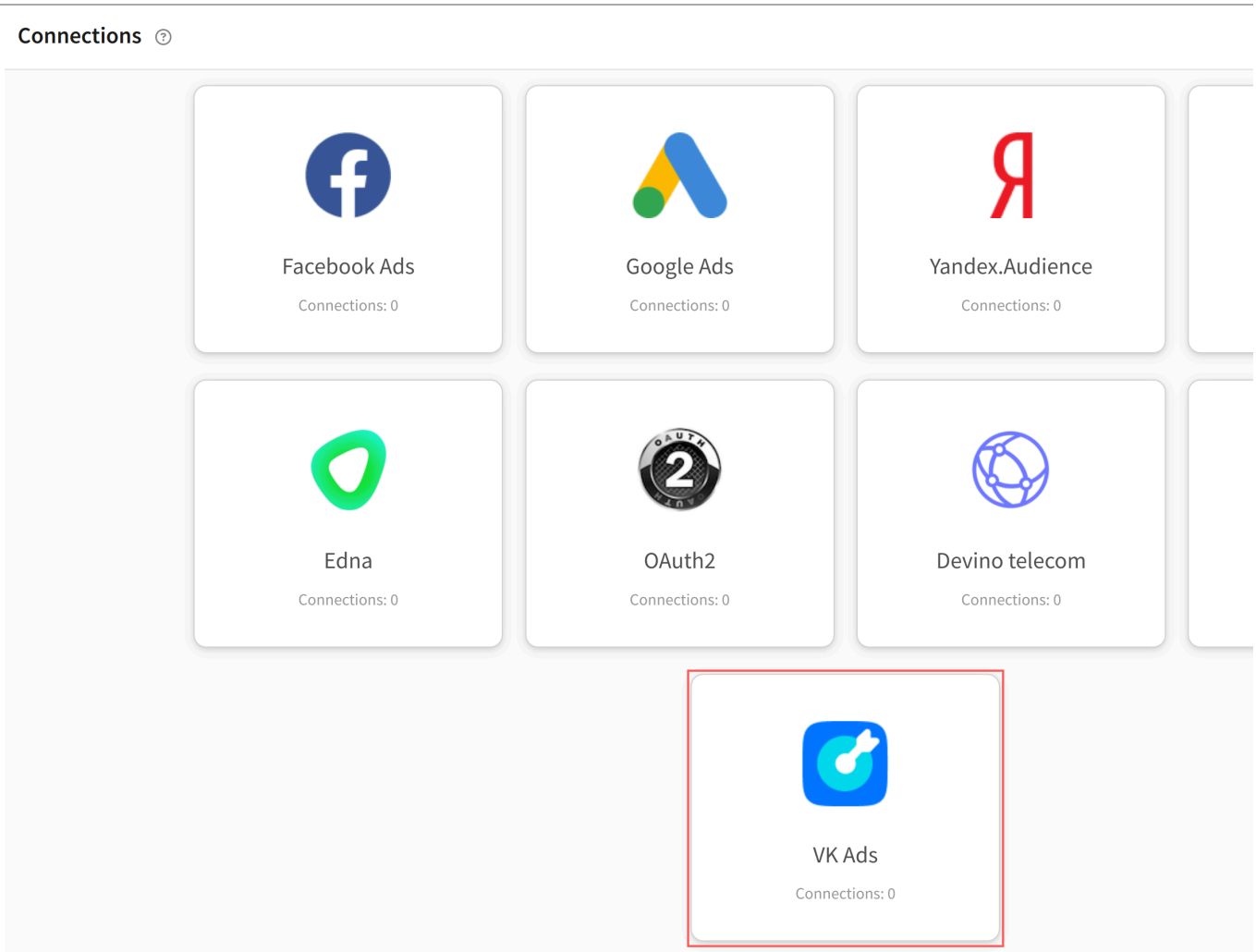
VK Ads™

	Added integration with **VK Ads™. The new feature allows you to transfer data segments from the Altcraft Pla connection](/en/user-guide/settings/connections/vk-connection/) and [configuring integrations](/en/user-guide/inte**



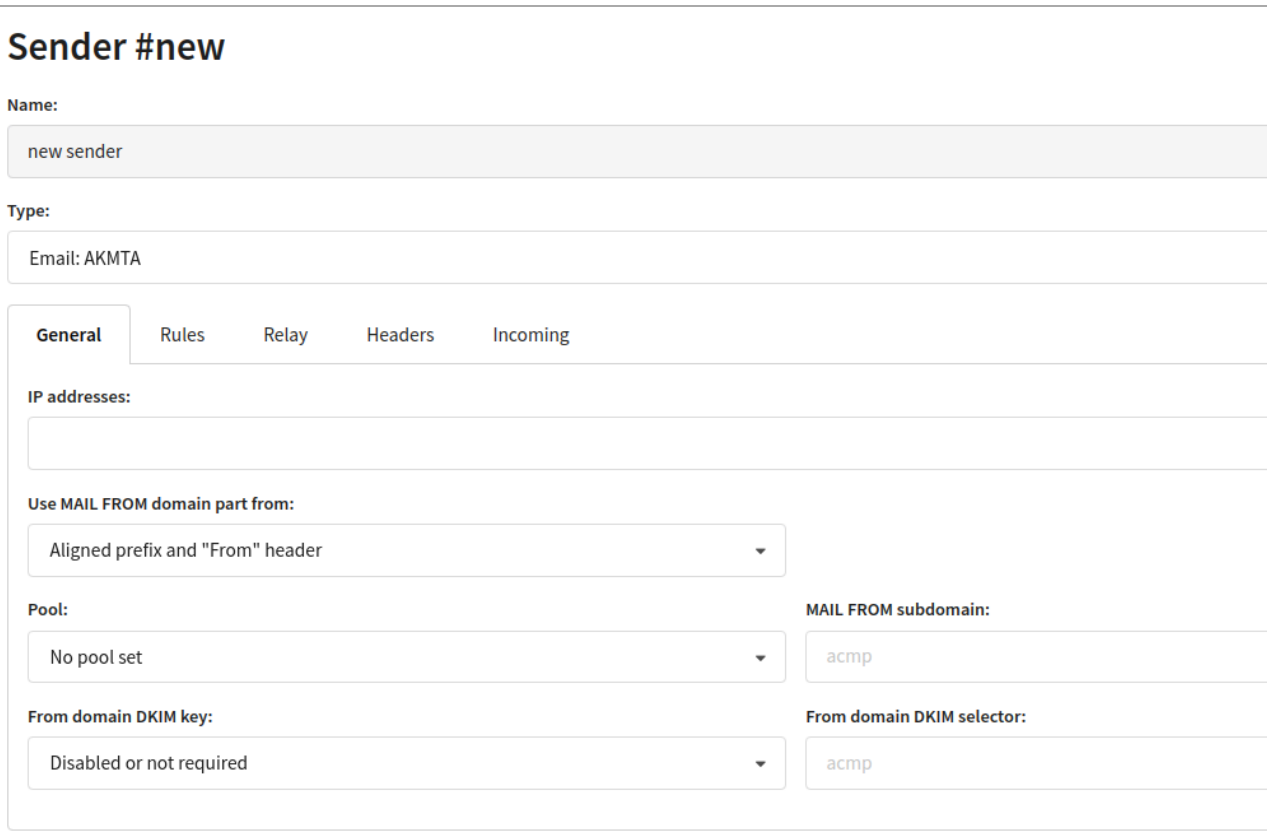
Updated connections menu in the dashboard

Removed Viber and WhatsApp* menu items, added separate buttons for creating connections using edi SMSTraffic.



Updates of the administrative panel

- **When creating an AKMTA sender, a new type was added for the MAIL FROM domain part - Aligned p setting allows you to simultaneously track statistics and perform SPF alignment. To use it, you nee**
- **Added pools. A pool is a set of resources needed to send emails from a specific domain. Detailed in it up can be read here.**
- **AKMTA settings in the “Senders Management” section are divided into General, Rules, Relay, Head**



Minor changes

- **Improved search in the user list: added search by email and phone number, fixed various errors ass**
- **The user's last name is no longer visible to support specialists in Jivo.**
- **Removed the limit on the number of displayed columns in data tables.**
- **The ability to view and copy a column value has been added in cases where it does not fit complet**

Performance improvements

- **Optimized API request for bulk import and launch of profiles into the script**
- **Fixed extremely high memory consumption for akmtad**
- **Fixed freezes in the visual editor when clicking on elements**

Interface updates and fixes

- **Fixed display of unavailable actions for frozen accounts.**
- **Updated interface in the "Pixels" and "Click map" modules.**
- **The "% Opens" column is now available again in channel reports.**
- **Fixed errors with displaying the menu of additional fields.**
- **Now the connection creation/update time is displayed correctly.**
- **The scrollbar in pop-up menus is now displayed correctly.**
- **The audit log report is now uploaded correctly.**
- **Some names and translations have been corrected.**

Corrections

- **The non-delivery report now correctly displays the reason of failure.**
- **Fixed a bug where variables were not inserted into the pop-up editor field.**
- **When mass profiles updating, push subscriptions are added only to the specified provider.**
- **Fixed various errors in the "Loyalty" section.**
- **Fixed a situation where it was impossible to add text to an empty "Text" element in the visual edit**
- **Fixed errors that occurred when deleting a database and associated segments.**
- **Improved logging.**
- **Saving a target in trigger campaigns now works correctly.**
- **Special characters are now correctly escaped in LDAP.**
- **Fixed a bug where sending more than 10 SMS on the MTS gateway did not work.**

- **Errors related to incorrect filling of date and time in various elements have been fixed.**
- **Fixed errors related to analytics, such as false clicks from bots, counting hardbounce responses, a**
- **An error message about the number of characters in the SMS has been exceeded has been added t**
- **Now you can select the status “Unconfirmed” in trigger campaigns with the event type “When addi**
- **Fixed repeated “context canceled” errors in procworkflow logs.**
- **Entity deletion is now correctly recorded as an update.**
- **Fixed panics in procintegras, webcontrol, proctask and regen_preview_msgs.**
- **Fixed panics when publishing a pop-up and when setting up 360dialog.**

References

PL-15764 PL-15754 PL-15741 PL-15735 PL-15731 PL-15730 PL-15727 PL-15725 PL-15711 PL-15703 PL-1 PL-15650 PL-15644 PL-15634 PL-15633 PL-15630 PL-15612 PL-15608 PL-15604 PL-15601 PL-15596 PL-1 PL-15553 PL-15509 PL-15333 PL-15286 PL-15016 PL-14793 PL-14439 PL-14291 PL-13485 PL-11892 PL-1 PL-15056 PL-1500 PL-14992 PL-14950 PL-14900 PL-14718 PL-14547 PL-13257 PL-13012 PL-12585 PL-12 PL-15633 PL-15630 PL-15612 PL-15604 PL-15574 PL-15509 PL-11972

---

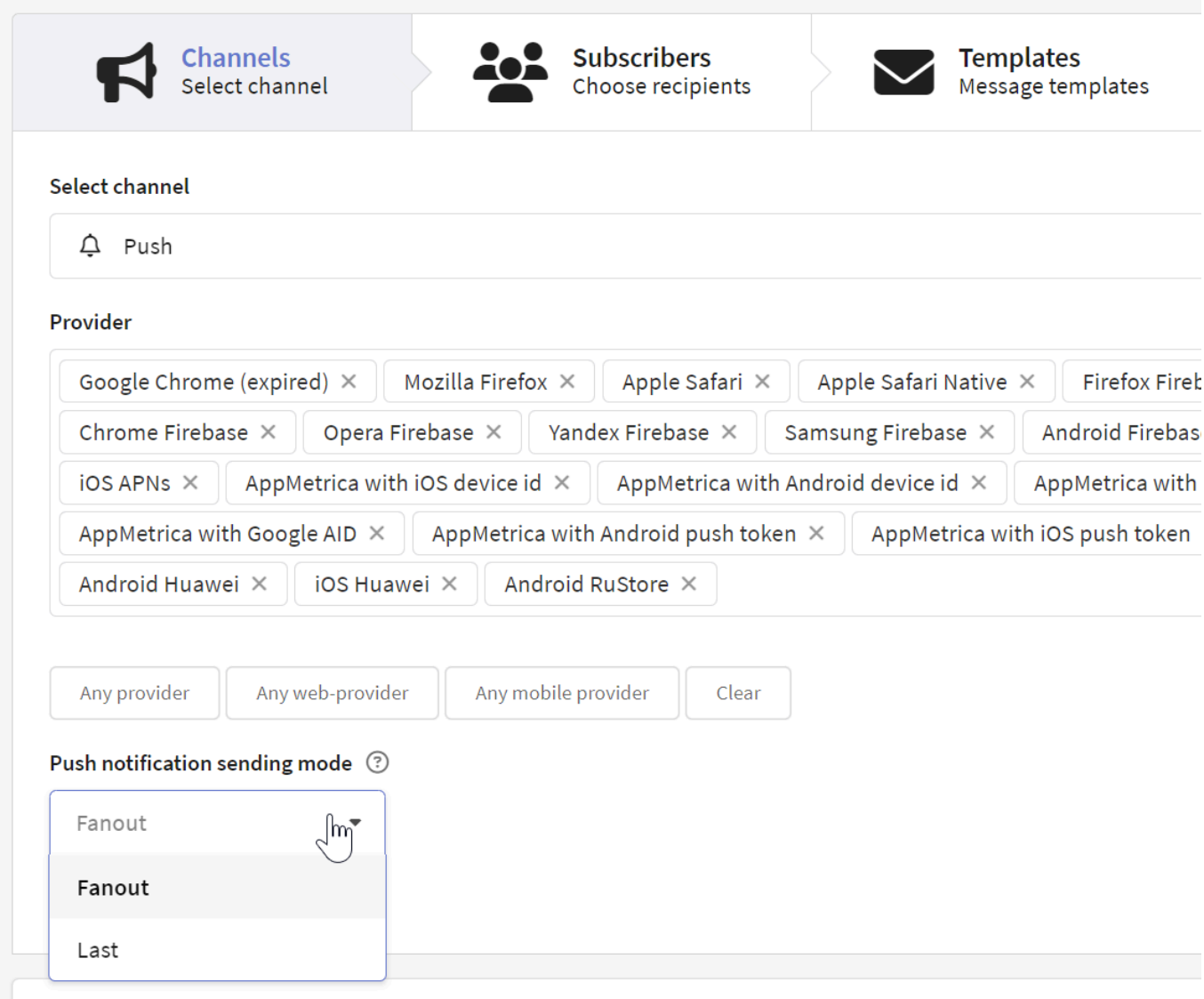
v2024.1.68.626 - New push sending mode, Fixes

Major changes

New mode for sending push notifications

When setting up campaigns using the push channel, you can now choose the sending mode for push m

- **fanout (default) — osends messages to all subscriptions of profiles that match the campaign’s filte**
- **last — sends the message to the subscription with the most recent creation date.**



Changes in the statistics table

Since the summary report was returning an incorrect number of events, the structure of the `channel_rep` must recreate the table when updating:

```
./aktool stats delete channel_report
./aktool stats init channel_report
./aktool stats fill channel_report
```

> **CAUTION**
>
> **You must stop the `procevent` service before starting the migration.**

Minor changes

- **Improved validation of links in push templates. An error “Number of invalid links in the message” is**
- **In the campaign log, the entry “Number of processed subscribers” has been renamed to “Number o**
- **Added the ability to save master users without specifying roles.**
- **Logging for relay sending has been added to syslog.**
- **Improved validation of fields in the Telegram Bot channel template.**
- **Updated the logic of the `./aktool stats` command, used for working with the report storage. Here’s `./aktool stats fill table_name`. Additionally, `channel_report` and `traffic_report` can now be deleted sepa**

Bug fixes

- **Fixed the operation of the segment by goals with a complex condition.**
- **In the scenarios, JSON filter now supports the OR operator and arithmetic operations.**
- **In the campaign settings, when deleting a segment used for sending, the correct segment is now d**
- **Fixed bugs in searching in resources, roles, and the market.**
- **In test SMS messages, the date is inserted in the specified format.**
- **In the quick segment, when selecting profiles by phone number, spaces between digits do not affe**
- **Images are now correctly displayed in both desktop and mobile versions in the visual image editor.**
- **The footer of the email with the unsubscribe link is displayed in the visual editor.**
- **Links in push templates after adding UTM tags redirect the user to the right page.**
- **It is now impossible to save an email template with an attachment larger than 12 MB.**
- **In forms, the option “Hide submit button” now works correctly and does not prevent form submissi the mandatory “One option” element.**
- **Export of the profile history of a static segment created during import now exports data without e**
- **In scenarios, the “Condition” element with the “Wait for condition for a specified time” option enab**
- **For one push message, the same `send_message_id` is now returned for all event types in event capture**
- **Fixed an error when deleting a profile via the API.**
- **Fixed a situation where a push subscription could be created via API without specifying `subscriptio`**
- **Fixed the 2FA mechanism for administrators.**
- **Exporting audit logs in the admin panel no longer causes errors.**
- **Changes in sender settings with the type Rapporto SMS are now successfully saved.**
- **Fixed a situation where AKMTA logs contained duplicate rejection responses.**
- **Performance of some platform processes has been optimized.**
- **Some interface flaws have been fixed.**

References

PL-15986 PL-15983 PL-15964 PL-15959 PL-15952 PL-15886 PL-15876 PL-15849 PL-15828 PL-15823 PL-1 PL-15739 PL-15719 PL-15715 PL-15684 PL-15658 PL-15649 PL-15609 PL-15577 PL-15486 PL-15432 PL-1 PL-10471 PL-10464 PL-10450 PL-10097 PL-9688 PL-7740 PL-15918 PL-15917 PL-15914 PL-15912 PL-159 15891 PL-15885 PL-15882 PL-15881 PL-15871 PL-15863 PL-15862 PL-15858 PL-15839 PL-15816 PL-1580 15786 PL-15785 PL-15781 PL-15778 PL-15770 PL-15746 PL-15745 PL-15744 PL-15740 PL-15738 PL-1573 15669 PL-15642 PL-15632 PL-15622 PL-15606 PL-15594 PL-15592 PL-15551 PL-15460 PL-15098 PL-1506 13304 PL-13120 PL-12999 PL-11158 PL-10533

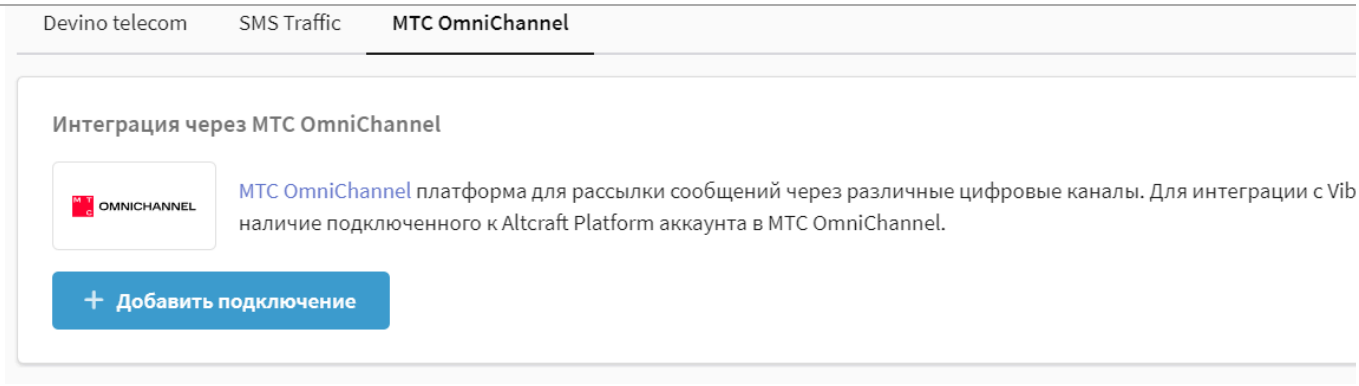
---

v2024.1.68.1354 - MTS OmniChannel integration, bug fixes

Major changes

Integration with MTS OmniChannel

Added support for integration with MTS OmniChannel. Now you can use this platform as a gateway to s



Before setting up the integration, you need to connect your MTS OmniChannel account with Altcraft Pl
read more about setting up the integration here.

Other changes

- **The "Pixels" section has been moved to the "Web Layer" menu item.**
- **Updated geoip data for April 2024.**
- **Improved platform performance.**
- **Added option to remove validation for hidden fields in forms.**

Fixes

- **Fixed a bug related to duplicate tracking links.**
- **Scenario items with a condition are now saved with the correct id.**
- **Frozen accounts now have view permission restriction again.**
- **Fixed an issue where form validation errors were displayed in error level logs.**
- **Checking allowed domains for redirects when opening a pixel now works correctly.**
- **Improved traffic report performance when using grouping by pixel targets.**
- **Fixed incorrect platform behavior during deleting profile databases when the user role does not ha**
- **Improved operation of critical push mode with APNS.**
- **Validation when using segments with link data collection and manual profile creation now works c**
- **Platform login history now displays correct data.**
- **Fixed errors in logs.**
- **Popups work better on mobile display.**
- **Integrations in the platform are now correctly saved.**
- **Fixed a bug related to creating Facebook* connections.**
- **Fixed some analytics and reporting bugs.**
- **Correct mx is now used for sending to Yahoo mailboxes.**
- **Fixed bugs in Yandex.Appmetrica integration.**
- **Form opening and filling events in profile history are now displayed correctly.**
- **Fixed validation of fields in forms.**
- **Fixed errors when previewing multi-page forms.**
- **`click_url` in the web version of trigger push emails is now saved correctly.**
- **Fixed `runtime error` panic when sending a campaign with hidden copy.**
- **Fixed `form_tst.go` panic.**

References

PL-15888 PL-12880 PL-15993 PL-15991 PL-15865 PL-14353 PL-15869 PL-15926 PL-15840 PL-15979 PL-1
PL-13837 PL-15334 PL-16037 PL-15779 PL- 15939 PL-15662 PL-16032 PL-13906 PL-15800 PL-16112 PL-1
PL-15907 PL-14312 PL-16013 PL-14015 PL-15104 PL-15989 PL-15712 PL-15693 PL-15700

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated a banned in the Russian Federation.
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

API / Developer Guide: Contents

Contents of this section.

API interaction

Matching

Importing the API collection in Postman

List of API endpoints

Import profile

Update profile

Import multiple profiles

Update multiple profiles

Add multiple profiles

Add profile to database

Import profile to RabbitMQ

Get profile data

Uploading profiles to a file

Delete profile

Subscription fields functional update

Database fields functional update

Merging multiple profiles

Unsubscribe profile from resource

Profile splitting

Get data for multiple profiles

Add or edit subscription

Get all profile subscriptions

Get all subscriptions from multiple profiles

Get profile subscription

Delete profile subscription

Restore deleted subscription

Suspend all subscriptions

Unsuspend all suspended subscriptions

Get profile action history

Get multiple profiles action history

Profile relations

Attach relation

Detach relation

Modify relation properties

Overwrite relation properties

Get profile relations info

Get profile relations info

Get database statistics

Update statistics on database

Get database list

Get database information

Get database fields

Database wipe
Get resource statistics
Update statistics on resource
Get resources list
Get resource information
Get resource subscription fields
Create segment
Get statistics on resource
Update statistics on segment
Add or remove profile
Get profile data in a static segment
Update segment
Get segment information
Get segments list
Delete segment
Create suppression list
Update suppression list
Get suppression list info
Get the list of suppression lists
Delete suppression list
Upload suppression list data to file
Suppression list actions
Check if email is suppressed s
Add email to suppression list s
Add multiple emails to suppression list s
Remove email from suppression list s
Remove all emails from suppression list s
Check if domain is suppressed s
Add domain to suppression list s
Add multiple domains to suppression list s
Remove domain from suppression list s
Remove all domains from suppression list s
Check if phone number is suppressed su
Add phone number to suppression list su
Add multiple phones to suppression list su
Remove phone number from suppression list su
Remove all phone numbers from suppression list su
Get templates list
Get template info
Delete template
Add template
Update template
Channel object
Get campaign information

Get campaign list
Activate campaign
Complete campaign
Deactivate campaign
Get campaign status
Activate mailing
Deactivate mailing
Get mailing list
Get mailing information
Get mailing log
Clone mailing
Delete mailing
Get mailing status
Get broadcasts list
Get broadcast information
Create broadcast mailing
Update broadcast mailing
Launch a broadcast mailing
Get regular mailings list
Get regular mailing information
Create regular mailing
Update regular mailing
Launch a regular mailing
Get trigger mailings list
Get trigger mailing information
Create trigger mailing
Update trigger mailing
Trigger launch (API call)
Profile import + trigger mailing launch
Task for bulk trigger launch
Task for bulk profiles import + trigger launch
Bulk trigger launch
Bulk profiles import + trigger mailing launch
Clone a trigger mailing
Data array ca
Engage profile in scenario
Import and engage profile in scenario
Batch import and engage profiles in a scenario
Task for batch import and engaging profiles in the scenario
Get scenarios list
Activate scenario
Deactivate scenario
Get scenario information
Change scenario priority

Get profile tier in a loyalty program
Export points transactions
Expiring points for a period
Get profile account transactions
Get trigger promotions list
Accrue points to a member
Redeem member points
Commit temporary transaction
Preliminary Order Calculation
Order Confirmation
Roll back temporary transaction
Cancel points transaction
Get points account balance
Register member in a loyalty program
Batch adding participants to loyalty program
Task for batch add participants to loyalty program
Remove member from loyalty program
Get form information
Get form list
Export form fill data by user
Export form fill data
Publish form
Unpublish form
Delete form
Import promo codes
Get promo code information
Activate promo code
Update promo code
Attach promo code
Detach promo code
Get all promo codes
Goals and goal values registration
Processing and adding a subscription
Add app push events
Order data object
Product data object mar
SKU data object n
Categories array m
Custom fields array mar
Import order and item statuses
Get orders list
Delete order
Get order status
Update order line status

Import products, SKUs and categories
Import SKUs and categories
Get products list
Get SKUs list
Delete products
Delete SKU
Get summary report
Get soft bounces report
Get undeliveries report
Get senders list
Get virtual senders list
Get virtual sender information
Clone virtual sender
Create virtual sender
Update virtual sender
Delete virtual sender
Add segmentation query
Update segmentation query
Get segmentation query information
Get segmentation queries list
Delete segmentation query
Add template query
Update template query
Get template query information
Get template queries list
Delete template query
AKMTA object
Content object
Email rule object
File object
Profile data object
Sender object
Sender types
SMS rule object
Start schedule object
Subscription object
Trigger types
Upload file
Get message web version
Push providers
Deduplication of requests
How to send API request with RabbitMQ
List of gender identifications
Obtain valid values for fields: browsers, devices, tz, oses, languages

mSDK
Managing JWT and Role Token
Quick Start
SDK Functionality
SDK Configuration
Public SDK API
Firebase Cloud Messaging
Huawei Mobile Services
RuStore
Quick Start
SDK Functionality
SDK Configuration
Public SDK API
Apple Push Notification Service
Firebase Cloud Messaging
Huawei Mobile Services
Quick Start
SDK Configuration
SDK Functionality
Public SDK API
Provider setup
Quick Start
SDK Configuration
SDK Functionality
SDK Public API
Provider Setup

API interaction

> TIP
>
> **You can download the Altcraft API collection for Postman using this link.**

Requests format

Requests are sent by POST method using HTTP/1.1 protocol. The method is also mentioned on each re
Request parameters are placed in the sent structure. The response format is determined only by the A
The input data format must be indicated in the Content-Type HTTP header.
Possible header values:

- **application/json — JSON format**

Symbols must have UTF-8 coding.

```
POST https://example.com/api/v1.1/<method name>
```

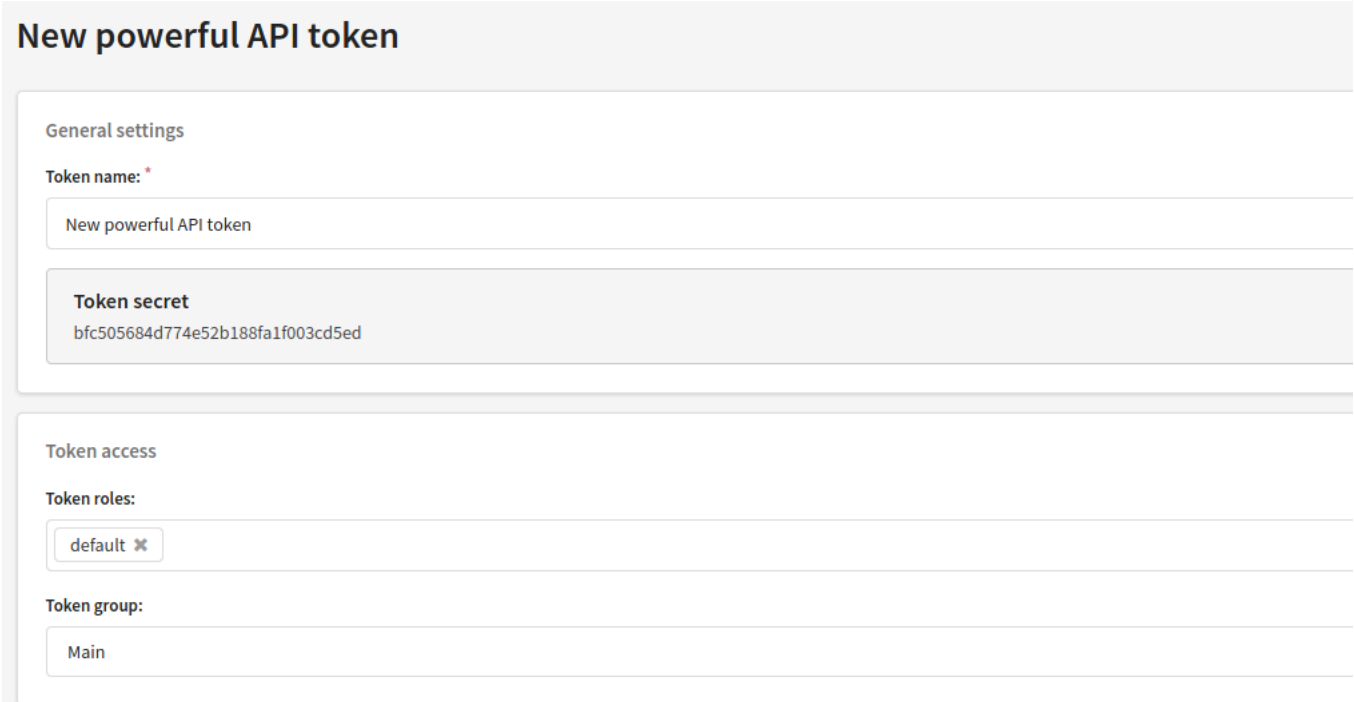
If you use an Altcraft Platform Cloud account, send API requests to the URL `ru.altkraft.com`. In the case hardware (On-premises), use your own URL.
A description of the JSON format can be found in RFC 7159: http://tools.ietf.org/html/rfc7159.
Don't forget to encode characters that cannot be directly written in JSON.

Authorization

For authorization when sending a request, you need to pass a token in the Authorization HTTP header. `<token>`, where instead of `<token>` insert your API token:

```
Authorization: Bearer bfc505684d774e52b188fa1f003cd5ed
```

API token can be created in the user panel, in section "Settings" — "Tokens". Master user rights are ne



API token is automatically generated after saving. You can also select a token name and configure acc objects available for this token.

Response format

The response format is determined by the Accept HTTP header.
Response example (Successful operation):

```
{
        "error": 0,
        "error_text": "Successful operation",
        "profile_id": "5f4fa1a5ce9448665fef548e"
}
```

The following parameters are given in responses:

- `error` **— error code;**
- `error_text` **— error description;**

- `profile_id` — **profile identifier (for successful operation).**

INFO

Code 200 is the success status code in HTTP transport. However, if there is information about an erro additional details about the error are provided in a JSON object in the `error` field with explanations in

Response codes

Code	Description
0	Operation is successful
400	Incorrect request
401	API token required
402	Tariff limit reached
403	No rights for this action
404	The object is not found in the token vision zone
409	Another record with the same unique attributes already exists
413	The database is not available in the selected resource
415	Requested Content-Type is not supported
429	The number of requests set in the configuration file has been exceeded (`API_MAX_REQUESTS_COUNT`)
435	Ambiguous search, there are several objects with given attributes
441	The object belongs to another group
450	Request validation error
500	Internal service error
501	The method does not exist

XML tags attributes

XML queries are built on top of JSON queries, so in some situations it is necessary to write additional a

- `array='true'` **- an attribute indicating that the tag is an array. Used when there is either 1 element ir all.**
- `string='true'` **- attribute is used to indicate that the value used has a string data type. Required whe the word “true”/“false”.**
- `json='true'` **- attribute needed when we need to pass json request inside xml request (for example, requesting data on several profiles).**

Request deduplication

If the connection fails at the moment of receiving the data, a second request may be sent. The platforr request if it modifies the data in order to avoid duplicate events. Read more about repeated requests ir

Matching

The `matching` field specifies the search mode for the profile to update. The default value is `email`.
When you update data, you should follow a general matching approach.

Search mode	matching	Required fields	Type	Example	
By email, from profile or subscription	"email"	email	string	"john@example.com"	Email address
By email from profile	"email_profile"	email	string	"john@example.com"	Email addressc
By email from subscriptions	"email_sub"	email	string	"john@example.com"	Email address
By phone, from profile or subscription	"phone"	phone	string	"+79000000000"	Phone number
By phone from subscriptions	"phone_sub"	phone	string	"+79000000000"	Phone number
By profile ID (only on update!)	"profile_id"	profile_id	string	"abcdefghijklmnqrstuvwxyz"	Profile ID
By push subscription	"push_sub"	provider	string	"android-firebase"	Provider
		subscription_id	string	"1234"	Push subscription ID
By custom field	"custom"	field_name	string	"CRM_ID"	The name of the custom profile field f
		field_value	int/string	"example123"	The value of the custom search field. Search by combination of field values separated by commas: "tag_1, tag_2".
Search for a profile by subscription to the custom channel	"custom_sub"	field_name	string	"CRM_ID"	The name of the custom profile field f
		field_value	int/string	"example123"	The value of the custom search field. Search by combination of field values separated by commas: "tag_1, tag_2".
By email or phone, from profile	"email_phone"	email / phone	string	"john@example.com" / "+79000000000"	Email address / Phone number
By email or phone, from subscriptions	"email_phone_sub"	email / phone	string	"john@example.com" / "+79000000000"	Email address / Phone number

INFO

Unclear matching

If the field you are passing in the request body is not unique, i.e. the value of this field is the same for will return you the error: “Unclear matching”.

INFO

Duplicate unique data

The “Duplicate unique data” error occurs if you use a custom unique field for matching when importin uniqueness of the system field.

For example, a unique custom field `client_id` is used for matching. The database has a profile with fie

- `email: test@example.com`,
- `client_id: 100`.

You import another profile with fields:

- `email: test@example.com`,
- `client_id: 101`.

The request will return a “Duplicate unique data” error, because the uniqueness of the `email` system f
In such cases, it is recommended to pass the `email` field in the `subscription` object:

```
{
    "db_id": 1,
    "matching": "custom",
    "field_name": "client_id",
    "field_value": "100",
    "data": {
            "_fname": "John",
            "_lname": "Doe",
            "subscriptions": [
                {
                    "channel": "email",
                    "email": "test@example.com",
                    "resource_id": 1
                }
            ]
        }
}
```

Examples of importing/updating a subscription with certain matching

email, email_profile, email_sub, email_phone, email_phone_sub

```
{
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "data": {
        "subscriptions": [
            {
                "channel": "email",
                "email": "example@example.com",
                "resource_id": 1
            }
        ]
     }
}
```

phone, phone_sub, email_phone, email_phone_sub

```
{
     "db_id": 1,
     "matching": "phone",
     "phone": "+790000000000",
     "data": {
        "subscriptions": [
                            {
                "channel": "sms",
                "phone": "+790000000000",
                "resource_id": 1
            }
        ]
     }
}
```

profile_id

```
{
     "db_id": 1,
     "matching": "profile_id",
     "profile_id": "012452gf62fcd71fjh8ec0",
     "data": {
        "subscriptions": [
            {
                "channel": "email",
                "email": "example@example.com",
                "resource_id": 1
            }
        ]
    }
}
```

push_sub

```
{
     "db_id": 1,
     "matching": "push_sub",
     "provider": "android-firebase",
     "subscription_id": "<token>",
     "data": {
        "subscriptions": [
            {
                "channel": "push",
                "subscription_id": "<token>",
                "provider": "android-firebase",
                "resource_id": 1
            }
        ]
    }
}
```

custom, custom_sub

```
{
    "db_id": 1,
         "matching": "custom",
    "field_name": "CRM_id",
    "field_value": "12345",
    "data": {
       "subscriptions": [
           {
               "channel": "email",
               "email": "example@example.com",
               "resource_id": 1
           }
       ]
   }
}
```


TIP

The `subscription` object contains data about the profile's subscriptions to the resource. Learn more.

Importing the API collection in Postman

This guide will help you import and configure the API methods collection to work with Altcraft Platform steps below, you can quickly add API methods in Postman, configure environment variables, and start with the platform.

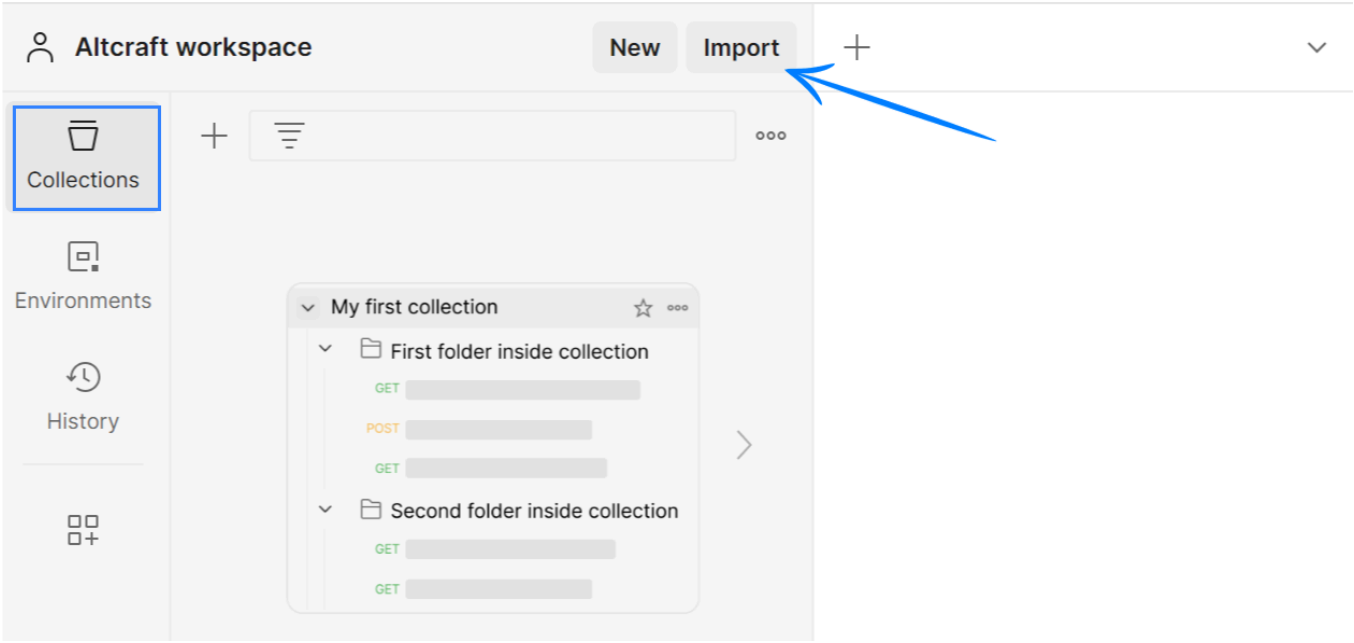
Importing the collection

1. First, make sure you have Postman installed. You can download it from the official website.

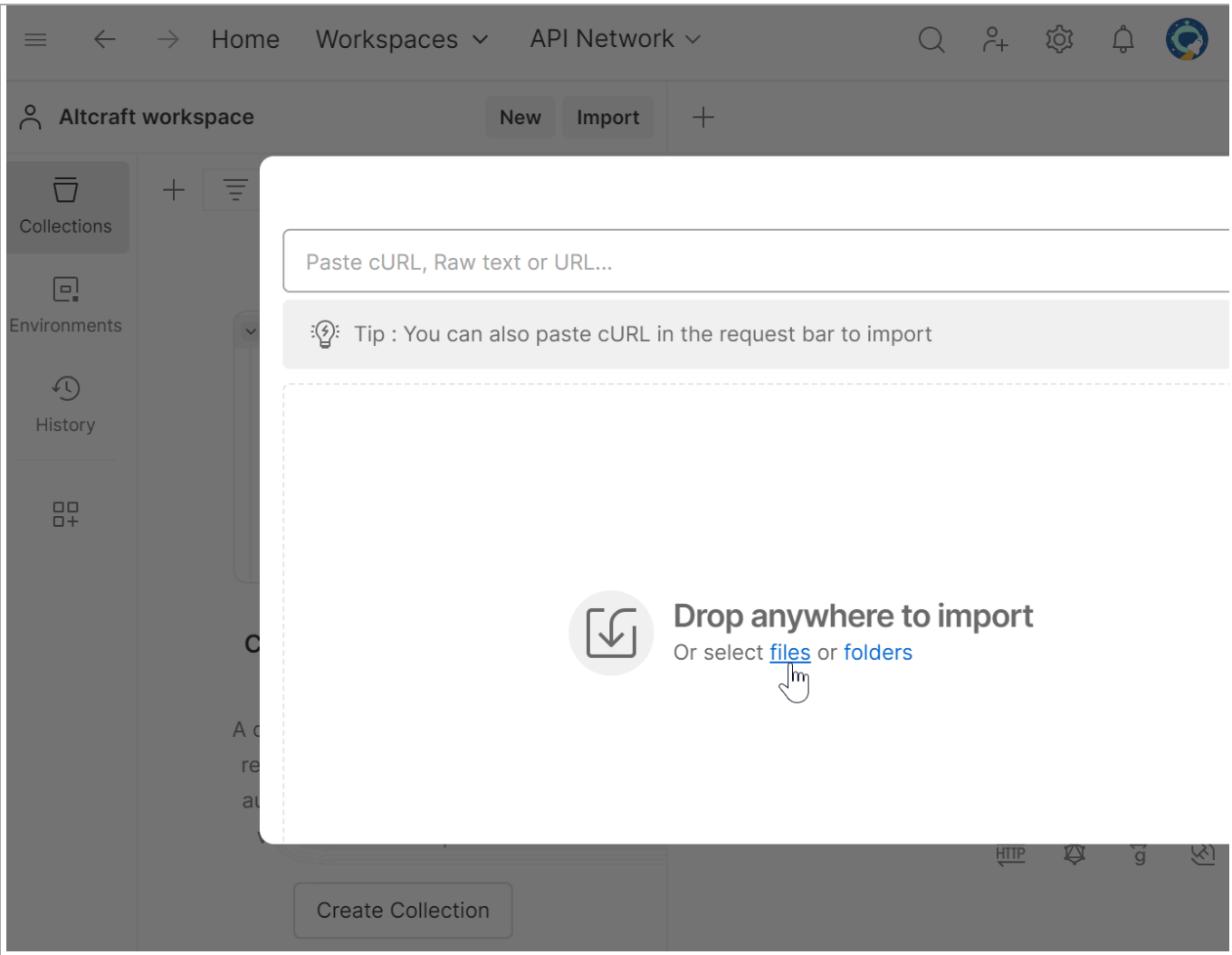
2. Download the API methods collection to your device.

> TIP
>
> **If the file doesn't start downloading, right-click the link and choose "Save link as".**

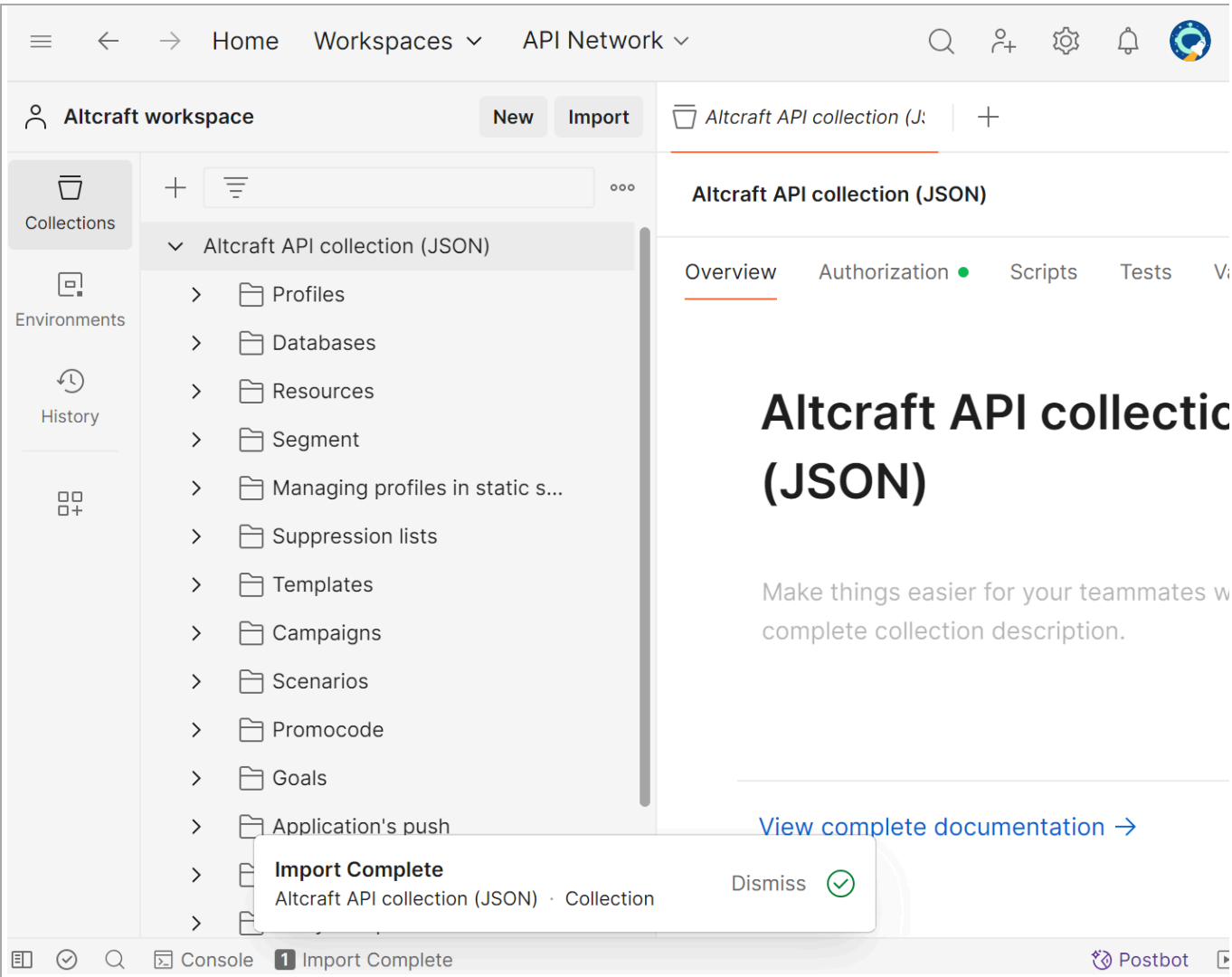
3. In Postman, go to the "Collections" section and click the "Import" button:



4. Import the collection file via the upload window:

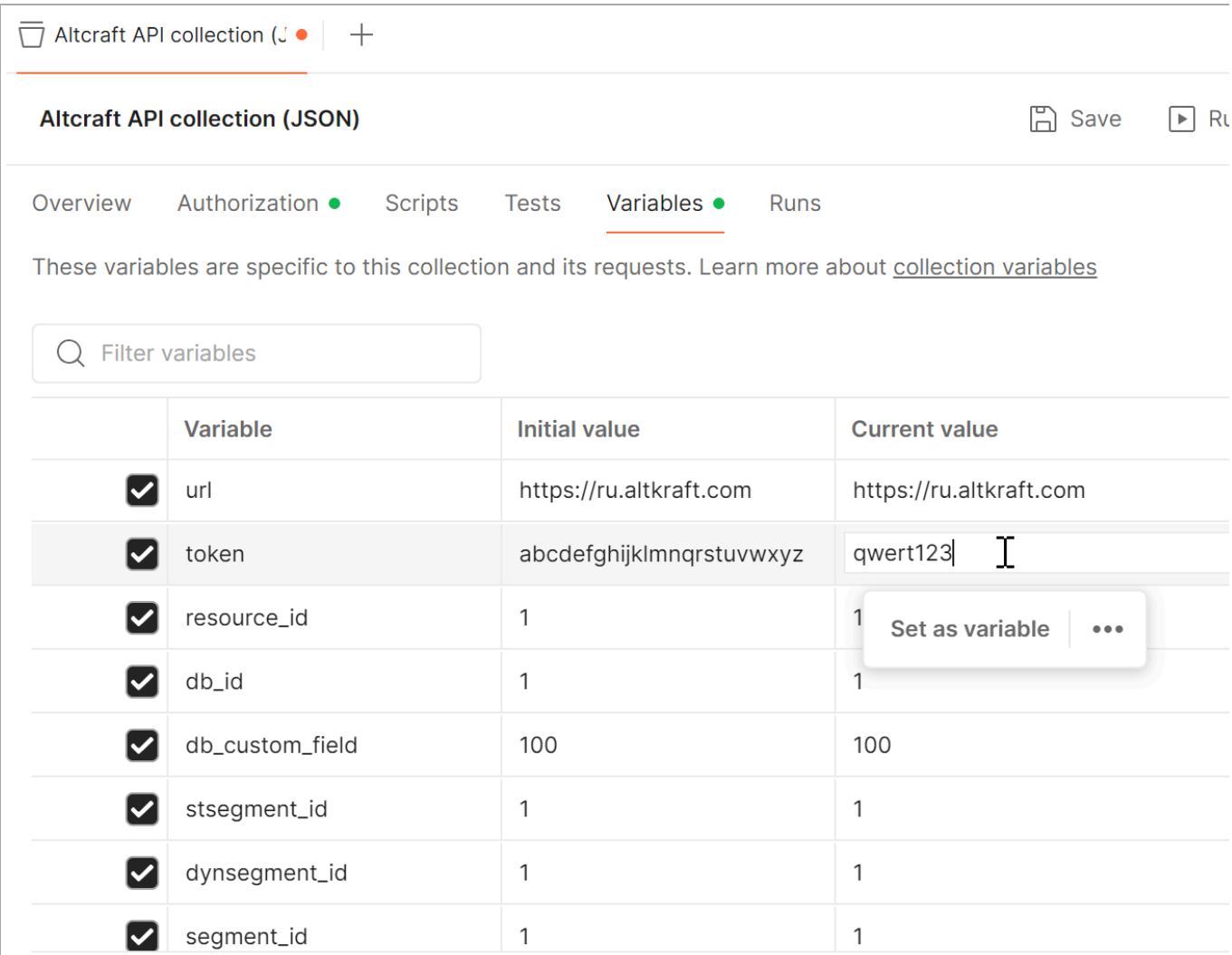


After the import is complete, all API methods will appear in the sidebar:



5. The collection includes Postman environment variables. Go to the “Variables” section to view the lis

6. The “Initial value” column contains default values for each variable. These act as examples. Replace account in the “Current value” column.



7. Save the changes by clicking the “Save” button or using Ctrl + S.

CAUTION

Make sure to save any changes to the variable list after editing.

Collection Variables List

Postman Variable	Description
url	**The URL address of the platform's control panel** **If you are using a cloud account for Altcraft Platform, use `ru.altkraft.com`. For On-premises installations, use your own URL.**
token	**The platform token** **(created in the "Settings" section => "Tokens")**
	Core Platform Objects
resource_id	**Resource ID**
db_id	**Profile database ID**
db_custom_field	**Name of the custom field in the profile database** **Note that all collection requests use a custom field of type "Integer". If you are working with other field types, make sure to insert values in the c**
stsegment_id	**Static segment ID**
dynsegment_id	**Dynamic segment ID**
segment_id	**Static or dynamic segment ID**
suplist_id	**Suppression list ID**
template_id	**Message template ID**
campaign_id	**Mailing ID**
workflow_id	**Workflow ID**
pixel_id	**Pixel ID**
loyalty_id	**Loyalty program ID**
vsender_id	**Virtual sender ID**
	System Parameters
profile_id	**Profile ID**
smid	**Message send ID** **(can be found in the email header `X-AKMTA-FSMID`)**

Postman Variable	Description
subscription_id	**Profile push subscription ID (push token)**
rtoken	**Resource token** **(can be found in the resource editing window on the "Details" tab — button ![img](/img/attachments/icons/icon_resource_details.png) in the top rig**
launch_id	**Internal send ID, unique within the mailing**
	Market
product_id	**Additional product identifier**
sku_id	**Additional SKU identifier**
order_external_id	**Additional order identifier (string)**
order_internal_id	**Order ID (integer)**
line_id	**Additional order line item ID**
market_endpoint	**Additional endpoint identifier**
market_region	**Additional region identifier**
market_category	**Additional product or SKU category identifier**
market_currency	**Currency identifier**

List of API endpoints

Profiles

Import or update profile	
Import profile	/api/v1.1/profiles/import
Import multiple profiles	/api/v1.1/profiles/import_batch
Add profile to database	/api/v1.1/profiles/add
Add multiple profiles	/api/v1.1/profiles/add_batch
Update profile	/api/v1.1/profiles/update
Update multiple profiles	/api/v1.1/profiles/update_batch
Get profile data	/api/v1.1/profiles/get/
Get data for multiple profiles	/api/v1.1/profiles/list
Uploading profiles to a file	/api/v1.1/profiles/get_all
Delete profile	/api/v1.1/profiles/delete
Merging multiple profiles	/api/v1.1/profiles/merge
Profile splitting	/api/v1.1/profiles/split
Unsubscribe profile from resource	/api/v1.1/profiles/message_unsub
Action history	
Get profile action history	/api/v1.1/subscribers/history_get
Get multiple profiles action history	/api/v1.1/subscribers/history_get_all
Import subscribers	/api/v1.1/subscribers/import
Update subscribers	/api/v1.1/subscribers/update
Profile relations	
Attach relation	/api/v1.1/relations/attach
Detach relation	/api/v1.1/relations/detach
Modify relation properties	/api/v1.1/relations/strengthen
Overwrite relation properties	/api/v1.1/relations/edit
Get profile relations info	/api/v1.1/relations/get_counter
Get profile relation data	/api/v1.1/relations/get
Subscriptions	
Add or edit subscription	/api/v1.1/subscriptions/set
Get all profile subscriptions	/api/v1.1/subscriptions/list
Get all subscriptions from multiple profiles	/api/v1.1/subscriptions/list_all
Get profile subscription	/api/v1.1/subscriptions/get
Delete profile subscription	/api/v1.1/subscriptions/delete
Restore deleted subscription	/api/v1.1/subscriptions/undelete
Suspend all subscriptions	/api/v1.1/subscriptions/suspend_all
Unsuspend all suspended subscriptions	/api/v1.1/subscriptions/unsuspend_all

Databases

Get database list	/api/v1.1/databases/list
Get database information	/api/v1.1/databases/get
Get database fields	/api/v1.1/databases/fields_get
Database wipe	/api/v1.1/databases/wipe
Get profile subscription statistics for a database	/api/v1.1/databases/get_counters
Refresh database subscription statistics	/api/v1.1/databases/refresh_counters

Resources

Get resources list	/api/v1.1/resources/list
Get resource information	/api/v1.1/resources/get
Get resource subscription fields	/api/v1.1/resources/fields_get
Get profile subscription statistics for a resource	/api/v1.1/resources/get_counters
Refresh resource subscription statistics	/api/v1.1/resources/refresh_counters
Subscribe to resource push notifications	/api/v1.1/resources/push_subscribe

Segments

Managing segments	
Create segment	/api/v1.1/segments/add
Update segment	/api/v1.1/segments/update
Get segment information	/api/v1.1/segments/get
Get segments list	/api/v1.1/segments/list
Delete segment	/api/v1.1/segments/delete
Refresh segment subscription statistics	/api/v1.1/segments/refresh_counters
Get profile subscription statistics for a segment	/api/v1.1/segments/get_counters
Managing profiles in a static segment	

Add profile	/api/v1.1/profiles/segstatic/add
Remove profile	/api/v1.1/profiles/segstatic/remove
Get profile data in a static segment	/api/v1.1/profiles/segstatic/list

Suppression lists

Managing suppression lists	
Create suppression list	/api/v1.1/suppresses/add
Update suppression list	/api/v1.1/suppresses/update
Get suppression list info	/api/v1.1/suppresses/get
Get the list of suppression lists	/api/v1.1/suppresses/list
Delete suppression list	/api/v1.1/suppresses/delete
Upload suppression list data to file	/api/v1.1/suppresses/export
Adding, checking and deleting an email address	
Check if email is suppressed	/api/v1.1/suppresses/email_check
Add email to suppression list	/api/v1.1/suppresses/email_add
Add multiple emails to suppression list	/api/v1.1/suppresses/emails_add
Remove email from suppression list	/api/v1.1/suppresses/email_remove
Remove all emails from suppression list	/api/v1.1/suppresses/emails_remove
Adding, checking and deleting an email domain	
Check if domain is suppressed	/api/v1.1/suppresses/domain_check
Add domain to suppression list	/api/v1.1/suppresses/domain_add
Add multiple domains to suppression list	/api/v1.1/suppresses/domains_add
Remove domain from suppression list	/api/v1.1/suppresses/domain_remove
Remove all domains from suppression list	/api/v1.1/suppresses/domains_remove
Adding, checking and deleting a phone number	
Check if phone number is suppressed	/api/v1.1/suppresses/phone_check
Add phone number to suppression list	/api/v1.1/suppresses/phone_add
Add multiple phones to suppression list	/api/v1.1/suppresses/phones_add
Remove phone number from suppression list	/api/v1.1/suppresses/phone_remove
Remove all phone numbers from suppression list	/api/v1.1/suppresses/phones_remove

Templates

Add template	/api/v1.1/templates/add
Update template	/api/v1.1/templates/update
Get template info	/api/v1.1/templates/get
Get templates list	/api/v1.1/templates/list
Delete template	/api/v1.1/templates/delete

Mailing

Managing mailings	
Activate mailing	/api/v1.1/campaigns/activate
Deactivate mailing	/api/v1.1/campaigns/deactivate
Get mailing information	/api/v1.1/campaigns/get
Get mailing status	/api/v1.1/campaigns/get_status
Get mailing log	/api/v1.1/campaigns/log_v2
Get mailing list	/api/v1.1/campaigns/list
Clone mailing	/api/v1.1/campaigns/clone
Delete mailing	/api/v1.1/campaigns/delete
Broadcast mailings	
Create broadcast mailing	/api/v1.1/campaigns/broadcasts/add
Update broadcast mailing	/api/v1.1/campaigns/broadcasts/update
Launch a broadcast mailing	/api/v1.1/campaigns/broadcasts/start
Get broadcast information	/api/v1.1/campaigns/broadcasts/get
Get broadcasts list	/api/v1.1/campaigns/broadcasts/list
Regular mailings	
Create regular mailing	/api/v1.1/campaigns/regulars/add
Update regular mailing	/api/v1.1/campaigns/regulars/update
Launch a regular mailing	/api/v1.1/campaigns/regulars/start
Get regular mailing information	/api/v1.1/campaigns/regulars/get
Get regular mailings list	/api/v1.1/campaigns/regulars/list
Trigger mailings	
Create trigger mailing	/api/v1.1/campaigns/triggers/add
Update trigger mailing	/api/v1.1/campaigns/triggers/update
Clone a trigger mailing	/api/v1.1/campaigns/triggers/clone
Trigger launch (API call)	/api/v1.1/campaigns/triggers/start
Bulk trigger launch	/api/v1.1/campaigns/triggers/start_batch

Task for bulk trigger launch	/api/v1.1/campaigns/triggers/start_batch_task
Profile import + trigger mailing launch	/api/v1.1/campaigns/triggers/import_and_start
Bulk profiles import + trigger mailing launch	/api/v1.1/campaigns/triggers/import_and_start_bat
Task for bulk profiles import + trigger launch	/api/v1.1/campaigns/triggers/import_and_start_bat
Get trigger mailing information	/api/v1.1/campaigns/triggers/get
Get trigger mailings list	/api/v1.1/campaigns/triggers/list
Bulk trigger launch (alternative path)	/api/v1.1/triggers/start_batch
Task for bulk trigger launch (alternative path)	/api/v1.1/triggers/start_batch_task
Bulk profiles import + trigger launch (alternative path)	/api/v1.1/triggers/import_and_start_batch
Task for bulk profiles import + trigger launch (alternative path)	/api/v1.1/triggers/import_and_start_batch_task

Campaigns

Campaign Management	
Get campaign information	/api/v1.1/scampaigns/get
Get campaign list	/api/v1.1/scampaigns/list
Activate campaign	/api/v1.1/scampaigns/activate
Deactivate campaign	/api/v1.1/scampaigns/deactivate
Complete campaign	/api/v1.1/scampaigns/completed
Get campaign status	/api/v1.1/scampaigns/get_status

Automation scenarios

Engage profile in scenario	/api/v1.1/workflows/start
Import and engage profile in scenario	/api/v1.1/workflows/import_and_start
Batch import and engage profiles in a scenario	/api/v1.1/workflows/import_and_start_batcl
Task for batch import and engaging profiles in the scenarioй	/api/v1.1/workflows/import_and_start_batcl
Get scenarios list	/api/v1.1/workflows/list
Activate scenario	/api/v1.1/workflows/activate
Deactivate scenario	/api/v1.1/workflows/deactivate

Promo codes

Import promo codes	/api/v1.1/promocodes/import
Get promo code information	/api/v1.1/promocodes/get
Activate promo code	/api/v1.1/promocodes/activate
Update promo code	/api/v1.1/promocodes/update
Attach promo code	/api/v1.1/promocodes/attach
Detach promo code	/api/v1.1/promocodes/detach
Detach promo code	/api/v1.1/promocodes/export_all

Loyalty Program

Points and Transactions Management	
Get points balance	/api/v1.1/loyalty/account/balance
Get profile account transactions	/api/v1.1/loyalty/account/transactions
Get expiring points for a period	/api/v1.1/loyalty/account/expiring_points
Get profile loyalty tier	/api/v1.1/loyalty/account/tiers
Accrue points to member	/api/v1.1/loyalty/accrue_points
Redeem member points	/api/v1.1/loyalty/redeeme_points
Cancel points transaction	/api/v1.1/loyalty/cancel_transaction
Export points transactions	/api/v1.1/loyalty/transactions_export
Member Management	
Register member in loyalty program	/api/v1.1/loyalty/register_member
Delete member from loyalty program	/api/v1.1/loyalty/delete_member
Batch register members	/api/v1.1/loyalty/register_member_batch
Task for batch member registration	/api/v1.1/loyalty/register_member_batch_task
Temporary Transactions and Preorders	
Get trigger promotions list	/api/v1.1/loyalty/trigger_promotions
Preorder	/api/v1.1/market/orders/preorder
Confirm order	/api/v1.1/market/orders/preorder_commit
Confirm temporary transaction	/api/v1.1/loyalty/commit_transaction
Cancel temporary transaction	/api/v1.1/loyalty/rollback_transaction

Goals

Goals and goal values registration	/api/v1.1/goals/register

Application push notifications

Add app push events	/api/v1.1/pushes/post_events
Add app push events (crypto)	/api/v1.1/pushes/crypto_post_events

Market

Orders	
Import order and item statuses	/api/v1.1/market/orders/import
Get orders list	/api/v1.1/market/orders/list
Delete order	/api/v1.1/market/orders/delete
Get order status	/api/v1.1/market/orders/get_status
Update order line status	/api/v1.1/market/orders/update_order_line_status
Products and SKU	
Import products, SKUs and categories	/api/v1.1/market/products/import
Import SKUs and categories	/api/v1.1/market/sku/import
Get products list	/api/v1.1/market/products/list
Get SKUs list	/api/v1.1/market/sku/list
Delete product	/api/v1.1/market/products/delete
Delete SKU	/api/v1.1/market/sku/delete

Analytic reports

Get summary report	/api/v1.1/reports/summary
Get soft bounces report	/api/v1.1/reports/bounces
Get undeliveries report	/api/v1.1/reports/undelivered

Senders

Senders list	
Get senders list	/api/v1.1/senders/list
Virtual senders	
Create virtual sender	/api/v1.1/admin/vsenders/add
Update virtual sender	/api/v1.1/admin/vsenders/update
Get virtual sender information	/api/v1.1/admin/vsenders/get
Get virtual senders list	/api/v1.1/admin/vsenders/list
Clone virtual sender	/api/v1.1/admin/vsenders/clone
Delete virtual sender	/api/v1.1/admin/vsenders/delete

External datatables queries

Segmentation queries	
Add segmentation query	/api/v1.1/datatables/query_segments/add
Update segmentation query	/api/v1.1/datatables/query_segments/update
Delete segmentation query	/api/v1.1/datatables/query_segments/delete
Get segmentation queries list	/api/v1.1/datatables/query_segments/list
Get segmentation query info	/api/v1.1/datatables/query_segments/get
Templates queries	
Add templates query	/api/v1.1/datatables/query_templates/add
Update templates query	/api/v1.1/datatables/query_templates/update
Delete templates query	/api/v1.1/datatables/query_templates/delete
Get templates queries list	/api/v1.1/datatables/query_templates/list
Get templates query info	/api/v1.1/datatables/query_templates/get

Miscellaneous

Upload file	/api/v1.1/file_upload
Get message web version	/api/v1.1/webver/get
Check task status	/api/v1.1/tasks/status
Dictionary	/api/v1.1/dictionary

Import profile

Description

Imports a customer profile. It also can be used to update an existing profile.
Can delete custom fields, add or delete tags, increase or decrease field values. See update functions f

> **TIP**
>
> **To launch a trigger mailing right after the import you can use Import + trigger request.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/import`

Request parameters

> **CAUTION**
>
> - **For JSON body use the following header: Content-Type: application/json.**
> - **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	Required	Description
db_id	**int**	**1**	**Yes**	**Database ID**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No, if matched by email**	**Profile matching mode By default — email For each type of matching, certain fields must be passed in the reques**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by _regip or _ip from data object**
add_to_segments	**array**	`[1, 2]`	**No**	**Add a profile to a segment**
remove_from_segments	**array**	`[1, 2]`	**No**	**Remove a profile from a segment**
data	**JSON object**	`{ “_fname”: “John”, “_lname”: “Doe” }`	**Yes**	**Profile fields data, including resource subscription data (`subscriptions`)**

Subscriptions array

The `subscriptions` array stores data about the profile’s subscriptions to resources. One object is one su

Parameter	Type	Example	Description
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status It is recommended to pass the parameter in the request only in cases where it is necessary to update the subscrip**
priority	**int**	**1**	**Subscription priority If the subscription priority is passed in the request, the "Increase priority for new subscriptions" option in the [resc work.**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**System and custom subscription fields. You can get information about available subscription fields in a resource using the /v1.1/resources/fields_get metho**
cats	**array**	`[ "category_1", "category_2" ]`	**Resource categories to which to subscribe the profile.**
Email channel			
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**
Telegram Bot channel			

Parameter	Type	Example	Description
channel	**string**	**"telegram_bot"**	**Channel type**
cc_data	**object**	{ "id": 1 }	**Telegram Bot chat ID**
WhatsApp* channel			
channel	**string**	**"whatsapp"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on WhatsApp***
Viber channel			
channel	**string**	**"viber"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on Viber**

Request example

Importing a profile and adding an email channel subscription

JSON

```
{
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": ["+79000000000"],
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <matching>email</matching>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Importing a profile with extended data

JSON

```
{
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",

        "email": "example@example.com",
        "phones": ["+79000000000"],

        "_bdate": "1990-02-22T21:00:00Z",
        "_sex": 0,

        "_regdate": "2019-03-14T22:00:00Z",
        "_regip": "94.231.119.122",
        "_ip": "94.231.119.122",

        "_tz": "Europe/Moscow",
        "_postal_code": "390000",

        "_os": "Windows 10",
        "_browser": "Firefox",

        "_vendor": "form_#31",

         "custom_field": "custom_value",

         "Custom_Tags": ["tag_1", "tag_2", "tag_3"],
         "Custom_Int": 42,
         "Custom_Float": 3.14,
         "Custom_Bool": true,
         "Custom_Enum": "option_a",
         "Custom_Date": "2025-01-15",
         "Custom_Datetime": "2025-01-15T10:30:00Z",
         "Custom_Object": {"key": "value"},

         "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```


XML

```
<xml>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <_bdate>1990-07-06</_bdate>
        <_sex>0</_sex>
        <_regdate>2023-07-06</_regdate>
        <_regip>192.168.0.179</_regip>
        <_ip>192.168.0.179</_ip>
        <_tz>Europe/Moscow</_tz>
        <_postal_code>390000</_postal_code>
        <_os>Windows 10</_os>
        <_browser>Firefox</_browser>
        <_vendor>form_#31</_vendor>
         <custom_field>custom_value</custom_field>
         <Custom_Tags>tag_1, tag_2, tag_3</Custom_Tags>
         <Custom_Int>42</Custom_Int>
         <Custom_Float>3.14</Custom_Float>
         <Custom_Bool>true</Custom_Bool>
         <Custom_Enum>option_a</Custom_Enum>
         <Custom_Date>2025-01-15</Custom_Date>
         <Custom_Datetime>2025-01-15T10:30:00Z</Custom_Datetime>
         <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Custom field value formats

Depending on the custom field type, the value can be passed in different formats:

Field type	Value format
Tags	**String array or a string with tags separated by comma. Both formats are supported: the array is passed as-is, the string is split by comma. When passing a string, the only valid separator is a comma. In both cases, leading and trailing spaces of each tag are trimmed.**
String	**String. Numeric and boolean values are automatically converted to string**
Integer	**Number or string containing a number**
Float	**Number or string containing a number**
Boolean	**Boolean value**
List	**String (option name from the list) or number (ordinal position of the option, starting from 1)**
Date	**String in `YYYY-MM-DD` format, ISO 8601, unix timestamp (number or string)**
Date/time	**Unix timestamp (number) or ISO 8601 string**
Object	**JSON object**

Importing a profile and adding subscriptions to multiple channels

JSON

```
{
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "detect_geo": true,
    "data": {
       "_fname": "Olly",
       "_lname": "Lambert",
       "email": "example@example.com",
       "subscriptions": [
           {
               "resource_id": 1,
               "channel": "email",
               "email": "example@example.com"
           },
           {
               "resource_id": 2,
               "channel": "sms",
               "phone": "+79000000000"
           },
           {
               "resource_id": 1,
               "channel": "push",
               "subscription_id": "a81c264a938b475",
               "provider": "android-firebase",
               "custom_fields": {
                   "_browser_name": "Firefox",
                   "_device_type": "web",
                   "custom_field_1": "test value"
               },
               "cats": [
                   "category_1",
                   "category_2"
               ]
           },
           {
               "resource_id": 1,
               "channel": "whatsapp",
               "cc_data": {
                   "phone": "+79000000000"
               }
           },
           {
               "resource_id": 1,
               "channel": "viber",
               "cc_data": {
                   "phone": "+79000000000"
               }
           },
           {
               "resource_id": 1,
               "channel": "telegram_bot",
               "cc_data": {
                   "id": 12345
               }
           },
           {
               "resource_id": 1,
               "channel": "notify",
               "cc_data": {
                   "phone": "+79000000000"
               }
           }
       ]
   }
}
```


XML

```
<xml>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>sms</channel>
            <phone>+79000000000</phone>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <subscription_id>a81c264a938b475</subscription_id>
            <provider>android-firebase</provider>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>whatsapp</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>viber</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>telegram_bot</channel>
            <cc_data>
              <id>12345</id>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>notify</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>c1are24fu213lwith21that2axe9<profile_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile ID**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Update profile

Description

Updates a customer profile.
Can delete custom fields, add or delete tags, increase or decrease field values. See update functions f

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/update`

Request parameters

CAUTION

- **For JSON body use the following header: Content-Type: application/json.**
- **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	Required	Description
db_id	**int**	**1**	**Yes**	**Database ID**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No, if matched by email**	**Profile matching mode By default — email For each type of matching, certain fields must be passed in the reques**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by _regip or _ip from data object**
add_to_segments	**array**	`[1, 2]`	**No**	**Add a profile to a segment**
remove_from_segments	**array**	`[1, 2]`	**No**	**Remove a profile from a segment**
data	**JSON object**	`{ “_fname”: “John”, “_lname”: “Doe” }`	**Yes**	**Profile fields data, including resource subscription data (`subscriptions`)**

Subscriptions array

The `subscriptions` array stores data about the profile’s subscriptions to resources. One object is one sul

Parameter	Type	Example	Description
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status It is recommended to pass the parameter in the request only in cases where it is necessary to update the subscrip**
priority	**int**	**1**	**Subscription priority If the subscription priority is passed in the request, the "Increase priority for new subscriptions" option in the [resc work.**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**System and custom subscription fields. You can get information about available subscription fields in a resource using the /v1.1/resources/fields_get metho**
cats	**array**	`[ "category_1", "category_2" ]`	**Resource categories to which to subscribe the profile.**
Email channel			
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**
Telegram Bot channel			
channel	**string**	**"telegram_bot"**	**Channel type**
cc_data	**object**	`{ "id": 1 }`	**Telegram Bot chat ID**
WhatsApp* channel			

Parameter	Type	Example	Description
channel	**string**	**"whatsapp"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on WhatsApp***
Viber channel			
channel	**string**	**"viber"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on Viber**

Request example

Updating basic profile data and adding an email channel subscription

JSON

```
{
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": ["+79000000000"],
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <matching>email</matching>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Updating additional profile data

JSON

```
{
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",

        "email": "example@example.com",
        "phones": ["+79000000000"],

        "_bdate": "1990-02-22T21:00:00Z",
        "_sex": 0,

        "_regdate": "2019-03-14T22:00:00Z",
        "_regip": "94.231.119.122",
        "_ip": "94.231.119.122",

        "_tz": "Europe/Moscow",
        "_postal_code": "390000",

        "_os": "Windows 10",
        "_browser": "Firefox",

        "_vendor": "form_#31",

        "custom_field": "custom_value"
    }
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <_bdate>1990-07-06</_bdate>
        <_sex>0</_sex>
        <_regdate>2023-07-06</_regdate>
        <_regip>192.168.0.179</_regip>
        <_ip>192.168.0.179</_ip>
        <_tz>Europe/Moscow</_tz>
        <_postal_code>390000</_postal_code>
        <_os>Windows 10</_os>
        <_browser>Firefox</_browser>
        <_vendor>form_#31</_vendor>
        <custom_field>custom_value</custom_field>
    </data>
</xml>
```

Adding subscriptions to multiple channels to a profile

JSON

```
{
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "detect_geo": true,
    "data": {
       "_fname": "Olly",
       "_lname": "Lambert",
       "email": "example@example.com",
       "subscriptions": [
           {
               "resource_id": 1,
               "channel": "email",
               "email": "example@example.com"
           },
           {
               "resource_id": 2,
               "channel": "sms",
               "phone": "+79000000000"
           },
           {
               "resource_id": 1,
               "channel": "push",
               "subscription_id": "a81c264a938b475",
               "provider": "android-firebase",
               "custom_fields": {
                   "_browser_name": "Firefox",
                   "_device_type": "web",
                   "custom_field_1": "test value"
               },
               "cats": [
                   "category_1",
                   "category_2"
               ]
           },
           {
               "resource_id": 1,
               "channel": "whatsapp",
               "cc_data": {
                   "phone": "+79000000000"
               }
           },
           {
               "resource_id": 1,
               "channel": "viber",
               "cc_data": {
                   "phone": "+79000000000"
               }
           },
           {
               "resource_id": 1,
               "channel": "telegram_bot",
               "cc_data": {
                   "id": 12345
               }
           },
           {
               "resource_id": 1,
               "channel": "notify",
               "cc_data": {
                   "phone": "+79000000000"
               }
           }
       ]
   }
}
```


XML

```
<xml>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>sms</channel>
            <phone>+79000000000</phone>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <subscription_id>a81c264a938b475</subscription_id>
            <provider>android-firebase</provider>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>whatsapp</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>viber</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>telegram_bot</channel>
            <cc_data>
              <id>12345</id>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>notify</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>c1are24fu213lwith21that2axe9<profile_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile ID**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Import multiple profiles

Description

Adds or updates multiple profiles to or in a database.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/import_batch`

Request parameters

CAUTION

- **For JSON body use the following header: Content-Type: application/json.**
- **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	Required	Description
data	**JSON array**	`[ { "_fname": "John", "_lname": "Doe", "custom_id": "abcdefghijklmnqrstuvwxyz", "email": "john@example.com", "subscriptions": [ { "channel": "email", "email_data": { "email": "john@example.com" }, "resource_id": 1 } ] } ]`	**Yes**	**Profile fields data**
db_id	**int**	**1**	**Yes**	**Database ID**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions. By default – false**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by `_regip` or `_ip` from data object**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email**	**Profile matching mode By default - email For each type of matching, certain fields must be passed in the requ**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom field value.** TIP **Searching by combination of tags is available for<a href='/en/user- type profile fields. Values are recorded as a string: "tag-1, tag-**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Subscriptions array

The `subscriptions` array stores data about the profile's subscriptions to resources. One object is one sul

Parameter	Type	Example	Description
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status It is recommended to pass the parameter in the request only in cases where it is necessary to updat**
priority	**int**	**1**	**Subscription priority**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**System and custom subscription fields. You can get information about available subscription fields in a resource using the /v1.1/resources/fi**
cats	**array**	`[ "category_1", "category_2" ]`	**Resource categories to which to subscribe the profile.**
Email channel			

Parameter	Type	Example	Description
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**
Telegram Bot channel			
channel	**string**	**"telegram_bot"**	**Channel type**
cc_data	**object**	`{ "id": 1 }`	**Telegram Bot chat ID**
WhatsApp* channel			
channel	**string**	**"whatsapp"**	**Channel type**
cc_data	**object**	`{ "phone": "+79000000000" }`	**Profile phone number on WhatsApp***
Viber channel			
channel	**string**	**"viber"**	**Channel type**
cc_data	**object**	`{ "phone": "+79000000000" }`	**Profile phone number on Viber**

Request example

JSON

```
{
    "db_id": 1,
    "matching": "custom",
    "field_name": "Custom_ID",
    "skip_triggers": true,
    "data": [
        {
            "_fname": "Lala",
            "_lname": "Doe",
            "Custom_ID": 77,
            "email": "example@example.com",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 28,
                    "custom_fields": {
                        "custom_field": "test value"
                    },
                    "cats": [
                        "category_1",
                        "category_2"
                    ]
                }
            ]
        },
        {
            "_fname": "Victor",
            "Custom_ID": 88,
            "email": "example@example.com"
        }
    ]
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Lala</_fname>
        <_lname>Doe</_lname>
        <Custom_ID>77</Custom_ID>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>28</resource_id>
            <custom_fields>
                <custom_field>test value</custom_field>
            </custom_fields>
            <cats>category_1</cats>
            <cats>category_2</cats>
        </subscriptions>
    </data>
    <data>
        <_fname>Alex</_fname>
        <Custom_ID>345</Custom_ID>
        <email>example@example.com</email>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "abcdefghijklmnqrstuvwxyz"
    }
  ]
}
```

XML

```
<xml>
 <error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  </result>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array of object**	**Update results by profiles** **The order of the objects in the returned `result` array repeats the order of the objects in the input `data` array.**
result.error	**int**	**Error code, for profile**
result.error_text	**string**	**Error text, for profile**
result.profile_id	**string**	**Profile ID**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Update multiple profiles

Description

Updates (only) multiple profiles to or in a database.

> **INFO**
>
> **If you pass the `status` and `priority` parameters of the `subscriptions` object with a value of `0`, then the**
> **In order not to change or update the values of these parameters, do not pass them, or set the values t**

> **CAUTION**
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible totransfer more data using multiple streams.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/update_batch`

Request parameters

Parameter	Type	Example	Required	Description
data	**JSON array**	`[ { "_fname": "John", "_lname": "Doe", "custom_id": "abcdefghijklmnqrstuvwxyz", "email": "john@example.com", "subscriptions": [ { "channel": "email", "email_data": { "email": "john@example.com" }, "resource_id": 1 } ] } ]`	**Yes**	**Profile fields data**
db_id	**int**	**1**	**Yes**	**Database ID**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions. By default – false**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email**	**Profile matching mode By default - email For each type of matching, certain fields must be passed in the requ**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** **TIP** **Searching by combination of tags is available for<a href='/en/user- type profile fields. Values are recorded as a string: "tag-1, tag-**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Subscriptions array

The `subscriptions` array stores data about the profile's subscriptions to resources. One object is one sul

Parameter	Type	Example	Description
resource_id	**int**	**1**	**Resource ID**

Parameter	Type	Example	Description
status	**string**	**"subscribed"**	**Subscription status** It is recommended to pass the parameter in the request only in cases where it is necessary to update the subscription status.

Request example

JSON

```
{
    "db_id": 1,
    "matching": "custom",
    "field_name": "Custom_ID",
    "skip_triggers": true,
    "data": [
        {
            "_fname": "Lala",
            "_lname": "Doe",
            "Custom_ID": 77,
            "email": "example@example.com",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 28,
                    "custom_fields": {
                        "custom_field": "test value"
                    },
                    "cats": [
                        "category_1",
                        "category_2"
                    ]
                }
            ]
        },
        {
            "_fname": "Victor",
            "Custom_ID": 88,
            "email": "example@example.com"
        }
    ]
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Lala</_fname>
        <_lname>Doe</_lname>
        <Custom_ID>77</Custom_ID>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>28</resource_id>
            <custom_fields>
                <custom_field>test value</custom_field>
            </custom_fields>
            <cats>category_1</cats>
            <cats>category_2</cats>
        </subscriptions>
    </data>
    <data>
        <_fname>Alex</_fname>
        <Custom_ID>345</Custom_ID>
        <email>example@example.com</email>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "error": 0,
            "error_text": "Successful operation",
            "profile_id": "abcdefghijklmnqrstuvwxyz"
        },
        {
            "error": 0,
            "error_text": "Successful operation",
            "profile_id": "abcdefghijklmnqrstuvwzzz"
        }
    ]
}
```


XML


```
<xml>
<error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  </result>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwzzz</profile_id>
  </result>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile ID**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Add multiple profiles

Description

Adds multiple new profiles to a database.
Profiles list is contained in `"data"` parameter as a JSON array. Server response will contain `"result"` pa
profile IDs.

> CAUTION
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible totransfer more data using multiple streams.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/add_batch/`

Request parameters

Parameter	Type	Example	Required
data	**JSON object**	`[ { "_fname": "John", "_lname": "Doe", "subscriptions": [ { "channel": "email", "email_data": { "email": "john@example.com" }, "resource_id": 1 } ] } ]`	**Yes**
db_id	**int**	**1**	**Yes**
skip_triggers	**bool**	**true**	**No**
skip_invalid_subscriptions	**bool**	**true**	**No**

Request example

JSON

```
{
    "db_id": 1,
    "skip_triggers": true,
    "data": [
        {
            "_fname": "John",
            "_lname": "Doe",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 27
                }
            ]
        }
    ]
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <skip_triggers>true</skip_triggers>
    <data array='true'>
        <_fname>Fillipp</_fname>
        <_lname>Doe</_lname>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>1</resource_id>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "663a138cee44bb50a67115cd"
    }
  ]
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>663a138cee44bb50a67115cd</profile_id>
  </xml>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile ID**

Add profile to database

> **TIP**
>
> **The request does not look for existing records and adds any data other than those that duplicate the recommend using the Import profile request in most cases.**

Description

Adds a new customer profile to a database.

> **CAUTION**
>
> **If no data is specified — an empty profile will be created.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/add`

Request parameters

Parameter	Type	Example	Required	
db_id	**int**	**1**	**Yes**	**Databa**
data	**JSON object**	{ “_fname”: “John”, “_lname”: “Doe” }	**Yes**	**Profile**
skip_triggers	**bool**	**true**	**No**	**Skip in By defa**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip in By defa**

Request example

JSON

```
{
    "db_id": 1,
    "data": {
        "_fname": "Fillipp",
        "_lname": "Doe"
    },
    "skip_triggers": true
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Fillipp</_fname>
        <_lname>Doe</_lname>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "profile_id": "6638e1e8ee44bb50a67115c4"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>6638e1e8ee44bb50a67115c4</profile_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Import profile to RabbitMQ

Description

Imports or updates customer profiles via RabbitMQ message broker queue. Useful for extensive data tr
You will need a queue named “`database_import`”. Therefore it must be available in the same message bro
system.
Import results are processed in another queue – “`database_import_result`”. The default queue time-to-live i
reconfigure TTL in `main.json` configuration file:

```
RABBITMQ_X_MESSAGE_TTL : "%seconds%"
```

Use one of these parameters to identify and match profiles:

- **email**
- **field_name, field_value**
- **phone**
- **profile_id**

To identify your requests you can use an extra field – “`feedback_id`”. It will appear in server response an
be logged in “`job_server.log`”.

Message parameters

Parameter	Type	Example	Required	Description
account_id	**int**	**1**	**Yes**	**Account ID**
data	**JSON object**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**	**Profile fields data**
db_id	**int**	**1**	**Yes**	**Database ID**
feedback_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**No**	**Request ID**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
update_only	**bool**	**true**	**No**	**Only update. By default – false**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/user-guide/profiles-and- fields. Values are recorded as a string: "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Message example

```
{
    "account_id": 1,
    "data": {
        "_fname": "John",
        "_lname": "Doe"
    },
    "db_id": 1,
    "email": "example@example.com",
    "feedback_id": "abcdefghijklmnqrstuvwxyz",
    "update_only": true
}
```

Response example

```
{
    "error": 0,
    "error_text": "",
    "feedback_id": "abcdefghijklmnqrstuvwxyz",
    "matching_field": "email",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
feedback_id	**string**	**Request ID**
matching_field	**string**	**Profile matching**
profile_id	**string**	**Profile ID**

Get profile data

Description

Retrieves customer profile data, including database fields and subscriptions.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/get/`

Request parameters

Parameter	Type	Example	Required	Description
db_id	**int**	**1**	**Yes**	**Database ID**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"example@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/user-guide/profiles-and-data Values are recorded as a string: "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "db_id": 1,
    "email": "example@example.com"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <email>example@example.com</email>
</xml>
```

Response example

JSON

```
{
    "data": {
        "LOYALTY_SCORE": 500,
        "New_field_7": "00016",
        "_acid": "FBFLR5Z8BxgvNPxBJZgL",
        "_bdate": "1990-07-06T11:15:25.526Z",
        "_browser": "Chrome",
        "_city": "Beijing",
        "_click_exist": false,
        "_click_push_exist": false,
        "_click_sms_exist": false,
        "_confirm_exist": false,
        "_country": "CN",
        "_createtime": "2024-04-08T10:48:05.418Z",
        "_deliv_date": "2024-04-18T10:18:29.809Z",
        "_deliv_exist": true,
        "_deliv_push_exist": false,
        "_deliv_sms_date": "2024-05-06T10:00:05.548Z",
        "_deliv_sms_exist": true,
        "_domaingroup": "other",
        "_emaildomain": "example.com",
        "_fname": "Christopher",
        "_ip": "106.12.21.70",
        "_is_desktop": true,
        "_is_test": true,
        "_lat": 34.7732,
        "_lname": "Lambert",
        "_lon": 113.722,
        "_md5": "1118e73b34d3f9a9030b9e4da6dcfb29",
        "_open_date": "2024-04-18T10:18:29.727Z",
        "_open_exist": true,
        "_open_push_exist": false,
        "_os": "Windows 10",
        "_pixel_open_exist": false,
        "_platform": "Other",
        "_postal_code": "17",
        "_ps_wf": [
            12,
            11,
            10
        ],
        "_ps_wfs": [
            {
                "di": "2024-04-08T10:48:07.093Z",
                "do": "2024-04-13T10:48:22.359Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 12
            },
            {
                "di": "2024-04-08T19:35:04.15Z",
                "do": "2024-04-12T19:37:19.821Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 11
            },
            {
                "di": "2024-04-16T14:42:49.063Z",
                "do": "2024-04-17T14:42:52.438Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 10
            }
        ],
        "_read_date": "2024-04-18T10:18:29.727Z",
        "_read_exist": true,
        "_regdate": "2024-04-08T10:48:05.418Z",
        "_region": "Beijing",
        "_reply_exist": false,
        "_send_date": "2024-04-10T09:32:06.275960407Z",
        "_send_exist": true,
        "_send_push_exist": false,
        "_send_sms_date": "2024-04-12T12:17:05.995339508Z",
        "_send_sms_exist": true,
        "_sex": 1,
        "_status": 0,
        "_subscribetime": "2024-04-08T10:48:05.418Z",
```

```
        "_tz": "Asia/Shanghai",
        "_undeliv_date": "2024-04-17T11:49:42.098Z",
        "_undeliv_exist": true,
        "_updatetime": "2024-04-19T14:25:22.185Z",
        "_xxh": "5ce324d8407a8e2b",
        "_xxm": 8507,
        "db_id": 1,
        "email": "example@example.com",
        "id": "6613cb65dc4594c2934db63f",
        "phones": [
            "+79568150396"
        ],
        "subscriptions": [
            {
                "resource_id": 24,
                "channel": "email",
                "email_data": {
                    "email": "example@example.com",
                    "domain": "example.com",
                    "domain_group": "other",
                    "md5": "1118e73b34d3f9a9030b9e4da6dcfb29"
                },
                "priority": 0,
                "status": "unsubscribed",
                "reg_info": {
                    "date": "2024-04-10T09:28:55.743Z"
                },
                "hash_id": "4a3c5813",
                "is_delete": false
            },
            {
                "resource_id": 25,
                "channel": "sms",
                "phone_data": {
                    "phone": "+79568150396",
                    "md5": "06ec7f059dae6f002d06a2f62a4320d8"
                },
                "priority": 0,
                "status": "subscribed",
                "reg_info": {
                    "date": "2024-04-12T12:16:44.086Z"
                },
                "hash_id": "f67a97c1",
                "is_delete": false
            }
        ],
        "unsub": {
            "c": 115,
            "dt": "2024-04-18T10:18:29.727Z",
            "m": 15
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <CRM_ID>66</CRM_ID>
        <LOYALTY_SCORE>500</LOYALTY_SCORE>
        <_acid>HWbGhUN1jKAdY8iNjdYf</_acid>
        <_browser>Chrome</_browser>
        <_city>St Petersburg</_city>
        <_click_exist>false</_click_exist>
        <_click_push_exist>false</_click_push_exist>
        <_click_sms_exist>false</_click_sms_exist>
        <_confirm_exist>false</_confirm_exist>
        <_country>RU</_country>
        <_createtime>2024-04-08T10:24:09.087Z</_createtime>
        <_deliv_date>2024-05-06T10:00:06.623Z</_deliv_date>
        <_deliv_exist>true</_deliv_exist>
        <_deliv_push_exist>false</_deliv_push_exist>
        <_deliv_sms_date>2024-05-06T10:00:05.274Z</_deliv_sms_date>
        <_deliv_sms_exist>true</_deliv_sms_exist>
        <_domaingroup>other</_domaingroup>
        <_emaildomain>example.com</_emaildomain>
        <_fname>Alina</_fname>
        <_ip>92.53.90.8</_ip>
        <_is_desktop>true</_is_desktop>
        <_is_mobile>true</_is_mobile>
        <_is_test>true</_is_test>
        <_lat>59.939095</_lat>
        <_lname>Lambert</_lname>
        <_lon>30.315868</_lon>
        <_md5>c5278382ce8972b4cd23b447c0e58989</_md5>
        <_open_date>2024-05-06T10:00:06.481Z</_open_date>
        <_open_exist>true</_open_exist>
        <_open_push_exist>false</_open_push_exist>
        <_os>Windows 10</_os>
        <_pixel_open_exist>false</_pixel_open_exist>
        <_platform>Other</_platform>
        <_postal_code>1042</_postal_code>
        <_ps_wf>12</_ps_wf>
        <_ps_wf>11</_ps_wf>
        <_ps_wfs>
            <di>2024-04-08T10:24:10.069Z</di>
            <do>2024-04-19T10:24:43.381Z</do>
            <ln>8</ln>
            <s>1</s>
            <wId>12</wId>
        </_ps_wfs>
        <_ps_wfs>
            <di>2024-04-08T19:35:04.091Z</di>
            <do>2024-04-18T19:37:37.428Z</do>
            <ln>8</ln>
            <s>1</s>
            <wId>11</wId>
        </_ps_wfs>
        <_read_date>2024-05-06T10:00:06.481Z</_read_date>
        <_read_exist>true</_read_exist>
        <_regdate>2024-04-08T10:24:09.087Z</_regdate>
        <_region>Saint Petersburg</_region>
        <_reply_exist>false</_reply_exist>
        <_sbounce_date>2024-04-28T10:00:03.452Z</_sbounce_date>
        <_sbounce_exist>true</_sbounce_exist>
        <_send_date>2024-04-10T09:32:05.531223661Z</_send_date>
        <_send_exist>true</_send_exist>
        <_send_push_exist>false</_send_push_exist>
        <_send_sms_date>2024-04-18T11:14:02.172438444Z</_send_sms_date>
        <_send_sms_exist>true</_send_sms_exist>
        <_status>0</_status>
        <_subscribetime>2024-04-08T10:24:09.087Z</_subscribetime>
        <_tz>Europe/Moscow</_tz>
        <_undeliv_date>2024-04-17T11:49:39.091Z</_undeliv_date>
        <_undeliv_exist>true</_undeliv_exist>
        <_undeliv_sms_date>2024-04-22T10:00:02.62Z</_undeliv_sms_date>
        <_undeliv_sms_exist>true</_undeliv_sms_exist>
        <_updatetime>2024-04-18T12:36:31.7Z</_updatetime>
        <_xxh>507a01069df7046e</_xxh>
        <_xxm>7902</_xxm>
        <db_id>1</db_id>
        <email>example@example.com</email>
        <id>6613c5c94a5d12bd6cc09171</id>
        <phones>+79006574081</phones>
        <subscriptions>
            <channel>email</channel>
```

```
            <email_data>
                <domain>example.com</domain>
                <domain_group>other</domain_group>
                <email>example@example.com</email>
                <md5>c5278382ce8972b4cd23b447c0e58989</md5>
            </email_data>
            <hash_id>fcc69fe5</hash_id>
            <is_delete>false</is_delete>
            <priority>0</priority>
            <reg_info>
                <date>2024-04-10T09:28:55.706Z</date>
            </reg_info>
            <resource_id>24</resource_id>
            <status>subscribed</status>
        </subscriptions>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Main fields

Parameter	Type	Description
`error`	**int**	**Error code. 0 — successful operation**
`error_text`	**string**	**Error text**
`data`	**object**	**Profile data**

data object

Contains information about the profile: contact data, geolocation, data on interactions with messages,

Parameter	Type	Description
`id`	**string**	**Unique profile identifier (MongoDB ObjectId)**
`db_id`	**int**	**Database identifier**
`email`	**string**	**Profile email address**
`phones`	**array**	**Profile phone numbers**
`_status`	**int**	**Global subscription status**
`_is_test`	**bool**	**Test profile flag. Returns `true` if the profile was created in test mode**
`gcg_is_exists`	**bool**	**Duplicate of the `_flags.gcg_is_exists` flag: whether the profile is currently in the global control group**
`_fname`	**string**	**First name**
`_lname`	**string**	**Last name**
`_bdate`	**string**	**Birth date in RFC 3339 format**
`_sex`	**int**	**Gender (according to the gender identities list)**
`_avatar_set`	**bool**	**Indicates whether the profile avatar is set**
`_city`	**string**	**City. Returned when geolocation data is available**
`_country`	**string**	**Country in TLD format. Returned when geolocation data is available**
`_region`	**string**	**Region. Returned when geolocation data is available**
`_postal_code`	**string**	**Postal code. Returned when geolocation data is available**
`_lat`	**float**	**Geographic latitude. Returned when geolocation data is available**
`_lon`	**float**	**Geographic longitude. Returned when geolocation data is available**
`_tz`	**string**	**Time zone**
`_regdate`	**string**	**Profile registration date in RFC 3339 format**
`_createtime`	**string**	**Profile creation time in RFC 3339 format**
`_subscribetime`	**string**	**Subscription time in RFC 3339 format**
`_updatetime`	**string**	**Time of the last profile update in RFC 3339 format**
`_ip`	**string**	**IP address. Returned when tracking data is available**
`_browser`	**string**	**Browser. Returned when tracking data is available**
`_os`	**string**	**Operating system. Returned when tracking data is available**
`_platform`	**string**	**Platform. Returned when tracking data is available**
`_is_desktop`	**bool**	**Indicates whether a desktop device is used. Returned when tracking data is available**
`_is_mobile`	**bool**	**Indicates whether a mobile device is used. Returned when tracking data is available**
`_emaildomain`	**string**	**Email address domain**
`_domaingroup`	**string**	**Domain group**
`_acid`	**string**	**Unique Altcraft identifier**
`_md5`	**string**	**MD5 hash of the email address**
`_xxh`	**string**	**XXH hash profile identifier**
`_xxm`	**int**	**XXH hash profile identifier (numeric representation)**
`_send_exist`	**bool**	**Indicates whether an email was sent**
`_send_date`	**string**	**Date of the last email send in RFC 3339 format. Returned when a send exists**
`_deliv_exist`	**bool**	**Indicates whether an email was delivered**
`_deliv_date`	**string**	**Date of the last email delivery in RFC 3339 format. Returned when a delivery exists**
`_open_exist`	**bool**	**Indicates whether an email was opened**
`_open_date`	**string**	**Date of the last email open in RFC 3339 format. Returned when an open exists**
`_read_exist`	**bool**	**Indicates whether an email was read**
`_read_date`	**string**	**Date of the last email read in RFC 3339 format. Returned when a read exists**
`_click_exist`	**bool**	**Indicates whether a link in an email was clicked**
`_click_date`	**string**	**Date of the last link click in an email in RFC 3339 format. Returned when a click exists**
`_confirm_exist`	**bool**	**Indicates whether the subscription was confirmed**
`_confirm_date`	**string**	**Date of the last subscription confirmation in RFC 3339 format. Returned when a confirmation exists**
`_reply_exist`	**bool**	**Indicates whether an email was replied to**
`_reply_date`	**string**	**Date of the last email reply in RFC 3339 format. Returned when a reply exists**
`_sbounce_exist`	**bool**	**Indicates whether a soft bounce (temporary delivery failure) occurred**
`_sbounce_date`	**string**	**Date of the last soft bounce in RFC 3339 format. Returned when a soft bounce exists**
`_undeliv_exist`	**bool**	**Indicates whether an email was not delivered**
`_undeliv_date`	**string**	**Date of the last email non-delivery in RFC 3339 format. Returned when a non-delivery exists**
`_pixel_open_exist`	**bool**	**Indicates whether the tracking pixel fired**
`_pixel_date`	**string**	**Date of the last pixel firing in RFC 3339 format. Returned when a firing exists**

Parameter	Type	Description
`_send_sms_exist`	**bool**	**Indicates whether an SMS was sent**
`_send_sms_date`	**string**	**Date of the last SMS send in RFC 3339 format. Returned when a send exists**
`_deliv_sms_exist`	**bool**	**Indicates whether an SMS was delivered**
`_deliv_sms_date`	**string**	**Date of the last SMS delivery in RFC 3339 format. Returned when a delivery exists**
`_click_sms_exist`	**bool**	**Indicates whether a link in an SMS was clicked**
`_click_sms_date`	**string**	**Date of the last link click in an SMS in RFC 3339 format. Returned when a click exists**
`_undeliv_sms_exist`	**bool**	**Indicates whether an SMS was not delivered**
`_undeliv_sms_date`	**string**	**Date of the last SMS non-delivery in RFC 3339 format. Returned when a non-delivery exists**
`_send_push_exist`	**bool**	**Indicates whether a push notification was sent**
`_send_push_date`	**string**	**Date of the last push notification send in RFC 3339 format. Returned when a send exists**
`_open_push_exist`	**bool**	**Indicates whether a push notification was opened**
`_open_push_date`	**string**	**Date of the last push notification open in RFC 3339 format. Returned when an open exists**
`_read_push_exist`	**bool**	**Indicates whether a push notification was read**
`_read_push_date`	**string**	**Date of the last push notification read in RFC 3339 format. Returned when a read exists**
`_deliv_push_exist`	**bool**	**Indicates whether a push notification was delivered**
`_deliv_push_date`	**string**	**Date of the last push notification delivery in RFC 3339 format. Returned when a delivery exists**
`_click_push_exist`	**bool**	**Indicates whether a push notification was clicked**
`_click_push_date`	**string**	**Date of the last push notification click in RFC 3339 format. Returned when a click exists**
`_ps_wf`	**array**	**Identifiers of marketing scenarios the profile participated in**
`_ps_wfs`	**array of objects**	**Details of participation in marketing scenarios (see the _ps_wfs object)**
`_segs`	**array**	**Identifiers of static segments the profile belongs to**
`_flags`	**object**	**Profile flags (see the _flags object)**
`subscriptions`	**array of objects**	**Profile subscriptions to resources (see the subscriptions array)**
`unsub`	**object**	**Profile unsubscription data (see the unsub object)**

NOTE

In addition to standard fields, custom database fields may be present in the response. Their names ar of the specific database. Information about database fields can be obtained via `/api/v1.1/databases/fiel`

subscriptions array

Contains profile subscriptions to resources. One object is one subscription.

Parameter	Type	Description
`resource_id`	**int**	**Resource identifier**
`channel`	**string**	**Channel type (`email`, `sms`, `push`, `whatsapp`, `viber`, `telegram_bot`, `notify`)**
`email_data`	**object**	**Email subscription data. Returned for the `email` channel**
`phone_data`	**object**	**SMS subscription data. Returned for the `sms` channel**
`priority`	**int**	**Subscription priority**
`status`	**string**	**Subscription status**
`reg_info`	**object**	**Subscription registration information**
`hash_id`	**string**	**Subscription hash identifier**
`is_delete`	**bool**	**Indicates whether the subscription is deleted**

email_data — Email subscription data

Parameter	Type	
`email`	**string**	**Email address**
`domain`	**string**	**Email domain**
`domain_group`	**string**	**Domain group**
`md5`	**string**	**Email MD5 hash**

phone_data — SMS subscription data

Parameter	Type	Des
`phone`	**string**	**Phone number**
`md5`	**string**	**Phone MD5 hash**

reg_info — Subscription registration information

Parameter	Type	Description
`date`	**string**	**Registration date in RFC 3339 format**

unsub object

Contains profile unsubscription data. Returned when unsubscription data is available.

Parameter	Type	Description
`c`	**int**	**Number of unsubscriptions**
`dt`	**string**	**Date of the last unsubscription in RFC 3339 format**
`m`	**int**	**Identifier of the message the profile unsubscribed from**

_flags object

Contains profile flags: participation statuses in the global control group and channel action flags.

Parameter	Type	Description
`channel_actions.email`	**object**	**Email channel action flags (** `send` , `deliv` , `open` , `read` , `click` , `confirm` , `confirmsub` , `reply` , `unsub` **)**
`channel_actions.push`	**object**	**Push channel action flags (** `send` , `deliv` , `open` , `click` **)**
`channel_actions.sms`	**object**	**SMS channel action flags (** `send` , `deliv` , `click` , `unsub` **)**
`gcg_is_exists`	**bool**	**Indicates whether the profile is currently in the global control group**
`gcg_ca_is_exists`	**bool**	**Indicates whether the profile is in the global control group audience**
`gcg_once_is_exists`	**bool**	**Indicates whether the profile has ever been in the global control group**
`gcg_is_from_file`	**bool**	**Indicates whether the profile was added to the global control group using the “from file” method**
`scamp_once_exist`	**bool**	**Indicates whether the profile has ever been in a campaign audience**
`web_actions.pixel_open`	**bool**	**Indicates whether the web pixel fired**

_ps_wfs object (array element)

Contains information about the profile’s participation in marketing automation scenarios.

Parameter	Type	Description
`wId`	**int**	**Automation scenario identifier**
`di`	**string**	**Date the profile entered the scenario in RFC 3339 format**
`do`	**string**	**Date the profile exited the scenario in RFC 3339 format. Returned when the participation is completed**
`et`	**string**	**Reason for exiting the scenario**
`ln`	**int**	**Number of the last passed step (node) of the scenario**
`s`	**int**	**Profile participation status in the scenario**

Uploading profiles to a file

Description

Retrieves information about all database or segment profiles.
To get the result, you need to check the task status and download the resulting file.

Step 1. Creating profiles export task

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/get_all`

Request parameters

Parameter	Type	Example	Required	
source_type	**string**	**“database”** **“segment”**	**Yes**	**Data source type**
source_id	**int**	**1**	**Yes**	**Data source ID**
file_format	**string**	**“csv”**	**No**	**File format for ex** **Default is csv.**
custom_request	**JSON object**	`{"occupation":"doctor"}`	**No**	**Profile fields for**

Request example

JSON

```
{
    "source_type": "database",
    "source_id": 1,
    "file_format": "csv"
}
```

XML

```
<xml>
  <source_type>database</source_type>
  <source_id>1</source_id>
  <file_format>csv</file_format>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
task_id	**string**	**Export task ID**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 2. Checking export status and retrieving link

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON

```
{
    "task_id": "task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5"
}
```

XML

```
<xml>
  <task_id>task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "success",
        "progress": 100,
        "result": {
            "source_id": 1,
            "total": 10,
            "type": "database",
            "urls": [
                "/api/download/6b248de2-39b9-4715-9720-5f4e23f4a296/1_database_export.csv"
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
<data>
    <status>success</status>
    <progress>100</progress>
    <result>
      <source_id>1</source_id>
      <total>10</total>
      <type>database</type>
      <urls>/api/download/6b248de2-39b9-4715-9720-5f4e23f4a296/1_database_export.csv</urls>
    </result>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Desc
status	**string**	**Task status**
progress	**int**	**Task progress**
total	**int**	**Number of profiles**
urls	**array**	**Download link**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 3. Downloading the file

The link to download the file looks like following:

`https://example.com/api/download/6b248de2-33b9-4715-9720-5f7e23f4a296/1_database_export.csv` **, where:**

- `https://example.com` — **platform domain**
- `api/download/6b248de2-33b9-4715-9720-5f7e23f4a296/1_database_export.csv` — **link from the** `urls` **parameter of t**

To download the file with profiles, send a GET request to the specified link with the authorization head

Delete profile

Description

This request deletes customer profile data, including its events history and relations.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/delete`

Request parameters

Parameter	Type	Example	Required	Description
db_id	**int**	**1**	**Yes**	**Database ID**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/user-guide/profiles-and-data Values are recorded as a string: "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

CAUTION

**Special permission is necessary for deleting profiles. It can be configured in user role, and this role c
can read more about roles [in this article](#).**

Request example

JSON

```
{
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com"
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <matching>email</matching>
  <email>example@example.com</email>
</xml>
```

Response example

JSON

```
{
    "data": {
        "profile_id": "5f33d33ece94481aa9382110"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <profile_id>5f33d33ece94481aa9382110</profile_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
data	**JSON object**	**Profile data**
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile identifier**

Subscription fields functional update

When you use requests to update profile's subscription fields, you can include functions in field values. A function here is a set of an operator and a value:

```
{
    "action": "set",
    "value": "x",
}
```

`action` operator can be:

action	Usage	Field type
set	**Set is a default operator for request parameters**	**All types: string, int, float, boolean**
delete	**Deletes a field from a profile. For custom fields only.**	**All types: string, int, float, boolean**
incr	**Increases field value. Integer type fields only**	**int, float**

TIP

You can find a description of the resource subscription fields at this link.

Example request /v1.1/subscriptions/set to update subscription system fields

JSON

```
{
    "channel": "email",
    "db_id": 1,
    "email": "example@example.com",
    "profile_id": "6527e4e2097c54377ee0e41d",
    "resource_id": 1,
    "status": "subscribe",
     "cats": [
        "category_1",
        "category_2"
    ],
    "custom_fields": {
        "_ad_id": {
            "action":"set",
            "value":"qwerty123"
        },
        "_ad_track":{
            "action":"set",
            "value":true
        },
        "_app_id": {
            "action":"delete"
        },
        "_app_iid": {
            "action":"set",
            "value":"example iid"
        },
        "_app_ver": {
            "action":"set",
            "value": "v1.2.3.4"
        },
        "_browser_name":{
            "action":"set",
            "value":"Chrome"
        },
        "_browser_ver": {
            "action":"set",
            "value":"4.5.6.7.8"
        },
        "_device_model": {
            "action":"set",
            "value":"Device model"
        },
        "_device_name":{
            "action":"set",
            "value":"Device name"
        },
        "_device_type": {
            "action":"set",
            "value":"Mobile"
        },
        "_os": {
            "action":"set",
            "value":"Android"
        },
        "_os_language": {
            "action":"set",
            "value":"ru"
        },
        "_os_tz": {
            "action":"set",
            "value":"Europe/Moscow"
        },
        "_os_ver": {
            "action":"set",
            "value": "v1.2.3.4"
        }
    }
}
```


XML

```
<xml>
    <channel>email</channel>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <profile_id>66102b984a5d12bd6cc09132</profile_id>
    <resource_id>29</resource_id>
    <status>subscribe</status>
    <cats>category_1</cats>
    <cats>category_2</cats>
    <custom_fields>
        <_ad_id>
            <action>set</action>
            <value>qwerty123</value>
        </_ad_id>
        <_ad_track>
            <action>set</action>
            <value>true</value>
        </_ad_track>
        <_app_id>
            <action>delete</action>
        </_app_id>
        <_app_iid>
            <action>set</action>
            <value>example iid</value>
        </_app_iid>
        <_app_ver>
            <action>set</action>
            <value>v1.2.3.4</value>
        </_app_ver>
        <_browser_name>
            <action>set</action>
            <value>Chrome</value>
        </_browser_name>
        <_browser_ver>
            <action>set</action>
            <value>4.5.6.7.8</value>
        </_browser_ver>
        <_device_model>
            <action>set</action>
            <value>Device model</value>
        </_device_model>
        <_device_name>
            <action>set</action>
            <value>Device name</value>
        </_device_name>
        <_device_type>
            <action>set</action>
            <value>Mobile</value>
        </_device_type>
        <_os>
            <action>set</action>
            <value>Android</value>
        </_os>
        <_os_language>
            <action>set</action>
            <value>ru</value>
        </_os_language>
        <_os_tz>
            <action>set</action>
            <value>Europe/Moscow</value>
        </_os_tz>
        <_os_ver>
            <action>set</action>
            <value>v1.2.3.4</value>
        </_os_ver>
    </custom_fields>
</xml>
```

Example request /v1.1/subscriptions/set to update custom subscription fields

JSON

```
{
    "channel": "email",
    "db_id": 1,
    "email": "example@example.com",
    "profile_id": "6527e4e2097c54377ee0e41d",
    "resource_id": 1,
    "status": "subscribe",
    "cats": [
        "category_1",
        "category_2"
    ],
    "custom_fields": {
        "custom_str_field": {
            "action": "set",
            "value": "example value"
        },
        "custom_int_field_1": {
            "action": "incr",
            "value": 1000
        },
        "custom_int_field_2": {
            "action": "incr",
            "value": -1000
        },
        "custom_bool_field": {
            "action": "delete"
        }
    }
}
```

XML

```
<xml>
    <channel>email</channel>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
    <resource_id>29</resource_id>
    <status>subscribe</status>
    <cats>category_1</cats>
    <cats>category_2</cats>
    <custom_fields>
        <custom_str_field>
            <action>set</action>
            <value>example value</value>
        </custom_str_field>
        <custom_int_field_1>
            <action>incr</action>
            <value>1000</value>
        </custom_int_field_1>
        <custom_int_field_2>
            <action>incr</action>
            <value>-1000</value>
        </custom_int_field_2>
        <custom_bool_field>
            <action>delete</action>
        </custom_bool_field>
    </custom_fields>
</xml>
```

Example request /v1.1/profiles/import to update subscription system fields

JSON

```
{
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "data": {
        "_fname": "John",
        "_lname": "Doe",
        "email": "example@example.com",
        "subscriptions": [
            {
                "channel": "push",
                "subscription_id": "9f135a38-aa7e-4bf3-8d75-de971d597cf0",
                "provider": "MR",
                "resource_id": 1,
                "custom_fields": {
                    "_ad_id": {
                        "action": "set",
                        "value": "qwerty123"
                    },
                    "_ad_track": {
                        "action": "set",
                        "value": true
                    },
                    "_app_id": {
                        "action": "set",
                        "value": "123"
                    },
                    "_app_iid": {
                        "action": "set",
                        "value": "123"
                    },
                    "_app_ver": {
                        "action": "set",
                        "value": "v1.2.3.4"
                    },
                    "_browser_name": {
                        "action": "set",
                        "value": "Chrome"
                    },
                    "_browser_ver": {
                        "action": "set",
                        "value": "4.5.6.7.8"
                    },
                    "_device_model": {
                        "action": "set",
                        "value": "Device model"
                    },
                    "_device_name": {
                        "action": "set",
                        "value": "Device name"
                    },
                    "_device_type": {
                        "action": "set",
                        "value": "Mobile"
                    },
                    "_os": {
                        "action": "set",
                        "value": "Android"
                    },
                    "_os_language": {
                        "action": "set",
                        "value": "ru"
                    },
                    "_os_tz": {
                        "action": "set",
                        "value": "Europe/Moscow"
                    },
                    "_os_ver": {
                        "action": "set",
                        "value": "v1.2.3.4"
                    }
                }
            }
        ]
    }
}
```


XML

```
<xml>
    <db_id>1</db_id>
    <matching>email</matching>
    <email>example@example.com</email>
    <data>
        <_fname>Fillippina</_fname>
        <_lname>Doe</_lname>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>push</channel>
            <subscription_id>9f135a38-aa7e-4bf3-8d75-de971d597cf0</subscription_id>
            <provider>MR</provider>
            <resource_id>26</resource_id>
            <custom_fields>
                <_ad_id>
                    <action>set</action>
                    <value>qwerty123</value>
                </_ad_id>
                <_ad_track>
                    <action>set</action>
                    <value>true</value>
                </_ad_track>
                <_app_id>
                    <action>delete</action>
                </_app_id>
                <_app_iid>
                    <action>set</action>
                    <value>example iid</value>
                </_app_iid>
                <_app_ver>
                    <action>set</action>
                    <value>v1.2.3.4</value>
                </_app_ver>
                <_browser_name>
                    <action>set</action>
                    <value>Chrome</value>
                </_browser_name>
                <_browser_ver>
                    <action>set</action>
                    <value>4.5.6.7.8</value>
                </_browser_ver>
                <_device_model>
                    <action>set</action>
                    <value>Device model</value>
                </_device_model>
                <_device_name>
                    <action>set</action>
                    <value>Device name</value>
                </_device_name>
                <_device_type>
                    <action>set</action>
                    <value>Mobile</value>
                </_device_type>
                <_os>
                    <action>set</action>
                    <value>Android</value>
                </_os>
                <_os_language>
                    <action>set</action>
                    <value>ru</value>
                </_os_language>
                <_os_tz>
                    <action>set</action>
                    <value>Europe/Moscow</value>
                </_os_tz>
                <_os_ver>
                    <action>set</action>
                    <value>v1.2.3.4</value>
                </_os_ver>
            </custom_fields>
        </subscriptions>
    </data>
</xml>
```

Example request /v1.1/profiles/import to update custom subscription fields

JSON

```
{
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "data": {
        "_fname": "John",
        "_lname": "Doe",
        "email": "example@example.com",
        "subscriptions": [
            {
                "channel": "email",
                "channel_sid": "email",
                "email": "example@example.com",
                "resource_id": 1
            },
            {
                "channel": "sms",
                "phone": "+79203214111",
                "resource_id": 1
            },
            {
                "channel": "push",
                "subscription_id": "9f135a38-aa7e-4bf3-8d75-de971d597cf0",
                "provider": "MR",
                "resource_id": 1,
                "custom_fields": {
                    "custom_str_field": {
                        "action": "set",
                        "value": "example value"
                    },
                    "custom_int_field": {
                        "action": "incr",
                        "value": 1000
                    },
                    "custom_bool_field": {
                        "action": "delete"
                    }
                }
            }
        ]
    }
}
```


XML

```
<xml>
    <db_id>1</db_id>
    <matching>email</matching>
    <email>example@example.com</email>
    <data>
        <_fname>Fillippina</_fname>
        <_lname>Doe</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <channel>email</channel>
            <channel_sid>email</channel_sid>
            <email>example@example.com</email>
            <resource_id>1</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>sms</channel>
            <phone>+79203214111</phone>
            <resource_id>1</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>push</channel>
            <subscription_id>9f135a38-aa7e-4bf3-8d75-de971d597cf0</subscription_id>
            <provider>MR</provider>
            <resource_id>1</resource_id>
            <custom_fields>
                <custom_str_field>
                    <action>set</action>
                    <value>example value</value>
                </custom_str_field>
                <custom_int_field>
                    <action>incr</action>
                    <value>1000</value>
                </custom_int_field>
                <custom_bool_field>
                    <action>delete</action>
                </custom_bool_field>
            </custom_fields>
        </subscriptions>
    </data>
</xml>
```

Database fields functional update

When you use requests to update profile fields, you can include functions in field values.
A function here is a set of an operator and a value:

```
{
    "action": "set",
    "value": "x",
}
```

Action operator can be:

action	Usage	Field t
set	**Set is a default operator for request parameters**	*
unset	**Deletes a field from a profile (for all fields except `email`)**	*
incr	**Increases field value. Integer type fields only**	**int, float**
add	**Adds tags to a profile field**	**tags, arr_object**
delete	**Deletes tags from a profile field**	**tags, arr_object**
upsert	**Sets the value from the *value* field to the property specified in the *submatch* field**	**object**

INFO

For `object` and `arr_object` data types, you must pass the `value` and `submatch` fields.
`submatch` is the path to the property whose value needs to be updated.

Request examples

profiles/update increases one value and decreases the other. The same operator with different value si

JSON

```
{
    "db_id": 24,
    "profile_id": "5a5b64a969d4265b958b9ac3",
    "data": {
        "checks": {
            "action": "incr",
            "value": 1000
        },
        "returns": {
            "action": "incr",
            "value": -1000
        }
    }
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <profile_id>6613c4754a5d12bd6cc0915f</profile_id>
  <data>
    <checks>
      <action>incr</action>
      <value>1000</value>
    </checks>
    <returns>
      <action>incr</action>
      <value>-1000</value>
    </returns>
  </data>
</xml>
```

profiles/update adds tags to profile “category” field:

JSON

```
{
    "db_id": 1,
    "profile_id": "5a5b64a969d4265b958b9ac3",
    "data": {
        "category": {
            "action": "add",
            "value": [
                "cars",
                "cats"
            ]
        }
    }
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
  <data>
    <category>
      <action>add</action>
      <value array='true'>cars</value>
      <value>cats</value>
    </category>
  </data>
</xml>
```

An example of a profiles/update request that will update the “x” field in an object:

JSON

```
{
  "db_id": 1,
  "profile_id": "6613c4bf4a5d12bd6cc0916b",
  "data": {
    "myObject": {
       "action": "upsert",
       "value": "y",
       "submatch": "x"
    }
  }
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <profile_id>66102b984a5d12bd6cc0912f</profile_id>
    <data>
        <myObject>
            <action>upsert</action>
            <value>y</value>
            <submatch>x</submatch>
        </myObject>
    </data>
</xml>
```

profiles/update sets a new value for all elements of the object that contain the specified property (subı

Request

JSON

```
{
    "db_id": 1,
    "profile_id": "66102b984a5d12bd6cc0912f",
    "data": {
        "myObject": {
            "action": "upsert",
            "value": {
                "x": "y",
                "z": 1
            },
            "submatch": "x"
        }
    }
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <profile_id>66102b984a5d12bd6cc09132</profile_id>
  <data>
    <myObject>
      <action>upsert</action>
      <value>
        <x>y</x>
        <z>1</z>
      </value>
      <submatch>x</submatch>
    </myObject>
  </data>
</xml>
```

Object

JSON

```
{
    {
       "A": 1,
       "x": false,
    },
    {
       "B": "test"
    },
    {
       "x": "test",
       "C": true
    }
 }
```

XML

```
<xml>
    <value>
        <A>1</A>
        <x>false</x>
    </value>
    <value>
        <B>test</a>
    </value>
    <value>
        <x>test</x>
        <c>true</c>
    </value>
</xml>
```

Updated object

JSON

```
{
    {
       "A": 1,
       "x": {"x": "y", "z": 1},
    },
    {
       "B": "test"
    },
    {
       "x": {"x": "y", "z": 1},
       "C": true
    }
}
```

XML

```
<xml>
    <value>
        <A>1</A>
        <x>
            <m>y</m>
            <z>1</z>
        </x>
    </value>
    <value>
        <B>test</B>
    </value>
    <value>
        <x>
            <m>y</m>
            <z>1</z>
        </x>
        <c>true</c>
    </value>
</xml>
```

INFO

If fields with type `object` and `object_arr` do not have a property that you want to change, this property

Merging multiple profiles

Description

Combining multiple profiles into a new profile or existing one.

> **CAUTION**
>
> - **The profiles generated by merging are not imports. Import trigger does not work on them.**
> - **Email: localhost - as an absent field, it is created if necessary and is not copied to another profile**
> - **Not localhost email is copied like any other field, but in order to avoid any conflicts, it disappears to empty (that is, localhost).**

URL

Method: POST

`https://example.com/api/v1.1/profiles/merge`

Request parameters

Parameter	Type	Example	Required	Descri

data	**JSON object**	"data":{
		"_fname": "John",
		"_lname": "Doe"
		}
			No	**Profile data that will change after merging**

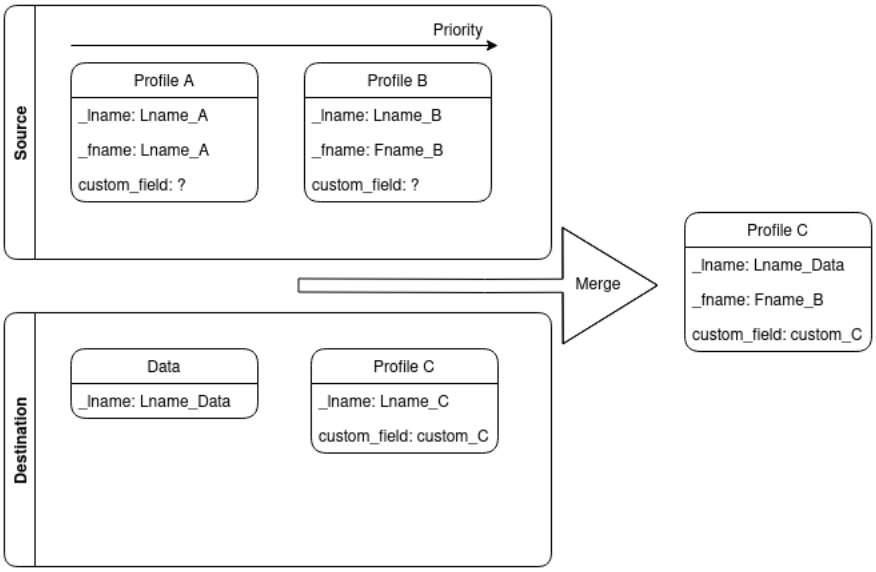
db_id	**int**	**1**	**Yes**	**Database ID**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events By default – false**

source	**array**	"source": [
		{
		"db_id": 1,
		"email":
		"test1@mail.ru"
		},
		{
		"db_id": 2,
		"email":
		"test2@mail.ru"
		}
		]
			Yes	**Contains matching data about the profiles to be merged**

destination	**JSON object**	"destination":{
		"db_id": 3,
		"email":
		"test3@mail.ru"
		}
			Yes	**Contains the matching data of the profile resulting from the**

matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email from the profile or subscriptions**	**Profile matching mode By default - email For each type of matching, certain fields must be passed in**
field_name	**string**	**custom_ID**	**Yes, if "matching": "custom"**	**Name of custom database field if matching : custom . In the required for search.**
field_value	**int/string**	**custom_string**	**Yes, if "matching": "custom"**	**Value of custom database field if matching : custom .**
subscriptions	**bool**	**true**	**No**	**Copy and merge subscriptions. Subscriptions to resource ca copied. By default – true**
history	**bool**	**true**	**No**	**Copy history and activity flags By default – true**
static_segments	**bool**	**true**	**No**	**Whether to make changes to static segments or not (source will enter) By default – true**
overwrite	**bool**	**true**	**No**	**Whether to copy the fields of source profiles or not. If the destination profile has empty fields, in any case, thes which is the highest.**
delete	**bool**	**true**	**No**	**Whether to remove source profiles after merging or not**

Copying profile fields



Request example

JSON

```
{
    "matching": "custom",
    "field_name": "field_1",
    "source": [
        {
            "db_id": 1,
            "field_value": "1",
            "data": {
                "_fname": "Profile1",
                "_lname": "LnameProfile1"
            }
        },
        {
            "db_id": 1,
            "matching": "email",
            "email": "test2@mail.ru",
            "data": {
                "_fname": "Profile2",
                "_lname": "LnameProfile2"
            }
        }
    ],
    "destination": {
        "db_id": 2,
        "matching": "email",
        "email": "test3@mail.ru",
        "data": {
            "_fname": "Profile3",
            "_lname": "LnameProfile3"
        }
    },
    "subscriptions": true,
    "static_segments": true,
    "history": false
}
```

XML

```
<xml>
    <matching>custom</matching>
    <field_name>field_1</field_name>
    <source>
        <db_id>1</db_id>
        <field_value>1</field_value>
        <data>
            <_fname>Profile1</_fname>
            <_lname>LnameProfile1</_lname>
        </data>
    </source>
    <source>
        <db_id>1</db_id>
        <matching>email</matching>
        <email>test2@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </source>
    <destination>
        <db_id>2</db_id>
        <matching>email</matching>
        <email>test3@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </destination>
    <subscriptions>true</subscriptions>
    <static_segments>true</static_segments>
    <history>false</history>
</xml>
```

Response example

JSON

```
{
    "destination_id": {
        "db_id": 1,
        "profile_id": "6613cb65dc4594c2934db63b"
    },
    "error": 0,
    "error_text": "",
    "source_ids": [
        {
            "db_id": 1,
            "profile_id": "6613cb65dc4594c2934db63f"
        },
        {
            "db_id": 1,
            "profile_id": "6613cb65dc4594c2934db63b"
        }
    ]
}
```

XML

```
<xml>
    <destination_id>
        <db_id>2</db_id>
        <profile_id>60a7855930b8bc8535b2acfb</profile_id>
    </destination_id>
    <error>0</error>
    <error_text></error_text>
    <source_ids>
        <db_id>1</db_id>
        <profile_id>60a7839330b8bc86a5bc5f62</profile_id>
    </source_ids>
    <source_ids>
        <db_id>1</db_id>
        <profile_id>60a7845230b8bc8535b2acf7</profile_id>
    </source_ids>
</xml>
```

Response parameters

Parameter	Type	Description
destination_id	**JSON object**	**Contains data about the profile resulting from the merge:** `db_id` **- database ID** `profile_id` **- profile ID**
source_ids	**JSON object**	**Contains data about merged profiles as a list:** `db_id` **- database ID** `profile_id` **- profile ID**
error	**int**	**Error code**
error_text	**string**	**Error text**

Unsubscribe profile from resource

Description

Sends the information of the profile unsubscribtion to the platform. The subscription status changes to allows to unsubscribe a profile from certain categories of resource.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/message_unsub`

Request parameters

Parameter	Type	Example	Required	
rtoken	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**Resour**
smid	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**ID of th**
categories	**array**	**[“cat_to_unsub”]**	**No**	**Resour**

Request example

JSON

```
{
  "rtoken":"2fHLS9YR9daS-a90b371eecc2952f",
  "smid": "fGmXiWU9Y4NSIpCwl4QG6QXAAAQEQsQQQEQEAABCRA",
  "categories": ["cat_to_unsub"]
}
```

XML

```
<xml>
  <rtoken>2fHLS9YR9daS-a90b371eecc2952f</rtoken>
  <smid>fGmXiWU9Y4NSIpCwl4QG6QXAAAQEQsQQQEQEAABCRA</smid>
  <categories>cat_to_unsub</categories>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "profile_id": "658997338e8f4032d92fd56d"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>658997338e8f4032d92fd56d</profile_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile ID**

Profile splitting

Description

Splitting a profile into multiple new or existing profiles.

> **CAUTION**
>
> **The profiles generated by splitting are not imports. Import trigger does not work on them.**

URL

Method: POST

`https://example.com/api/v1.1/profiles/split`

Request parameters

Parameter	Type	Example	Required	Desc

data	**JSON object**	“data”:{
		“_fname”: “John”,
		“_lname”: “Doe”
		}
			No	**Profile data that will change after merging**

skip_triggers	**bool**	**true**	**No**	**Skip trigger events By default – false**

source	**JSON object**	“source”:{
		“db_id”: 1,
		“email”: “test1@mail.ru”
		}
			Yes	**Contains matching data about the profiles to be split**

destination	**array**	“destination”: [
		{
		“db_id”: 3,
		“email”:
		“test3@mail.ru”
		}
		]
			Yes	**Contains the matching data of the profile resulting from**

matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No, if matched by email from the profile or subscriptions**	**Profile matching mode By default - email For each type of matching, certain fields must be passe**
field_name	**string**	**custom_ID**	**Yes, if “matching”: “custom”**	**Name of custom database field if matching : custom . In required for search.**
field_value	**int/string**	**custom_string**	**Yes, if “matching”: “custom”**	**Value of custom database field if matching : custom .**
subscriptions	**bool**	**true**	**No**	**Copy and split subscriptions. Subscriptions to resource copied. By default – true**
history	**bool**	**true**	**No**	**Copy history and activity flags By default – true**
static_segments	**bool**	**true**	**No**	**Whether to make changes to static segments or not (so profile will enter) By default – false**

subscription_filters	**JSON object**	“subscription_filters”: [
		{
		“resource_id”: 1,
		“channel”: “push”,
		“provider”: “Safari”
		}
		]
			No	**Filter to copy specific subscriptions.
				If not specified, all subscriptions are copied.**

overwrite	**bool**	**true**	**No**	**Whether to copy the fields of source profiles or not. If the destination profiles have empty fields, these fields case.**
delete	**bool**	**true**	**No**	**Whether to delete source profiles after splitting or not**

Request example

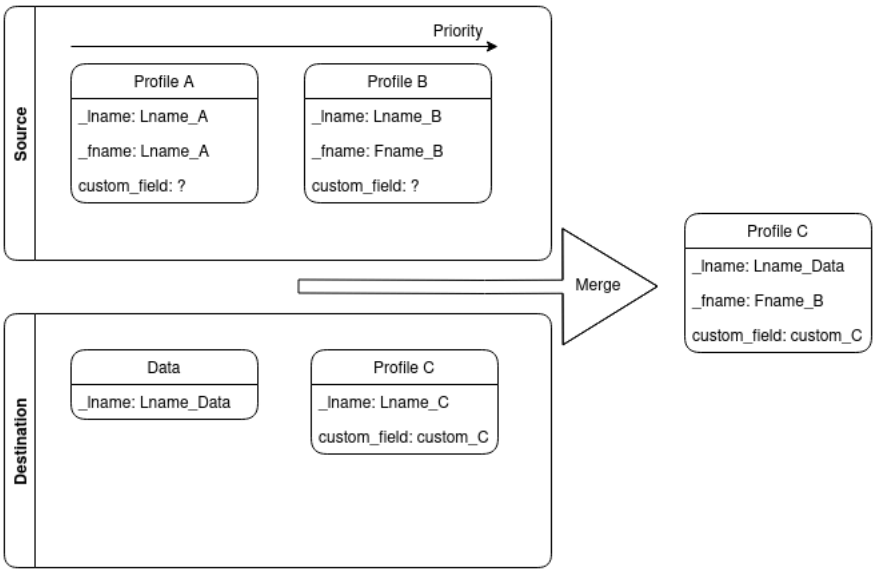
JSON

```
{
    "matching": "custom",
    "field_name": "field_1",
    "source": {
        "db_id": 1,
        "field_value": 1,
        "data": {
            "_fname": "Profile1",
            "_lname": "LnameProfile1"
        }
    },
    "destination": [
        {
            "db_id": 2,
            "matching": "email",
            "email": "test2@mail.ru",
            "data": {
                "_fname": "Profile2",
                "_lname": "LnameProfile2"
            }
        },
        {
            "db_id": 2,
            "field_value": 3,
            "data": {
                "_fname": "Profile3",
                "_lname": "Lname3"
            },
            "subscription_filters": [
                {
                    "resource_id": 1,
                    "channel": "push",
                    "provider": "Safari"
                },
                {
                    "resource_id": 2,
                    "channel": "email"
                }
            ]
        }
    ],
    "subscriptions": true,
    "static_segments": true,
    "history": false
}
```


XML

```
<xml>
    <matching>custom</matching>
    <field_name>field_1</field_name>
    <delete>true</delete>
    <source>
        <db_id>1</db_id>
        <field_value>1</field_value>
        <data>
            <_fname>Profile1</_fname>
            <_lname>LnameProfile1</_lname>
        </data>
    </source>
    <destination>
        <db_id>2</db_id>
        <matching>email</matching>
        <email>test2@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </destination>
    <destination>
        <db_id>2</db_id>
        <field_value>3</field_value>
        <data>
            <_fname>Profile3</_fname>
            <_lname>LnameProfile3</_lname>
        </data>
        <subscription_filters>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <provider>Safari</provider>
        </subscription_filters>
        <subscription_filters>
            <resource_id>2</resource_id>
            <channel>email</channel>
        </subscription_filters>
    </destination>
    <subscriptions>true</subscriptions>
    <static_segments>true</static_segments>
    <history>false</history>
</xml>
```

Copying profile fields



Response example

JSON

```
{
    "destination_ids": [
        {
            "db_id": 2,
            "profile_id": "60a7b150fa3adb6f1179fe7f"
        },
        {
            "db_id": 2,
            "profile_id": "60a7b150fa3adb6f1179fe80"
        }
    ],
    "error": 0,
    "error_text": "",
    "source_id": {
        "db_id": 1,
        "profile_id": "60a7a45bfa3adb7132dcceaa"
    }
}
```


XML


```
<xml>
    <destination_ids>
        <db_id>1</db_id>
        <profile_id>66432f6cee44bb50a6711634</profile_id>
    </destination_ids>
    <destination_ids>
        <db_id>1</db_id>
        <profile_id>6613cb65dc4594c2934db63f</profile_id>
    </destination_ids>
    <error>0</error>
    <source_id>
        <db_id>1</db_id>
        <profile_id>6643405b41c3a674897dd9cc</profile_id>
    </source_id>
</xml>
```


Response parameters

Parameter	Type	Description
destination_ids	**JSON object**	**Contains data about the profile resulting from the split:** **db_id - database ID** **profile_id - profile ID**
source_id	**JSON object**	**Contains data about split profiles as a list:** **db_id - database ID** **profile_id - profile ID**
error	**int**	**Error code**
error_text	**string**	**Error text**

Get data for multiple profiles

Description

Gets data for multiple profiles in the database.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/list`

Request parameters

Parameter	Type	Example	Required	Description
db_id	**int**	**20**	**Yes**	**Database ID**
from_id	**string**	**“0014a2feb331ace2.20200124134127”**	**No**	**The pointer to the next page, taken from the previous request, or passed empty for the first on**
limit	**int**	**3**	**No**	**Limit on the number of entries. Default - 1000**

query	**JSON объект**	“query”:{
		“$or”:[
		{
		“_lname”: “Black”
		},
		{
		“_city”: “Bay City”
		}
		]
		}
			No	**Refinement of the search for profiles by parameters in the form of a query to the database in . MongoDB documentation.
				
				To get a list of fields in the database, use the “Get database fields” request.**

fields	**Array of string**	“fields”: [“_id”, “_lname”, “_fname”, “_city”, “subscriptions”],	**No**	**Contains fields for filtering the displayed profile data.**

Request example

JSON

```
{
    "db_id": 1,
    "from_id": "65117f62736a0e15d273fae9",
    "limit": 1,
    "query": {
        "$or": [
            {
                "_lname": "Doe"
            },
            {
                "_city": "Moscow"
            }
        ]
    },
    "fields": [
        "_id",
        "_lname",
        "_fname",
        "_city",
        "subscriptions"
    ]
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <from_id>65117f62736a0e15d273fae9</from_id>>
    <limit>1</limit>
    <fields>_id</fields>
    <fields>_lname</fields>
    <fields>_fname</fields>
    <fields>_city</fields>
    <fields>subscriptions</fields>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "_city": "Moscow",
            "_fname": "Anna",
            "_lname": "Murphy",
            "id": "65117f62736a0e15d273faee",
            "subscriptions": [
                {
                    "resource_id": 5,
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.edu",
                        "domain": "example.edu",
                        "domain_group": "other",
                        "md5": "7dbeb0466131e377e58988a97196d18e"
                    },
                    "priority": 0,
                    "status": "unsubscribed",
                    "reg_info": {
                        "date": "2024-01-09T12:12:18.172Z"
                    },
                    "hash_id": "e70dbabe",
                    "is_delete": false
                },
                {
                    "resource_id": 26,
                    "channel": "push",
                    "push_data": {
                        "subscription_id": "abcdefghijklmnqrstuvwxyz",
                        "provider": "Firefox"
                    },
                    "priority": 0,
                    "status": "subscribed",
                    "reg_info": {
                        "date": "2024-04-18T12:30:44.592Z"
                    },
                    "hash_id": "e70dbabe",
                    "is_delete": false
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": "65117f62736a0e15d273faf4"
}
```


XML


```
<xml>
    <data>
        <_city>Toronto</_city>
        <_fname>Juan</_fname>
        <_lname>Wallace</_lname>
        <id>65117f62736a0e15d273fae9</id>
        <subscriptions>
            <channel>email</channel>
            <email_data>
                <domain>example.com</domain>
                <domain_group>other</domain_group>
                <email>example@example.com</email>
                <md5>18450b1edfef75279fc4f4f60c78d97e</md5>
            </email_data>
            <hash_id>a0cc2672</hash_id>
            <is_delete>false</is_delete>
            <priority>0</priority>
            <reg_info>
                <date>2024-01-09T12:12:18.172Z</date>
            </reg_info>
            <resource_id>5</resource_id>
            <status>hardbounced</status>
        </subscriptions>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>65117f62736a0e15d273faea</next_from_id>
</xml>
```

Response parameters

Main fields

Parameter	Type	Description
`error`	**int**	**Error code. 0 — successful operation**
`error_text`	**string**	**Error text**
`next_from_id`	**string**	**The value must be passed in the next request to get another piece of data until the result output contains 0 history lines.**
`data`	**array of objects**	**Array of profile objects (see data object)**

data object

Each element of the `data` array contains information about the profile: contact data, geolocation, data and other information.

Parameter	Type	Description
`id`	**string**	**Unique profile identifier (MongoDB ObjectId)**
`db_id`	**int**	**Database identifier**
`email`	**string**	**Profile email address**
`phones`	**array**	**Profile phone numbers**
`_status`	**int**	**Global subscription status**
`_is_test`	**bool**	**Test profile flag. Returns `true` if the profile was created in test mode**
`gcg_is_exists`	**bool**	**Duplicate of the `_flags.gcg_is_exists` flag: whether the profile is currently in the global control group**
`_fname`	**string**	**First name**
`_lname`	**string**	**Last name**
`_bdate`	**string**	**Birth date in RFC 3339 format**
`_sex`	**int**	**Gender (according to the gender identities list)**
`_avatar_set`	**bool**	**Indicates whether the profile avatar is set**
`_city`	**string**	**City. Returned when geolocation data is available**
`_country`	**string**	**Country in TLD format. Returned when geolocation data is available**
`_region`	**string**	**Region. Returned when geolocation data is available**
`_postal_code`	**string**	**Postal code. Returned when geolocation data is available**
`_lat`	**float**	**Geographic latitude. Returned when geolocation data is available**
`_lon`	**float**	**Geographic longitude. Returned when geolocation data is available**
`_tz`	**string**	**Time zone**
`_regdate`	**string**	**Profile registration date in RFC 3339 format**
`_createtime`	**string**	**Profile creation time in RFC 3339 format**
`_subscribetime`	**string**	**Subscription time in RFC 3339 format**
`_updatetime`	**string**	**Time of the last profile update in RFC 3339 format**
`_ip`	**string**	**IP address. Returned when tracking data is available**
`_browser`	**string**	**Browser. Returned when tracking data is available**
`_os`	**string**	**Operating system. Returned when tracking data is available**
`_platform`	**string**	**Platform. Returned when tracking data is available**
`_is_desktop`	**bool**	**Indicates whether a desktop device is used. Returned when tracking data is available**
`_is_mobile`	**bool**	**Indicates whether a mobile device is used. Returned when tracking data is available**
`_emaildomain`	**string**	**Email address domain**
`_domaingroup`	**string**	**Domain group**
`_acid`	**string**	**Unique Altcraft identifier**
`_md5`	**string**	**MD5 hash of the email address**
`_xxh`	**string**	**XXH hash profile identifier**
`_xxm`	**int**	**XXH hash profile identifier (numeric representation)**
`_send_exist`	**bool**	**Indicates whether an email was sent**
`_send_date`	**string**	**Date of the last email send in RFC 3339 format. Returned when a send exists**
`_deliv_exist`	**bool**	**Indicates whether an email was delivered**
`_deliv_date`	**string**	**Date of the last email delivery in RFC 3339 format. Returned when a delivery exists**
`_open_exist`	**bool**	**Indicates whether an email was opened**
`_open_date`	**string**	**Date of the last email open in RFC 3339 format. Returned when an open exists**
`_read_exist`	**bool**	**Indicates whether an email was read**
`_read_date`	**string**	**Date of the last email read in RFC 3339 format. Returned when a read exists**
`_click_exist`	**bool**	**Indicates whether a link in an email was clicked**
`_click_date`	**string**	**Date of the last link click in an email in RFC 3339 format. Returned when a click exists**
`_confirm_exist`	**bool**	**Indicates whether the subscription was confirmed**
`_confirm_date`	**string**	**Date of the last subscription confirmation in RFC 3339 format. Returned when a confirmation exists**
`_reply_exist`	**bool**	**Indicates whether an email was replied to**
`_reply_date`	**string**	**Date of the last email reply in RFC 3339 format. Returned when a reply exists**
`_sbounce_exist`	**bool**	**Indicates whether a soft bounce (temporary delivery failure) occurred**
`_sbounce_date`	**string**	**Date of the last soft bounce in RFC 3339 format. Returned when a soft bounce exists**
`_undeliv_exist`	**bool**	**Indicates whether an email was not delivered**
`_undeliv_date`	**string**	**Date of the last email non-delivery in RFC 3339 format. Returned when a non-delivery exists**

Parameter	Type	Description
`_pixel_open_exist`	**bool**	**Indicates whether the tracking pixel fired**
`_pixel_date`	**string**	**Date of the last pixel firing in RFC 3339 format. Returned when a firing exists**
`_send_sms_exist`	**bool**	**Indicates whether an SMS was sent**
`_send_sms_date`	**string**	**Date of the last SMS send in RFC 3339 format. Returned when a send exists**
`_deliv_sms_exist`	**bool**	**Indicates whether an SMS was delivered**
`_deliv_sms_date`	**string**	**Date of the last SMS delivery in RFC 3339 format. Returned when a delivery exists**
`_click_sms_exist`	**bool**	**Indicates whether a link in an SMS was clicked**
`_click_sms_date`	**string**	**Date of the last link click in an SMS in RFC 3339 format. Returned when a click exists**
`_undeliv_sms_exist`	**bool**	**Indicates whether an SMS was not delivered**
`_undeliv_sms_date`	**string**	**Date of the last SMS non-delivery in RFC 3339 format. Returned when a non-delivery exists**
`_send_push_exist`	**bool**	**Indicates whether a push notification was sent**
`_send_push_date`	**string**	**Date of the last push notification send in RFC 3339 format. Returned when a send exists**
`_open_push_exist`	**bool**	**Indicates whether a push notification was opened**
`_open_push_date`	**string**	**Date of the last push notification open in RFC 3339 format. Returned when an open exists**
`_read_push_exist`	**bool**	**Indicates whether a push notification was read**
`_read_push_date`	**string**	**Date of the last push notification read in RFC 3339 format. Returned when a read exists**
`_deliv_push_exist`	**bool**	**Indicates whether a push notification was delivered**
`_deliv_push_date`	**string**	**Date of the last push notification delivery in RFC 3339 format. Returned when a delivery exists**
`_click_push_exist`	**bool**	**Indicates whether a push notification was clicked**
`_click_push_date`	**string**	**Date of the last push notification click in RFC 3339 format. Returned when a click exists**
`_ps_wf`	**array**	**Identifiers of marketing scenarios the profile participated in**
`_ps_wfs`	**array of objects**	**Details of participation in marketing scenarios (see the _ps_wfs object)**
`_segs`	**array**	**Identifiers of static segments the profile belongs to**
`_flags`	**object**	**Profile flags (see the _flags object)**
`subscriptions`	**array of objects**	**Profile subscriptions to resources (see the subscriptions array)**
`unsub`	**object**	**Profile unsubscription data (see the unsub object)**

NOTE

In addition to standard fields, custom database fields may be present in the response. Their names ar of the specific database. Information about database fields can be obtained via `/api/v1.1/databases/fiel`

NOTE

The request supports the `fields` parameter to filter returned fields. If `fields` is not specified, all avail

subscriptions array

Contains profile subscriptions to resources. One object is one subscription.

Parameter	Type	Description
`resource_id`	**int**	**Resource identifier**
`channel`	**string**	**Channel type (`email` , `sms` , `push` , `whatsapp` , `viber` , `telegram_bot` , `notify`)**
`email_data`	**object**	**Email subscription data. Returned for the `email` channel**
`phone_data`	**object**	**SMS subscription data. Returned for the `sms` channel**
`priority`	**int**	**Subscription priority**
`status`	**string**	**Subscription status**
`reg_info`	**object**	**Subscription registration information**
`hash_id`	**string**	**Hash identifier of the subscription**
`is_delete`	**bool**	**Indicates whether the subscription is deleted**

email_data — Email subscription data

Parameter	Type	Descrip
`email`	**string**	**Email address**
`domain`	**string**	**Email domain**
`domain_group`	**string**	**Domain group**
`md5`	**string**	**MD5 hash of the email**

phone_data — SMS subscription data

Parameter	Type	Description
`phone`	**string**	**Phone number**
`md5`	**string**	**MD5 hash of the phone number**

reg_info — Subscription registration information

Parameter	Type	Description
`date`	**string**	**Registration date in RFC 3339 format**

unsub object

Contains profile unsubscription data. Returned when unsubscription data is available.

Parameter	Type	Description
`c`	**int**	**Number of unsubscriptions**
`dt`	**string**	**Date of the last unsubscription in RFC 3339 format**
`m`	**int**	**Identifier of the campaign from which the unsubscription occurred**

_flags object

Contains profile flags: global control group participation statuses and channel action flags.

Parameter	Type	Description
`channel_actions.email`	**object**	**Email channel action flags (** `send` , `deliv` , `open` , `read` , `click` , `confirm` , `confirmsub` , `reply` , `unsub` **)**
`channel_actions.push`	**object**	**Push channel action flags (** `send` , `deliv` , `open` , `click` **)**
`channel_actions.sms`	**object**	**SMS channel action flags (** `send` , `deliv` , `click` , `unsub` **)**
`gcg_is_exists`	**bool**	**Indicates whether the profile is currently in the global control group**
`gcg_ca_is_exists`	**bool**	**Indicates whether the profile is in the global control group audience**
`gcg_once_is_exists`	**bool**	**Indicates whether the profile was ever in the global control group**
`gcg_is_from_file`	**bool**	**Indicates whether the profile was added to the global control group via “from file” method**
`scamp_once_exist`	**bool**	**Indicates whether the profile was in a campaign audience at least once**
`web_actions.pixel_open`	**bool**	**Indicates whether the web pixel fired**

_ps_wfs object (array element)

Contains information about the profile’s participation in marketing automation scenarios.

Parameter	Type	Description
`wId`	**int**	**Marketing scenario identifier**
`di`	**string**	**Date the profile entered the scenario in RFC 3339 format**
`do`	**string**	**Date the profile exited the scenario in RFC 3339 format. Returned when participation is complete**
`et`	**string**	**Exit reason from the scenario**
`ln`	**int**	**Number of the last completed step (node) in the scenario**
`s`	**int**	**Profile participation status in the scenario**

Add or edit subscription

Description

Adds or edits profile subscriptions. Changes subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/set`

Request parameters

Parameter	Type	Example	Required	Description
db_id	**int**	**1**	**Yes**	**Database ID**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**	**Profile ID**
resource_id	**int**	**1**	**Yes**	**Resource ID**
status	**string**	**"subscribed"**	**No**	**Subscription status.** **We recommend that you pass the parameter only when you need to update the subscripti**
priority	**int**		**No**	**Subscription priority**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**No**	**System and custom subscription fields.** **You can get information about available subscription fields in a resource using the /v1.1/r**
cats	**array**	`[ "category_1", "category_2" ]`	**No**	**Resource categories to which you need to subscribe a profile**
date	**string**	**2018-08-22T00:00:00.000Z**	**No**	**Registration date in RFC 3339 format**
ip	**string**	**10.250.13.5**	**No**	**Registration IP**
url	**string**	**https://example.com/**	**No**	**Registration URL**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false.**

Additionally for the Email channel:

Parameters	Type	Example	
channel	**string**	**“email”**	**Ch**
email	**string**	**“john@example.com”**	**Em**

Additionally for the Push channel:

Parameters	Type	Example
channel	**string**	**“push”**
provider	**string**	**“Firefox”**
subscription_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Additionally for the SMS channel:

Parameters	Type	Example	
channel	**string**	**“sms”**	**Channel**
phone	**string**	**“+79000000000”**	**Phone nu**

Additionally for the Telegram channel:

Parameters	Type	Example	Description
channel	**string**	**“telegram_bot”**	**Channel type**
cc_data	**object**	`{ “id”: 1 }`	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameters	Type	Example	Description
channel	**string**	**“whatsapp”**	**Channel type**
cc_data	**object**	`{ “phone”: “+79000000000” }`	**WhatsApp* user’s phone number**

Request example

JSON

```
{
  "db_id": 1,
  "resource_id": 1,
  "channel": "email",
  "email": "example@example.com",
  "priority": 10,
  "profile_id": "abcdefghijklmnqrstuvwxyz",
  "status": "suspended"
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <resource_id>29</resource_id>
  <channel>email</channel>
  <email>example@example.com</email>
  <priority>10</priority>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <status>suspended</status>
</xml>
```

Response example

JSON

```
{
  "data": {
    "action": "subscribed",
    "is_new_channel": false,
    "is_new_contact": false,
    "is_new_subscription": true,
    "is_restored": false,
    "is_safe": true
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <action>subscribed</action>
    <is_new_channel>false</is_new_channel>
    <is_new_contact>false</is_new_contact>
    <is_new_subscription>true</is_new_subscription>
    <is_restored>false</is_restored>
    <is_safe>true</is_safe>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

CAUTION

**Please note that the “Increase priority for new subscriptions” option in the resource works only if a pr
when adding a new subscription. If the priority for the new subscription is passed in the API request,
subscriptions for the profile will not change.**

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
action	**string**	**Subscription update result**
is_safe	**bool**	**Safe and successful operation (no hard bounces etc)**
is_new_subscription	**bool**	**New for this resource**
is_new_channel	**bool**	**New for this channel of an existing resource**
is_new_contact	**bool**	**New subscription for an existing channel**
is_restored	**bool**	**A previously deleted subscription restored**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Get all profile subscriptions

Description

Retrieves all subscriptions of a single profile.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/list`

Request parameters

Parameter	Type	Example	Requ
db_id	**int**	**1**	**Yes**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**

Request example

JSON

```
{
    "db_id": 1,
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "resource_id": 29,
            "channel": "email",
            "email": "example@example.com",
            "priority": 10,
            "status": "suspended",
            "custom_fields": {
                "custom_int_field_2": 20000
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <channel>email</channel>
        <custom_fields>
            <custom_int_field_2>20000</custom_int_field_2>
        </custom_fields>
        <email>example@example.com</email>
        <priority>10</priority>
        <resource_id>29</resource_id>
        <status>suspended</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
channel	**string**	**Channel type (email, push or sms)**
priority	**int**	**Subscription priority (0 for the highest priority)**
resource_id	**int**	**Subscription resource ID**
status	**string**	**Subscription status**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resources/fields_get method**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	Type	
email	**string**	**Email address**

Additionally for the Push channel:

Parameter	Type	
provider	**string**	**Provider type**
subscription_id	**string**	**Subscription ID**

Additionally for the SMS channel:

Parameter	Type	
phone	**string**	**Phone number**

Additionally for the Telegram channel:

Parameter	Type	Description
cc_data	**object**	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameter	Type	Description
cc_data	**object**	**WhatsApp* user's phone number**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Get all subscriptions from multiple profiles

Description

Retrieves all subscription entries from several customer profiles.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/list_all`

Request parameters

Parameter	Type	Example	Required	
resource_id	**int**	**1**	**Yes**	**Resource ID**
db_id	**int**	**1**	**No**	**Database ID**
channel	**string**	**“email”**	**No**	**Channel type**
status	**string**	**“subscribed”**	**No**	**Subscription status**
from_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**First subscription i**
limit	**int**	**1**	**No**	**Response data limi**

Request example

JSON

```
{
    "db_id": 1,
    "resource_id": 29,
    "from_id": "abcdefghijklmnqrstuvwxyz",
    "limit": 1
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <resource_id>1</resource_id>
  <from_id>abcdefghijklmnqrstuvwxyz</from_id>
  <limit>1</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "db_id": 1,
            "profile_id": "6643052fee44bb50a671161d",
            "_xxh": "026e82a364dd17e6",
            "hash_id": "fcf06ea1",
            "resource_id": 29,
            "channel": "email",
            "email": "zanitrevitteu-5912@1xp.fr",
            "priority": 0,
            "status": "subscribed",
            "custom_fields": {
                "custom_int_field_2": 20000
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": "cPydNpzRRGLhjKziMHRQ3awnGtyo8hjzn7Jar5mhd"
}
```

XML

```
<xml>
    <data>
        <_xxh>026e82a364dd17e6</_xxh>
        <channel>email</channel>
        <custom_fields>
            <custom_int_field_2>20000</custom_int_field_2>
        </custom_fields>
        <db_id>1</db_id>
        <email>zanitrevitteu-5912@1xp.fr</email>
        <hash_id>fcf06ea1</hash_id>
        <priority>0</priority>
        <profile_id>6643052fee44bb50a671161d</profile_id>
        <resource_id>29</resource_id>
        <status>subscribed</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>cPydNpzRRGLhjKziMHRQ3awnGtyo8hjzn7Jar5mhdm</next_from_id>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**
resource_id	**int**	**Subscription resource ID**
channel	**string**	**channel type**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resources/fields_get method**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	Type	
email	**string**	**Email address**

Additionally for the Push channel:

Parameter	Type	
provider	**string**	**Provider type**
subscription_id	**string**	**Subscription ID**

Additionally for the SMS channel:

Parameters	Type	
phone	**string**	**Phone number**

Additionally for the Telegram channel:

Parameter	Type	Description
cc_data	**object**	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameter	Type	Description
cc_data	**object**	**WhatsApp* user's phone number**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Get profile subscription

Description

Retrieves detailed information about one profile subscription.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/get`

Request parameters

Parameter	Type	Example	Re
db_id	**int**	**1**	**Yes**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
resource_id	**int**	**1**	**Yes**

Additionally for the Email channel:

Parameter	Type	Example	
channel	**string**	**“email”**	**Cha**
email	**string**	**“john@example.com”**	**Ema**

Additionally for the Push channel:

Parameters	Type	Example
channel	**string**	**“push”**
provider	**string**	**“Firefox”**
subscription_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Additionally for the SMS channel:

Parameters	Type	Example	
channel	**string**	**“sms”**	**Channel**
phone	**string**	**“+79000000000”**	**Phone nu**

Additionally for the Telegram channel:

Parameters	Type	Example	Descrip
channel	**string**	**“telegram_bot”**	**Channel type**
cc_data	**object**	`{ “id”: 1 }`	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameters	Type	Example	
channel	**string**	**“telegram_bot”**	**Channel type**
cc_data	**object**	`{ “phone”: “+79000000000” }`	**WhatsApp* user’s phone number**

Request example

JSON

```
{
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <resource_id>26</resource_id>
  <channel>push</channel>
  <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
  "data": {
    "channel": "push",
    "priority": 10,
    "provider": "Firefox",
    "resource_id": 1,
    "status": "subscribed",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "custom_fields": {
      "_device_type": "mob",
      "_os": "Android",
      "custom_field_1": "test value"
    },
    "cats": ["news_category", "transaction_category"]
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
    <data>
        <channel>push</channel>
        <priority>0</priority>
        <provider>Firefox</provider>
        <resource_id>26</resource_id>
        <status>subscribed</status>
        <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
channel	**string**	**Channel type (email, push or sms)**
priority	**int**	**Subscription priority (0 is the highest priority)**
resource_id	**int**	**Subscription resource ID**
status	**string**	**Subscription status**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resources/fields_get method**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	Type	
email	**string**	**Email address**

Additionally for the Push channel:

Parameter	Type	
provider	**string**	**Provider type**
subscription_id	**string**	**Subscription ID**

Additionally for the SMS channel:

Parameter	Type	
phone	**string**	**Phone number**

Additionally for the Telegram channel:

Parameter	Type	Description
cc_data	**object**	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameter	Type	Description
cc_data	**object**	**WhatsApp* user's phone number**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Delete profile subscription

Description

Deletes profile subscription.
Subscription remains but gets `"is_delete : true"` property.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/delete`

Request parameters

Parameter	Type	Example	Re
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
resource_id	**int**	**1**	**Yes**
db_id	**int**	**1**	**Yes**

Additionally for the Email channel:

Parameters	Type	Example	
channel	**string**	**“email”**	**Ch**
email	**string**	**“john@example.com”**	**Em**

Additionally for the Push channel:

Parameters	Type	Example
channel	**string**	**“push”**
provider	**string**	**“Firefox”**
subscription_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Additionally for the SMS channel:

Parameters	Type	Example	
channel	**string**	**“sms”**	**Channel**
phone	**string**	**“+79000000000”**	**Phone nu**

Additionally for the Telegram channel:

Parameters	Type	Example	Description
channel	**string**	**“telegram_bot”**	**Channel type**
cc_data	**object**	`{ “id”: 1 }`	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameters	Type	Example	Description
channel	**string**	**“whatsapp”**	**Channel type**
cc_data	**object**	`{ “phone”: “+79000000000” }`	**WhatsApp* user’s phone number**

Request example

JSON

```
{
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <resource_id>26</resource_id>
    <channel>push</channel>
    <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Restore deleted subscription

Description

Restores deleted subscription.
Changes `"is_delete"` parameter from `"false"` to `"true"`.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/undelete`

Request parameters

Parameter	Type	Example	Re
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
resource_id	**int**	**1**	**Yes**
db_id	**int**	**1**	**Yes**

Additionally for the Email channel:

Parameters	Type	Example	
channel	**string**	**“email”**	**Ch**
email	**string**	**“john@example.com”**	**Em**

Additionally for the Push channel:

Parameters	Type	Example
channel	**string**	**“push”**
provider	**string**	**“Firefox”**
subscription_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Additionally for the SMS channel:

Parameters	Type	Example	
channel	**string**	**“sms”**	**Channel**
phone	**string**	**“+79000000000”**	**Phone nu**

Additionally for the Telegram channel:

Parameters	Type	Example	Description
channel	**string**	**“telegram_bot”**	**Channel type**
cc_data	**object**	`{ “id”: 1 }`	**Chat ID between Telegram bot and user**

Additionally for the WhatsApp* channel:

Parameters	Type	Example	Description
channel	**string**	**“whatsapp”**	**Channel type**
cc_data	**object**	`{ “phone”: “+79000000000” }`	**WhatsApp* user’s phone number**

Request example

JSON

```
{
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <resource_id>26</resource_id>
    <channel>push</channel>
    <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared

Suspend all subscriptions

Description

Changes profile subscription status from `"subscribed"` to `"suspended"`.
Does not affect any other subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/suspend_all`

Request example

Parameter	Type	Example	Required	
db_id	**int**	**1**	**Yes**	**Dat**
channel	**int**	**“email”**	**No**	**Cha**
resource_id	**int**	**1**	**No**	**Res**

If the `"channel"` parameter is passed, all subscriptions within the specified channel will be suspended.
If the `"resource_id"` parameter is passed, all subscriptions within the specified resource will be suspend
If both the `"channel"` and `"resource_id"` parameters are passed, subscriptions for the specified channel v suspended.

Request example

JSON

```
{
    "db_id": 1,
    "channel": "email"
}
```

XML

```
<xml>
  <db_id>1</db_id>
  <channel>email</channel>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "updated": 4080
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <updated>1344</updated>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
updated	**int**	**Number of subscriptions suspended**

Unsuspend all suspended subscriptions

Description

Changes profile subscription status from `"suspended"` to `"subscribed"`.
Does not affect any other subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/unsuspend_all`

Request parameters

Parameter	Type	Example	Required	
db_id	**int**	**1**	**Yes**	**Dat**
channel	**int**	**“email”**	**No**	**Cha**
resource_id	**int**	**1**	**No**	**Res**

If `"channel"` parameter is specified, all channel subscriptions for a database will be unsuspended.
If `"resource_id"` parameter is specified, all resource subscriptions for a database will be unsuspended.
If both parameters are specified — resource and channel subscriptions will be unsuspended.
If none of these parameters are specified — all database subscriptions will be unsuspended.

Request example

JSON

```
{
    "db_id": 1
}
```

XML

```
<xml>
  <db_id>1</db_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "updated": 5428
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <updated>5428</updated>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
updated	**int**	**Number of subscriptions unsuspended**

Get profile action history

Description

Retrieves a single profile action history.
The request supports pagination using the `limit` and `from_id` parameters. By default, up to 1000 events events than the specified `limit`, the response will include the `next_from_id` field — the ID of the next eve events, pass the `next_from_id` value from the previous response as the `from_id` parameter in your next re made without `from_id` or with `from_id` set to `"0"`.

Request URL

Method: POST

`https://example.com/api/v1.1/subscribers/history_get`

Request parameters

Parameter	Type	Example	Required	Description
list_id	**Int**	**20**	**Yes**	**Database ID**
xxh	**string**	**“0eb51aefd919f90a”**	**No, of you use `matching` for search**	**xx hash profile ID**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No, of you use `xxh` for search**	**Profile matching mode By default — email For each type of matching, certain fields must be passed in the request body. Learn more abou**
date_from	**string**	**“2016-11-25”**	**No**	**Time period start YYYY-MM-DD Default — current date minus 30 days. If `date_to` is set but `date_from` is not, the left-side period restriction is removed — all events up t date restriction.**
date_to	**string**	**“2016-11-29”**	**No**	**Time period end YYYY-MM-DD**
limit	**Int**	**1000**	**No**	**Maximum number of events in the response Default — 1000**
from_id	**string**	**“0”**	**No**	**Event ID to start output from Default — “0” To continue pagination, pass the value from the `next_from_id` field of the previous response**

Request example

JSON

```
{
  "list_id": 20,
  "xxh": "0eb51aefd919f90a",
  "limit": 1000,
  "from_id": "0"
}
```

XML

```
<xml>
  <list_id>20</list_id>
  <xxh>0eb51aefd919f90a</xxh>
  <limit>1000</limit>
  <from_id>0</from_id>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "xxh": "9202595093f237d4",
            "profile_id": "66f66973096b3b454bbbccec",
            "email": "example@example.com",
            "list_id": 66,
            "action": "send",
            "count": 1,
            "datetime": "2024-09-27T15:52:57.004Z",
            "message_id": 5,
            "campaign_id": 137,
            "event_id": "66f6aaa9da00a9db1c0b2683",
            "smid": "uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5",
            "campaign_name": "New astral mailing",
            "message_name": "testing",
            "subscription": {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub1@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "other"
                },
                "cats": []
            },
            "channel": "email",
            "subscriptions": [
                {
                    "resource_id": 5,
                    "resource_name": "res1",
                    "channel": "email",
                    "email": "sub1@yandex.ru",
                    "priority": 0,
                    "status": "subscribed",
                    "custom_fields": {
                        "_device_type": "other"
                    },
                    "cats": []
                },
                {
                    "resource_id": 5,
                    "resource_name": "res1",
                    "channel": "email",
                    "email": "sub2@yandex.ru",
                    "priority": 0,
                    "status": "subscribed",
                    "custom_fields": {
                        "_device_type": "mob"
                    },
                    "cats": []
                }
            ]
        }
    ],
    "next_from_id": "13",
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
        <item>
            <xxh>9202595093f237d4</xxh>
            <profile_id>66f66973096b3b454bbbccec</profile_id>
            <email>example@example.ru</email>
            <list_id>66</list_id>
            <action>send</action>
            <count>1</count>
            <datetime>2024-09-27T15:52:57.004Z</datetime>
            <message_id>5</message_id>
            <campaign_id>137</campaign_id>
            <event_id>66f6aaa9da00a9db1c0b2683</event_id>
            <smid>uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5</smid>
            <campaign_name>New astral mailing</campaign_name>
            <message_name>testing</message_name>
            <subscription>
                <resource_id>5</resource_id>
                <resource_name>res1</resource_name>
                <channel>email</channel>
                <email>sub1@yandex.ru</email>
                <priority>0</priority>
                <status>subscribed</status>
                <custom_fields>
                    <_device_type>other</_device_type>
                </custom_fields>
            </subscription>
            <channel>email</channel>
            <subscriptions>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub1@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>other</_device_type>
                    </custom_fields>
                </subscription>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub2@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>mob</_device_type>
                    </custom_fields>
                </subscription>
            </subscriptions>
        </item>
    </data>
    <next_from_id>13</next_from_id>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
data	**string**	**An array of received data regarding events in the profile history**
data.xxh	**string**	**Profile xxhash ID**
data.profile_id	**string**	**Profile ID**
data.email	**string**	**Profile email**
data.list_id (db_id)	**int**	**Database ID**
data.action	**string**	**Profile action (More details)**
data.count	**int**	**Number of actions taken in this second**
data.datetime	**string**	**Date and time of the action in RFC 3339 (ISO 8601) format**
data.message_id	**int**	**Message template ID** **Returns 0 if the action does not involve using a template.**
data.campaign_id	**int**	**Mailing ID** **Returns 0 if the action does not involve using a mailing.**
data.event_id	**string**	**Event ID**

Parameter	Type	Description
data.smid	string	Unique identifier of the send within the mailing Returns an empty string if the action does not involve sending a message.
data.campaign_name	string	Name of the mailing Returns an empty string if the action does not involve using a mailing.
data.message_name	string	Name of the message template Returns an empty string if the action does not involve using a template.
data.subscription	string	Object containing information about the subscription under which the sending was performed
data.channel	string	Communication channel in which the event was recorded. Returns an empty string if the channel is not involved.
data.subscriptions	array of objects	All subscriptions of the profile. For CSV format, it will be represented as a JSON string.
next_from_id	string	ID of the next event for pagination continuation. If the number of events exceeds the `limit` value, pass this value as the `from_id` par events. If there are no more events, the field will be empty or absent.
error	int	Error code
error_text	string	Error message

Profile history events (action)

Email channel events	
send	Email sent
deliv	Email delivered
undeliv	Email not delivered
open	Email opened
read	Email read. The message was opened by the user for 8 seconds or more
click	Email clicked. Registered when the user clicks a link in the email. Each link click is registered as a separate click event, so if a profile clicks on be recorded.
confirm	Click on the subscription confirmation link in the email (confirm-link)
subscribe_email	Subscribed to the Email channel of the resource
unsubscribe_email	Unsubscribed from the Email channel of the resource
reply	Received a reply to the Email
complain	Received a Complain status
hbounce	Received a Hard Bounce status
SMS channel events	
send_sms	SMS message sent to the gateway for subscriber delivery
deliv_sms	SMS message delivered
undeliv_sms	SMS message not delivered
click_sms	Non-unique click on the link in the SMS message
subscribe_sms	Subscribed to the SMS channel of the resource
unsubscribe_sms	Unsubscribed from the SMS channel of the resource
Push channel events	
send_push	Push sent
deliv_push	Push delivered
open_push	Non-unique opening of the push message
click_push	Click on the Push message
undeliv_push	Push message not delivered
subscribe_push	Subscribed to the Push channel of the resource
unsubscribe_push	Unsubscribed from the Push channel of the resource
Telegram Bot Channel Events	
telegram_bot_send	Telegram bot event: Telegram bot sent
telegram_bot_deliv	Telegram bot event: Telegram bot delivered
telegram_bot_undeliv	Telegram bot event: Telegram bot not delivered
telegram_bot_click	Telegram bot event: Telegram bot click
telegram_bot_subscribe	Telegram bot event: Telegram bot subscription
WhatsApp* channel events	
whatsapp_send	Message sent for delivery to the WhatsApp* chat
whatsapp_deliv	Message delivered to the WhatsApp* chat
whatsapp_undeliv	Message not delivered to the WhatsApp* chat
whatsapp_click	Non-unique click on the link in the WhatsApp* chat message
whatsapp_read	Message read by the user
whatsapp_subscribe	WhatsApp* channel event: WhatsApp* subscription
Viber channel events	
viber_send	Message sent for delivery to the Viber chat
viber_deliv	Message delivered to the recipient
viber_undeliv	Message not delivered, rejected by Viber or Devino.Online
viber_click	Click on the link in the message. If the recipient clicks on the same link several times, each click event is recorded.
viber_read	Message read by the user

viber_subscribe	**Viber channel event: Viber subscription**
Events for all channels	
offence	**Message not sent due to sending policy restrictions**
suppress	**Message not sent by the mailing because the profile is on the stop list**
Subscriber Profile Import Events	
import_manual	**Profile created manually**
import_api	**Profile created by API import**
import_file	**Profile created by import from a file**
import_form	**Profile created by import from a form**
import_push	**Profile created as a result of push import**
import_popup	**Profile created by import via popup**
Unsubscribe Events	
unsub_api	**Subscriber unsubscribed from emails via API**
unsub_manual	**Unsubscribed manually (global profile status changed to "Unsubscribed")**
Pixel events	
pixel_open	**Goal achieved**
Promo codes events	
promocode_attach	**Promo code attached**
promocode_detach	**Promo code detached**
promocode_activate	**Promo code activated**
Segment events	
segs_add	**Entered a static segment**
segs_remove	**Exited a static segment**
Profile Relationship Events	
rel_attach	**Profile receives a connection with another profile**
rel_detach	**Profile loses a connection with another profile**
rel_strengthen	**Weight of the connection between profiles increases**
Form Events	
form_load	**Form loaded**
form_page_show	**Form page loaded**
form_post	**Form filled and submitted**
form_abandon	**Form abandoned. The profile loads the form page and does not close it for an hour, without submitting the form.**
form_bounce	**Error in filling out the form (attempted re-filling, technical issues, etc.)**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared***

Get multiple profiles action history

Description

Retrieves multiple profiles action history.

Request URL

Method: POST

`https://example.com/api/v1.1/subscribers/history_get_all`

Request parameters

Parameter	Type	Example	Required	Description
list_id	**int**	**20**	**Yes**	**Database ID**
from_id	**string**	**“005a06ded879ecd7.20170406”**	**No**	**The first profile in response (xxhash)**
limit	**int**	**300**	**No**	**Response profiles array limit By default - 1000**
date_from	**string**	**“2016-11-25”**	**Yes**	**Time period start YYYY-MM-DD**
date_to	**string**	**“2016-11-29”**	**Yes**	**Time period end YYYY-MM-DD**
fields	**string array**	**[“CustomID”]**	**No**	**Returns a field (`fields`) with profile data from the listed fields in the response. For CSV form**

Request example

JSON

```
{
    "list_id": 4,
    "limit": 3,
    "date_from": "2024-07-15",
    "date_to": "2024-10-09",
    "format": "json",
    "from_id": "",
    "fields": [
        "CustomNumber",
        "CustomEnum"
    ]
}
```

XML

```
<xml>
  <list_id>4</list_id>
  <limit>3</limit>
  <date_from>2024-07-15</date_from>
  <date_to>2024-10-09</date_to>
  <format>xml</format>
  <from_id></from_id>
  <fields>CustomNumber</fields>
  <fields>CustomEnum</fields>
</xml>
```

Response example

JSON

```
{
"data": [
    {
        "xxh": "9202595093f237d4",
        "profile_id": "66f66973096b3b454bbbccec",
        "email": "example@example.com",
        "list_id": 66,
        "action": "send",
        "count": 1,
        "datetime": "2024-09-27T15:52:57.004Z",
        "message_id": 5,
        "campaign_id": 137,
        "event_id": "66f6aaa9da00a9db1c0b2683",
        "smid": "uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5",
        "campaign_name": "New astral mailing",
        "message_name": "testing",
        "subscription": {
            "resource_id": 5,
            "resource_name": "res1",
            "channel": "email",
            "email": "sub1@yandex.ru",
            "priority": 0,
            "status": "subscribed",
            "custom_fields": {
                "_device_type": "other"
            },
            "cats": []
        },
        "channel": "email",
        "subscriptions": [
            {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub1@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "other"
                },
                "cats": []
            },
            {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub2@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "mob"
                },
                "cats": []
            }
        ]
    }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
        <item>
            <xxh>9202595093f237d4</xxh>
            <profile_id>66f66973096b3b454bbbccec</profile_id>
            <email>example@example.ru</email>
            <list_id>66</list_id>
            <action>send</action>
            <count>1</count>
            <datetime>2024-09-27T15:52:57.004Z</datetime>
            <message_id>5</message_id>
            <campaign_id>137</campaign_id>
            <event_id>66f6aaa9da00a9db1c0b2683</event_id>
            <smid>uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5</smid>
            <campaign_name>New astral mailing</campaign_name>
            <message_name>testing</message_name>
            <subscription>
                <resource_id>5</resource_id>
                <resource_name>res1</resource_name>
                <channel>email</channel>
                <email>sub1@yandex.ru</email>
                <priority>0</priority>
                <status>subscribed</status>
                <custom_fields>
                    <_device_type>other</_device_type>
                </custom_fields>
            </subscription>
            <channel>email</channel>
            <subscriptions>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub1@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>other</_device_type>
                    </custom_fields>
                </subscription>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub2@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>mob</_device_type>
                    </custom_fields>
                </subscription>
            </subscriptions>
        </item>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
data	**string**	**An array of received data regarding events in the history of profiles**
data.xxh	**string**	**Profile xxhash ID**
data.profile_id	**string**	**Profile ID**
data.email	**string**	**Profile email**
data.list_id (db_id)	**int**	**Database ID**
data.action	**string**	**Profile action (More details)**
data.count	**int**	**Number of actions taken in this second**
data.datetime	**string**	**Date and time of the action in RFC 3339 (ISO 8601) format**
data.message_id	**int**	**Message template ID** **Returns 0 if the action does not involve using a template.**
data.campaign_id	**int**	**Mailing ID** **Returns 0 if the action does not involve using a mailing.**
data.event_id	**string**	**Event ID**

Parameter	Type	Description
data.smid	string	Unique identifier of the send within the mailing Returns an empty string if the action does not involve sending a message.
data.campaign_name	string	Name of the mailing Returns an empty string if the action does not involve using a mailing.
data.message_name	string	Name of the message template Returns an empty string if the action does not involve using a template.
data.subscription	string	Object containing information about the subscription under which the sending was performed
data.channel	string	Communication channel in which the event was recorded. Returns an empty string if the channel is not involved.
data.subscriptions	array of objects	All subscriptions of the profile. For CSV format, it will be represented as a JSON string.
data.fields	object	Additional profile fields specified in the fields parameter of the request. For the CSV format, it will be presented as
next_from_id	string	The first profile in the next batch of the databas
error	int	Error code
error_text	string	Error message

Profile relations

TIP

Profile relations are used to describe interactions between your customers, or between customers an sale. You can find more information in this User documentation article.

Attach relation

Description

Creates a relation between two profiles. In case a relation already exists an error will be returned.

URL

Method: POST

`https://example.com/api/v1.1/relations/attach`

Request parameters

Parameter	Type	Example	Required	Descript
rel_id	**int**	**3**	**Yes**	**Relation ID**
profile_a	**object**	`{"profile_id": "c1wefohoi1312"}`	**Yes**	**Database A profile - relation recipient (object).**
profile_b	**object**	`{"profile_id": "c1wefohoi13123"}`	**Yes**	**Database B profile - relation initiator (subject).**
weights	**object**	`{"user_like": 2, "donate": 343, "is_online": true}`	**Yes**	**Relation properties**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/ tag type profile fields. Values are recorded as a string: "t**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
    <rel_id>3</rel_id>
    <profile_a>
        <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
    </profile_a>
    <profile_b>
        <profile_id>5d441038439e7363f3c26539</profile_id>
    </profile_b>
    <weights>
        <user_like>2</user_like>
        <donate>343</donate>
        <is_online>true</is_online>
    </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
  <error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	Type	Des
error	**int**	**Error code**
error_text	**string**	**Error description**
profile_a	**string**	**Database A profile**
profile_b	**string**	**Database B profile**
relation_id	**int**	**Relation ID**

Detach relation

Description

Removes a relation between two profiles.

URL

Method: POST

`https://example.com/api/v1.1/relations/detach`

Request parameters

Parameter	Type	Example	Required	Description
rel_id	**int**	**3**	**Yes**	**Relation ID**
profile_a	**object**	`{"profile_id": "c1wefohoi1312"}`	**Yes**	**Database A profile - relation recipient (object).**
profile_b	**object**	`{"profile_id": "c1wefohoi13123"}`	**Yes**	**Database B profile - relation initiator (subject).**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/user-gu type profile fields. Values are recorded as a string: "tag-1, tag-2**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    }
}
```

XML

```
<xml>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```


XML


```
<xml>
    <error>0</error>
    <error_text></error_text>
    <profile_a>5c910f6bc484d023b1309bee</profile_a>
    <profile_b>5c910938c484d023b1309bac</profile_b>
    <relation_id>1</relation_id>
</xml>
```


Response parameters

Parameter	Type	Des
error	**int**	**Error code**
error_text	**string**	**Error description**
profile_a	**string**	**Database A profile**
profile_b	**string**	**Database B profile**
relation_id	**int**	**Relation ID**

Modify relation properties

Description

Modifies (increases or decreases) relation properties' values by those you send in `weights` parameter o positive and negative values.

> **NOTE**
>
> **If no relation is found between the two profiles, a new relation will be created.**

> **TIP**
>
> **If you need to replace properties' values instead of increasing/decreasing, use Overwrite relation pro**

URL

Method: POST.

`https://example.com/api/v1.1/relations/strengthen`

Request parameters

Parameter	Type	Example	Required	Descript
rel_id	**int**	**3**	**Yes**	**Relation ID**
profile_a	**object**	`{"profile_id": "c1wefohoi1312"}`	**Yes**	**Database A profile - relation recipient (object).**
profile_b	**object**	`{"profile_id": "c1wefohoi13123"}`	**Yes**	**Database B profile - relation initiator (subject).**
weights	**object**	`{"user_like": 2, "donate": 343, "is_online": true}`	**Yes**	**Relation properties.** **If you need to weaken the property, pass a negative value: -2 o**
Profile matching mode				
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode** **By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions** **"matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for<a href='/en/ tag type profile fields. Values are recorded as a string: "t**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
  <weights>
    <user_like>2</user_like>
    <donate>343</donate>
    <is_online>true</is_online>
  </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
  <error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	Type	Des
error	**int**	**Error code**
error_text	**string**	**Error description**
profile_a	**string**	**Database A profile**
profile_b	**string**	**Database B profile**
relation_id	**int**	**Relation ID**

Overwrite relation properties

Description

Replaces relation properties' values with the ones you send with this request.

> **NOTE**
>
> **In case no relation between the specified profiles exists — a new relation will be created.**

> **TIP**
>
> **If you need to increase or decrease relation properties, use Modify relation properties endpoint.**

URL

Method: POST.

`https://example.com/api/v1.1/relations/edit`

Request parameters

Parameter	Type	Example	Required	
rel_id	**int**	**3**	**Yes**	**Relation ID**
profile_a	**object**	{"profile_id": "c1wefohoi1312"}	**Yes**	**Database A profile - relation recipient (object).**
profile_b	**object**	{"profile_id": "c1wefohoi13123"}	**Yes**	**Database B profile - relation initiator (subject).**
weights	**object**	{"user_like": 2, "donate": 343, "is_online": true}	**Yes**	**Relation properties**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value. Searching by combination of tags is available fo "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
  <weights>
    <user_like>2</user_like>
    <donate>343</donate>
    <is_online>true</is_online>
  </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
<error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	Type	Des
error	**int**	**Error code**
error_text	**string**	**Error description**
profile_a	**string**	**Database A profile**
profile_b	**string**	**Database B profile**
relation_id	**int**	**Relation ID**

Get profile relations info

Description

Get information about a Database A profile's relations (initiated by Database B profiles) and their prope

URL

Method: POST.

`https://example.com/api/v1.1/relations/get_counter`

Request parameters

Parameter	Type	Example	Required	Desc
rel_id	**int**	**3**	**Yes**	**Relation ID**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value. Searching by combination of tags is available for tag type 2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 3,
    "profile_id": "5d63dbdb439e7306b596c5d0"
}
```

XML

```
<xml>
  <rel_id>3</rel_id>
  <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "profile_id": "669e6968ebf4ac52b623a997",
        "profile_xxh": "5af1023c84f66195",
        "props": {}
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <profile_id>669e6968ebf4ac52b623a997</profile_id>
        <profile_xxh>5af1023c84f66195</profile_xxh>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**object**	**Profile relations info**
data.profile_id	**string**	**Profile identifier**
data.profile_xxh	**string**	**Hashed profile ID**
data.props	**object**	**Relation properties**

Relation properties: numeric

Parameter	Type	Description
top	**object**	**Property top records**
sum	**int/float**	**Sum total of property values**

Get profile relations info

Description

The request brings a list of all existing links for the specified profile from the database A, including ide links and their characteristics.

URL

Method: POST.

`https://example.com/api/v1.1/relations/get`

Request parameters

Parameter	Type	Example	Required	Desc
rel_id	**int**	**3**	**Yes**	**Relation ID**
profile_id	**string**	**"669e6968ebf4ac52b623a997"**	**Yes, if no other search type is specified**	**Profile identifier from database A for the search**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value. Searching by combination of tags is available for tag typ tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

Request example

JSON

```
{
    "rel_id": 2,
    "profile_id": "669e6968ebf4ac52b623a997"
}
```

XML

```
<xml>
  <rel_id>2</rel_id>
  <profile_id>669e6968ebf4ac52b623a997</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "profile_id_a": "669e6968ebf4ac52b623a997",
            "xxh_a": "5af1023c84f66195",
            "profile_id_b": "6751b8f1c5e3db9bcb75befa",
            "xxh_b": "879cebbb0e6b6723",
            "db_b": 9,
            "props": {
                "exr": {
                    "value": 11323123
                }
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 1
}
```

XML

```
<xml>
    <data>
        <profile_id_a>669e6968ebf4ac52b623a997</profile_id_a>
            <xxh_a>5af1023c84f66195</xxh_a>
            <profile_id_b>6751b8f1c5e3db9bcb75befa</profile_id_b>
            <xxh_b>879cebbb0e6b6723</xxh_b>
            <db_b>9</db_b>
            <props>
                <exr>
                    <value>11323123</value>
                </exr>
            </props>
        </relation>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>1</total_count>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error message**
data	**object**	**Information about profile relations**
data.profile_id_a	**string**	**ID of profile A (the profile to which the relation is directed)**
data.xxh_a	**string**	**Hashed ID of profile A**
data.profile_id_b	**string**	**ID of profile B (the profile initiating the relation)**
data.xxh_b	**string**	**Hashed ID of profile B**
data.db_b	**int**	**ID of the database containing profile B**
data.props	**object**	**Connection properties**
data.total_count	**int**	**Number of relations with other profiles**

Get database statistics

Description

Get the statistics of subscriptions to a specific resource within the base.

URL

Method: POST

`https://example.com/api/v1.1/databases/get_counters`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Database ID**
resource_id	**int**	**1**	**Yes**	**Resource ID**
format	**int**	**“json”**	**No**	**Response data format By default – json**

Request example

JSON

```
{
    "id": 5,
    "resource_id": 24,
    "format": "json"
}
```

XML

```
 <xml>
     <id>5</id>
     <resource_id>24</resource_id>
     <format>json</format>
 </xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "all": 1336,
                "subscribed": 925,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 411,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 925,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <channel>email</channel>
        <status>
            <all>1336</all>
            <subscribed>925</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>411</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>925</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>sms</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>push</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
data	**array of objects**	**Array of received resource channel data**
error	**int**	**Error code**
error_text	**string**	**Error text**

Update statistics on database

Description

Update subscription statistics of profiles in the database.
Returns a response with task ID without waiting for execution.

URL

Method: POST

`https://example.com/api/v1.1/databases/refresh_counters`

Request parameters

```
    <tr>
        <td>format</td>
        <td>JSON object </td>
        <td>"json"</td>
        <td>No</td>
        <td> Response data format<br/>By default – json </td>
    </tr>
    <tr>
        <td>feedback</td>
        <td>JSON object </td>
        <td>
"feedback": {
    "type": "JSON",
    "url": "http://example.com/receiver"
}
            </td>
        <td>No</td>
        <td>Parameter defining the method of sending data about statistics update. <br/> Parameter "type" - the format or method used to se
statistics data will be sent.<br/></td>
    </tr>
</tbody>
```

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**D**
resource_id	**int**	**1**	**Yes**	**R**

Request example

JSON

```
{
    "id": 5,
    "resource_id": 24,
    "format": "json",
    "feedback": {
        "type": "JSON",
        "url": "http://example.com/receiver"
    }
}
```

XML

```
<xml>
     <id>5</id>
     <resource_id>24</resource_id>
     <format>json</format>
     <feedback>
         <type>JSON</type>
         <url>http://example.com/receiver</url>
     </feedback>
 </xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "6908adeb-346e-4f5e-9f68-df062bf795cf"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>6908adeb-346e-4f5e-9f68-df062bf795cf</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
 </xml>
```

Response parameters

Parameter	Type	Description
data	**JSON objects**	**Contains the task identifier - task_id**
error	**int**	**Error code**
error_text	**string**	**Error text**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "task_id": "930bfdb0-a5a8-4207-a77d-6b989bdea326"
}
```

XML

```
<xml>
     <task_id>930bfdb0-a5a8-4207-a77d-6b989bdea326</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "email",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 411,
                            "invalid": 0,
                            "subscribed": 925,
                            "subscribed_sendable": 925,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "sms",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    }
                ],
                "in_progress": false,
                "last_count_time": "counted at 2024-04-17T21:36:01Z in 82.746959ms",
                "refresh_task_id": "",
                "resource_id": 24,
                "total_count": 1336,
                "updated": "2024-04-17T21:36:01.449Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>email</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>411</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>925</subscribed>
                    <subscribed_sendable>925</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>sms</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <last_count_time>counted at 2024-04-17T21:36:01Z in 82.746959ms</last_count_time>
            <refresh_task_id></refresh_task_id>
            <resource_id>24</resource_id>
            <total_count>1336</total_count>
            <updated>2024-04-17T21:36:01.449Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get database list

Description

Retrieves a list of databases.

Request URL

Method: POST

`https://example.com/api/v1.1/databases/list/`

Request parameters

Parameter	Type	Example	Required	
format	**int**	**“json”**	**No**	**Response data forma By default – json**
from_id	**int**	**1**	**No**	**First database in the By default – 1**
limit	**int**	**1**	**No**	**Limit for response e By default – 1000**
ui_tags	**[]String**	**[“tag1”, “tag2”]**	**No**	**List of tags for searc**
groups	**[]String**	**[“group1”, “group2”]**	**No**	**List of access group**
groups_id	**[]int**	**[0, 1]**	**No**	**List of access group**

Request example

JSON

```
{
  "limit": 2
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <limit>2</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Database #1",
      "ui_tags": [],
      "Groups": [
        {
          "Id": 0,
          "Name": "main"
        }
      ]
    },
    {
      "id": 2,
      "name": "Database #2",
      "ui_tags": ["tag1", "tag2"],
      "Groups": [
        {
          "Id": 1,
          "Name": "subaccount"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 6,
  "total_count": 4
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>Database #1</name>
    <Groups>
      <Id>0</Id>
      <Name>main</Name>
    </Groups>
  </data>
  <data>
    <id>2</id>
    <name>Database #2</name>
    <ui_tags>tag1</ui_tags>
    <ui_tags>tag2</ui_tags>
    <Groups>
      <Id>1</Id>
      <Name>subaccount</Name>
    </Groups>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <next_from_id>6</next_from_id>
  <total_count>4</total_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**
total_count	**int**	**Total database count**
id	**int**	**Database ID**
name	**string**	**Database name**
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of groups**

Get database information

Description

Retrieves a database information.

Request URL

Method: POST

`https://example.com/api/v1.1/databases/get/`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Database ID**
format	**int**	**“json”**	**No**	**Response data format By default – json**

Request example

JSON

```
{
  "id": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "id": 1,
    "name": "B2C customers",
    "suppress_id": 1,
    "description": "May contain info"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>B2C customers</name>
    <suppress_id>1</suppress_id>
    <description>May contain info</description>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Database ID**
name	**string**	**Database name**
suppress_id	**int**	**Associated suppression list ID**
description	**string**	**Database description**

Get database fields

Description

Retrieves database fields information. The types of fields and allowed values are described in [this arti](#)

Request URL

Method: POST

`https://example.com/api/v1.1/databases/fields_get/`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Database ID**
format	**int**	**“json”**	**No**	**Response data format By default – json**

Request example

JSON

```
{
  "id": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "category": "system",
      "title": "Id",
      "name": "_id",
      "required": true,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "default",
      "title": "FirstName",
      "name": "_fname",
      "required": false,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_String",
      "name": "Custom_String",
      "required": false,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Int",
      "name": "Custom_Int",
      "required": false,
      "submanager": false,
      "type": "int",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Bool",
      "name": "Custom_Bool",
      "required": false,
      "submanager": false,
      "type": "bool",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Date",
      "name": "Custom_Date",
      "required": false,
      "submanager": false,
      "type": "date",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Tags",
      "name": "Custom_Tags",
      "required": false,
      "submanager": false,
      "type": "tags",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Enum",
      "name": "Custom_Enum",
      "required": false,
      "submanager": false,
      "type": "enum",
      "enum_values": [
        {
          "name": "GMail",
          "value": 2
        },
        {
          "name": "HotMail",
          "value": 3
```

```
        },
        {
          "name": "Yahoo!",
          "value": 1
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <category>system</category>
    <title>Id</title>
    <name>_id</name>
    <required>true</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>default</category>
    <title>FirstName</title>
    <name>_fname</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_String</title>
    <name>Custom_String</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Int</title>
    <name>Custom_Int</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>int</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Bool</title>
    <name>Custom_Bool</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>bool</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Date</title>
    <name>Custom_Date</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>date</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Tags</title>
    <name>Custom_Tags</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>tags</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Enum</title>
    <name>Custom_Enum</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>enum</type>
    <enum_values>
      <name>GMail</name>
      <value>2</value>
    </enum_values>
    <enum_values>
      <name>HotMail</name>
      <value>3</value>
    </enum_values>
    <enum_values>
      <name>Yahoo!</name>
      <value>1</value>
    </enum_values>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
```

```
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
category	**string**	**Field category**
title	**string**	**Field title**
name	**string**	**Field name**
required	**bool**	**Required flag**
submanager	**bool**	**Subscription manager access**
type	**string**	**Field type**
enum_values		**Enum list values**

Database wipe

Description

Wipes all data from a database:

1. All customer or seed profiles.
2. All associated data except statistics.

To use this request you need to reconfigure `main.json` file in `config` directory:

```
{
    "ALLOW_DATA_WIPE": true
}
```

Request URL

Method: POST

`https://example.com/api/v1.1/databases/wipe`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**database ID**
format	**int**	**“json”**	**No**	**Response data format By default – json**

Request example

JSON

```
{
  "id": 1
}
```

XML

```
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "count_action": 0,
    "count_profile": 0
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <count_action>0</count_action>
    <count_profile>0</count_profile>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
data	**JSON object**	**Data object**
data.count_action	**int**	**Profile actions wiped**
data.count_profile	**int**	**Profiles wiped**
error	**int**	**Error code**

Parameter	Type	
error_text	**string**	**Error text**

Get resource statistics

Description

Get the statistics of subscriptions on resource.

> **INFO: Note**
>
> **The `get_counters` and `refresh_counters` methods work together. Statistics for a resource do not update a the `refresh_counters` method, which initiates data recalculation and returns a task ID. After the task co be checked via `tasks/status`), the updated data will be available in the `get_counters` method.**

URL

Method: POST

`https://example.com/api/v1.1/resources/get_counters`

Request parameters

Parameter	Type	Example	Required	
list_id	**int**	**1**	**Yes**	**Database ID**
id	**int**	**1**	**Yes**	**Resource ID**
format	**JSON object**	**“json”**	**No**	**Response data form By default – json**

Request example

JSON

```
{
    "list_id": 24,
    "id": 5,
    "format": "json"
}
```

XML

```
 <xml>
     <list_id>24</list_id>
     <id>5</id>
     <format>json</format>
 </xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "all": 1336,
                "subscribed": 925,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 411,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 925,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <channel>email</channel>
        <status>
            <all>1336</all>
            <subscribed>925</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>411</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>925</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>sms</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>push</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
data	**array of objects**	**Array of received resource channel data**
error	**int**	**Error code**
error_text	**string**	**Error text**

Update statistics on resource

Description

Update subscription statistics of resource.
Returns a response with task ID without waiting for execution.

URL

Method: POST

`https://example.com/api/v1.1/resources/refresh_counters`

Request

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Resource ID**
list_id	**int**	**1**	**Yes**	**Database ID**
format	**JSON object**	**"json"**	**No**	**Response data format By default – json**
feedback	**JSON object**	**```json "feedback": { "type": "JSON", "url": "http://example.com/receiver" } ```**	**No**	**Parameter defining the method Parameter "type" - the format "url" - the URL to which the st**

Request example

JSON

```
{
    "id": 24,
    "list_id": 1,
    "format": "json",
    "feedback": {
        "type": "JSON",
        "url": "http://example.com/receiver"
    }
}
```

XML

```
<xml>
     <id>24</id>
     <list_id>1</list_id>
     <format>json</format>
     <feedback>
         <type>JSON</type>
         <url>http://example.com/receiver</url>
     </feedback>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "c9c3ec48-1bef-4969-9c1a-df39f0c179bb"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>c9c3ec48-1bef-4969-9c1a-df39f0c179bb</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
data	**JSON objects**	**Contains the task identifier - task_id**
error	**int**	**Error code**
error_text	**string**	**Error text**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "task_id": "c9c3ec48-1bef-4969-9c1a-df39f0c179bb"
}
```

XML

```
<xml>
     <task_id>c9c3ec48-1bef-4969-9c1a-df39f0c179bb</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "sms",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "email",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 434,
                            "invalid": 0,
                            "subscribed": 842,
                            "subscribed_sendable": 842,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 60
                        }
                    }
                ],
                "in_progress": false,
                "last_count_time": "counted at 2024-04-19T13:31:08Z in 231.374251ms",
                "list_id": 1,
                "refresh_task_id": "",
                "updated": "2024-04-19T13:31:08.927Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>sms</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>email</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>434</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>842</subscribed>
                    <subscribed_sendable>842</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>60</unsubscribed>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <last_count_time>counted at 2024-04-19T13:31:08Z in 231.374251ms</last_count_time>
            <list_id>1</list_id>
            <refresh_task_id></refresh_task_id>
            <updated>2024-04-19T13:31:08.927Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get resources list

Description

Retrieves all created resources list.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**1**	**No**	**The first ID in the resp By default – 1**
limit	**int**	**1**	**No**	**Resources list limit. By default – 1000**
groups	**[]String**	`[ “group1”, “group2” ]`	**No**	**List of access groups**
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access groups I**

Request example

JSON

```
{
  "from_id": 1,
  "limit": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <from_id>1</from_id>
  <limit>1</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Resource",
      "groups": [
        {
          "id": 1,
          "name": "subaccount"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 4,
  "total_count": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>Resource</name>
    <groups>
      <id>1</id>
      <name>subaccount</name>
    </groups>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <next_from_id>4</next_from_id>
  <total_count>1</total_count>
</xml>
```

Response parameters

Parameter	Type	description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**
id	**int**	**Resource ID**
name	**string**	**Resource name**
total_count	**int**	**Total resources count**
groups	**[]String**	**List of groups**

Get resource information

Description

Retrieves detailed information about a single resource, including channels and associated databases.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Res**

Request example

JSON

```
{
  "id": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "channels": ["email", "push"],
    "dbs_ids": [1],
    "id": 1,
    "name": "Resource",
    "trk_domain": "tracking.example.com",
    "url": "https://blog.example:8080"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <channels>email</channels>
    <channels>push</channels>
    <dbs_ids>1</dbs_ids>
    <id>1</id>
    <name>Resource</name>
    <trk_domain>tracking.example.com</trk_domain>
    <url>https://blog.example:8080</url>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
channel	**JSON array**	**Enabled channels**
dbs_ids	**JSON array**	**Associated databases**

Parameter	Type	Description
id	**int**	**Resource ID**
name	**string**	**Resource name**
trk_domain	**string**	**Subscribers actions tracking domain**
url	**string**	**Resource URL**

Get resource subscription fields

Description

Get information about system and additional subscription fields.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/fields_get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Resource ID**
channel	**string**	**“email” “sms” “push” “telegram_bot” “whatsapp”**	**Yes**	**Resource subscription channel**

Request example

JSON

```
{
  "id": 1,
  "channel": "email"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
  <channel>email</channel>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "category": "custom",
      "title": "New_field_1",
      "name": "api_new_field_1",
      "required": false,
      "type": "int"
    },
    {
      "category": "system",
      "title": "Device type",
      "name": "_device_type",
      "required": false,
      "type": "unicode",
      "dictionary": "devices"
    },
    {
      "category": "system",
      "title": "Device model",
      "name": "_device_model",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Device name",
      "name": "_device_name",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Platform",
      "name": "_os",
      "required": false,
      "type": "unicode",
      "dictionary": "oses"
    },
    {
      "category": "system",
      "title": "Platform version",
      "name": "_os_ver",
      "required": false,
      "type": "version"
    },
    {
      "category": "system",
      "title": "Platform time zone",
      "name": "_os_tz",
      "required": false,
      "type": "unicode",
      "dictionary": "tz"
    },
    {
      "category": "system",
      "title": "Platform language",
      "name": "_os_language",
      "required": false,
      "type": "unicode",
      "dictionary": "languages"
    },
    {
      "category": "system",
      "title": "Ad tracking allowed",
      "name": "_ad_track",
      "required": false,
      "type": "bool"
    },
    {
      "category": "system",
      "title": "Ad identifier",
      "name": "_ad_id",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application ID",
      "name": "_app_id",
```

```
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application installation ID",
      "name": "_app_iid",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application version",
      "name": "_app_ver",
      "required": false,
      "type": "version"
    },
    {
      "category": "system",
      "title": "Browser",
      "name": "_browser_name",
      "required": false,
      "type": "unicode",
      "dictionary": "browsers"
    },
    {
      "category": "system",
      "title": "Browser version",
      "name": "_browser_ver",
      "required": false,
      "type": "version"
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <category>custom</category>
    <title>New_field_1</title>
    <name>api_new_field_1</name>
    <required>false</required>
    <type>int</type>
  </data>
  <data>
    <category>system</category>
    <title>Device type</title>
    <name>_device_type</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>devices</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Device model</title>
    <name>_device_model</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Device name</title>
    <name>_device_name</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Platform</title>
    <name>_os</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>oses</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Platform version</title>
    <name>_os_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <data>
    <category>system</category>
    <title>Platform time zone</title>
    <name>_os_tz</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>tz</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Platform language</title>
    <name>_os_language</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>languages</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Ad tracking allowed</title>
    <name>_ad_track</name>
    <required>false</required>
    <type>bool</type>
  </data>
  <data>
    <category>system</category>
    <title>Ad identifier</title>
    <name>_ad_id</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application ID</title>
    <name>_app_id</name>
```

```
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application installation ID</title>
    <name>_app_iid</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application version</title>
    <name>_app_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <data>
    <category>system</category>
    <title>Browser</title>
    <name>_browser_name</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>browsers</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Browser version</title>
    <name>_browser_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
category	**string**	**Subscription field category** `"system","custom"`
title	**string**	**UI field name**
name	**string**	**API field name**
required	**bool**	**Required flag (true)**
type	**string**	**Field type**
default_value	*	**Field default value if it exists**

dictionary	**string**	**Field name used in the request /v1.1/dictionary (a request to obtain the values of system fields that are displayed in the interface as a drop-**
		
		`"browsers"`
		`"devices"`
		`"tz"`
		`"oses"`
		`"languages"`

Subscription system fields

Parameter	Type	Example	Description
_device_type	**string**	**“mob”**	**Device type. Recommended to pass specific values. Get the list of valid values through the request** `/v1.1/dictionary`
_device_model	**string**	**“MHA-L29”**	**Device model.**
_device_name	**string**	**“HUAWEI Mate 9”**	**Device name.**
_os	**string**	**“Android”**	**Platform’s operating system. Recommended to pass specific values. Get the list of valid values through the request**
_os_ver	**string**	**“v11.0”**	**OS version.**
_os_tz	**string**	**“+0300”**	**Platform time zone. Recommended to pass specific values. Get the list of valid values through the request** `/v1.1/dic`
_os_language	**string**	**“ru”**	**Platform language. Recommended to pass specific values. Get the list of valid values through the request** `/v1.1/dict`
_ad_track	**bool**	**true**	**Ad tracking allowed.**
_ad_id	**string**	**“ad 123”**	**Ad identifier.**
_app_id	**string**	**“app 123”**	**Application ID.**
_app_iid	**string**	**“example iid”**	**Installation identifier of the application.**
_app_ver	**string**	**“v1.2.3.4”**	**Application version.**
_browser_name	**string**	**“Chrome”**	**Browser. Recommended to pass specific values. Get the list of valid values through the request** `/v1.1/dictionary ("br`
_browser_ver	**string**	**“4.5.6.7.8”**	**Browser version.**

Create segment

Description

Create a new segment.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/add/`

Request parameters

Parameter	Type	Example	Required	Description
name	**string**	**“New segment”**	**Yes**	**Segment name**
list_id	**int**	**5**	**Yes, for static segments**	**Database ID from which profiles will be taken. For dynamic and updat**
segment	**JSON array**		**Yes, for dynamic and updatable segments**	**Segmentation conditions. The array must contain at least one object.**
is_refresh_daily	**bool**	**true**	**No**	**Flag to activate segment schedule. By default false. Used only for upd**

Segment type is determined by a combination of parameters:

- `is_static` **is** `false` **— the segment is dynamic**
- `is_static` **is** `true` **and no conditions — the segment is static**
- `is_static` **is** `true` **and conditions present — the segment is updatable. Schedule activity is determir parameter**

Parameters of the `segment` array

Parameter	Type	Example	Required	Description
condition	**string**	**“and”**	**Yes**	**Logical condition for combining queries. Possible values:** `and`, `or`
limit	**int**	**0**	**No**	**Limit the number of profiles in the segment (0 - no limit)**
list_id	**int**	**2**	**Yes**	**Database ID**
percent_from	**int**	**0**	**No**	**Start of the sample percentage range (0-99)**
percent_to	**int**	**99**	**No**	**End of the sample percentage range (1-100)**
queries	**JSON array**	**-**	**Yes**	**Array of filter conditions**

Parameters of the `queries` object

An object that is part of the `queries` array.

Parameter	Type	Example	Required	Description
compo	**string**	**“default”**	**Yes**	**Source of the value for comparison**
name	**string**	**“_sex”, “_region”**	**Yes**	**Field name for filtering**
operator	**string**	**“equal”**	**Yes**	**Comparison operator. Available values depend on the field type.**
type	**string**	**“enum”**	**Yes**	**Field type**
value	**array**	**[{ “name”: “default”, “value”: 2 }]**	**Yes**	**Value for filtering. Format depends on the field type.**

Request example

For static segments

JSON

```
{
    "name": "Static segment",
    "is_static": true,
    "list_id": 5,
    "is_refresh_daily": false
}
```

XML

```
<xml>
  <list_id>5</list_id>
  <name>Static segment</name>
  <is_static>true</is_static>
  <is_refresh_daily>false</is_refresh_daily>
</xml>
```

For dynamic segments

JSON

```
{
  "name": "Dynamic segment",
  "is_static": false,
  "segment": [
    {
      "condition": "or",
      "limit": 0,
      "list_id": 2,
      "percent_from": 0,
      "percent_to": 99,
      "queries": [
        {
          "compo": "default",
          "name": "_sex",
          "operator": "equal",
          "type": "enum",
          "value": [
            {
              "name": "default",
              "value": 1
            }
          ]
        },
        {
          "compo": "default",
          "name": "_sex",
          "operator": "equal",
          "type": "enum",
          "value": [
            {
              "name": "default",
              "value": 2
            }
          ]
        }
      ]
    }
  ]
}
```

XML

```
<xml>
  <name>Dynamic segment</name>
  <is_static>false</is_static>
  <segment>
    <condition>or</condition>
    <limit>0</limit>
    <list_id>2</list_id>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries>
      <compo>default</compo>
      <name>_sex</name>
      <operator>equal</operator>
      <type>enum</type>
      <value>
        <name>default</name>
        <value>1</value>
      </value>
    </queries>
    <queries>
      <compo>default</compo>
      <name>_sex</name>
      <operator>equal</operator>
      <type>enum</type>
      <value>
        <name>default</name>
        <value>2</value>
      </value>
    </queries>
  </segment>
</xml>
```

For updatable segments

JSON

```
{
  "name": "Updatable segment",
  "is_static": true,
  "is_refresh_daily": true,
  "segment": [
        {
            "condition": "or",
            "limit": 0,
            "list_id": 2,
            "percent_from": 0,
            "percent_to": 99,
            "queries": [
                {
                    "compo": "default",
                    "name": "_sex",
                    "operator": "equal",
                    "type": "enum",
                    "value": [
                        {
                            "name": "default",
                            "value": 1
                        }
                    ]
                },
                {
                    "compo": "default",
                    "name": "_sex",
                    "operator": "equal",
                    "type": "enum",
                    "value": [
                        {
                            "name": "default",
                            "value": 2
                        }
                    ]
                }
            ]
        }
    ]
}
```

XML

```
<xml>
  <name>Updatable segment</name>
  <is_static>true</is_static>
  <is_refresh_daily>true</is_refresh_daily>
  <segment>
    <condition>or</condition>
    <limit>0</limit>
    <list_id>2</list_id>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries>
      <compo>default</compo>
      <name>_sex</name>
      <operator>equal</operator>
      <type>enum</type>
      <value>
        <name>default</name>
        <value>1</value>
      </value>
    </queries>
    <queries>
      <compo>default</compo>
      <name>_sex</name>
      <operator>equal</operator>
      <type>enum</type>
      <value>
        <name>default</name>
        <value>2</value>
      </value>
    </queries>
  </segment>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 1
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	D
error	**int**	**Error code**
error_text	**string**	**Error message**
saved_id	**int**	**New segment ID**

Get statistics on resource

Description

Get statistics on profile subscriptions in segment.

> **INFO: Please note**
>
> **The `get_counters` and `refresh_counters` methods work together. Segment statistics are not updated auto `refresh_counters` method, which initiates data recalculation and returns a task ID. After the task is suc checked via `tasks/status`), the updated data becomes available in the `get_counters` method.**

URL

Method: POST

`https://example.com/api/v1.1/segments/get_counters`

Request parameters

Parameter	Type	Example	Required	De
list_id	**int**	**3**	**Yes, for dynamic segment**	**Database ID** **The parameter is used only for the dynamic segment**
resource_id	**int**	**2**	**Yes**	**Resource ID**
id	**int**	**1**	**Yes**	**Segment ID**
format	**int**	**“json”**	**No**	**Response data format** **By default – json**

Request example

JSON

```
{
    "list_id": 1,
    "resource_id": 24,
    "id": 7,
    "format": "json"
}
```

XML

```
<xml>
     <list_id>1</list_id>
     <resource_id>24</resource_id>
     <id>7</id>
     <format>json</format>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "subscribed_sendable": 209,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "subscribed_sendable": 108,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
     <data>
         <channel>email</channel>
         <status>
             <subscribed_sendable>209</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <data>
         <channel>push</channel>
         <status>
             <subscribed_sendable>0</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <data>
         <channel>sms</channel>
         <status>
             <subscribed_sendable>108</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
data	**array of objects**	**Array of received resource channel data**
error	**int**	**Error code**
error_text	**string**	**Error text**

Update statistics on segment

Description

Update the subscription statistics of the profiles in the segment.
Returns a response with task ID without waiting for execution.

> **INFO: Please note**
>
> **The `get_counters` and `refresh_counters` methods work together. Segment statistics are not updated auto `refresh_counters` method, which initiates data recalculation and returns a task ID. After the task is suc checked via `tasks/status`), the updated data becomes available in the `get_counters` method.**

URL

Method: POST

`https://example.com/api/v1.1/segments/refresh_counters`

Request parameters

Parameter	Type	Example	Required	De
list_id	**int**	**3**	**Yes, for dynamic segment**	**Database ID** **The parameter is used only for the dynamic segment**
resource_id	**int**	**2**	**Yes**	**Resource ID**
id	**int**	**1**	**Yes**	**Segment ID**
format	**int**	**“json”**	**No**	**Response data format** **By default – json**

Request example

JSON

```
{
    "list_id": 1,
    "resource_id": 24,
    "id": 7,
    "format": "json"
}
```

XML

```
<xml>
     <list_id>1</list_id>
     <resource_id>24</resource_id>
     <id>7</id>
     <format>json</format>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "55a345c8-3d79-47ef-b03e-b84ce8e0c126"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>55a345c8-3d79-47ef-b03e-b84ce8e0c126</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```

Parameter	Type	Description
data	**JSON objects**	**Contains the task identifier – task_id**
error	**int**	**Error code**
error_text	**string**	**Error text**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "task_id": "930bfdb0-a5a8-4207-a77d-6b989bdea326"
}
```

XML

```
<xml>
     <task_id>930bfdb0-a5a8-4207-a77d-6b989bdea326</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "email",
                        "status": {
                            "subscribed_sendable": 209,
                            "unconfirmed_sendable": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "subscribed_sendable": 0,
                            "unconfirmed_sendable": 0
                        }
                    },
                    {
                        "channel": "sms",
                        "status": {
                            "subscribed_sendable": 0,
                            "unconfirmed_sendable": 0
                        }
                    }
                ],
                "in_progress": false,
                "list_id": 1,
                "refresh_task_id": "",
                "resource_id": 27,
                "updated": "2024-04-20T09:30:01.583Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>email</channel>
                <status>
                    <subscribed_sendable>209</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <subscribed_sendable>0</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <channels>
                <channel>sms</channel>
                <status>
                    <subscribed_sendable>0</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <list_id>1</list_id>
            <refresh_task_id></refresh_task_id>
            <resource_id>27</resource_id>
            <updated>2024-04-20T09:30:01.583Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Add or remove profile

Description

Adds a profile to a static segment or removes a profile by ID (`profile_id`).

Request URL

Method: POST
Add a profile to a segment: `https://example.com/api/v1.1/profiles/segstatic/add`
Remove a profile from a segment: `https://example.com/api/v1.1/profiles/segstatic/remove`

Request parameters

Parameter	Type	Example	Required	
segment_id	**int**	**1**	**Yes**	**Static segment ID**
db_id	**int**	**1**	**Yes**	**Profiles database ID**
profile_id	**String**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**Profile ID in the datebase**
skip_triggers	**bool**	**true**	**No**	**Disables triggers and scenarios configured for e By default – false.**

Request example

JSON

```
{
  "segment_id": 2,
  "db_id": 8,
  "profile_id": "abcdefghijklmnqrstuvwxyz",
  "skip_triggers": true
}
```

XML

```
<xml>
  <segment_id>2</segment_id>
  <db_id>8</db_id>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <skip_triggers>true</skip_triggers>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get profile data in a static or updatable segment

Description

Get the data of profiles that are in a static or updatable segment, page by page.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/segstatic/list`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Static segment ID**
from_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**ID of the first profile to uplo**
from_db	**int**	**1**	**Yes for from_id**	**ID of the database of the fir**
count	**int**	**20**	**No**	**Number of records per page**
fields	**[string]**	`[“custom_field_1”]`	**No**	**List of fields to extract from If empty — no fields are retu**

Request example

JSON

```
{
   "id": 13,
   "from_id": "abcdefghijklmnqrstuvwxyz",
   "from_db": 3,
   "count": 25,
   "fields": [email]
}
```

XML

```
<xml>
    <id>33</id>
    <from_id>abcdefghijklmnqrstuvwxyz</from_id>
    <from_db>75</from_db>
    <count>25</count>
    <fields array="true">email</fields>
</xml>
```

Response example

JSON

```
{
   "next_id": 26,
   "data": [  {
       "profile_id": "abcdefghijklmnqrstuvwxyz",
       "db_id": 3,
       "xxh": "abcdefghijklmxyz",
       "captured": "ISODateString",
       "profile": {profile Object}
   }
]}
```

XML

```
<xml>
  <next_id>26</next_id>
  <data>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <db_id>3</db_id>
    <xxh>abcdefghijklmxyz</xxh>
    <captured>ISODateString</captured>
    <profile></profile>
  </data>
</xml>
```

Response parameters

Parameter	Type	Descr
error	**int**	**Error code**
error_text	**string**	**Error text**
next_id	**int**	**Next profile in the request**
data	**JSON object**	**Customer profile fields**

Update segment

Description

Updates segment building rules.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/update/`

Request parameters

Parameter	Type	Example	Required
list_id	**int**	**5**	**Required for static segments**
id	**int**	**1**	**Yes**
name	**string**	**“New segment”**	**Yes**
segment	**JSON array**		**Yes**

Request example

> **TIP**
>
> **When updating segmentation conditions, the specified profile database cannot be changed**

For static segments

JSON

```
{
    "list_id": 5,
    "name": "Static segment 2",
    "is_static": true,
    "is_refresh_daily": false
}
```

XML

```
<xml>
  <id>2</id>
  <list_id>5</list_id>
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>false</is_refresh_daily>
</xml>
```

For dynamic segments

When updating a dynamic segment, the presence of segmentation rules in the `segment` field is required.

JSON

```
{
    "id": 14,
    "list_id": 6,
    "name": "Static segment 2",
    "is_static": false,
    "segment": [
        {
            "condition": "and",
            "limit": 0,
            "percent_from": 1,
            "list_id": 6,
            "percent_to": 99,
            "queries": [
                {
                    "compo": "domaingroup",
                    "name": "email",
                    "operator": "equal",
                    "type": "email",
                    "value": [
                        {
                            "name": "domaingroup",
                            "value": "Yahoo! (US)"
                        }
                    ]
                }
            ]
        }
    ]
}
```


XML


```
<xml>
  <id>14</id>
  <name>New segment 2</name>
  <is_static>false</is_static>
  <segment array="">
    <list_id>6</list_id>
    <condition>and</condition>
    <limit>0</limit>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries array="">
      <compo>domaingroup</compo>
      <name>email</name>
      <operator>equal</operator>
      <type>email</type>
      <value array="">
        <name>domaingroup</name>
        <value>Yahoo! (US)</value>
      </value>
    </queries>
  </segment>
</xml>
```


For updatable segments

JSON

```
{
    "id": 28,
    "list_id": 6,
    "name": "Static segment 2",
    "is_static": true,
    "is_refresh_daily": true,
    "segment": [
        {
            "condition": "and",
            "limit": 0,
            "percent_from": 1,
            "list_id": 6,
            "percent_to": 99,
            "queries": [
                {
                    "compo": "domaingroup",
                    "name": "email",
                    "operator": "equal",
                    "type": "email",
                    "value": [
                        {
                            "name": "domaingroup",
                            "value": "Yahoo! (US)"
                        }
                    ]
                }
            ]
        }
    ]
}

#### XML

```xml
<xml>
  <id>28</id>
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>true</is_refresh_daily>
  
    <list_id>6</list_id>
    <condition>and</condition>
    <limit>0</limit>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries array="">
      <compo>domaingroup</compo>
      <name>email</name>
      <operator>equal</operator>
      <type>email</type>
      <value array="">
        <name>domaingroup</name>
        <value>Yahoo! (US)</value>
      </value>
    </queries>
  
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 1
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML
```

```
<xml>
    <data>
        <saved_id>14</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Segment ID**

Get segment information

Description

Retrieves a segment detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/get/`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**No**	**Se**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
    <id>10</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "count": 100,
        "id": 1,
        "name": "Segment A",
        "segment": [
            {
                "condition": "or",
                "limit": 0,
                "list_id": 1,
                "percent_from": 0,
                "percent_to": 99,
                "queries": [
                    {
                        "child": [],
                        "compo": "domaingroup",
                        "condition": "",
                        "name": "email",
                        "operator": "equal",
                        "type": "email",
                        "value": [
                            {
                                "name": "domaingroup",
                                "value": "Yahoo! (US)"
                            }
                        ]
                    }
                ]
            }
        ]
      },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <count>0</count>
        <description>""</description>
        <id>14</id>
        <name>"Test segment1"</name>
        <segment>
            <condition>"or"</condition>
            <limit>0</limit>
            <list_id>1</list_id>
            <percent_from>0</percent_from>
            <percent_to>99</percent_to>
            <queries>
                <compo>"default"</compo>
                <condition>""</condition>
                <name>"custom_tags"</name>
                <object_path>""</object_path>
                <object_type>""</object_type>
                <operator>"in_data"</operator>
                <type>"subscription_email"</type>
                <value>
                    <name>"url"</name>
                    <value>"http:127.0.0.1:8080/json.email.subscriptions"</value>
                </value>
                <value>
                    <name>"field"</name>
                    <value>"subscriptions"</value>
                </value>
                <value>
                    <name>"resource_id"</name>
                    <value>"1"</value>
                </value>
            </queries>
        </segment>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Descript
error	**int**	**Error code**
error_text	**string**	**Error text**
count	**int**	**Profiles in the segment**
id	**int**	**Segment ID**
list_id	**int**	**Database id**
name	**string**	**Segment name**
type	**string**	**Type of a segment**
segment	**JSON array**	**segment details**
descr	**string**	**Description of the segment**

Get segments list

Description

Retrieves all segments list.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/list/`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**1**	**No**	**The first ID in the By default – 1**
limit	**int**	**1**	**No**	**Segments list limi By default – 1000**
ui_tags	**[string]**	`[ “tag1”, “tag2” ]`	**No**	**List of tags for se**
groups	**[string]**	`[ “group1”, “group2” ]`	**No**	**List of access gro**
groups_id	**[int]**	`[ 0, 1 ]`	**No**	**List of access gro**

Request example

JSON

```
{
   "limit": 2
}
```

XML

```
<xml>
   <limit>1</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Segment A",
      "ui_tags": ["tag1", "tag2"],
      "Groups": [
        {
          "Id": 1,
          "Name": "subaccount"
        }
      ]
    },
    {
      "id": 2,
      "name": "Segment B",
      "ui_tags": [],
      "Groups": [
        {
          "Id": 0,
          "Name": "main"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 2,
  "total_count": 2
}
```

XML

```
<xml>
    <data>
        <groups>
            <id>1</id>
            <name>"subaccount"</name>
        </groups>
        <id>2</id>
        <name>"Women age 35-60"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>5</next_from_id>
    <total_count>9</total_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**
total_count	**int**	**Total segments count**
id	**int**	**Segment ID**
list_id	**int**	**Database id**
name	**string**	**Segment name**
ui_tags	**[string]**	**List of tags**
type	**string**	**Type of a segment**
groups	**[string]**	**List of groups**

Delete segment

Description

Deletes a segment.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/delete`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Se**

Request example

JSON

```
{
  "id": 1
}
```

XML

```
<xml>
    <id>10</id>
</xml>
```

Response parameters

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Create suppression list

Description

Creates a new suppression list.

Request URL

Method POST

`https://example.com/api/v1.1/suppresses/add`

Request parameters

Parameter	Type	Example	Required	
name	**string**	**“New suppression list”**	**Yes**	**New suppression list**

Request example

JSON

```
{
    "name": "New suppression list"
}
```

XML

```
<xml>
  <name>New suppression list</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 1
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<result>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</result>
```

Response parameters

Parameter	Type	Decription
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**ID of a new suppression list**

Update suppression list

Description

Renames a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/update`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Да**	**Suppression list ID**
name	**string**	**“Suppression list v.2”**	**Да**	**Suppression list new n**

Request example

JSON

```
{
    "id": 2,
    "name": "New suppression list"
}
```

XML

```
<xml>
  <id>2</id>
  <name>Suppression list v.2</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 2
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>2</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get suppression list info

Description

Retrieves suppression list details.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id": 2
}
```

XML

```
<xml>
  <id>1</id>
</xml>
```

JSON

```
{
    "data": {
        "id": 3,
        "name": "New suppression list",
        "is_global": false,
        "description": ""
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <description>""</description>
        <id>1</id>
        <is_global>true</is_global>
        <name>"Global suppress list"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Suppression list ID**
name	**string**	**Suppression list name**
is_global	**bool**	**Global suppression list flag**

Get the list of suppression lists

Description

Retrieves multiple suppression lists information.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**1**	**No**	**The first ID in the resulting li** **By default –** `1`
limit	**int**	**5**	**No**	**Suppression lists limit** **By default –** `1000`
ui_tags	**[]String**	`[ "tag1", "tag2" ]`	**No**	**List of tags for search** **By default –** `[]`
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access groups** **By default –** `[]`
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access groups IDs** **By default –** `[]`

Request example

JSON

```
{
    "from_id": 1,
    "limit":2
}
```

XML

```
<xml>
    <from_id>1</from_id>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "name": "Global suppress list",
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 3,
            "name": "New suppression list",
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 5
}
```


XML


```
<xml>
    <data>
        <groups>
            <id>0</id>
            <name>"main"</name>
        </groups>
        <id>1</id>
        <name>"Global suppress list"</name>
    </data>
    <data>
        <groups>
            <id>0</id>
            <name>"main"</name>
        </groups>
        <id>3</id>
        <name>"New suppression list"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>5</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Suppression list ID**
name	**string**	**Suppression list name**
total_count	**int**	**Total suppression lists count**
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of groups**

Delete suppression list

Description

Deletes a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/delete`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id": 2
}
```

XML

```
<xml>
  <id>4</id>
</xml>
```

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Upload suppression list data to file

Description

Uploads a CSV file containing MD5 data from a suppression list.
To get the result, you need to check the task status and download the resulting file.

Step 1 - creating export task

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/export`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**
type	**string**	**“emails” “phones” “domains” “all”**	**No**	**Type of contact data to export. By default — all.**

Request example

JSON

```
{
    "id": 1,
    "type": "emails"
}
```

XML

```
<xml>
  <id>3</id>
  <type>emails</type>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "c110f0-06d0-4695-9aee-a9bab742d"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>c110f0-06d0-4695-9aee-a9bab742d</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
task_id	**string**	**Export task ID**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 2 - checking export status and getting link

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example	Rec
task_id	**string**	**“c110f0-06d0-4695-9aee-a9bab742d”**	**Yes**

Request example

JSON

```
{
    "task_id": "c110f0-06d0-4695-9aee-a9bab742d"
}
```

XML

```
<xml>
  <task_id>c110f0-06d0-4695-9aee-a9bab742d</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "hashed_file": "b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip",
            "is_api": true,
            "regular_file": "",
            "suppress_id": 1,
            "total": 32
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <status>completed</status>
    <progress>100</progress>
    <result>
      <hashed_file>suppress/b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip</hashed_file>
      <is_api>true</is_api>
      <regular_file></regular_file>
      <suppress_id>1</suppress_id>
      <total>32</total>
    </result>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
status	**string**	**Task status**
progress	**int**	**Task progress**
hashed_file	**string**	**Encrypted path to the file with data in MD5 format**
is_api	**bool**	**Export via API**
suppress_id	**int**	**Suppression list ID**
total	**int**	**Number of lines (email addresses and domains, phones) in the suppression list**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 3 - downloading result file

To download the file, send a GET request to a URL composed of the platform’s base URL, the entity pre the step 2 response:

`https://example.com/api/download/suppress/` **+ value of** `hashed_file`

For example:

`https://example.com/api/download/suppress/abCDefghIklMnop57498257/suppress_list_name.md5.zip`

where:

- `https://example.com` **- platform domain**
- `api/download/suppress/` **- fixed prefix for suppression list data export**
- `abCDefghIklMnop57498257/suppress_list_name.md5.zip` **- value of the** `hashed_file` **parameter from the step 2 re name)**

The request must include the authorization header `Authorization: Bearer <token>`**.**

Suppression list actions

Email address

- **Check if email is suppressed**
- **Add email to suppression list**
- **Add multiple emails to suppression list**
- **Remove email from suppression list**
- **Remove all emails from suppression list**

Email domain

- **Check if domain is suppressed**
- **Add domain to suppression list**
- **Add multiple domains to suppression list**
- **Remove domain from suppression list**
- **Remove all domains from suppression list**

Phone number

- **Check if phone number is suppressed**
- **Add phone number to suppression list**
- **Add multiple phones to suppression list**
- **Remove phone number from suppression list**
- **Remove all phone numbers from suppression list**

Check if email is suppressed

Description

Checks if an email address is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_check`

Request parameters

Parameter	Type	Example	Required	
email	**string**	**“john@example.com”**	**Yes**	**Email address t**
id	**int**	**1**	**Yes**	**Suppression list**

Request example

JSON

```
{
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
  <email>john@example.com</email>
  <id>1</id>
</xml>
```

JSON

```
{
    "domain_matched": false,
    "email_matched": true,
    "error": 0,
    "error_text": "Successful operation",
    "matched": true,
    "phone_matched": false
}
```

XML

```
<xml>
    <domain_matched>false</domain_matched>
    <email_matched>true</email_matched>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <matched>true</matched>
    <phone_matched>false</phone_matched>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**error text**
domain_matched	**bool**	**true, if email domain is on the suppressed domains list**
email_matched	**bool**	**true, if email domain is on the suppressed emails list**
matched	**bool**	**true, if email address is on the suppressed emails or domains list**

Add email to suppression list

Description

Adds an email address to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_add`

Request parameters

Parameter	Type	Example	Required	
email	**string**	**“john@example.com”**	**Yes**	**Email**
id	**int**	**1**	**Yes**	**suppr**

Request example

JSON

```
{
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
    <email>john@example.com</email>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "john@example.com": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <john@example.com>false</john@example.com>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Add multiple emails to suppression list

Description

Adds single or multiple email addresses to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/emails_add`

Request parameters

Parameter	Type	Example	Required	
emails	**JSON array**	`["john@example.com"]`	**Yes**	**Email**
id	**int**	**1**	**Yes**	**Suppr**

Request example

JSON

```
{
    "emails": [
        "example@example.com",
        "example1@example.com",
        "example2@example.com"
    ],
    "id": 3
}
```

XML

```
<xml>
  <emails>example@example.com</emails>
  <emails>example1@example.com</emails>
  <emails>example2@example.com</emails>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3,
    "total_duplicated": 0,
    "total_imported": 3,
    "total_invalid": 0
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
    <total_duplicated>0</total_duplicated>
    <total_imported>3</total_imported>
    <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
total_count	**int**	**Total number of emails**
total_duplicated	**int**	**Emails previously added to the stop list**
total_imported	**int**	**Emails added**

Parameter	Type	Description
total_updated	**int**	**Emails updated (if matched)**

Remove email from suppression list

Description

Removes an email address from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**12**	**Yes**	**Su**
email	**string**	**“example@example.com”**	**Yes**	**En**

> **CAUTION**
>
> **You can use MD5 hash for an email address.**

Request example

JSON

```
{
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
    <email>john@example.com</email>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Remove all emails from suppression list

Description

Removes all email addresses from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/emails_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "removed_count": 1000
}
```

XML

```
<result>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <removed_count>1000</removed_count>
</result>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
remove_count	**int**	**Number of removed email addresses**

Check if domain is suppressed

Description

Check if an email domain is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_check`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppressio**
domain	**string**	**“example.com”**	**Yes**	**Email doma**

Request example

JSON

```
{
    "domain": "example.com",
    "id": 1
}
```

XML

```
<xml>
  <domain>example.com</domain>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "matched": true
}
```

XML

```
<xml>
    <domain_matched>true</domain_matched>
    <email_matched>false</email_matched>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <matched>true</matched>
    <phone_matched>false</phone_matched>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
matched	**bool**	**true, if domain is on the suppression list**

Add domain to suppression list

Description

Adds an email domain to a specified suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_add`

Request parameters

Parameter	Type	Example	Required	
domain	**string**	**“example.com”**	**Yes**	**Email doma**
id	**int**	**1**	**Yes**	**Suppressio**

Request example

JSON

```
{
    "domain": "example.com",
    "id": 1
}
```

XML

```
<xml>
  <domain>example.com</domain>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "example.com": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <example.com>false</example.com>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Add multiple domains to suppression list

Description

Adds one or multiple email domains to a specified suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domains_add`

Request parameters

Parameter	Type	Example	Required	
domains	**JSON array**	**[“example.com”]**	**Yes**	**Emai**
id	**int**	**1**	**Yes**	**Supp**

Request example

JSON

```
{
    "domains": [
        "example.com",
        "example1.com"
    ],
    "id": 3
}
```

XML

```
<xml>
  <domains>example1.com</domains>
  <domains>example2.com</domains>
  <domains>example3.com</domains>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3,
    "total_duplicated": 0,
    "total_imported": 3,
    "total_invalid": 0
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
    <total_duplicated>0</total_duplicated>
    <total_imported>3</total_imported>
    <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
total_count	**int**	**Total number of domains**
total_duplicated	**int**	**Domains previously added to the stop list**
total_imported	**int**	**Domains added**
total_updated	**int**	**Domains updated (if matched)**

Remove domain from suppression list

Description

Removes an email domain from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppressio**
domain	**string**	**“example.com”**	**Yes**	**Email doma**

> **CAUTION**
>
> **You can use domain MD5-hash sum.**

Request example

JSON

```
{
    "domain": "example.com",
    "id": 3
}
```

XML

```
<xml>
  <domain>test.com</domain>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Remove all domains from suppression list

Description

Removes all email domains from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domains_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id": 3
}
```

XML

```
<xml>
    <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "removed_count": 100
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <removed_count>100</removed_count>
</xml>
```

Response parameters

Parameter	Type	Descriptio
error	**int**	**Error code**
error_text	**string**	**Error text**
removed_count	**int**	**Number of domains removed**

Check if phone number is suppressed

Description

Checks if a phone number is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_check`

Request parameters

Parameter	Type	Example	Required	
value	**string**	**“+79000000001”**	**Yes**	**Phone number to che**
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id":1,
    "value":"+79000000001"
}
```

XML

```
<xml>
  <id>1</id>
  <value>+79000000001</value>
</xml>
```

Response example

JSON

```
{
    "domain_matched": false,
    "email_matched": false,
    "error": 0,
    "error_text": "Successful operation",
    "matched": true,
    "phone_matched": true
}
```

XML

```
<xml>
  <domain_matched>false</domain_matched>
  <email_matched>false</email_matched>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <matched>true</matched>
  <phone_matched>true</phone_matched>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
domain_matched	**bool**	**true, if value is in the suppressed domains list**
email_matched	**bool**	**true, if value is in the suppressed emails list**
phone_matched	**bool**	**true, if value is in the list of suppressed phone numbers**
matched	**bool**	**true, if value is in the suppressed emails, domains or phones list**

Add phone number to suppression list

Description

Adds a phone number to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_add`

Request parameters

Parameter	Type	Example	Required	
value	**string**	**“+79000000001”**	**Yes**	**Phone nur**
id	**int**	**1**	**Yes**	**suppressi**

Request example

JSON

```
{
  "id": 1,
  "value":"+79000000001"
}
```

XML

```
<xml>
    <id>1</id>
    <value>+79000000001</value>
</xml>
```

Response example

JSON

```
{
    "data": {
        "+79000000991": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data><+79000000001>false</+79000000001>
</data>
<error>0</error>
<error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**obj**	**Phone number added to the suppression list**

Add multiple phones to suppression list

Description

Adds single or multiple phone numbers to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phones_add`

Request parameters

Parameter	Type	Example	Required	
values	**JSON array**	[“+79009009001”, “+79009009002”]	**Yes**	**List of phone n**
id	**int**	**1**	**Yes**	**Suppression lis**

Request example

JSON

```
{
    "id": 1,
    "values": [
        "+79009009001",
        "+79009009002"
    ]
}
```

XML

```
<xml>
  <id>1</id>
  <values>+79009009001</values>
  <values>+79009009002</values>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 2,
    "total_duplicated": 0,
    "total_imported": 2,
    "total_invalid": 0
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>2</total_count>
  <total_duplicated>0</total_duplicated>
  <total_imported>2</total_imported>
  <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
total_count	**int**	**Total number of phone numbers**
total_duplicated	**int**	**Phone numbers previously added to the stop list**
total_imported	**int**	**Phone numbers added**
total_invalid	**int**	**Number of invalid phone numbers**

Remove phone number from suppression list

Description

Removes a phone number from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**12**	**Yes**	**Suppressi**
value	**string**	**“+79009009001”**	**Yes**	**Phone nun**

Request example

JSON

```
{
    "id":1,
    "value":"+79009009001"
}
```

XML

```
<xml>
  <id>1</id>
  <value>+79009009001</value>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Remove all phone numbers from suppression list

Description

Removes all phone numbers from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phones_remove`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Suppression list ID**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "removed_count": 500
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <removed_count>4</removed_count>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "removed_count": 500
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <removed_count>500</removed_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**

Parameter	Type	Description
error_text	**string**	**Error text**
remove_count	**int**	**Number of removed phones**

Get templates list

Description

Retrieves a list of message templates.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**1**	**No**	**The first template ID in the list. By default – 1**
limit	**int**	**100**	**No**	**Message list limit. By default – 1000**
channel	**string**	**“email” “sms” “push” “telegram_bot” “telegram_group” “custom_channel”**	**No**	**The channel that uses the template By default – all channels**
ui_tags	**[string]**	`[ "tag1", "tag2" ]`	**No**	**List of tags to search By default –** `[]`
groups	**[string]**	`[ "group1", "group2" ]`	**No**	**List of groups By default –** `[]`
groups_id	**[int]**	`[ 0, 1 ]`	**No**	**List of group IDs By default –** `[]`

Request example

JSON

```
{
           "limit": 2
}
```

XML

```
<xml>
  <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "update_time": "2025-05-21T09:17:53.362Z",
            "name": "Message A",
                        "channels": [
                "email"
            ]
                   },
        {
            "id": 2,
            "name": "Message B",
                        "channels": [
                "push"
            ]
                   }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 10
}
```

XML

```
<xml>
  <data>
    <id>1</id>
    <update_time>2025-05-21T09:17:53.362Z</update_time>
    <name>Message A</name>
    <channels>email</channels>
  </data>
  <data>
    <id>2</id>
    <name>Message B</name>
    <channels>push</channels>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>10</total_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
total_count	**int**	**Total number of templates**
id	**int**	**Message template ID**
name	**string**	**Template name**
channels	**array**	**Channels for which the template is configured**
is_draft	**bool**	**If this is a draft, the template cannot be sent.**
is_confirm	**bool**	**Is there a confirmation link in the body of the email or not.**
ui_tags	**[]String**	**List of tags**
groups	**array of objects**	**List of groups (group ID + group name)**
update_time	**string**	**When the scenario was updated**

Get template info

Description

Retrieves a single message template information.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/get`

Request parameters

Parameter	Type	Example	Required	Description
id	int	1	Yes	Template ID
channel_sid	array of strings	[“email”, “sms”, “cc_channel”]	No	Channel filter. If specified, returns only templates for the specified channels. By d
get_variable	boolean	true	No	If true, the response includes a list of variables (params) used in the template. By

Available variable types

Profile data

`lead.fname` , `lead.lname` , `lead.email` , `lead.phone`
`lead.birthdate` , `lead.gender` , `lead._city` , `lead._country`
`lead._xxh` , `lead_custom.*`

Parameters

`campid` , `workflow_id` , `workflow_name` , `msg_type`
`attribute_value.*` , `node_id`

Date and time

`datenow` , `yearlong` , `hourse` , `minutes` , `ampm`

Apicontent variables

`apicontent.field_1` , `apicontent.field_2` , `apicontent.field_N`

Segmentation

`segname`

Learn more about template variables in this article.

Request example

JSON

```
{
    "id": 1,
    "channel_sid": ["email", "sms", "cc_channel"],
    "get_variable": true
}
```

XML

```
<xml>
  <id>1</id>
  <channel_sid>email</channel_sid>
  <channel_sid>sms</channel_sid>
  <channel_sid>cc_channel</channel_sid>
  <get_variable>true</get_variable>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 1,
        "update_time": "2025-05-21T09:17:53.362Z",
        "name": "example template name",
        "description": "example template description",
        "channels": [
            {
                "id": 1,
                "sid": "email",
                "fields": {
                    "text": "example text version",
                    "html": "example html content",
                    "subject": "example subject",
                    "from_email": "example@example.com",
                    "from_name": "example from name",
                    "replyto_email": "reply@example.com",
                    "replyto_name": "example reply name",
                    "is_not_inline": true,
                    "attach": []
                },
                "params": [
                    "lead.fname",
                    "datenow",
                    "segname",
                    "apicontent.field_name",
                    "workflow_name"
                ],
                "presets": null
            },
            {
                "id": 2,
                "sid": "sms",
                "fields": {
                    "sms": "example sms content",
                    "sms_max_count": 0,
                    "add_sms_url_scheme": false
                },
                "params": [
                    "lead.fname",
                    "datenow",
                    "segname",
                    "apicontent.field_name",
                    "workflow_name"
                ],
                "presets": null
            }
        ],
        "json_url": "https://example.com/examples/message.json",
        "is_draft": true,
        "is_confirm": true,
        "ui_tags": [
            "example_tag_1",
            "example_tag_2"
        ],
        "groups": [
            {
                "id": 2,
                "name": "example_group_1"
            },
            {
                "id": 3,
                "name": "example_group_2"
            }
        ],
        "suppress_id": 1
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
    <id>1</id>
    <update_time>2025-05-21T09:17:53.362Z</update_time>
    <name>example template name</name>
    <description>example template description</description>
    <channels>
      <id>1</id>
      <sid>email</sid>
      <fields>
        <text>example text version</text>
        <html>example html content</html>
        <subject>example subject</subject>
        <from_email>example@example.com</from_email>
        <from_name>example from name</from_name>
        <replyto_email>reply@example.com</replyto_email>
        <replyto_name>example reply name</replyto_name>
        <is_not_inline>true</is_not_inline>
        <attach array='true'></attach>
      </fields>
      <params>lead.fname</params>
      <params>datenow</params>
      <params>segname</params>
      <params>apicontent.field_name</params>
      <params>workflow_name</params>
      <presets></presets>
    </channels>
    <channels>
      <id>2</id>
      <sid>sms</sid>
      <fields>
        <sms>example sms content</sms>
        <sms_max_count>0</sms_max_count>
        <add_sms_url_scheme>false</add_sms_url_scheme>
      </fields>
      <params>lead.fname</params>
      <params>datenow</params>
      <params>segname</params>
      <params>apicontent.field_name</params>
      <params>workflow_name</params>
    </channels>
    <json_url>https://example.com/examples/message.json</json_url>
    <is_draft>true</is_draft>
    <is_confirm>true</is_confirm>
    <ui_tags>example_tag_1</ui_tags>
    <ui_tags>example_tag_2</ui_tags>
    <groups>
      <id>2</id>
      <name>example_group_1</name>
    </groups>
    <groups>
      <id>3</id>
      <name>example_group_2</name>
    </groups>
    <suppress_id>1</suppress_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Template ID**
name	**string**	**Template name**
description	**string**	**Template description**
channels	**array of objects**	**Template settings data for each channel**
channels[].id	**int**	**Channel ID in template**
channels[].sid	**string**	**System channel identifier (email, sms, push, etc.)**
channels[].fields	**object**	**Template content settings for channel**
channels[].params	**array of strings**	**List of variables used in the template. Returned only when `get_variable: true` in request.**
channels[].presets	**object|null**	**Preset channel settings**
json_url	**string**	**Dynamic JSON URL**
is_draft	**bool**	**Draft template flag. If this is a draft, the template cannot be sent.**

Parameter	Type	Description
is_confirm	**bool**	**Confirm subscription link flag. Is there a confirmation link in the body of the email or not.**
ui_tags	**[]String**	**List of tags to search**
groups	**array of objects**	**List of groups**
suppress_id	**int**	**Suppression list ID**
update_time	**string**	**When the template was updated**

Delete template

Description

Deletes a message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/delete`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Tem**

Request example

JSON

```
{
         "id": 1
}
```

XML

```
<xml>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": 97,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>97</data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descrip
data	**int**	**Deleted template ID**
error	**int**	**Error code**
error_text	**string**	**Error text**

Add template

Description

Adds a new message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/add`

Request parameters

Parameter	Type	Example	Required	
name	**string**	`"Subscription confirmation template"`	**Yes**	**Template name**
descr	**string**	`"Template used in Double Opt-In scenario"`	**No**	**Template description**

channels	**[]channels**	<pre>“channels”: [
		{
		“sid”: “email”,
		“fields”:{
		“text”: “Hello, world!”,
		“html”: “Hello, world!”,
		“subject”: “Subject line”,
		“from_email”: “”,
		“from_name”: “”,
		“replyto_email”: “”,
		“replyto_name”: “”
		}
		}
		]</pre>
			Yes	**Template settings for a certain cha**
				
				Learn more

json_url	**string**	`"https://altcraft.com/public/assets/example/test.json"`	**No**	**Link to dynamic JSON content**
ui_tags	**[string]**	`["tag_1","tag_2"]`	**No**	**List of tags**
groups	**[int]**	`[1,2]`	**No**	**List of group IDs**
is_draft	**bool**	`false`	**No**	**Draft template flag.** **If this is a draft, the template cann**
is_confirm	**bool**	`true`	**No**	**Confirm subscription link flag.** **Is there a confirmation link in the t**
suppress_id	**int**	`2`	**No**	**Suppression list ID**

Request example

INFO

Every template should contain unsubscribe link.

JSON

```
{
    "name": "Subscription confirmation template",
    "descr":"Template used in Double Opt-In scenario",
    "channels": [
        {
            "sid": "email",
            "fields": {
                "text": "Hello, world!",
                "html": "<html>Hello, world!</html>",
                "subject": "Тема письма",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "is_not_inline": true
            }
        }
    ]
    "json_url": "https://altcraft.com/example/message.json",
    "ui_tags": ["tag_1","tag_2"],
    "groups": [1,2],
         "is_draft": false,
    "is_confirm": false,
    "suppress_id": 2
}
```

XML

```
<xml>
    <name>Template</name>
    <descr>Template description</descr>
    <channels array = 'true'>
        <fields>
            <body>Hello, world!</body>
            <click_url>https://example.com</click_url>
            <icon>https://example.com/images/picture.jpg</icon>
            <title>Welcome</title>
        </fields>
        <sid>push</sid>
    </channels>
    <is_confirm>false</is_confirm>
    <is_draft>false</is_draft>
    <json_url>https://altcraft.com/example/message.json</json_url>
    <suppress_id>2</suppress_id>
    <groups>1</groups>
    <groups>2</groups>
    <ui_tags>tag_1</ui_tags>
    <ui_tags>tag_2</ui_tags>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 10
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>10</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descrip
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**string**	**Created template ID**

Update template

Description

Add a new message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/update`

Request parameters

Parameter	Type	Example	Required	
id	**int**	`1`	**Yes**	**Template ID**
name	**string**	`"Subscription confirmation template"`	**Yes**	**Template name**
descr	**string**	`"Template used in Double Opt-In scenario"`	**No**	**Template description**

channels	**[]channels**	<pre>“channels”: [
		  {
		  “sid”: “email”,
		  “fields”:{
		  “text”: “Hello, world!”,
		  “html”: “Hello, world!”,
		  “subject”: “Subject line”,
		  “from_email”: “”,
		  “from_name”: “”,
		  “replyto_email”: “”,
		  “replyto_name”: “”
		  }
		  }
		]</pre>
			Yes	**Template settings for a certain cha**
				
				Learn more

json_url	**string**	`"https://altcraft.com/public/assets/example/test.json"`	**No**	**Link to dynamic JSON content**
ui_tags	**[string]**	`["tag_1","tag_2"]`	**No**	**List of tags**
groups	**[int]**	`[1,2]`	**No**	**List of group IDs**
is_draft	**bool**	`false`	**No**	**Draft template flag.** **If this is a draft, the template cann**
is_confirm	**bool**	`true`	**No**	**Confirm subscription link flag.** **Is there a confirmation link in the t**
suppress_id	**int**	`2`	**No**	**Suppression list ID**

Request example

JSON

```
{
  "id": 10,
  "name": "Название шаблона",
  "descr": "Описание шаблона",
  "channels": [
    {
      "sid": "email",
      "fields": {
        "text": "Текстовая версия письма",
        "html": "<html>Hello, world!</html>",
        "subject": "Шаблон сообщения",
        "from_email": "",
        "from_name": "",
        "replyto_email": "",
        "replyto_name": "",
        "is_not_inline": true
      }
    }
  ],
  "json_url": "https://altcraft.com/public/assets/example/test.json",
  "ui_tags": ["tag_1", "tag_2"],
  "groups": [],
  "is_draft": false,
  "is_confirm": false,
  "suppress_id": 4
}
```

XML

```
<xml>
  <channels array = 'true'>
    <fields>
      <body>Hello, world!</body>
      <click_url>https://example.com</click_url>
      <icon>https://example.com/images/picture.jpg</icon>
      <title>Welcome</title>
    </fields>
    <sid>push</sid>
  </channels>
  <descr>Template description</descr>
  <groups>1</groups>
  <groups>2</groups>
  <is_confirm>false</is_confirm>
  <is_draft>false</is_draft>
  <json_url>https://altcraft.com/example/message.json</json_url>
  <id>10</id>
  <name>Template</name>
  <suppress_id>2</suppress_id>
  <ui_tags>tag_1</ui_tags>
  <ui_tags>tag_2</ui_tags>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 10
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>10</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descript
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**string**	**Updated template ID**

Channel object

Used in API requests related to adding and updating message templates.

JSON

```
{
        "sid": "email",
        "fields": {
                "text": "Hello, world!",
                "html": "<html>Hello, world!</html>",
                "subject": "Welcome"
        }
},
{
        "sid": "sms",
        "fields": {
                "sms": "Hello, world!"
        }
},
{
        "sid": "push",
        "fields": {
                "title": "Welcome",
                "body": "Hello, world!",
                "icon": "https://example.com/images/picture.jpg",
                "click_url": "https://example.com"
        }
}
```

Description

Parameter	Type	Example	Description
sid	**string**	**“email”** **“sms”** **“push”** **“telegram_bot”** **“telegram_group”** **“custom_channel”**	**Channel name**

fields	**object**	`“fields”:{`
		`“text”: “Hello, world”,`
		`“html”: “Hello, world!”,`
		`“subject”: “Welcome”`
		`}`
			Template settings for a certain channel

Fields object

Each `channel` object contains a `fields` object:

Email fields

JSON

```
{
        "sid": "email",
        "fields": {
                "text": "Hello, world!",
                "html": "<html>Hello, world!</html>",
                "subject": "Subject line",
                "from_email": "test@example.com",
                "from_name": "Jenny's List",
                "replyto_email": "test_reply@example.com",
                "replyto_name": "Jenny",
                "is_not_inline": true
        }
}
```

Description

Parameter	Type	Example	D
text	**string**	`"Hello, world!"`	**Text version of the message**
html	**string**	`"<html>Hello, world!</html>"`	**HTML version of the message**
is_amp	**bool**	`false`	**Is there an amp version in the template or not**
amp	**string**	`"<html>Hello, world!</html>"`	**AMP HTML version of the message**
subject	**string**	`"Weather alert"`	**Subject line**
from_email	**string**	`"test@example.com"`	**Override “from-email”**

Parameter	Type	Example	D
from_name	**string**	`"Jenny's List"`	**Override “from-name”**
replyto_email	**string**	`"test_reply@example.com"`	**Override “replyto_email”**
replyto_name	**string**	`"Jenny"`	**Override “replyto_name”**
is_not_inline	**bool**	`true`	**Do not convert CSS to inline**

SMS fields

JSON

```
{
	"sid": "sms",
	"fields": {
		"sms": "Hello, world!",
		"sms_max_count": 0,
		"add_sms_url_scheme": true
	}
}
```

Description

Parameter	Type	Example	Description
sms	**string**	`"Hello, world!"`	**Message text**
sms_max_count	**int**	`0`	**SMS parts limit(/en/user-guide/message-templates/managing-templates/sms#character-limit)**
add_sms_url_scheme	**bool**	`true`	**Add “http://” to links**

Push fields

JSON

```
{
	"sid": "push",
	"fields": {
		"title": "Welcome",
		"body": "Hello, world!",
		"icon": "https://example.com/images/icon.jpg",
		"click_url": "https://example.com",
		"custom_push_json_raw": "{}"
	}
}
```

Description

Parameter	Type	Example	Description
title	**string**	`"Welcome"`	**Notification title**
body	**string**	`"Hello, world!"`	**Notification text**
icon	**string**	`"https://example.com/images/test.jpg"`	**URL where the notification ico**
click_url	**string**	`"https://example.com"`	**Click-to-go link**
custom_push_json_raw	**string**	`"{}"`	**Custom JSON to send to mobil**

Notify fields

JSON

```
{
  "sid": "notify",
  "fields": {
    "template": {
      "id": 1,
      "name": "message_after_purchase",
      "type": "NOTIFY",
      "status": "NOT_SENT",
      "is_draft": true,
      "edna_account_id": "",
      "notify_service_name": "altcraft_testing",
      "vk_data": {
        "body": "Notify message content",
        "buttons": [],
        "attachments": [],
        "enabled": true,
        "inline_keyboard": false,
        "hide_keyboard": false,
        "reply_enabled": false
      },
      "ok_data": {
        "enabled": true,
        "body": "Notify message content"
      },
      "sms_data": {
        "enabled": true,
        "body": "Notify message content",
        "type": "advert"
      },
      "routes": [
        {
          "route": "vk",
          "deliv_timeout": 86400,
          "read_timeout": 86400
        },
        {
          "route": "ok",
          "deliv_timeout": 86400,
          "read_timeout": 86400
        },
        {
          "route": "sms"
        }
      ]
    },
    "info": {
      "is_obsolete": false,
      "not_found": false
    },
    "params": {
      "name": "{{$.Lead._fname}}",
      "order_number": "{{$.Lead.order_number}}"
    }
  }
}
```


XML

```
<xml>
  <sid>notify</sid>
  <fields>
                <info>
                    <is_obsolete>false</is_obsolete>
                    <not_found>false</not_found>
                </info>
                <params>
                    <name>{{$.Lead._fname}}</name>
                    <order_number>{{$.Lead.order_number}}</order_number>
                </params>
                <template>
                    <id>1</id>
                    <is_draft>true</is_draft>
                    <name>message_after_purchase</name>
                    <notify_service_name>altcraft_testing</notify_service_name>
                    <ok_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                    </ok_data>
                    <routes>
                        <deliv_timeout>86400</deliv_timeout>
                        <read_timeout>86400</read_timeout>
                        <route>vk</route>
                    </routes>
                    <routes>
                        <deliv_timeout>86400</deliv_timeout>
                        <read_timeout>86400</read_timeout>
                        <route>ok</route>
                    </routes>
                    <routes>
                        <route>sms</route>
                    </routes>
                    <sms_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                        <type>advert</type>
                    </sms_data>
                    <status>NOT_SENT</status>
                    <type>NOTIFY</type>
                    <vk_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                        <hide_keyboard>false</hide_keyboard>
                        <inline_keyboard>false</inline_keyboard>
                        <reply_enabled>false</reply_enabled>
                    </vk_data>
                </template>
            </fields>
</xml>
```

Description

Parameter	Type	Example	
name	**string**	**“message_after_purchase”**	**Template name**
type	**string**	**“NOTIFY”**	**Template type**
status	**string**	**“NOT_SENT”**	**Sending status of the template**
is_draft	**boolean**	**true**	**Indicates if the template is a draft**
edna_account_id	**string**	**“123456”**	**Edna account ID**
notify_service_name	**string**	**“altcraft_testing”**	**Name of the notification service**
body	**string**	**“Notify message content”**	**VK message body**
buttons	**array**	**[]**	**VK message buttons**
attachments	**array**	**[]**	**VK message attachments**
enabled	**boolean**	**true**	**Whether VK message is enabled**
inline_keyboard	**boolean**	**false**	**Whether VK inline keyboard is used**
hide_keyboard	**boolean**	**false**	**Whether VK keyboard should be hidde**
reply_enabled	**boolean**	**false**	**Whether replies to VK message are all**
enabled	**boolean**	**true**	**Whether OK message is enabled**
body	**string**	**“Notify message content”**	**OK message body**
enabled	**boolean**	**true**	**Whether SMS message is enabled**
body	**string**	**“Notify message content”**	**SMS message body**
type	**string**	**“advert”**	**SMS message type**
route	**string**	**“vk”**	**Delivery channel**
deliv_timeout	**integer**	**86400**	**Delivery timeout (in seconds)**

Parameter	Type	Example	
read_timeout	**integer**	**86400**	**Read timeout (in seconds)**
route	**string**	**“ok”**	**Second delivery channel**
route	**string**	**“sms”**	**Third delivery channel**
is_obsolete	**boolean**	**false**	**Whether the template is obsolete**
not_found	**boolean**	**false**	**Whether the template was found**

Telegram fields

JSON

```
{
  "sid": "telegram_bot",
  "fields": {
    "image": {
      "attach": [
        {
          "content": "/9j/bcvb4234hgdffsd45",
          "filekey": "67863659182463338076 2bee",
          "name": "example.jpg",
          "size": 65101,
          "type": "image_path"
        }
      ]
    },
    "image_caption": "Telegram bot message description",
    "markup_style": "HTML"
  }
}
```

XML

```
<xml>
  <sid>telegram_bot</sid>
  <fields>
    <image>
      <attach>
          <content>/9j/bcvb4234hgdffsd45</content>
          <filekey>67863659182463338076 2bee</filekey>
          <name>example.jpg</name>
          <size>65101</size>
          <type>image_path</type>
      </attach>
    </image>
    <image_caption>Telegram bot message description</image_caption>
    <markup_style>HTML</markup_style>
  </fields>
</xml>
```

Description

Parameter	Type	Example	
legacy_button_link	**string**	**“https://example.com”**	**Link attached to the button**
legacy_button_name	**string**	**“button1”**	**Button label**
markdown_type	**string**	**“HTML”**	**Text formatting type (HTML/Markdow**
text	**string**	**“example”**	**Main message text**
content	**string**	**“/9j/bcvb4234hgdffsd45”**	**File content in binary (base64)**
filekey	**string**	**“67863659182463338076 2bee”**	**Unique file identifier**
name	**string**	**“example.jpg”**	**Name of the attached file**
size	**integer**	**65101**	**File size in bytes**
type	**string**	**“image_path”**	**Attachment type (e.g., path to file)**
image_caption	**string**	**“Image description”**	**Caption for the image**
audio_caption	**string**	**“Audio file description”**	**Caption for the audio file**
video_caption	**string**	**“Video file description”**	**Caption for the video file**
markup_style	**string**	**“HTML”**	**Markup style for Telegram (HTML/Ma**

Whatsapp fields

JSON

```
{
  "id": 9,
  "sid": "whatsapp",
  "fields": {
    "type": "TEMPLATE",
    "template": {
      "type": "360DIALOG",
      "name": "initiate_message_en",
      "hsm_external_id": "c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en",
      "language": "en",
      "wa_account_id": "+79999999999",
      "hsm_hash_sum": "0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1",
      "is_invalid": false,
      "hsm_not_found": false,
      "account_not_found": false
    },
    "raw_hsm": {
      "dialog_template": {
        "components": [
          {
            "type": "BODY",
            "text": "Hello! Thank you for waiting. I needed some time to clarify the information regarding your request. Could you please l
discuss the details? 😀"
          }
        ],
        "name": "initiate_message_en",
        "namespace": "c466caf6_c340_4d85_8659_6c9685618a9f",
        "category": "ISSUE_RESOLUTION",
        "language": "en",
        "rejected_reason": "NONE",
        "status": "approved"
      },
      "edna_template": null
    }
  }
}
```


XML


```
<xml>
  <id>9</id>
  <sid>whatsapp</sid>
  <fields>
    <type>TEMPLATE</type>
    <template>
      <type>360DIALOG</type>
      <name>initiate_message_en</name>
      <hsm_external_id>c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en</hsm_external_id>
      <language>en</language>
      <wa_account_id>+79999999999</wa_account_id>
      <hsm_hash_sum>0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1</hsm_hash_sum>
      <is_invalid>false</is_invalid>
      <hsm_not_found>false</hsm_not_found>
      <account_not_found>false</account_not_found>
    </template>
    <raw_hsm>
      <dialog_template>
        <components>
          <component>
            <type>BODY</type>
            <text>Hello! Thank you for waiting. I needed some time to clarify the information regarding your request. Could you please let
the details? 😀</text>
          </component>
        </components>
        <name>initiate_message_en</name>
        <namespace>c466caf6_c340_4d85_8659_6c9685618a9f</namespace>
        <category>ISSUE_RESOLUTION</category>
        <language>en</language>
        <rejected_reason>NONE</rejected_reason>
        <status>approved</status>
      </dialog_template>
    </raw_hsm>
  </fields>
</xml>
```


Description

Parameter	Type	Example	
type	**string**	**“360DIALOG”**	**WhatsApp template t**
name	**string**	**“initiate_message_en”**	**Template name**
hsm_external_id	**string**	**“c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en”**	**External template ID**
language	**string**	**“en”**	**Template language**
wa_account_id	**string**	**“+79999999999”**	**WhatsApp account ID**
hsm_hash_sum	**string**	**“0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1”**	**Template hash**
is_invalid	**boolean**	**false**	**Whether the template**
hsm_not_found	**boolean**	**false**	**Whether the HSM tem**
account_not_found	**boolean**	**false**	**Whether the account**
name (in raw_hsm)	**string**	**“initiate_message_en”**	**Template name in** `raw`
namespace	**string**	**“c466caf6_c340_4d85_8659_6c9685618a9f”**	**Template namespace**
category	**string**	**“ISSUE_RESOLUTION”**	**Template category**
status	**string**	**“approved”**	**Template status (e.g.,**
rejected_reason	**string**	**“NONE”**	**Reason for template r**
components.type	**string**	**“BODY”**	**Component type**
components.text	**string**	**“Hello! Thank you for waiting...”**	**Message body text**
edna_template	**string**	**“null”**	**Edna template, if pres**

Viber™ fields

JSON

```
"id": 10,
{
  "sid": "viber",
  "fields": {
    "series": [
      {
        "type": "TEXT",
        "text": "message text"
      },
      {
        "type": "IMAGE",
        "image": "/849074172/3/example.jpg"
      },
      {
        "type": "VIDEO",
        "thumbnail": "/848985842/3/67863234582463338076 2bee/example.jpg",
        "duration": 6,
        "file_size": 2,
        "action": "https://example.com/lib/preview/mp4/example.mp4"
      },
      {
        "type": "TEXT_BUTTON",
        "text": "Sample text",
        "caption": "button text",
        "action": "https://example.com"
      },
      {
        "type": "FILE",
        "file_type": "pdf",
        "file_name": "example.pdf",
        "action": "https://www.example.org/testfiles/resources/pdf/example.pdf"
      }
    ]
  }
}
```

XML

```
<xml>
    <id>10</id>
    <sid>viber</sid>
    <fields>
      <series>
        <element>
          <type>TEXT</type>
          <text>message text</text>
        </element>
        <element>
          <type>IMAGE</type>
          <image>/849074172/3/example.jpg</image>
        </element>
        <element>
          <type>VIDEO</type>
          <thumbnail>/848985842/3/6786323458246333807 62bee/example.jpg</thumbnail>
          <duration>6</duration>
          <file_size>2</file_size>
          <action>https://example.com/lib/preview/mp4/example.mp4</action>
        </element>
        <element>
          <type>TEXT_BUTTON</type>
          <text>Sample text</text>
          <caption>button text</caption>
          <action>https://example.com</action>
        </element>
        <element>
          <type>FILE</type>
          <file_type>pdf</file_type>
          <file_name>example.pdf</file_name>
          <action>https://www.example.org/testfiles/resources/pdf/example.pdf</action>
        </element>
      </series>
    </fields>
</xml>
```

Description

Parameter	Type	Example	
type	**string**	**“TEXT”**	**Type of message ele**
text	**string**	**“message text”**	**Text content of the m**
image	**string**	**“/849074172/3/example.jpg”**	**Path to the image**
thumbnail	**string**	**“/848985842/3/6786323458246333807 62bee/example.jpg”**	**Preview image for th**
duration	**integer**	**6**	**Video duration in sec**
file_size	**integer**	**2**	**File size in megabyte**
action	**string**	**“https://example.com/lib/preview/mp4/example.mp4”**	**Action URL (used for**
caption	**string**	**“button text”**	**Caption for the butto**
file_type	**string**	**“pdf”**	**Type of the file (e.g.,**
file_name	**string**	**“example.pdf”**	**Name of the attache**

**** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Get campaign information

Description

Retrieve detailed information about a campaign, including audience settings, LCG, marketing activities

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — View object**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings — Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/get
```

Request parameters

Parameter	Type	Example	Required	
id	**int**	**22**	**Yes**	**Campaign ID**
marketing_activities	**bool**	**false**	**No**	**Include the list of marketing activities default – false**

Request example

JSON

```
{
  "id": 22,
  "marketing_activities": true
}
```

XML

```
<xml>
  <id>22</id>
  <marketing_activities>true</marketing_activities>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 22,
      "is_delete": false,
      "createtime": "2025-08-22T09:00:21.654Z",
      "updatetime": "2025-08-22T11:08:10.216Z",
      "name": "Seasonal sale",
      "description": "Campaign for promoting the spring collection",
      "tags": [],
      "state": "active",
      "audience": {
        "state": "ready",
        "seg_id": 4,
        "list_id": 2,
        "extension_conf": {
          "workflows": {"ids": []},
          "triggers": {"ids": []},
          "broadcasts": {"ids": [1]},
          "regulars": {"ids": []},
          "abns": {"ids": []},
          "placements": {"ids": []}
        },
        "lcg": {
          "stratification_field_settings": [
            {
              "field_name": "_sex",
              "field_type": "enum",
              "condition_type": "list",
              "conditions": [
                {"list": [1]},
                {"list": [2]}
              ]
            },
            {
              "field_name": "_region",
              "field_type": "unicode",
              "condition_type": "list",
              "conditions": [
                {"list": ["Moscow", "Saint Petersburg"]},
                {"list": ["Kazan"]}
              ]
            }
          ],
          "ignore_entities": {
            "workflows": {"ids": []},
            "triggers": {"ids": []},
            "broadcasts": {"ids": [1]},
            "regulars": {"ids": []},
            "abns": {"ids": []},
            "placements": {"ids": []}
          },
          "size_formation_settings": {
            "limit": 133333,
            "percent": "16.00"
          }
        }
      },
      "active_period": {
        "settings": {
          "tz": "UTC",
          "end_at": "2025-08-24T12:03:21.93Z"
        },
        "history": {
          "start": "2025-08-22T09:00:47.786Z"
        }
      },
      "counters": {
        "segment": 631,
        "test_group": 530,
        "local_control_group": 101,
        "global_control_group": 0
      },
      "marketing_activities_counter": 2,
      "groups": [
        {
          "id": 0,
          "name": "main"
        }
      ],
```

```
      "attributes": {
        "season": "spring_2025",
        "branch": "marketing"
      },
      "marketing_activities": [
        {
          "workflows": [
            {
              "id": 9,
              "name": "Welcome Series",
              "is_active": true
            }
          ],
          "triggers": [
            {
              "id": 6,
              "name": "Thank You Purchase",
              "is_active": true
            }
          ],
          "broadcasts": [
            {
              "id": 1,
              "name": "Seasonal Sale Announcement",
              "is_active": true,
              "is_completed": false
            }
          ],
          "regulars": [
            {
              "id": 2,
              "name": "Weekly Newsletter",
              "is_active": true
            }
          ],
          "abns": [],
          "placements": []
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <id>22</id>
    <is_delete>false</is_delete>
    <createtime>2025-08-22T09:00:21.654Z</createtime>
    <updatetime>2025-08-22T11:08:10.216Z</updatetime>
    <name>Seasonal sale</name>
    <description>Campaign for promoting the spring collection</description>
    <state>active</state>
    <audience>
      <state>ready</state>
      <seg_id>4</seg_id>
      <list_id>2</list_id>
      <extension_conf>
        <workflows>
        </workflows>
        <triggers>
        </triggers>
        <broadcasts>
          <ids>
            <id>1</id>
          </ids>
        </broadcasts>
        <regulars>
        </regulars>
        <abns>
        </abns>
        <placements>
        </placements>
      </extension_conf>
      <lcg>
        <stratification_field_settings>
          <field>
            <field_name>_sex</field_name>
            <field_type>enum</field_type>
            <condition_type>list</condition_type>
            <conditions>
              <condition>
                <list>
                  <value>1</value>
                </list>
              </condition>
              <condition>
                <list>
                  <value>2</value>
                </list>
              </condition>
            </conditions>
          </field>
        </stratification_field_settings>
        <ignore_entities>
          <broadcasts>
            <ids>
              <id>1</id>
            </ids>
          </broadcasts>
        </ignore_entities>
        <size_formation_settings>
          <limit>133333</limit>
          <percent>16.00</percent>
        </size_formation_settings>
      </lcg>
    </audience>
    <active_period>
      <settings>
        <tz>UTC</tz>
        <end_at>2025-08-24T12:03:21.93Z</end_at>
      </settings>
      <history>
        <start>2025-08-22T09:00:47.786Z</start>
      </history>
    </active_period>
    <counters>
      <segment>631</segment>
      <test_group>530</test_group>
      <local_control_group>101</local_control_group>
      <global_control_group>0</global_control_group>
    </counters>
```

```
    <marketing_activities_counter>2</marketing_activities_counter>
    <groups>
      <group>
        <id>0</id>
        <name>main</name>
      </group>
    </groups>
    <attributes>
      <season>spring_2025</season>
      <branch>marketing</branch>
    </attributes>
  </data>
</xml>
```

Returned parameters

Top-level fields

Parameter	Type	Description
`error`	**int**	**Error code. 0 – successful execution**
`error_text`	**string**	**Error description**
`data`	**array**	**Array of campaign objects**

The data object

Contains the main campaign information.

Parameter	Type	Description
`id`	**int**	**Campaign ID**
`name`	**string**	**Campaign name**
`description`	**string**	**Campaign description**
`list_id`	**int**	**Profile database ID**
`seg_id`	**int**	**Segment ID**
`state`	**string**	**Current campaign status**
`createtime`	**string**	**Campaign creation date and time**
`updatetime`	**string**	**Last update date and time**
`is_delete`	**boolean**	**Deletion flag**
`tags`	**array**	**Campaign tags**
`groups`	**array**	**Access groups**
`marketing_activities_counter`	**int**	**Number of marketing activities (mailings, scenarios, placements)**
`attributes`	**object**	**Campaign attributes**
`counters`	**object**	**Profile counters for the audience, Test group, Local and Global control groups**

The counters object

Contains profile counts for different groups.

Parameter	Type	Description
`segment`	**int**	**Number of profiles in the segment**
`test_group`	**int**	**Number of profiles in the test group (TG)**
`local_control_group`	**int**	**Number of profiles in the Local control group (LCG)**
`global_control_group`	**int**	**Number of profiles in the Global control group (GCG) of the profile database**

The active_period object

Contains information about the campaign timeframe.

Parameter	Type	Description
`settings`	**object**	**Planned active period settings**
`history`	**object**	**Actual active period data**

settings

Parameter	Type	Description
`tz`	**string**	**Campaign time zone**
`start_at`	**string**	**Planned activation date**
`end_at`	**string**	**Planned completion date**

history

Parameter	Type	Description
`start`	**string**	**Actual activation date**

The audience object

Contains campaign audience settings and information.

Parameter	Type	Description
`state`	**string**	**Audience building status**
`seg_id`	**int**	**Campaign segment ID**
`list_id`	**int**	**Profile database ID**
`extension_conf`	**object**	**Audience expansion activities**
`lcg`	**object**	**Local control group settings**

extension_conf

Activities included in audience expansion.

Parameter	Type	Description
`workflows`	**object**	**Scenarios that expand the audience**
`triggers`	**object**	**Triggered mailings that expand the audience**
`broadcasts`	**object**	**Broadcast mailings that expand the audience**
`regulars`	**object**	**Regular mailings that expand the audience**
`abns`	**object**	**A/B tests that expand the audience**
`placements`	**object**	**Placements that expand the audience**

Nested extension_conf objects

Each object contains:

Parameter	Type	
`ids`	**array**	**Activity ID list**

lcg (Local control group)

Parameter	Type	
`stratification_field_settings`	**array**	**Stratification settings**
`ignore_entities`	**object**	**Activities that ignore LCG**
`size_formation_settings`	**object**	**LCG size settings**

stratification_field_settings

Parameter	Type	Description
`field_name`	**string**	**Field name for stratification**
`field_type`	**string**	**Field type**
`condition_type`	**string**	**Condition type for stratification**
`conditions`	**array**	**Stratification conditions**

ignore_entities

Activities that ignore LCG audience building.

Parameter	Type	Description
`workflows`	**object**	**Scenarios that ignore LCG**
`triggers`	**object**	**Triggers that ignore LCG**
`broadcasts`	**object**	**Broadcast mailings that ignore LCG**
`regulars`	**object**	**Regular mailings that ignore LCG**
`abns`	**object**	**A/B tests that ignore LCG**
`placements`	**object**	**Placements that ignore LCG**

size_formation_settings

LCG size settings.

Parameter	Type	Description
`limit`	**int**	**LCG size limit**
`percent`	**string**	**LCG share of the audience in percent**

The marketing_activities object

Contains the list of all campaign marketing activities.

Parameter	Type	De
`workflows`	**array**	**Scenarios**
`triggers`	**array**	**Triggers**
`broadcasts`	**array**	**Broadcasts**
`regulars`	**array**	**Regular mailings**
`abns`	**array**	**A/B tests**
`placements`	**array**	**Placements**

Common fields of marketing activities

Parameter	Type	Descri
`id`	**int**	**Activity ID**
`name`	**string**	**Marketing activity name**
`is_active`	**boolean**	**Activity status**
`is_completed`	**boolean**	**Completion status**

Get campaign list

Description

Retrieve the list of campaigns — combinations of mailings and scenarios into unified marketing strateg

> **CAUTION: Access rights**
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings — Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/list
```

Request parameters

Parameter	Type	Example	Required	Descriptio
from_id	**int**	**1**	**No**	**ID of the first campaign in the response default – 1**
limit	**int**	**2**	**No**	**Limit on the number of returned items default – 1000**
state	**string**	**"active"**	**No**	**Campaign status. One of: "cloning", "inactive", "waiting", "active", "**
ui_tags	**[]string**	**["tag1", "tag2"]**	**No**	**Tag list to filter campaigns default – []**
groups	**[]string**	**["group1", "group2"]**	**No**	**Group names to filter campaigns default – []**
groups_id	**[]int**	**[0, 1]**	**No**	**Group IDs to filter campaigns default – []**
attributes	**JSON object**	`"attributes": \{   "season": ["spring_2025", "summer_2025"] \}`	**No**	**Attribute values to filter campaigns**

Request example

JSON

```
{
  "limit": 1
}
```

XML

```
<xml>
  <limit>1</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 42,
      "is_delete": false,
      "createtime": "2025-08-22T07:15:29.711Z",
      "updatetime": "2025-08-22T09:13:24.264Z",
      "name": "Seasonal sale",
      "description": "",
      "tags": [],
      "state": "inactive",
      "audience": {
        "state": "ready",
        "seg_id": 124,
        "list_id": 49,
        "extension_conf": {
          "workflows": {"ids": []},
          "triggers": {"ids": []},
          "broadcasts": {"ids": []},
          "regulars": {"ids": [556]},
          "abns": {"ids": []},
          "placements": {"ids": []}
        },
        "lcg": {
          "ignore_entities": {
            "workflows": {"ids": []},
            "triggers": {"ids": []},
            "broadcasts": {"ids": []},
            "regulars": {"ids": []},
            "abns": {"ids": []},
            "placements": {"ids": []}
          },
          "size_formation_settings": {
            "limit": 0,
            "percent": "10.00"
          }
        }
      },
      "active_period": {
        "settings": {
          "tz": "Europe/Moscow",
          "end_at": "2025-08-23T10:36:25.099+03:00"
        },
        "history": {
          "start": "2025-08-22T12:13:18.813+03:00",
          "stop": "2025-08-22T12:13:24.264+03:00"
        }
      },
      "control_group_counter": 3002,
      "marketing_activities_counter": 1,
      "refresh_data": {
        "state": "completed",
        "progress": 100,
        "task_start_date": "2025-08-22T09:13:19.926Z",
        "task_end_date": "2025-08-22T09:13:20.048Z"
      },
      "group_id": 0,
      "groups_id": [0],
      "groups": [
        {
          "id": 0,
          "name": "main"
        }
      ],
      "attributes": {
        "season": "spring_2025",
        "branch": "marketing"
      }
    }
  ],
  "error": 0,
  "error_text": "",
  "total_count": 5
}
```


XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <id>42</id>
    <is_delete>false</is_delete>
    <createtime>2025-08-22T07:15:29.711Z</createtime>
    <updatetime>2025-08-22T09:13:24.264Z</updatetime>
    <name>Seasonal sale</name>
    <state>inactive</state>
    <audience>
      <state>ready</state>
      <seg_id>124</seg_id>
      <list_id>49</list_id>
      <extension_conf>
        <workflows><ids/></workflows>
        <triggers><ids/></triggers>
        <broadcasts><ids/></broadcasts>
        <regulars><ids><id>556</id></ids></regulars>
        <abns><ids/></abns>
        <placements><ids/></placements>
      </extension_conf>
      <lcg>
        <ignore_entities>
          <workflows><ids/></workflows>
          <triggers><ids/></triggers>
          <broadcasts><ids/></broadcasts>
          <regulars><ids/></regulars>
          <abns><ids/></abns>
          <placements><ids/></placements>
        </ignore_entities>
        <size_formation_settings>
          <limit>0</limit>
          <percent>10.00</percent>
        </size_formation_settings>
      </lcg>
    </audience>
    <active_period>
      <settings>
        <tz>Europe/Moscow</tz>
        <end_at>2025-08-23T10:36:25.099+03:00</end_at>
      </settings>
      <history>
        <start>2025-08-22T12:13:18.813+03:00</start>
        <stop>2025-08-22T12:13:24.264+03:00</stop>
      </history>
    </active_period>
    <control_group_counter>3002</control_group_counter>
    <marketing_activities_counter>1</marketing_activities_counter>
    <refresh_data>
      <state>completed</state>
      <progress>100</progress>
      <task_start_date>2025-08-22T09:13:19.926Z</task_start_date>
      <task_end_date>2025-08-22T09:13:20.048Z</task_end_date>
    </refresh_data>
    <group_id>0</group_id>
    <groups_id>0</groups_id>
    <groups>
      <group>
        <id>0</id>
        <name>main</name>
      </group>
    </groups>
    <attributes>
      <season>spring_2025</season>
      <branch>marketing</branch>
    </attributes>
  </data>
  <total_count>5</total_count>
</xml>
```


Returned parameters

Top-level fields

Parameter	Type	Description
`error`	**int**	**Error code. 0 – successful execution**
`error_text`	**string**	**Error description**
`total_count`	**int**	**Total number of campaigns**

Parameter	Type	Description
next_from_id	int	Pointer for retrieving the next batch of data
data	array	Array of campaign objects

The data object

Contains core campaign information.

Parameter	Type	Description
id	int	Campaign ID
name	string	Campaign name
description	string	Campaign description
list_id	int	Profile database ID
seg_id	int	Segment ID
state	string	Current campaign status
createtime	string	Campaign creation date and time
updatetime	string	Last update date and time
is_delete	boolean	Indicates whether the campaign is deleted
tags	array	Campaign tags
groups	array	User groups that have access to the campaign
group_id	int	Main campaign group ID
groups_id	array	Array of campaign group IDs
marketing_activities_counter	int	Number of marketing activities (mailings, scenarios, placements)
control_group_counter	int	Number of profiles in the control group
attributes	object	Campaign attributes
refresh_data	object	Information about the campaign data refresh process

The active_period object

Contains information about the campaign timeframe.

Parameter	Type	Description
settings	object	Planned active period settings
history	object	Actual active period data

settings — planned campaign timing

Parameter	Type	Description
tz	string	Campaign time zone
start_at	string	Planned campaign start date
end_at	string	Planned campaign end date

history — actual campaign data

Parameter	Type	Description
start	string	Actual campaign start date
stop	string	Actual campaign stop date

The audience object

Contains campaign audience settings and information.

Parameter	Type	Description
state	string	Audience building status
seg_id	int	Segment ID
list_id	int	Profile database ID
extension_conf	object	Audience expansion settings via marketing activities
lcg	object	Local control group settings

extension_conf

Activities that expand the campaign audience.

Parameter	Type	Description
workflows	object	Scenarios that expand the audience
triggers	object	Triggered mailings that expand the audience
broadcasts	object	Broadcast mailings that expand the audience
regulars	object	Regular mailings that expand the audience
abns	object	A/B tests that expand the audience
placements	object	Placements that expand the audience

Each of these objects contains an array of IDs of the corresponding activities.

lcg — Local control group settings

Parameter	Type	Descri
ignore_entities	object	Activities that ignore LCG
size_formation_settings	object	Local control group size settings

ignore_entities — activities excluded from LCG audience building

Parameter	Type	Description
`workflows`	**object**	**Scenarios that ignore LCG**
`triggers`	**object**	**Triggers that ignore LCG**
`broadcasts`	**object**	**Broadcast mailings that ignore LCG**
`regulars`	**object**	**Regular mailings that ignore LCG**
`abns`	**object**	**A/B tests that ignore LCG**
`placements`	**object**	**Placements that ignore LCG**

size_formation_settings — LCG size settings

Parameter	Type	Description
`limit`	**int**	**Maximum number of profiles in LCG**
`percent`	**string**	**LCG share of the total audience in percent**

The refresh_data object

Contains information about the campaign data refresh process.

Parameter	Type	Description
`state`	**string**	**Data refresh task status**
`progress`	**int**	**Task progress in percent**
`task_start_date`	**string**	**Data refresh task start date and time**
`task_end_date`	**string**	**Data refresh task end date and time**

The groups object

Contains information about access groups for the campaign.

Parameter	Type	Desc
`id`	**int**	**User group ID**
`name`	**string**	**User group name**

Activate campaign

Description

Request to activate a campaign.

> **CAUTION: Access rights**
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/activate
```

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Campaign ID**
activate_date	**string**	**"2025-03-15T10:00:00Z"**	**Yes**	**Activation date in RFC 3339 fo**
deactivate_date	**string**	**"2025-03-20T15:30:00Z"**	**No**	**Deactivation date in RFC 3339**
completed_date	**string**	**"2025-04-01T18:45:00Z"**	**No**	**Completion date in RFC 3339**

> **TIP**
>
> **If you send `start_date` without a date value, the campaign is activated immediately.**

Request example

JSON

```
{
  "id": 1,
  "start_date": "2025-04-01T18:45:00Z",
  "deactivate_date": "2025-05-01T18:45:00Z",
  "completed_date": "2025-05-01T18:45:00Z"
}
```

XML

```
<xml>
  <id>1</id>
  <start_date>2025-04-01T18:45:00Z</start_date>
  <deactivate_date>2025-05-01T18:45:00Z</deactivate_date>
  <completed_date>2025-05-01T18:45:00Z</completed_date>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error description**
data	**null**	**No data returned**

Complete campaign

Description

Complete a campaign immediately. Completion stops sending messages for running mailings and switc scenarios to the “inactive” status.

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/completed
```

Request parameters

Parameter	Type	Example	Required	
id	**int**	**18**	**Yes**	**Camp**

Request example

JSON

```
{
  "id": 18
}
```

XML

```
<xml>
  <id>18</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error description**
data	**null**	**No data returned**

> WARNING
>
> **Completing a campaign is irreversible. After completion, the campaign cannot be restarted; you can c**

Deactivate campaign

Description

Deactivate a campaign immediately. Deactivation stops sending messages from running mailings.

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/deactivate
```

Request parameters

Parameter	Type	Example	Required	
id	**int**	**18**	**Yes**	**Camp**

Request example

JSON

```
{
  "id": 18
}
```

XML

```
<xml>
  <id>18</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error description**
data	**null**	**No data returned**

> INFO
>
> **Deactivation temporarily pauses the campaign. All settings and data are preserved for potential react**

Get campaign status

Description

Retrieve the current status of a campaign.

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — View object**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/get_status
```

Request parameters

Parameter	Type	Example	Required	
id	**int**	**10**	**Yes**	**Camp**

Request example

JSON

```
{
  "id": 10
}
```

XML

```
<xml>
  <id>10</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "status": "active"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <status>active</status>
  </data>
</xml>
```

Returned parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error description**
data	**object**	**Object containing status information**

Status fields

Parameter	Type	Description
status	**string**	**Current campaign status**

Possible statuses

Status	Description
cloning	**Being cloned**
inactive	**Inactive**
waiting	**Awaiting activation**
active	**Active**
completed	**Completed**
deleted	**Deleted**

Activate mailing

Description

Activates a mailing.
Can be used to activate and launch a mailing instantly.
Also you can override mailing audience source for a different database or segment.
If `run_now` = `true`, the specified mailing launches instantly.
`override_source_type` and `override_source_id` are valid only if `run_now` = `true`.
`override_limit` parameter sets sending limitations for the current run.
If no override options are set - mailings will use their default audience sources and sending limitations

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/activate/`

Request parameters

Parameter	Type	Example	Required	Descriptio
id	**int**	**12**	**Yes**	**Mailing ID**
run_now	**bool**	**true**	**No**	**Launch mailing instantly**
override_source_type	**enum**	**“list” or “segment”**	**No**	**Override audience source**
override_source_id	**int**	**5**	**No**	**New audience source ID**
override_limit	**int**	**300**	**No**	**Override mailing limitations**
content	**JSON object**	`{"field":"value"}`	**No**	**Additional content for API content filter node and template apicor**
attach	**JSON array**	`[]`	**No**	**Attach file(s) to mailing messages**

Request example

JSON

```
{
  "id": 82,
  "override_source_id": 1,
  "override_source_type": "list",
  "run_now": true
}
```

XML

```
<xml>
    <id>82</id>
    <override_source_id>1</override_source_id>
    <override_source_type>list</override_source_type>
    <run_now>true</run_now>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**

Parameter	Type	
error_text	**string**	**Error text**

Deactivate mailing

Description

Deactivates a mailing.
Can also abort message sending, if a mailing is active:
`abort_if_running` **=** `true` **property immediately aborts all messages in sending queue. If the value is** `false` **though its already scheduled messages will be sent.**

Request URL

Method: POST
`https://example.com/api/v1.1/campaigns/deactivate`

Request parameters

Parameter	Type	Example	Required	
abort_if_running	**bool**	**true**	**No**	**Abort mailing sending By default – false**
id	**int**	**12**	**Yes**	**Mailing ID**

Request example

JSON

```
{
    "id": "15",
    "abort_if_running": "true"
}
```

XML

```
<xml>
    <id>15</id>
    <abort_if_running>true</abort_if_running>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get mailing list

Description

Retrieves mailing list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	`1`	**No**	**First mailing ir By default – 1**
limit	**int**	`2`	**No**	**Response mail By default – 10**
type	**string**	`"trigger"`	**No**	**Mailing type**
is_active	**bool**	`true`	**No**	**Mailing active**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`	**No**	**List of tags fo**
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access**
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access**
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`	**No**	**Attributes and**

Request example

JSON

```
{
  "limit": 2
}
```

XML

```
<xml>
  <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 5,
            "name": "New meteoric mailing",
            "type": "broadcast",
            "is_active": false,
            "subid": "",
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 7,
            "name": "Copy of New meteoric mailing",
            "type": "broadcast",
            "is_active": false,
            "subid": "",
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 8,
    "total_count": 134
}
```


XML


```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <id>5</id>
        <name>New meteoric mailing</name>
        <type>broadcast</type>
        <is_active>false</is_active>
        <subid></subid>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
    </data>
    <data>
        <id>7</id>
        <name>Copy of New meteoric mailing</name>
        <type>broadcast</type>
        <is_active>false</is_active>
        <subid></subid>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
    </data>
    <next_from_id>8</next_from_id>
    <total_count>134</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**

Parameter	Type	Description
id	**int**	**Mailing ID**
is_active	**bool**	**Mailing active flag**
name	**string**	**Mailing name**
type	**string**	**Mailing type**
total_count	**int**	**Total mailing count**
for_confirm	**bool**	**Messages with opt-in link flag**
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of access groups**
attributes	**object**	**Attributes assigned to the mailing and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Get mailing information

Description

Retrieves detailed mailing information.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/get`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes**

Request example

JSON

```
{
  "id": 2
}
```

XML

```
<xml>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 118,
        "name": "Daily weather notification",
        "description": "",
        "type": "regular",
        "update_time": "2024-05-20T09:17:53.362Z",
        "is_active": false,
        "is_completed": false,
        "data_id": 1,
        "data_type": "list",
        "content": [
            {
                "subject": "",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "message_id": 21
            }
        ],
        "status": "Inactive",
        "subid": "",
        "virtual_sender_id": 2,
        "providers": [
            "google",
            "mozilla",
            "safari",
            "safari-native",
            "firefox-firebase",
            "chrome-firebase",
            "opera-firebase",
            "yandex-firebase",
            "samsung-firebase",
            "android-firebase",
            "ios-firebase",
            "ios-apns",
            "yandex-Appmetrica-device-id-ios",
            "yandex-Appmetrica-device-id-android",
            "yandex-Appmetrica-ios-ifa",
            "yandex-Appmetrica-google-aid",
            "yandex-Appmetrica-android-token",
            "yandex-Appmetrica-ios-token",
            "android-huawei",
            "ios-huawei",
            "android-rustore"
        ],
        "push_sending_mode": "fanout",
        "resource_id": 26,
        "channel": "push",
        "for_confirm": false
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <id>118</id>
        <name>Daily weather notification</name>
        <description></description>
        <type>regular</type>
        <update_time>2024-05-20T09:17:53.362Z</update_time>
        <is_active>false</is_active>
        <is_completed>false</is_completed>
        <data_id>1</data_id>
        <data_type>list</data_type>
        <content>
            <subject></subject>
            <from_email></from_email>
            <from_name></from_name>
            <replyto_email></replyto_email>
            <replyto_name></replyto_name>
            <message_id>21</message_id>
        </content>
        <status>Inactive</status>
        <subid></subid>
        <virtual_sender_id>2</virtual_sender_id>
        <providers>google</providers>
        <providers>mozilla</providers>
        <providers>safari</providers>
        <providers>safari-native</providers>
        <providers>firefox-firebase</providers>
        <providers>chrome-firebase</providers>
        <providers>opera-firebase</providers>
        <providers>yandex-firebase</providers>
        <providers>samsung-firebase</providers>
        <providers>android-firebase</providers>
        <providers>ios-firebase</providers>
        <providers>ios-apns</providers>
        <providers>yandex-Appmetrica-device-id-ios</providers>
        <providers>yandex-Appmetrica-device-id-android</providers>
        <providers>yandex-Appmetrica-ios-ifa</providers>
        <providers>yandex-Appmetrica-google-aid</providers>
        <providers>yandex-Appmetrica-android-token</providers>
        <providers>yandex-Appmetrica-ios-token</providers>
        <providers>android-huawei</providers>
        <providers>ios-huawei</providers>
        <providers>android-rustore</providers>
        <push_sending_mode>fanout</push_sending_mode>
        <resource_id>26</resource_id>
        <channel>push</channel>
        <for_confirm>false</for_confirm>
    </data>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameters	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
content	**JSON array**	**Message templates list**
data_id	**int**	**Audience source ID**
data_type	**string**	**Audience source type**
id	**int**	**Mailing ID**
update_time	**string**	**Date the mailing was last edited**
updated_user	**string**	**Name of the user who edited the mailing**
is_active	**bool**	**Mailing active flag**
is_completed	**bool**	**Mailing finished flag**
name	**string**	**Mailing name**
resource_id	**int**	**Resource ID**
send_email_content	**bool**	**Email channel active**
send_push_content	**bool**	**Push channel active**
send_sms_content	**bool**	**SMS channel active**
push_sending_mode	**string**	**Push notification sending mode (push channels only)**
for_confirm	**bool**	**Messages with opt-in link flag**
status	**string**	**Mailing status**
subid	**string**	**Mailing SubID**

Parameters	Type	Description
type	**string**	**Mailing type**
virtual_sender_id	**int**	**Virtual sender ID**
descr	**string**	**Mailing description**
attributes	**object**	**Attributes assigned to the mailing and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Get mailing log

Description

Retrieves mailing logs.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/log_v2`

Request parameters

Parameter	Type	Example	Required
from_time	**string**	**“2018-08-10T00:00:00.000Z”**	**No**
id	**int**	**1**	**Yes**
limit	**int**	**2**	**No**

Request example

JSON

```
{
    "id": 1,
    "limit": 2
}
```

XML

```
<xml>
   <id>1</id>
   <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "level": "INFO",
            "message": "Mailing finished",
            "time": "2024-04-02T17:57:01.504Z",
            "state": "end",
            "value": ""
        },
        {
            "level": "INFO",
            "message": "Total time spent",
            "time": "2024-04-02T17:57:01.505Z",
            "state": "",
            "value": "138.351985ms"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 140
}
```

XML

```
<xml>
    <data>
        <level>"INFO"</level>
        <message>"Mailing finished"</message>
        <state>"end"</state>
        <time>"2024-05-28T16:40:59.207Z"</time>
        <value>""</value>
    </data>
    <data>
        <level>"INFO"</level>
        <message>"Total time spent"</message>
        <state>""</state>
        <time>"2024-05-28T16:40:59.208Z"</time>
        <value>"351.155547ms"</value>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>9</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
level	**string**	**Logging level**
message	**string**	**Log entry**
state	**string**	**Entry status**
time	**date**	**Entry registration date**
value	**string**	**Event associated value**

Logging levels:

- `DEBUG`
- `INFO`
- `WARNING`
- `ERROR`

Event entry status:

- `start` – **mailing start**
- `end` – **mailing finish**

Clone mailing

Description

Clones mailing content and settings to a newly created mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/clone`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Mailing ID**
name	**string**	**“Clone”**	**No**	**New mailing name By default – “Clone of ...”**

Request example

JSON

```
{
    "id": 1,
    "name": "Clone"
}
```

XML

```
<xml>
   <id>1</id>
   <name>Clone</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 87
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>88</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New mailing ID**

Delete mailing

Description

Deletes a mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/delete`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get mailing status

Description

Retrieves a mailing status — whether it is active or inactive.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/get_status`

Request parameters

Parameter	Type	Example	Required
id	**int**	**12**	**Yes**

Request example

JSON

```
{
    "id": 82
}
```

XML

```
<xml>
    <id>82</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "is_active": false,
        "is_completed": false,
        "status": "Inactive"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <is_active>false</is_active>
        <is_completed>false</is_completed>
        <status>"Inactive"</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descr
error	**int**	**Error code**
error_text	**string**	**Error text**
is_active	**bool**	**Mailing active flag**
is_completed	**bool**	**Mailing finished flag**
status	**string**	**Current mailing status**

Get broadcasts list

Description

Retrieves broadcast mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**2**	**No**	**First in the list** **By default – 1**
limit	**int**	**10**	**No**	**Response broa** **By default – 10**
type	**string**	**“broadcast”**	**Yes**	**Mailing type**
is_active	**bool**	**true**	**No**	**Retrieve active**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`	**No**	**List of tags for** **By default –** `[]`
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access** **By default –** `[]`
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of groups** **By default –** `[]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`	**No**	**Attributes and**

Request example

JSON

```
{
    "limit": 2
}
```

XML

```
<xml>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 3,
            "name": "New mailing 1",
            "type": "broadcast",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 5,
            "name": "New mailing 2",
            "type": "broadcast",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 17
}
```


XML


```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>3</id>
        <is_active>false</is_active>
        <name>New mailing 1</name>
        <type>broadcast</type>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>5</id>
        <is_active>false</is_active>
        <name>New mailing 2</name>
        <type>broadcast</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>17</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**
name	**string**	**Mailing name**
type	**string**	**Mailing type**
is_active	**bool**	**Mailing active flag**
for_confirm	**bool**	**Messages with opt-in link flag**
total_count	**int**	**Total mailings count**

Parameter	Type	Description
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of groups**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Get broadcast information

Description

Retrieves detailed information about a broadcast mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/get`

Request parameters

Parameter	Type	Example	Required
id	**int**	**11768**	**Yes**

Request example

JSON

```
{
  "id": "87"
}
```

XML

```
<xml>
  <id>87</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "name": "sms ",
        "is_active": false,
        "data_id": 13,
        "data_type": "segment",
        "content": [
            {
                "subject": "",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "message_id": 27
            }
        ],
        "send_channel": "sms",
        "channel": "sms",
        "push_sending_mode": "fanout",
        "resource_id": 36,
        "subid": "",
        "limit": 0,
        "aglimit_count": 0,
        "aglimit_percent": 0,
        "start_schedule": {
            "start_mins": [
                38
            ],
            "start_hours": [
                16
            ],
            "start_mdays": [],
            "start_wdays": [],
            "start_months": [],
            "start_years": [],
            "tz": "",
            "profile": false,
            "losers": false
        },
        "for_confirm": false,
        "policy_id": 0,
        "description": "",
        "virtual_sender_id": 0,
        "is_random_tracking_prefix": false,
        "custom_tracking_domains": [],
        "watcher_list_id": 0,
        "watcher_limit": 0,
        "blind_carbon_copy": [],
        "unconditional_sending": false,
        "skip_unique_checking": false,
        "sub_cats_filter": null,
        "sub_fields_filter": null,
        "id": 179,
        "update_time": "2024-05-28T13:40:57.3Z"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <aglimit_count>0</aglimit_count>
        <aglimit_percent>0</aglimit_percent>
        <channel>"sms"</channel>
        <content>
            <from_email>""</from_email>
            <from_name>""</from_name>
            <message_id>27</message_id>
            <replyto_email>""</replyto_email>
            <replyto_name>""</replyto_name>
            <subject>""</subject>
        </content>
        <data_id>13</data_id>
        <data_type>"segment"</data_type>
        <description>""</description>
        <for_confirm>false</for_confirm>
        <id>179</id>
        <is_active>false</is_active>
        <is_random_tracking_prefix>false</is_random_tracking_prefix>
        <limit>0</limit>
        <name>"sms "</name>
        <policy_id>0</policy_id>
        <push_sending_mode>"fanout"</push_sending_mode>
        <resource_id>36</resource_id>
        <send_channel>"sms"</send_channel>
        <skip_unique_checking>false</skip_unique_checking>
        <start_schedule>
            <losers>false</losers>
            <profile>false</profile>
            <start_hours>16</start_hours>
            <start_mins>38</start_mins>
            <tz>""</tz>
        </start_schedule>
        <subid>""</subid>
        <unconditional_sending>false</unconditional_sending>
        <update_time>"2024-05-28T13:40:57.3Z"</update_time>
        <virtual_sender_id>0</virtual_sender_id>
        <watcher_limit>0</watcher_limit>
        <watcher_list_id>0</watcher_list_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**
name	**string**	**Mailing name**
is_active	**bool**	**Mailing active flag**
resource_id	**int**	**Resource ID**
data_id	**int**	**Audience source ID**
data_type	**string**	**Audience source type**
content	**JSON array**	**Message template object**
send_channel	**string**	**Sending channel**
channel	**string**	**Channel**
for_confirm	**bool**	**Messages with opt-in link flag**
subid	**string**	**Mailing SubID**
start_schedule	**JSON object**	**Start schedule object**
limit	**int**	**Mailing cycle limitation**
aglimit_count	**int**	**Increase limit each cycle - number**
agllimit_percent	**float**	**Increase limit each cycle - percent**
policy_id	**int**	**Sending policy ID**
push_sending_mode	**string**	**Push notification sending mode (push channels only)**
descr	**string**	**Mailing description**
attributes	**object**	**Assigned attributes and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**
Smart settings		
virtual_sender_id	**int**	**Virtual sender ID** **(Smart accounts)**

Parameter	Type	Description
is_random_tracking_prefix	**bool**	**Random tracking domain prefix (Smart accounts)**
custom_tracking_domains	**JSON array**	**Tracking domains list (Smart accounts)**
blind_carbon_copy	**JSON array**	**Hidden copy email addresses (Smart accounts)**
unconditional_sending	**bool**	**Send messages regardless of subscription status (Smart accounts)**
watcher_list_id	**int**	**Mailing watchers database ID (Smart accounts)**
watcher_limit	**int**	**Watchers limit (Smart accounts)**

Create broadcast mailing

Description

Creates a new broadcast mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/add`

Request parameters

Parameter	Type	Example	Required	Description
name	**string**	**"Broadcast mailing name"**	**Yes**	**Mailing name**
is_active	**bool**	**true**	**No**	**Activate mailing on creation**
data_type	**string**	**"list"**	**Yes**	**Audience source type**
data_id	**int**	**12**	**Yes**	**Audience source ID**
resource_id	**int**	**24**	**No**	**Resource ID**
content	**JSON array**		**Yes**	**Message template object**
send_channel	**string**	**"email" "sms" "push"**	**Yes**	**Sending channel**
content_type	**string**	**"email"**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_email_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_push_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_sms_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
subid	**string**	**"Adv_1"**	**No**	**Mailing SubID**
start_schedule	**JSON object**		**No**	**Start schedule object**
limit	**int**	**0**	**No**	**Mailing cycle limitation**
aglimit_count	**int**	**0**	**No**	**Increase limit each cycle — number**
agllimit_percent	**float**	**99.0**	**No**	**Increase limit each cycle — percent**
policy_id	**int**	**1**	**No**	**Sending policy ID**
push_sending_mode	**string**	**"fanout" (default) "last"**	**No**	**Push notification sending mode (push channels only)**
attributes	**object**	**"season":"summer_2024"**	**No**	**Assigned attributes and their values**
Smart settings				
virtual_sender_id	**int**	**12**	**No**	**Virtual sender ID (Smart accounts)**
is_random_tracking_prefix	**bool**	**false**	**No**	**Random tracking domain prefix (Smart accounts)**
custom_tracking_domains	**JSON array**	**["trk.example.com", "trk2.example.com"]**	**No**	**Tracking domain list (Smart accounts)**
blind_carbon_copy	**JSON array**	**["john@example.com", "george@example.com"]**	**No**	**Hidden copy email addresses (Smart accounts)**
unconditional_sending	**bool**	**true**	**No**	**Send messages regardless of subscription status (Smart accounts)**
watched_list_id	**int**	**12**	**No**	**Mailing watchers database ID (Smart accounts)**
watcher_limit	**int**	**100**	**No**	**Watchers limit (Smart accounts)**

Request example

JSON

```
{
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "example@example.com",
            "from_name": "Victor",
            "message_id": 15,
            "replyto_email": "example@example.com",
            "replyto_name": "Victoria",
            "subject": "Hello"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "is_active": true,
    "limit": 1,
    "name": "New broadcast",
    "resource_id": 21,
    "send_channel": "email",
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name></from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name></replyto_name>
        <subject>Hello!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Broadcast</name>
    <resource_id>21</resource_id>
    <send_channel>email</send_channel>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 90
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>90</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New mailing ID**

Update broadcast mailing

Description

Updates mailing details, changes an existing mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/update`

Request parameters

Parameter	Type	Example	Required	
id	int	1	Yes	Mailing ID
name	string	"Broadcast mailing name"	Yes	Mailing name
is_active	bool	true	No	Activate or deactiva
data_type	string	"list"	Yes	Audience source typ
data_id	int	12	Yes	Audience source ID
resource_id	int	24	No	Resource ID
content	JSON array		Yes	Message template o
subid	string	"Adv_1"	No	Mailing SubID
start_schedule	JSON object		No	Start schedule objec
limit	int	0	No	Mailing cycle limitat
aglimit_count	int	0	No	Increase limit each
agllimit_percent	float	99.0	No	Increase limit each
policy_id	int	1	No	Sending policy ID
push_sending_mode	string	"fanout" (default) "last"	No	Push notification se
attributes	object	"season":"summer_2024"	No	Assigned attributes
Smart settings				
virtual_sender_id	int	12	No	Virtual sender ID (Smart accounts)
is_random_tracking_prefix	bool	false	No	Random tracking do (Smart accounts)
custom_tracking_domains	JSON array	["trk.example.com", "trk2.example.com"]	No	Tracking domains lis (Smart accounts)
blind_carbon_copy	JSON array	["john@example.com", "george@example.com"]	No	Hidden copy email a (Smart accounts)
unconditional_sending	bool	true	No	Send messages rega (Smart accounts)
watched_list_id	int	12	No	Mailing watchers da (Smart accounts)
watcher_limit	int	100	No	Watchers limit (Smart accounts)

INFO

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, c send_sms_content, send_push_content, the request will not return an error, but it will not update the s

Request example

JSON

```
{
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "example@example.com",
            "from_name": "Alexandra",
            "message_id": 14,
            "replyto_email": "example@example.com",
            "replyto_name": "Victoria",
            "subject": "Hi!"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "id": 90,
    "is_active": true,
    "limit": 10,
    "name": "Broadcast",
    "resource_id": 20,
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>broadcast1@altcraft.com</from_email>
        <from_name>Margorita</from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Victoria</replyto_name>
        <subject>Good morning!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <id>89</id>
    <is_active>true</is_active>
    <limit>10</limit>
    <name>Broadcast</name>
    <resource_id>20</resource_id>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 90
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>90</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Mailing ID**

Launch a broadcast mailing

Description

Launches a broadcast mailing instantly. A mailing must be active.
`override_source_type` **and** `override_source_id` **parameters override mailing audience for the run.**
`override_limit` **parameter sets sending limitations for the current run.**
If no override options are set — mailings will use their default audience sources and sending limitation

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/start`

Request parameters

Parameter	Type	Example	Required	Descripti
id	**int**	**12**	**Yes**	**Mailing ID**
override_source_type	**enum**	**“list” или “segment”**	**No**	**Override audience source**
override_source_id	**int**	**5**	**No**	**New audience source ID**
override_limit	**int**	**300**	**No**	**Override mailing limitations**
content	**JSON object**	`{"field":"value"}`	**No**	**Additional content for API content filter node and template apic**
attach	**JSON array**	**[]**	**No**	**Attach file(s) to mailing messages**

Request example

JSON

```
{
    "id": 90,
    "override_source_id": 1,
    "override_source_type": "list"
}
```

XML

```
<xml>
  <id>90</id>
  <override_source_id>1</override_source_id>
  <override_source_type>list</override_source_type>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get regular mailings list

Description

Retrieves regular mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/list/`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**11768**	**No**	**The first maili By default –1**
limit	**int**	**10**	**No**	**Mailings list li By default – 10**
is_active	**bool**	**true**	**No**	**Retrieve active**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`	**No**	**List of tags fo By default –** `[]`
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access By default –** `[]`
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of group I By default –** `[]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`	**No**	**Attributes and**

Request example

JSON

```
{
    "limit": 2
}
```

XML

```
<xml>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 15,
            "name": "Regular mailing 1",
            "type": "regular",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [
                "B2C"
            ],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ]
        },
        {
            "id": 78,
            "name": "Regular mailing 2",
            "type": "regular",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [
                "B2C"
            ],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3
}
```


XML


```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>15</id>
        <is_active>true</is_active>
        <name>Regular mailing 1</name>
        <type>regular</type>
        <ui_tags>B2C</ui_tags>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>78</id>
        <is_active>false</is_active>
        <name>Regular mailing 2</name>
        <type>regular</type>
        <ui_tags>B2C</ui_tags>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**

Parameter	Type	Description
name	**string**	**Mailing name**
type	**string**	**Mailing type**
is_active	**bool**	**Mailing active flag**
for_confirm	**bool**	**Messages with opt-in link flag**
total_count	**int**	**Total mailing count**
ui_tags	**[]String**	**Tag list**
groups	**[]String**	**Group list**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Get regular mailing information

Description

Retrieves a regular mailing detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/get`

Request parameters

Parameter	Type	Example	Required
id	**int**	**11768**	**Yes**

Request example

JSON

```
{
    "id": 15
}
```

XML

```
<xml>
    <id>15</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "name": "Regular mailing name",
        "is_active": true,
        "data_id": 2,
        "data_type": "segment",
        "content": [
            {
                "subject": "Hi!",
                "from_email": "broadcast@altcraft.com",
                "from_name": "",
                "replyto_email": "broadcast@altcraft.com",
                "replyto_name": "",
                "message_id": 15
            }
        ],
        "send_channel": "email",
        "channel": "email",
        "resource_id": 20,
        "subid": "",
        "limit": 10,
        "aglimit_count": 0,
        "aglimit_percent": 0,
        "start_schedule": {
            "start_mins": [
                10
            ],
            "start_hours": [
                12
            ],
            "start_mdays": [],
            "start_wdays": [],
            "start_months": [],
            "start_years": [],
            "tz": "",
            "profile": false,
            "losers": false
        },
        "for_confirm": false,
        "policy_id": 0,
        "description": "",
        "virtual_sender_id": 1,
        "is_random_tracking_prefix": false,
        "custom_tracking_domains": [
            "trk.example.com",
            "trk.example.org"
        ],
        "watcher_list_id": 0,
        "watcher_limit": 0,
        "blind_carbon_copy": [],
        "unconditional_sending": false,
        "skip_unique_checking": false,
        "sub_cats_filter": [],
        "sub_fields_filter": null,
        "id": 15,
        "update_time": "2024-04-03T15:35:54.024Z"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <aglimit_count>0</aglimit_count>
        <aglimit_percent>0</aglimit_percent>
        <channel>"email"</channel>
        <content>
            <from_email>"broadcast@altcraft.com"</from_email>
            <from_name>""</from_name>
            <message_id>15</message_id>
            <replyto_email>"broadcast@altcraft.com"</replyto_email>
            <replyto_name>""</replyto_name>
            <subject>"Hi!"</subject>
        </content>
        <custom_tracking_domains>"trk.example.com"</custom_tracking_domains>
        <custom_tracking_domains>"trk.example.org"</custom_tracking_domains>
        <data_id>2</data_id>
        <data_type>"segment"</data_type>
        <description>""</description>
        <for_confirm>false</for_confirm>
        <id>15</id>
        <is_active>false</is_active>
        <is_random_tracking_prefix>false</is_random_tracking_prefix>
        <limit>10</limit>
        <name>"Regular mailing name"</name>
        <policy_id>0</policy_id>
        <push_sending_mode>"fanout"</push_sending_mode>
        <resource_id>20</resource_id>
        <send_channel>"email"</send_channel>
        <skip_unique_checking>false</skip_unique_checking>
        <start_schedule>
            <losers>false</losers>
            <profile>false</profile>
            <start_hours>12</start_hours>
            <start_mins>10</start_mins>
            <tz>""</tz>
        </start_schedule>
        <subid>""</subid>
        <unconditional_sending>false</unconditional_sending>
        <update_time>"2024-05-20T09:17:53.326Z"</update_time>
        <virtual_sender_id>1</virtual_sender_id>
        <watcher_limit>0</watcher_limit>
        <watcher_list_id>0</watcher_list_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**
name	**string**	**Mailing name**
is_active	**bool**	**Mailing active flag**
resource_id	**int**	**Resource ID**
data_id	**int**	**Audience source ID**
data_type	**string**	**Audience source type**
content	**JSON array**	**Message template object**
send_channel	**string**	**Sending channel**
channel	**string**	**Channel**
for_confirm	**bool**	**Messages with opt-in link flag**
subid	**string**	**Mailing SubID**
start_schedule	**JSON object**	**Start schedule object**
limit	**int**	**Mailing cycle limitation**
aglimit_count	**int**	**Increase limit each cycle - number**
agllimit_percent	**float**	**Increase limit each cycle - percent**
policy_id	**int**	**Sending policy ID**
push_sending_mode	**string**	**Push notification sending mode (push channels only)**
descr	**string**	**Mailing description**
Smart settings		
virtual_sender_id	**int**	**Virtual sender ID (Smart accounts)**
is_random_tracking_prefix	**bool**	**Random tracking domain prefix (Smart accounts)**

Parameter	Type	Description
custom_tracking_domains	**JSON array**	**Tracking domains list** **(Smart accounts)**
blind_carbon_copy	**JSON array**	**Hidden copy email addresses** **(Smart accounts)**
unconditional_sending	**bool**	**Send messages regardless of subscription status** **(Smart accounts)**
watcher_list_id	**int**	**Mailing watchers database ID** **(Smart accounts)**
watcher_limit	**int**	**Watchers limit** **(Smart accounts)**
attributes	**object**	**Assigned attributes and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Create regular mailing

Description

Creates a new regular mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/add/`

Request parameters

Parameter	Type	Example	Required	Description
name	**string**	**"Broadcast mailing name"**	**Yes**	**Mailing name**
is_active	**bool**	**true**	**No**	**Activate mailing on creation**
data_type	**string**	**"list"**	**Yes**	**Audience source type**
data_id	**int**	**12**	**Yes**	**Audience source ID**
resource_id	**int**	**24**	**No**	**Resource ID**
content	**JSON array**		**Yes**	**Message template object**
send_channel	**string**	**"email" "sms" "push"**	**Yes**	**Sending channel**
content_type	**string**	**"email"**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_email_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_push_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
send_sms_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email chan recommended to use `send_channel`.**
subid	**string**	**"Adv_1"**	**No**	**Mailing SubID**
start_schedule	**JSON object**		**No**	**Start schedule object**
limit	**int**	**0**	**No**	**Mailing cycle limitation**
aglimit_count	**int**	**0**	**No**	**Increase limit each cycle — number**
agllimit_percent	**float**	**99.0**	**No**	**Increase limit each cycle — percent**
policy_id	**int**	**1**	**No**	**Sending policy ID**
push_sending_mode	**string**	**"fanout" (default) "last"**	**No**	**Push notification sending mode (push channels only)**
attributes	**object**	**"season":"summer_2024"**	**No**	**Assigned attributes and their values**
Smart settings				
virtual_sender_id	**int**	**12**	**No**	**Virtual sender ID (Smart accounts)**
is_random_tracking_prefix	**bool**	**false**	**No**	**Random tracking domain prefix (Smart accounts)**
custom_tracking_domains	**JSON array**	**["trk.example.com", "trk2.example.com"]**	**No**	**Tracking domain list (Smart accounts)**
blind_carbon_copy	**JSON array**	**["john@example.com", "george@example.com"]**	**No**	**Hidden copy email addresses (Smart accounts)**
unconditional_sending	**bool**	**true**	**No**	**Send messages regardless of subscription status (Smart accounts)**
watched_list_id	**int**	**12**	**No**	**Mailing watchers database ID (Smart accounts)**
watcher_limit	**int**	**100**	**No**	**Watchers limit (Smart accounts)**

Request example

JSON

```
{
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "regular@altcraft.com",
            "from_name": "Paul",
            "message_id": 15,
            "replyto_email": "regular@altcraft.com",
            "replyto_name": "Roman",
            "subject": "Hello"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "is_active": true,
    "limit": 1,
    "name": "Regular mailing #1",
    "resource_id": 20,
    "send_channel": "email",
    "start_schedule": {
         "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name>Paul</from_name>
        <message_id>15</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Roman</replyto_name>
        <subject>Hello</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Regular mailing #2</name>
    <resource_id>20</resource_id>
    <send_channel>email</send_channel>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 92
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>93</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New mailing ID**

Update regular mailing

Description

Updates a regular mailing. Changes mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/update`

Request parameters

Parameter	Type	Example	Required	
id	int	1	Yes	Mailing ID
name	string	"Broadcast mailing name"	Yes	Mailing name
is_active	bool	true	No	Activate or deactiva
data_type	string	"list"	Yes	Audience source typ
data_id	int	12	Yes	Audience source ID
resource_id	int	24	No	Resource ID
content	JSON array		Yes	Message template o
subid	string	"Adv_1"	No	Mailing SubID
start_schedule	JSON object		No	Start schedule objec
limit	int	0	No	Mailing cycle limitat
aglimit_count	int	0	No	Increase limit each
agllimit_percent	float	99.0	No	Increase limit each
policy_id	int	1	No	Sending policy ID
push_sending_mode	string	"fanout" (default) "last"	No	Push notification se
attributes	object	"season":"summer_2024"	No	Assigned attributes
Smart settings				
virtual_sender_id	int	12	No	Virtual sender ID (Smart accounts)
is_random_tracking_prefix	bool	false	No	Random tracking do (Smart accounts)
custom_tracking_domains	JSON array	["trk.example.com", "trk2.example.com"]	No	Tracking domains lis (Smart accounts)
blind_carbon_copy	JSON array	["john@example.com", "george@example.com"]	No	Hidden copy email a (Smart accounts)
unconditional_sending	bool	true	No	Send messages rega (Smart accounts)
watched_list_id	int	12	No	Mailing watchers da (Smart accounts)
watcher_limit	int	100	No	Watchers limit (Smart accounts)

INFO

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, c send_sms_content, send_push_content, the request will not return an error, but it will not update the s

Request example

JSON

```
{
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "regular@altcraft.com",
            "from_name": "Roman",
            "message_id": 14,
            "replyto_email": "regular@altcraft.com",
            "replyto_name": "Romana",
            "subject": "Hi!"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": {{db_id}},
    "data_type": "list",
    "id": 93,
    "is_active": true,
    "limit": 1,
    "name": "Regular mailing 1",
    "resource_id": 20,
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name>Roman</from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Romana</replyto_name>
        <subject>Hi!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <id>93</id>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Regular mailing 1</name>
    <resource_id>20</resource_id>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 93
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>93</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Mailing ID**

Launch a regular mailing

Description

Instantly starts a regular mailing if it is active. There is a one-time send.
`override_source_type` **and** `override_source_id` **parameters override mailing audience for the run.**
`override_limit` **parameter sets sending limitations for the current run.**
If no override options are set, mailings will use their default audience sources and sending limitations.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/start`

Request parameters

Parameter	Type	Example	Required	Descripti
id	**int**	**12**	**Yes**	**Mailing ID**
override_source_type	**enum**	**“list” или “segment”**	**No**	**Override audience source**
override_source_id	**int**	**5**	**No**	**New audience source ID**
override_limit	**int**	**300**	**No**	**Override mailing limitations**
content	**JSON object**	`{"field":"value"}`	**No**	**Additional content for API content filter node and template apic**
attach	**JSON array**	**[]**	**No**	**Attach file(s) to mailing messages**

Request example

JSON

```
{
    "id": 94,
    "override_source_id": 1,
    "override_source_type": "list"
}
```

XML

```
<xml>
    <id>93</id>
    <override_source_id>1</override_source_id>
    <override_source_type>list</override_source_type>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get trigger mailings list

Description

Retrieves trigger mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/list/`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**11768**	**No**	**The first maili By default – 1**
limit	**int**	**10**	**No**	**Trigger mailing By default – 10**
is_active	**bool**	**true**	**No**	**Retrieve only a**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`	**No**	**List of tags for**
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access**
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of group I**
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`	**No**	**Attributes and**

Request example

JSON

```
{
    "limit": 2
}
```

XML

```
<xml>
   <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 12,
            "name": "New trigger mailing 3",
            "type": "api_call",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 14,
            "name": "New planetary mailing",
            "type": "api_call",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ],
            "providers": [
                "google",
                "mozilla",
                "safari",
                "safari-native",
                "firefox-firebase",
                "chrome-firebase",
                "opera-firebase",
                "yandex-firebase",
                "samsung-firebase",
                "android-firebase",
                "ios-firebase",
                "ios-apns",
                "yandex-Appmetrica-device-id-ios",
                "yandex-Appmetrica-device-id-android",
                "yandex-Appmetrica-ios-ifa",
                "yandex-Appmetrica-google-aid",
                "yandex-Appmetrica-android-token",
                "yandex-Appmetrica-ios-token",
                "mr",
                "android-huawei",
                "ios-huawei",
                "android-rustore"
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 41
}
```


XML

```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>12</id>
        <is_active>true</is_active>
        <name>New trigger mailing 3</name>
        <type>api_call</type>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>14</id>
        <is_active>true</is_active>
        <name>New planetary mailing</name>
        <providers>google</providers>
        <providers>mozilla</providers>
        <providers>safari</providers>
        <providers>safari-native</providers>
        <providers>firefox-firebase</providers>
        <providers>chrome-firebase</providers>
        <providers>opera-firebase</providers>
        <providers>yandex-firebase</providers>
        <providers>samsung-firebase</providers>
        <providers>android-firebase</providers>
        <providers>ios-firebase</providers>
        <providers>ios-apns</providers>
        <providers>yandex-Appmetrica-device-id-ios</providers>
        <providers>yandex-Appmetrica-device-id-android</providers>
        <providers>yandex-Appmetrica-ios-ifa</providers>
        <providers>yandex-Appmetrica-google-aid</providers>
        <providers>yandex-Appmetrica-android-token</providers>
        <providers>yandex-Appmetrica-ios-token</providers>
        <providers>mr</providers>
        <providers>android-huawei</providers>
        <providers>ios-huawei</providers>
        <providers>android-rustore</providers>
        <type>api_call</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>41</total_count>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Trigger mailing ID**
name	**string**	**Mailing name**
type	**string**	**Trigger type**
for_confirm	**bool**	**Messages with opt-in links flag**
is_active	**bool**	**Mailing active flag**
ui_tags	**[]String**	**Tag list**
groups	**[]String**	**Group list**
workflow_id	**int**	**Scenario identifier in which the trigger mailing is used (The parameter is returned if the mailing is used in a scenario)**
node_id	**int**	**Scenario node identifier in which the trigger mailing is used (The parameter is returned if the mailing is used in a scenario)**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged (The parameter is returned if the mailing is engaged in a campaign)**

Get trigger mailing information

Description

Retrieves detailed information about a trigger mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Trigger mailing ID**

Request example

JSON

```
{
  "id": 12
}
```

XML

```
<xml>
    <id>12</id>
</xml>
```

Response example

JSON


```
{
  "data": {
    "call_url": "trk.test.test.altcraft.com/trg/465ea6fedf0a75fafe1a413943aae3696d95939038311994/<email|md5|xxh>",
    "content": [
      {
        "from_email": "",
        "from_name": "",
        "message_id": 41,
        "replyto_email": "",
        "replyto_name": "",
        "subject": ""
      }
    ],
    "send_channel": "email",
    "custom_tracking_domains": [],
    "id": 7,
    "is_active": true,
    "is_random_tracking_prefix": false,
    "list_id": 20,
    "name": "API call trigger",
    "resource_id": 44,
    "skip_minutes": 0,
    "subid": "API",
    "trigger_type": "api_call",
    "virtual_sender_id": 98
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
    <call_url>http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd63e50585b4bfd377365512f8ec196edd62f3de6b87e2969e3/
      <email|md5|xxh>
        </call_url>
          <channel>0</channel>
          <content>
              <message_id>8</message_id>
          </content>
          <for_confirm>false</for_confirm>
          <id>12</id>
          <is_active>true</is_active>
          <is_new_ch>false</is_new_ch>
          <is_new_contact>false</is_new_contact>
          <is_new_sub>false</is_new_sub>
          <is_random_tracking_prefix>false</is_random_tracking_prefix>
          <is_status_changed>false</is_status_changed>
          <list_id>1</list_id>
          <name>New trigger mailing 3</name>
          <policy_id>0</policy_id>
          <resource_id>4</resource_id>
          <send_channel>email</send_channel>
          <skip_minutes>0</skip_minutes>
          <skip_unique_checking>false</skip_unique_checking>
          <start_schedule>
              <losers>false</losers>
              <profile>false</profile>
          </start_schedule>
          <subscribe_resource_id>0</subscribe_resource_id>
          <trigger_type>api_call</trigger_type>
          <unconditional_sending>false</unconditional_sending>
          <update_time>2024-02-15T20:45:00.705Z</update_time>
          <virtual_sender_id>0</virtual_sender_id>
  </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Trigger mailing ID**
name	**string**	**Mailing name**
is_active	**bool**	**Mailing active flag**
content	**JSON array**	**Message template object**
trigger_type	**string**	**Trigger type. See below**
skip_minutes	**int**	**Waits for X minutes before mailing launch**
resource_id	**int**	**Resource ID**
send_channel	**string**	**Sending channel**
for_confirm	**bool**	**Messages with opt-in link flag**
policy_id	**int**	**Sending policy ID**
descr	**string**	**Mailing description**
push_sending_mode	**string**	**Push notification sending mode (push channels only)**
attributes	**object**	**Attributes assigned to mailing**
workflow_id	**int**	**Scenario identifier in which the trigger mailing is used (The parameter is returned if the mailing is used in a scenario)**
node_id	**int**	**Scenario node identifier in which the trigger mailing is used (The parameter is returned if the mailing is used in a scenario)**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged (The parameter is returned if the mailing is engaged in a campaign)**
Smart settings		
virtual_sender_id	**string**	**Virtual sender ID (Smart accounts)**
is_random_tracking_prefix	**bool**	**Random tracking domain prefix (Smart accounts)**
custom_tracking_domains	**JSON array**	**Tracking domains list (Smart accounts)**
blind_carbon_copy	**JSON array**	**Hidden copy email addresses (Smart accounts)**
unconditional_sending	**bool**	**Send messages regardless of subscription status (Smart accounts)**

Trigger type parameters

These parameters depend on trigger type specified for a mailing (`trigger_type`)

Parameter	Type	Description	api_call	api_import
action	string	Action based trigger type	No	No
update_query	JSON array		No	No
link_name	string	Message link name	No	No
message_id	int	Template ID	No	No
list_id	int	Database ID	Yes	Yes
subscribe_resource_id	int	Resource ID	No	No
channel	string	Trigger channel	No	No
is_new_sub	bool	New resource trigger	No	No
is_new_ch	bool	New channel trigger	No	No
is_new_contact	bool	New contact trigger	No	No
is_status_changed	bool	Subscription status change trigger	No	No
trigger_statuses	array(int)	Subscription status change options	No	No
pixel_id	int	Pixel ID	No	No
pixel_target	string	Pixel goal	No	No

trigger_type: action

Represents message action based trigger.

It has 3 additional parameters.

Two of them are: `"action": "<action name>"` and `"message_id": 123`. They specify associated message and a `"link_name": "String"` property cpecifies link name when `"action": "click_sms"` or `"action": "clicklink_sms"`.

- `"action": "open"` — **email opened.**
- `"action": "read"` — **email read**
- `"action": "click"` — **any link clicked in email message**
- `"action": "confirm"` — **opt-in link clicked**
- `"action": "click_sms"` — **any click in SMS message**
- `"action": "clicklink"` — **a specific email link clicked,** `"link_name": "String"`
- `"action": "clicklink_sms"` — **a specific SMS link clicked,** `"link_name": "String"`

trigger_type: api_import

Represents API profile import trigger.

Retrieves import database ID `"list_id": 32`.

trigger_type: pixel

Represents pixel activation trigger type.

Has two additional parameters:

- `"pixel_id": 64` — **Altcraft pixel unique ID.**
- `"pixel_target": "Target-name"` — **Pixel goal name.**

trigger_type: update

Represents profile fields update trigger.

Has two additional parameters:

- `"list_id": 32` — **profile database ID.**
- `"update_query": {"fieldname": "_postal_code", "fieldtype": "unicode", "operator": "equal", "value": "800028"}` —

Update query

- `"fieldname": "_postal_code"` — **field name. Custom fields are used by their custom names. System and**
- `"fieldtype": "unicode"` — **data type for custom and default database fields.**
- `"operator": "equal"` — **logical operator.**
- `"value": "800028"` — **value expected by trigger.**
- `"submatch": "object_key"` — **property path. When the value of this property changes, the trigger will fi arr_object field types.**

Field types

field types can be the following:

- `unicode` — **a string of characters or numbers,** `"a1b2c"`
- `int` — **an integer number,** `-1234`
- `bool` — **Boolean,** `true or false`
- `date` — **date in string format,** `"2000-10-24T00:00:00.000Z"`

- `tags` — **tags, an array of strings,** `["T1", "Tag2"]`
- `enum` — **a map composed of key-value pairs** `{"item_one": 1, "item_two": 2}`
- `object` — **object,** `{"A": "B", "C": "D"};`
- `arr_object` — **array of objects,** `[{"A": 1},{"B": 2}]`

Query logical operators

Operator	description	Line	Number	IP	Enum	
“equal”	**value gets equal**	**+**	**+**	**+**	**+**	
“not_equal”	**value gets not equal**	**+**	**+**	**+**	**+**	
“not_exists_or_empty”	**value unset or field removed**	**+**	**+**	**+**	**+**	
“not_empty_or_exists”	**value set or field added**	**+**	**+**	**+**	**+**	
“tag_added_any”	**any tag added**	**-**	**-**	**-**	**-**	
“tag_removed_any”	**any tag removed**	**-**	**-**	**-**	**-**	
“equal_property”	**if the value of the specified property becomes equal to**	**-**	**-**	**-**	**-**	
“not_equal_property”	**if the value of the specified property becomes not equal to**	**-**	**-**	**-**	**-**	
“added_element”	**if a new element is added to the array**	**-**	**-**	**-**	**-**	
“deleted_element”	**if one of the elements is removed from the array**	**-**	**-**	**-**	**-**	

Create trigger mailing

Description

Creates a new trigger mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/add/`

Request parameters

Parameter	Type	Example	Required	Description
name	**string**	**"New Awesome Mailing"**	**Yes**	**New mailing name**
is_active	**bool**	**true**	**No**	**Activates mailing instantly**
content	**JSON array**	`[ { "from_email": "", "from_name": "", "message_id": 12, "replyto_email": "", "replyto_name": "", "subject": "Weather alert" } ]`	**Yes**	**Message template object**
subid	**string**	**"Adv_1"**	**No**	**Mailing SubID**
trigger_type	**string**	**"api_call"**	**Yes**	**Trigger type**
skip_minutes	**int**	**0**	**No**	**Wait for X minutes before sending**
resource_id	**int**	**24**	**No**	**resource ID**
send_channel	**string**	**"email" "sms" "push"**	**Yes**	**Sending channel**
content_type	**string**	**"email"**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email channel for sen to use `send_channel`.**
send_email_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email channel for sen to use `send_channel`.**
send_sms_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email channel for sen to use `send_channel`.**
send_push_content	**bool**	**true**	**No**	**Deprecated. If you pass this parameter, the mailing will use the email channel for sen to use `send_channel`.**
policy_id	**int**	**1**	**No**	**Sending policy ID**
attributes	**object**	**"season":"summer_2024"**	**No**	**Assigned attributes and their values**
push_sending_mode	**string**	**"fanout" (default) "last"**	**No**	**Push notification sending mode (push channels only)**
Smart settings				
virtual_sender_id	**int**	**12**	**No**	**Virtual sender ID (Smart accounts)**
is_random_trk_prefix	**bool**	**false**	**No**	**Tracking domain random prefix. (Smart accounts)**
custom_trk_domain	**JSON array**	`["trk.example.com", "trk2.example.com"]`	**No**	**Tracking domains list (Smart accounts)**
blind_carbon_copy	**JSON array**	`["john@example.com", "george@example.com"]`	**No**	**Hidden copy email addresses (Smart accounts)**
unconditional_sending	**bool**	**true**	**No**	**Send messages ignoring subscription status (Smart accounts)**

INFO

Unconditional sending is available only for On-premises and Off-premises licensed instances. To activ object of `main.json` configuration file.

Trigger type parameters

Parameter	Type	Description	api_call	api_import
action	**string**	**Action based trigger**	**No**	**No**
update_query	**JSON array**	`{ "fieldname": "custom_ID", "fieldtype": "int", "operator": "equal", "value": "2000" }`	**No**	**No**
link_name	**string**	**Message link name**	**No**	**No**
message_id	**int**	**Template ID**	**No**	**No**
list_id	**int**	**Database ID**	**Yes**	**Yes**
subscribe_resource_id	**int**	**Resource ID**	**No**	**No**
channel	**string**	**Subscription channel**	**No**	**No**
is_new_sub	**bool**	**New resource trigger**	**No**	**No**
is_new_ch	**bool**	**New channel trigger**	**No**	**No**

Parameter	Type	Description	api_call	api_import
is_new_contact	**bool**	**New contact trigger**	**No**	**No**
is_status_changed	**bool**	**Subscription status change trigger**	**No**	**No**
trigger_statuses	**array(int)**	**Subscription status change options**	**No**	**No**
pixel_id	**int**	**Pixel ID**	**No**	**No**

Request example

JSON

```
{
  "name": "New trigger mailing №1",
  "content": [
    {
      "subject": "Maybe very important message",
      "from_email": "",
      "from_name": "James",
      "replyto_email": "",
      "replyto_name": "",
      "message_id": 14
    }
  ],
  "trigger_type": "api_call",
  "send_channel": "email",
  "list_id": 1
}
```

XML

```
<xml>
    <name>New trigger mailing №1</name>
    <content array='true'>
        <from_name>James</from_name>
        <message_id>15</message_id>
        <subject>Maybe very important message</subject>
    </content>
    <trigger_type>api_call</trigger_type>
    <send_channel>email</send_channel>
    <list_id>1</list_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 95,
        "trigger_link": "http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd6ac71fa5d068d355f65512f8ec196edd62f3de6b87e2969e3/<emai
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>96</saved_id>
        <trigger_link>http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd6fc63524eba02bff065512f8ec196edd62f3de6b87e2969e3/
            <email|md5|xxh>
            </trigger_link>
        </data>
        <error>0</error>
        <error_text>Successful operation</error_text>
    </xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New trigger mailing ID**

Parameter	Type	Description
trigger_link	**string**	**trigger link. Only for trigger_type = “api_call”**

Update trigger mailing

Description

Updates a trigger mailing. Changes mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/update/`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Trigger mailing ID**
name	**string**	**"New Awesome Mailing"**	**Yes**	**New mailing name**
is_active	**bool**	**true**	**No**	**Activates mailing instantly**
content	**JSON array**	`[ { "from_email": "", "from_name": "", "message_id": 12, "replyto_email": "", "replyto_name": "", "subject": "Weather alert" } ]`	**Yes**	**Message template object**
subid	**string**	**"Adv_1"**	**No**	**Mailing SubID**
trigger_type	**string**	**"api_call"**	**Yes**	**Trigger type**
skip_minutes	**int**	**0**	**No**	**Wait for X minutes before se**
resource_id	**int**	**24**	**No**	**resource ID**
policy_id	**int**	**1**	**No**	**Sending policy ID**
push_sending_mode	**string**	**"fanout" (default) "last"**	**No**	**Push notification sending mo**
attributes	**object**	**"season":"summer_2024"**	**No**	**Assigned attributes and thei**
Smart settings				
virtual_sender_id	**int**	**12**	**No**	**Virtual sender ID (Smart accounts)**
is_random_trk_prefix	**bool**	**false**	**No**	**Tracking domain random pre (Smart accounts)**
custom_trk_domain	**JSON array**	`["trk.example.com", "trk2.example.com"]`	**No**	**Tracking domains list (Smart accounts)**
blind_carbon_copy	**JSON array**	`["john@example.com", "george@example.com"]`	**No**	**Hidden copy email addresse (Smart accounts)**
unconditional_sending	**bool**	**true**	**No**	**Send messages ignoring sub (Smart accounts)**

NOTE

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, c send_sms_content, send_push_content, the request will not return an error, but it will not update the s

Trigger type parameters

Parameter	Type	Description	api_call	api_import
action	**string**	**Action based trigger type**	**No**	**No**
update_query	**JSON array**	`{ “fieldname”: “custom_ID”, “fieldtype”: “int”, “operator”: “equal”, “value”: “2000” }`	**No**	**No**
link_name	**string**	**Message link name**	**No**	**No**
message_id	**int**	**Template ID**	**No**	**No**
list_id	**int**	**Database ID**	**Yes**	**Yes**
subscribe_resource_id	**int**	**Subscription trigger ID**	**No**	**No**
channel	**string**	**Subscription channel**	**No**	**No**
is_new_sub	**bool**	**New resource trigger**	**No**	**No**
is_new_ch	**bool**	**New channel trigger**	**No**	**No**
is_new_contact	**bool**	**New contact trigger**	**No**	**No**
is_status_changed	**bool**	**Subscription status change trigger**	**No**	**No**
trigger_statuses	**array(int)**	**Subscription status change options**	**No**	**No**
pixel_id	**int**	**Pixel ID**	**No**	**No**

Request example

JSON

```
{
  "id": 42,
  "name": "New trigger mailing update",
  "content": [
    {
      "subject": "Maybe not very important message",
      "from_email": "",
      "from_name": "Alex",
      "replyto_email": "",
      "replyto_name": "",
      "message_id": 8
    }
  ],
  "trigger_type": "api_call",
  "list_id": 1
}
```

XML

```
<xml>
    <id>42</id>
    <name>New trigger mailing 3</name>
    <content array='true'>
        <from_name>James</from_name>
        <message_id>15</message_id>
        <subject>Maybe very important message</subject>
    </content>
    <trigger_type>api_call</trigger_type>
    <list_id>1</list_id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 42,
    "trigger_link": "http://trk.example.com:8095/trg/3a345a07f01657e7966566e657a34d023a345a07f01802e7/<email|md5|xxh>"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>42</saved_id>
        <trigger_link>http://trk.example.com:8095/trg/3a345a07f01657e7966566e657a34d023a345a07f01802e7/&lt;email|md5|xxh&gt;</trigger_link>
    </data>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Trigger mailing ID**
trigger_link	**string**	**Trigger mailing link. Only for trigger_type = “api_call”**

Trigger launch (API call)

Top priority instantly launched trigger mailings.

> TIP
>
> **To launch a trigger at the precise time of a new profile import you can use Import + trigger request.**

Version 1.1

Description

It has several advantages compared with API V. 1.0:

- **Requests can change mailing content for each launch.**
- **Sends messages to an audience instead of single profiles.**
- **Advanced audience matching — profile ID, custom fields, external database queries.**
- **API token via Authorization header for advanced security.**

Static request URL `/api/v1.1/campaigns/triggers/start/` **as compared to a dynamically generated link bound**

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/start/`

Request parameters

Parameter	Type	Example	Required	
id	**string**	**5511**	**Yes**	**Trigger mailing ID**
content	**JSON object**	`{ "field": "value" }`	**No**	**Additional content for API content filter and template apicontent variables**
attach	**JSON array**	`[]`	**No**	**Attach file(s) to mailing messages**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date in RFC3339 fo delivery failure event will be recorded**
custom_data	**JSON object**	`{ "x": "y"}`	**No**	**Data used for integration action hooks**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is a databases/66420#custom-fields'> tag "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**
custom_request	**JSON object**	`{ "email": { "$in": [ "asd@dsfsd.com", "zxcv@sdfsd.com" ] } }`	**When no matching specified**	**Custom MongoDB database query**

subscription_filter	**JSON object**	**Push example:**
		`"subscription_filter": { "subscription_id": "CAREFULWITHTHATAXE", "provider": "Chrome", "not_strict": true }`
		Email example:
		`"subscription_filter": { "email": "example@example.com", "not_strict": true }`
		SMS example:
		`"subscription_filter": { "phone": "+79106135133", "not_strict": false }`
			No	**Selects a profile's subscription to send**
				Not used if custom_request is set.
				`not_strict` **determines what to do when fitting subscription will be used instead**

Request example

This example triggers a mailing for a single subscriber:

JSON

```
{
  "id": 84,
  "matching": "profile_id",
  "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>84</id>
  <matching>profile_id</matching>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Send messages to subscribers with `Custom_IDs` from a list `["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiW`

JSON

```
{
  "id": 84,
  "custom_request":{"Custom_ID": {"$in": ["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"]}}
}
```

XML

```
<xml>
    <id>84</id>
    <custom_request>{"Custom_ID": {"$in": ["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"]}}</custom_request>
</xml>
```

Insert text `"New videos on your favorite channels"` to replace `{apicontent.email_title}` message content varial

Attach files “日本.txt” and US.txt".

JSON

```
{
    "id": 96,
    "matching": "profile_id",
    "profile_id": "6606e08f10bcdd88297057a7",
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "日本.txt"
        },
        {
            "data": "data:text/csv; base64,SEVMTE8K",
            "name": "US.txt"
        }
    ],
    "content": {
        "data_lines": [
            "Channel A",
            "Channel B"
        ],
        "email_title": "New videos on your favorite channels"
    }
}
```

XML

```
<xml>
  <id>96</id>
  <matching>profile_id</matching>
  <profile_id>6606e08f10bcdd88297057a7</profile_id>
  <attach>
    <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
    <name>日本.txt</name>
  </attach>
  <attach>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
  </attach>
  <content>
    <data_lines>Channel A</data_lines>
    <data_lines>Channel B</data_lines>
    <email_title>New videos on your favorite channels</email_title>
  </content>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Version 1.0

Description

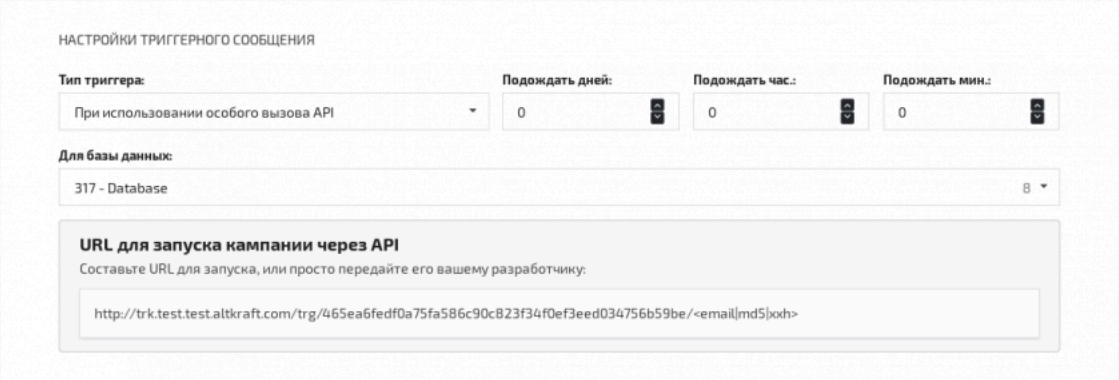
Methods: GET, POST

In this version, each API trigger has a unique link that is displayed after it is created.

`http://<your.tracking.domain>/trg/<base16_value_d1d93d9cca76a7c1742a2f23fbacd>/<email|md5|xxh>`

The link will change if you change the trigger database.

Using link 1.0 allows you to send a trigger to only one profile per request.



`"email"` **is a deprecated parameter.**

GET request

Sends a message with no optional parameters.

Matches profiles by email, md5 hash or XX hash. XX hash profile matching example:

`http://trk.example.org/trg/465ea6fedf0a75fa9ee0b5fb8f9d5fe89af34054909af8bb/dd78094ad0e36ca3`

POST JSON request

The same link is used as in the GET request.

`http://trk.example.org/trg/465ea6fedf0a75fa9ee0b5fb8f9d5fe89af34054909af8bb/dd78094ad0e36ca3`

With POST method you have an option to attach JSON data to message body.

You can override database field values for a mailing message or even update database fields.

JSON

```
{
    "id": 96,
    "matching": "profile_id",
    "profile_id": "6606e08f10bcdd88297057a7",
    "fields": [
        {
            "name": "FirstName",
            "value": "Jason",
            "option": 0
        },
        {
            "name": "Order",
            "value": "123abc456-78",
            "option": 1
        }
    ]
}
```

XML

```
<xml>
  <id>96</id>
  <matching>profile_id</matching>
  <profile_id>6606e08f10bcdd88297057a4</profile_id>
  <fields>
    <name>FirstName</name>
    <value>Jason</value>
    <option>0</option>
  </fields>
  <fields>
    <name>Order</name>
    <value>123abc456-78</value>
    <option>1</option>
  </fields>
</xml>
```

- `"name"` — profile database field name. If no field matches this name — it won't be used.
- `"value"` — profile database field value. Must match field data type.
- `"option"` — (0 | 1 | 2) determines data usage options.
- `"option": 0` — if a database field is not empty — request data will be ignored.
- `"option": 1` — request field value will be used in the message, but will not rewrite database value.
- `"option": 2` — request field value will be used in the message and will rewrite database value.

Profile import + trigger mailing launch

Description

A new customer profile import followed by a trigger mailing launch.

URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/import_and_start`

Request parameters

Parameters	Type	Example	Required	D
Profile import				
data	**JSON object**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**	**Profile fields data**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by _regip or _ip from data object**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for databases/66420#custom-fields'> tag type profile**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**
Trigger launch				
trigger_id	**int**	**13**	**Yes**	**Trigger mailing ID**
timeout	**JSON object**	`{ "days": 0, "hours": 0, "minutes": 5 }`	**No**	**Trigger start timeout. Specify the days, hours, and n**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date in RFC3339 format, after w failure event will be recorded**
custom_data	**JSON object**	`{ "sending_time": "12:31:12" }`	**No**	**Data used for integration Action hooks**
content	**JSON object**	`{ "page": "registration_form" }`	**No**	**Data you can use in a message template using {apic**
attach	**JSON objects array**	`{ "data": "data:text/csv; base64,SEVMTE8K", "name": "Attach.txt" }`	**No**	**A file you can attach to an Email message**

Request example

JSON

```
{
   "matching": "email",
   "email": "example@example.com",
   "detect_geo": true,
   "data": {
      "_fname": "Olly",
      "_lname": "Lambert",
      "email": "example@example.com",
      "phones": [
         "+79000000000"
      ],
      "_bdate": "1990-02-22T21:00:00Z",
      "custom_field": "custom_value",
      "subscriptions": [
         {
            "channel": "email",
            "email": "example@example.com",
            "resource_id": 21
         },
         {
            "channel": "phone",
            "phone": "+79000000000",
            "resource_id": 21
         }
      ]
   },
   "trigger_id": 96,
   "timeout": {
         "days": 0,
         "hours": 0,
         "minutes": 5
   },
   "custom_data": {
      "sending_time": "12:31:12"
   },
     "attach": [
      {
         "data": "data:text/csv; base64,5LuK5pel44GvCg==",
         "name": "Attach.txt"
      }
   ]
}
```


XML

```
<xml>
    <matching>email</matching>
    <email>example@example.com</email>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Garry</_fname>
        <_lname>Sallivan</_lname>
        <_bdate>1990-02-22T21:00:00Z</_bdate>
        <email>example@example.com</email>
        <phones array='true'>+79000000027</phones>
        <custom_field>custom_value</custom_field>
        <subscriptions>
            <channel>email</channel>
            <email>testtriggers90@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000000780</phone>
            <resource_id>21</resource_id>
        </subscriptions>
    </data>
    <trigger_id>96</trigger_id>
    <timeout>
        <days>0</days>
        <hours>0</hours>
        <minutes>5</minutes>
    </timeout>
    <custom_data>
        <sending_time>12:31:12</sending_time>
    </custom_data>
    <attach array='true'>
        <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
        <name>Attach.txt</name>
    </attach>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>660feee02984cf021df478f3</profile_id>
</xml>
```

CAUTION

Subscription to the Resource that is used by the trigger mailing has to be sent within profile data obje
will be sent back:

```
{
    "error": 400,
    "error_text": "No active subscriptions passed for trigger resource"
}
```

Response parameters

Parameter	Type	Descr
error	**int**	**Error code**
error_text	**string**	**Error description**
profile_id	**string**	**Imported profile ID**

Task for bulk trigger launch

Description

Launches a trigger mailing for multiple profiles.
This request is a batch version of the Trigger Launch (API call) request.
Returns a response with the task id, without waiting for completion.

> NOTE
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

URL request

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/start_batch_task`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Trigger mailing ID**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date in RFC3339 form delivery failure event will be recorded**
attach	**JSON array of File object**	`[]`	**No**	**File object**
content	**JSON object**	`{ "H1": "Weather alert", "H2": "Welcome" }`	**No**	**Data you can use in a message template u**
custom_data	**JSON object**	`{ "field": "value" }`	**No**	**Data used for integration Action hooks**
data	**array**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**	**A field in the database and its value, whic**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode. By default - email Learn more.**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only "matching":"email_sub" - subscription data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone" - by phone number from profile or subscriptions "matching":"phone_sub" - by phone number from subscriptions**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**		**"matching":"custom"**	**Name of the custom profile data field**
field_value			**"matching":"custom"**	**Value of the custom profile data field**
custom_request	**JSON object**	`{ "email": { "$in": [ "asd@dsfsd.com", "zxcv@sdfsd.com" ] } }`	**When no matching specified**	**Custom MongoDB database query**

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
task_id	**string**	**Task ID**

Request example (matching by email)

JSON

```
{
    "id": 96,
    "format": "json",
    "matching": "email",
    "data": [
        {
            "email": "example1@example.com"
        },
        {
            "email": "example2@example.com"
        }
    ]
}
```

XML

```
<xml>
    <id>96</id>
    <format>xml</format>
    <matching>email</matching>
    <data>
        <email>example1@example.com</email>
    </data>
    <data>
        <email>example2@altcraft.com</email>
    </data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "id": 96,
    "format": "json",
    "skip_triggers": false,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "field_value": "001"
        },
        {
            "field_value": "002"
        }
    ]
}
```

XML

```
<xml>
    <id>96</id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <data>
        <field_value>001</field_value>
    </data>
    <data>
        <field_value>002</field_value>
    </data>
</xml>
```

Response example

JSON

```
{
         "error": 0,
         "error_text": "Successful operation",
         "task_id": "30421978-7f6d-4f39-913f-fb6cad8e2477"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <task_id>392b7e46-e4d9-4a07-959a-1e541138b684</task_id>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
    "error": 400,
    "error_debug": "http: request body too large",
    "error_text": "Invalid request"
}
```

XML

```
<xml>
  <error>400</error>
  <error_debug>http: request body too large</error_debug>
  <error_text>"Invalid request"</error_text>
</xml>
```

Task for bulk profiles import + trigger launch

Description

Add or update multiple profiles and then send a trigger mailing.
This request is a batch version of the Profile import + trigger mailing launch request.
Returns a response with the task id, without waiting for completion.

> NOTE
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

Request URL

Метод: POST

`http://example.com/api/v1.1/campaigns/triggers/import_and_start_batch_task`

Request parameters

Parameters	Type	Example	Required	D
Profile import				
data	**JSON object**	[{ "data": { "_fname": "John", "_lname": "Lambert", "email": "profile4@example.com", "phones": "+790000000000" } }, { "data": { "_fname": "Tom", "_lname": "Hard", "email": "profile3@example.com", "phones": "+790000000001" } }]	**Yes**	**Profile fields data**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date in RFC3339 format, after failure event will be recorded**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by _regip or _ip from data object**
Profile matching mode				
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is available for databases/66420#custom-fields'> tag type profile 2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**
Trigger launch				
trigger_id	**int**	**13**	**Yes**	**Trigger mailing ID**
custom_data	**JSON object**	{ "sending_time": "12:31:12" }	**No**	**Data used for integration Action hooks**
content	**JSON object**	{ "page": "registration_form" }	**No**	**Data you can use in a message template using {ap**
attach	**JSON objects array**	{ "data": "data:text/csv; base64,SEVMTE8K", "name": "Attach.txt" }	**No**	**A file you can attach to an Email message**

Response parameters

Параметр	Тип	Описание
error_text	**string**	**Текст ошибки**
error	**int**	**Код ошибки**
task_id	**string**	**Идентификатор задачи**

Request example (matching by email)

JSON

```
{
    "matching": "email",
    "data": [
        {
            "data": {
                "_fname": "Victoria",
                "_lname": " Beckham",
                "email": "example1@example.com",
                "phones": "+79000000000",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example2@example.com",
                        "resource_id": 21
                    },
                    {
                        "channel": "phone",
                        "phone": "+79000005478",
                        "resource_id": 20
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "David",
                "_lname": " Beckham",
                "email": "example3@example.com",
                "phones": "+79000000001",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example14@example.com",
                        "resource_id": 21
                    },
                    {
                        "channel": "phone",
                        "phone": "+79000000000",
                        "resource_id": 20
                    }
                ]
            }
        }
    ],
    "trigger_id": 96,
    "timeout": {
        "days": 0,
        "hours": 0,
        "minutes": 5
    },
    "content": {
        "page": "registration_form"
    },
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "Attach.txt"
        }
    ]
}
```

XML

```
<xml>
    <matching>email</matching>
    <data>
        <_fname>Brooklyn</_fname>
        <_lname>Beckham</_lname>
        <email>example3@example.com</email>
        <phones>+79000849376</phones>
        <subscriptions>
            <channel>email</channel>
            <email>example9@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000005475</phone>
            <resource_id>20</resource_id>
        </subscriptions>
    </data>
    <data>
        <_fname>Romeo</_fname>
        <_lname>Beckham</_lname>
        <email>example32@example.com</email>
        <phones>+79000000651</phones>
        <subscriptions>
            <channel>email</channel>
            <email>@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000880000</phone>
            <resource_id>20</resource_id>
        </subscriptions>
    </data>
    <trigger_id>96</trigger_id>
    <timeout>
        <days>0</days>
        <hours>0</hours>
        <minutes>5</minutes>
    </timeout>
    <content>
        <page>registration_form</page>
    </content>
    <attach array='true'>
        <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
        <name>Attach.txt</name>
    </attach>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "custom",
    "field_name": "Custom_id",
    "data": [
        {
            "data": {
                "_fname": "Tom",
                "_lname": "Cruise",
                "email": "example@example.com",
                "phones": "+79000000000",
                "Custom_id": "0002"
            }
        },
        {
            "data": {
                "_fname": "Lili",
                "_lname": "Hard",
                "email": "example@example.com",
                "phones": "+79000000001",
                "Custom_id": "0001"
            }
        }
    ]
}
```

XML

```
<xml>
    <trigger_id>96</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <data>
        <_fname>Viola</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79000000000</phones>
        <Custom_ID>0002</Custom_ID>
    </data>
    <data>
        <_fname>Will</_fname>
        <_lname>Hard</_lname>
        <email>example@example.com</email>
        <phones>+79000000001</phones>
        <Custom_ID>0001</Custom_ID>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "task_id": "1c1a5861-6d4c-4d6f-bd6a-3ca4f46b217d"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <task_id>30421978-7f6d-4f39-913f-fb6cad8e2477</task_id>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
         "error":400,
         "error_debug":"http: request body too large",
         "error_text":"Invalid request"
}
```


XML


```
<xml>
    <error>400</error>
    <error_text>Invalid request</error_text>
</xml>
```


Checking task status

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON


```
{
    "task_id": "task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5"
}
```


XML


```
<xml>
    <task_id>d521abe0-1abf-4e1d-b247-a790be66a424</task_id>
</xml>
```


Response example

JSON


```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "error": 0,
            "error_text": "",
            "result": [
                {
                    "profile_id": "61e947333f62fc26d384452c"
                },
                {
                    "profile_id": "61e947333f62fc26d384452e"
                }
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <progress>100</progress>
        <result>
            <captured>0</captured>
            <status>ok</status>
        </result>
        <status>completed</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Bulk trigger launch

Description

Launches a trigger mailing for multiple profiles.
This request is a batch version of the Trigger Launch (API call) request.

> **NOTE**
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

URL request

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/start_batch`

Request parameters

Parameter	Type	Example	Required	De
trigger_id	**int**	**1**	**Yes**	**Trigger mailing ID**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date in RFC3339 format, after wh failure event will be recorded**
attach	**JSON array of File object**	`[]`	**No**	**File object**
content	**JSON object**	`{ "H1": "Weather alert", "H2": "Welcome" }`	**No**	**Data you can use in a message template using {apico**
custom_data	**JSON object**	`{ "field": "value" }`	**No**	**Data used for integration Action hooks**
data	**array**	`[ {"email": "profile1@example.com"}, {"email": "profile2@example.com"} ]`	**Yes**	**A field in the database and its value, which will be us**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email Learn more.**

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Profile IDs to which the mailing was sent**

Request example (matching by email)

JSON

```
{
    "trigger_id": 8,
    "format": "json",
    "matching": "email",
    "data": [
        {
            "email": "example1@example.com"
        },
        {
            "email": "example2@example.com"
        }
    ]
}
```

XML

```
<xml>
    <trigger_id>85</trigger_id>
    <format>xml</format>
    <matching>email</matching>
    <data>
        <email>example1@example.com</email>
    </data>
    <data>
        <email>example2@example.com</email>
    </data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "field_value": "003"
        },
        {
            "field_value": "002"
        }
    ]
}
```

XML

```
<xml>
  <trigger_id>85</trigger_id>
  <format>xml</format>
  <skip_triggers>false</skip_triggers>
  <matching>custom</matching>
  <field_name>Custom_ID</field_name>
  <data>
    <field_value>YY</field_value>
  </data>
    <data>
    <field_value>AA</field_value>
  </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "6606e08f10bcdd88297057a7"
        },
        {
            "profile_id": "66101437dc4594c2934da6da"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <result>
        <profile_id>66101437dc4594c2934da6da</profile_id>
    </result>
    <result>
        <profile_id>6606e08f10bcdd88297057a7</profile_id>
    </result>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
         "error":400,
         "error_debug":"http: request body too large",
         "error_text":"Invalid request"
}
```

XML

```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>"Invalid request"</error_text>
</xml>
```

Bulk profiles import + trigger mailing launch

Description

Add or update multiple profiles and then send a trigger mailing.
This request is a batch version of the Profile import + trigger mailing launch request.

> **NOTE**
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

Request URL

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/import_and_start_batch`

Request parameters

Parameters	Type	Example	Required	
Profile import				
data	**JSON object**	`[ { "data": { "_fname": "John", "_lname": "Lambert", "email": "profile4@example.com", "phones": "+790000000000" } }, { "data": { "_fname": "Tom", "_lname": "Hard", "email": "profile3@example.com", "phones": "+790000000001" } } ]`	**Yes**	**Profile fields data**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
skip_invalid_subscriptions	**bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**bool**	**true**	**No**	**Detect location by _regip**
subscription_filter	**JSON object**	**Push example:** `"subscription_filter": { "subscription_id": "CAREFULWITHTHATAXE", "provider": "Chrome", "not_strict": true }` **Email example:** `"subscription_filter": { "email": "example@example.com", "not_strict": true }` **SMS example:** `"subscription_filter": { "phone": "+79106135133", "not_strict": false }`	**No**	**Selects a profile's subscr** **Not used if custom_reque** `not_strict` **determines wh — another fitting subscrip**
Profile matching mode				
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscriptions**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions** **"matching":"email_profile" - profile data only**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field** TIP **Searching by combinati and-databases/66420#c recorded as a string: "t**
resource_id	**int**	**24**	**No**	**Resource ID for email and**
Trigger launch				
trigger_id	**int**	**13**	**Yes**	**Trigger mailing ID**
timeout	**JSON object**	`{ "days": 0, "hours": 0, "minutes": 5 }`	**No**	**Trigger start timeout. Spe period.**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**	**Message expiration date queue and a delivery failu**

Parameters	Type	Example	Required	

custom_data	**JSON object**	<pre>{
		  "sending_time": "12:31:12"
		}</pre>
			No	**Data used for integration**

content	**JSON object**	<pre>{
		  "page": "registration_form"
		}</pre>
			No	**Data you can use in a me**

attach	**JSON objects array**	<pre>{
		  "data": "data:text/csv; base64,SEVMTE8K",
		  "name": "Attach.txt"
		}</pre>
			No	**A file you can attach to a**

Response parameters

Parameter	Type	Description
error_text	**string**	**Error text**
error	**int**	**Error code**
result	**array**	**IDs of successfully imported profiles**

Request example (matching by email)

JSON

```
{
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "email",
    "data": [
        {
            "data": {
                "_fname": "Lili",
                "_lname": "Collins",
                "email": "example@example.com",
                "phones": "+79000000000",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "Eva",
                "_lname": "Collins",
                "email": "example@example.com",
                "phones": "+79000000001",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        }
    ]
}
```

XML

```
<xml>
    <trigger_id>96</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>email</matching>
<data>
    <_fname>Lili</_fname>
    <_lname>Collins</_lname>
    <email>example@example.com</email>
    <phones>+79000000000</phones>
        <subscriptions>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
</data>
<data>
    <_fname>Eva</_fname>
    <_lname>Collins</_lname>
    <email>example@example.com</email>
    <phones>+79000000001</phones>
    <subscriptions>
        <channel>email</channel>
        <email>example@example.com</email>
        <priority>0</priority>
        <resource_id>21</resource_id>
        <status>0</status>
    </subscriptions>
</data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "trigger_id": 85,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "data": {
                "_fname": "Adama",
                "_lname": "Fuller",
                "email": "example@example.com,
                "phones": "+79000000000",
                "Custom_ID": "345",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "Eva",
                "_lname": "Fuller",
                "email": "example@example.com",
                "phones": "+79000000001",
                "Custom_ID": "567",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        }
    ]
}
```


XML

```
<xml>
    <trigger_id>85</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>custom</matching>
    <field_name>custom_field</field_name>
    <data>
        <_fname>Irina</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+79000000000</phones>
        <custom_field>343</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
    <data>
        <_fname>Will</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+79000000001</phones>
        <custom_field>560</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "602103d65ae47a3bc65d5d83"
        },
        {
            "profile_id": "601ec71c5ae47a15d1c9b472"
        }
    ]
}
```

XML

```
<xml>
   <error>0</error>
   <error_text>Successful operation</error_text>
   <result>
      <element>
         <profile_id>602103d65ae47a3bc65d5d83</profile_id>
      </element>
      <element>
         <profile_id>601ec71c5ae47a15d1c9b472</profile_id>
      </element>
   </result>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
    "error": 400,
    "error_debug": "http: request body too large",
    "error_text": "Invalid request"
}
```

XML

```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>"Invalid request"</error_text>
</xml>
```

Clone a trigger mailing

Description

Clones content and settings of a trigger mailing to a newly created mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/clone`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Mailing ID**
name	**string**	**“Clone”**	**No**	**New mailing name By default – “Clone of ...”**
is_active	**bool**	**false**	**No**	**Mailing status**

Request example

JSON

```
{
    "id": 96,
    "name": "Clone trigger mailing #1",
    "is_active": false
}
```

XML

```
<xml>
    <name>Clone trigger mailing #2</name>
    <id>95</id>
    <is_active>true</is_active>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 97
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>96</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New mailing ID**

Data array

Description

The `data` array is used in the following queries:

- **Task for bulk trigger launch**
- **Task for bulk profiles import + trigger launch**
- **Bulk trigger launch**
- **Bulk profiles import + trigger mailing launch**

The array consists of objects, each of which contains information about a separate profile. As the key specify the data by which the profile will be searched in the database.

Parameters

Parameter	Type	Example	Required
email	**string**	**"john@example.com"**	**"matching":"email" - by profile and subscriptions** **"matching":"email_profile" - only by profile** **"matching":"email_sub" - only for subscriptions**
phone	**string**	**"+79000000000"**	**"matching":"phone" - by phone number from profile and subscriptions** **"matching":"phone_sub" - by phone number from subscriptions**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_value	**int/string**	**"12345"**	**"matching":"custom"**
custom_request	**JSON object**	`{ "email": { "$in": [ "asd@dsfsd.com", "zxcv@sdfsd.com" ] } }`	**When no matching specified**

Examples

email

JSON

```
 "data": [
                    {"email": "profile1@example.com"},
                    {"email": "profile2@example.com"}
          ]
```

XML

```
<data>
    <email>profile1@example.com</email>
</data>
<data>
    <email>profile2@example.com</email>
</data>
```

phone

JSON

```
 "data": [
                    {"phone": "+79000000000"},
                    {"phone": "+79000000001"}
          ]
```

XML

```
<data>
    <phone>+79000000000</phone>
</data>
<data>
    <phone>+79000000001</phone>
</data>
```

profile_id

JSON

```
 "data": [
                  {"profile_id": "60ae529b3f62fcdebdb159bb"},
                  {"profile_id": "843vfdkisfh934hfe834rhkk"}
         ]
```

XML

```
<data>
    <profile_id>+79000000000</profile_id>
</data>
<data>
    <profile_id>+79000000001</profile_id>
</data>
```

custom

JSON

```
"data": [
         {"field_value": "001"},
         {"field_value": "002"}
         ]
```

XML

```
<data>
    <field_value>001</field_value>
</data>
<data>
    <field_value>002</field_value>
</data>
```

Engage profile in scenario

Description

Engages a customer profile in scenario.

> **TIP**
>
> **When you launch the scenario using the API, the profiles will enter the “Default” start node.**

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/start`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**12**	**Yes**	**Workflow ID**
high_priority	**bool**	**true**	**No**	**Overrides the processing priority of the used. See High-priority scenarios**
content	**JSON-object**	**{}**	**No**	**Additional data to be used in the messa variable can be used in the "API call" a**
attach	**JSON-массив**	**[]**	**No**	**Attaches file(s) to the request**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field value.** TIP **Searching by combination of tags is a databases/66420#custom-fields'> tag "tag-1, tag-2".**
resource_id	**int**	**24**	**No**	**Resource ID for email and phone matching**

subscription_filter	**JSON object**	**Push example:**
		`"subscription_filter": { "subscription_id": "CAREFULWITHTHATAXE", "provider": "Chrome", "not_strict": true }`
		Email example:
		`"subscription_filter": { "email": "example@example.com", "not_strict": true }`
		SMS example:
		`"subscription_filter": { "phone": "+79106135133", "not_strict": false }`
			No	**Selects a profile's subscription to send**
				`not_strict` **determines what to do when fitting subscription will be used instead**

Request example 1

Engages a single profile matched by Profile ID.

JSON

```
{
  "id": 11,
  "matching": "profile_id",
  "profile_id": "5a74876369d42613b31947b8"
}
```

XML

```
<xml>
  <id>11</id>
  <matching>profile_id</matching>
  <profile_id>5a74876369d42613b31947b8</profile_id>
</xml>
```

Request example 2

**Overwrites message subject by placing “New videos on your favorite channels” for “{apicontent.email_t
is used in mailing node templates.**
Attaches “日本.txt” and “US.txt” files and a content object to be used in API content filter query.

JSON

```
{
    "id": 12,
    "matching": "profile_id",
    "profile_id": "66102b984a5d12bd6cc0912f",
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "日本.txt"
        },
        {
            "data": "data:text/csv; base64,SEVMTE8K",
            "name": "US.txt"
        }
    ],
    "content": {
        "data_lines": [
            "Channel A",
            "Channel B"
        ],
        "email_title": "New videos on your favorite channels"
    }
}
```

XML

```
<xml>
  <id>12</id>
  <matching>profile_id</matching>
  <profile_id>661027b04a5d12bd6cc09125</profile_id>
  <attach>
    <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
    <name>日本.txt</name>
  </attach>
  <attach>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
  </attach>
  <content>
    <data_lines>Channel A</data_lines>
    <data_lines>Channel B</data_lines>
    <email_title>New videos on your favorite channels</email_title>
  </content>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Import and engage profile in scenario

Description

Use this request to import customer profile and engage it in scenario.

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start`

Request parameters

Parameter	Type	Example	Required	Desc
content	**JSON-object**	`{}`	**No**	**Additional content for API content filter node and template apicontent variables**
workflow_id	**Int**	**1**	**Yes**	**Worklow identifier**
high_priority	**Bool**	**true**	**No**	**Overrides the processing priority of the created task. If no priority scenarios**
matching	**String**	**“email”**	**Yes**	**Profile matching mode**
profile_id	**String**	**“5f19953a03e4227216278ba4”**	*	**Import: matching fields**
email	**String**	**“importStart9219@example.com”**	*	**Import: matching fields**
phone	**String**	**“+79001234567”**	*	**Import: matching fields**
field_name	**String**	**“CRM_ID”**	*	**Import: matching fields**
field_value	**Any**	**“12345”**	*	**Import: matching fields**
data	**Object**	{ “_fname”: “John”, “_lname”: “Doe” }	**No**	**Import: Profile fields data**
skip_triggers	**Bool**	**true**	**No**	**Import options: skip triggers and workflows activated by i**
skip_invalid_subscriptions	**Bool**	**true**	**No**	**Import options: skip invalid subscriptions**
detect_geo	**Bool**	**true**	**No**	**Import options: detect geolocation by IP**
content	**Object**	“content”:{ “one”: “My email = importStart9219@example.com”, “two”: “Timestamp: 1595512477”, “three”: “Three” }	**No**	**Additional data to be used in the message template as {ap the “API call” and “API content filter” nodes**
attach	**array of{name, flie_key, data, mimetype}**		**No**	**Trigger options: attach files to message**

Request example

Import one customer profile and engage it in scenario.

JSON

```
{
  "data": {
    "subscriptions": [
      {
        "email": "example001@example.com",
        "resource_id": 7
      }
    ]
  },
  "matching": "email",
  "email": "example001@example.com",
  "content": {
    "one": "My email = example001@example.com",
    "two": "Timestamp: 1595512477",
    "three": "Three"
  },
  "workflow_id": 12
}
```

XML

```
<xml>
  <data>
    <subscriptions array='true'>
      <email>example001@example.com</email>
      <resource_id>7</resource_id>
    </subscriptions>
  </data>
  <matching>email</matching>
  <email>example001@example.com</email>
  <content>
    <one>My email = example001@example.com</one>
    <two>Timestamp: 1595512477</two>
    <three>Three</three>
  </content>
  <workflow_id>12</workflow_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "5eeb54a2ce9448518e85fdad"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>6613c13a4a5d12bd6cc09156</profile_id>
</xml>
```

Response parameters

Parameter	Type	D
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**Profile identifier**

Service responses

Code	Description
0	**Operation is successful**
400,450	**Incorrect request**
401	**API token required**
402	**Tariff limit reached**
403	**No rights for this action**
404	**Object is not found in token vision zone**
409	**Other record with same unique attributes already exists**
413	**Database is not available in selected resource**
415	**Requested Content-Type is not supported**
435	**Ambiguous search, there are several objects with given attributes**
441	**Object belongs to another group**
500	**Internal service error**
501	**Method does not exist**

Batch import and engage profiles in a scenario

Description

The request allows you to import multiple profiles and immediately send them to a scenario. It is a bat
and engage profile in scenario.

> **NOTE**
>
> **The request body is limited by default to 10MB, exception: files – 100MB**
> **It is not recommended to transfer more than 10,000 profiles in one request.**

URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start_batch`

Request parameters

Parameter	Type	Example	Required	
workflow_id	**Int**	**1**	**Yes**	**Workflow identifier**
high_priority	**Bool**	**true**	**No**	**Overrides the processing priority c is used. See High-priority scenario**
matching	**String**	**"email"**	**Yes**	**Profile matching mode. By default – email.**
profile_id	**String**	**"5f19953a03e4227216278ba4"**	**"matching":"profile_id"**	**Profile identifier**
email	**String**	**"importStart9219@example.com"**	**"matching":"email" - from profile or subscriptions** **"matching":"email_profile" - only from profile**	**Email**
phone	**String**	**"+79001234567"**	**"matching":"phone"**	**Phone number**
field_name	**String**	**"CRM_ID"**	**"matching":"custom"**	**The name of the custom profile fie**
field_value	**Any**	**"12345"**	**"matching":"custom"**	**The value of the custom profile fie** **Search by combination of tags is a**
data	**Object**	\{ "_fname": "John", "_lname": "Doe" }	**No**	**Profile data, including subscriptior**
skip_triggers	**Bool**	**true**	**No**	**Skip triggers and workflows activa By default – false**
skip_invalid_subscriptions	**Bool**	**true**	**No**	**Skip invalid subscriptions By default – false**
detect_geo	**Bool**	**true**	**No**	**Detect geolocation by IP (it is dete**
content	**Object**	"content":\{ "one": "My email = importStart9219@example.com", "two": "Timestamp: 1595512477", "three": "Three" }	**No**	**Additional data to be used in the n variable can be used in the "API ca**
attach	**array of \{name, flie_key, data, mimetype}**		**No**	**File to attach to an Email message**

Response parameters

Parameter	Type	
error_text	**string**	**Error text**
error	**int**	**Error code**
task_id	**string**	**Task identifier**

Request example – “matching”:“email”

JSON

```
{
  "workflow_id": 12,
  "matching": "email",
  "data": [
    {
      "data": {
        "_fname": "Alisa",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000000000"
      }
    },
    {
      "data": {
        "_fname": "Alex",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000000000"
      }
    }
  ]
}
```

XML

```
<xml>
  <workflow_id>10</workflow_id>
  <matching>email</matching>
  <data>
    <_fname>John</_fname>
    <_lname>Lambert</_lname>
    <email>example@example.com</email>
  </data>
  <data>
    <_fname>Bert</_fname>
    <_lname>Lambert</_lname>
    <email>example@example.com</email>
  </data>
</xml>
```

Request example – “matching”:“custom”

JSON

```
{
  "workflow_id": 12,
  "matching": "custom",
  "field_name": "CRM_ID",
  "data": [
    {
      "data": {
        "_fname": "Alina",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79006574081",
        "CRM_ID": "00066"
      }
    },
    {
      "data": {
        "_fname": "Victor",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000005422",
        "CRM_ID": "00076"
      }
    }
  ]
}
```

XML

```
<xml>
    <workflow_id>12</workflow_id>
    <matching>custom</matching>
    <field_name>New_field_7</field_name>
    <data>
        <_fname>Polina</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79006574082</phones>
        <New_field_7 string = "true" >00067</New_field_7>
    </data>
    <data>
        <_fname>Dariya</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79000005423</phones>
        <New_field_7 string = "true">00077</New_field_7>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "6613c4754a5d12bd6cc0915f"
        },
        {
            "profile_id": "6613c4754a5d12bd6cc09163"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <result>
        <profile_id>6613c4bf4a5d12bd6cc09167</profile_id>
    </result>
    <result>
        <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
    </result>
</xml>
```

Response example when a request limitation is triggered

JSON

```
{
  "error": 400,
  "error_debug": "http: request body too large",
  "error_text": "Invalid request"
}
```

XML

```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>Invalid request</error_text>
</xml>
```

Task for batch import and engaging profiles in the scenario

Description

Adding or updating multiple profiles and then sending them to the scenario. Returns a response with th execution.

> NOTE
>
> **The request body is limited by default to 10mb, exception: files are 100mb.**
>
> **It is not recommended to transmit more than 10,000 profiles per request.**

URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start_batch_task`

Request parameters

Parameter	Type	Example	Required	
workflow_id	Int	1	Yes	Workflow identifier
high_priority	bool	true	No	Overrides the processing priority of the used. See High-priority scenarios
matching	String	"email"	Yes	Profile matching mode. By default – email.
profile_id	String	"5f19953a03e4227216278ba4"	"matching":"profile_id"	Profile identifier
email	String	"importStart9219@example.com"	"matching":"email" - from profile or subscriptions "matching":"email_profile" - only from profile	Email
phone	String	"+79001234567"	"matching":"phone"	Phone number
field_name	String	"CRM_ID"	"matching":"custom"	The name of the custom profile field fo
field_value	Any	"12345"	"matching":"custom"	The value of the custom profile field fo Search by combination of tags is availa
data	Object	\{ "_fname": "John", "_lname": "Doe" }	No	Profile data, including subscriptions
skip_triggers	Bool	true	No	Skip triggers and workflows activated By default – false
skip_invalid_subscriptions	Bool	true	No	Skip invalid subscriptions By default – false
detect_geo	Bool	true	No	Detect geolocation by IP (it is detected
content	Object	"content":\{ "one": "My email = importStart9219@example.com", "two": "Timestamp: 1595512477", "three": "Three" }	No	Additional data to be used in the messa variable can be used in the "API call" a
custom_data	JSON object	"custom_data": \{ "field": "value", }	No	Add some data to be used in [Action ho The parameter can be passed either at
attach	array of \{name, flie_key, data, mimetype}		No	File to attach to an Email message (onl

Returned parameters

Parameter	Type	
error_text	string	Error text
error	int	Error code
task_id	string	Task identifier

Request example – “matching”:“email”

JSON

```
{
  "workflow_id": 12,
  "matching": "email",
  "data": [
    {
      "data": {
        "_fname": "Herbert",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79862014852"
      }
    },
    {
      "data": {
        "_fname": "Bert",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79862014853"
      }
    }
  ]
}
```

XML

```
<xml>
  <workflow_id>12</workflow_id>
  <matching>email</matching>
  <data>
      <_fname>Herb</_fname>
      <_lname>Limbert</_lname>
      <email>example@example.com</email>
      <phones>+79847076617</phones>
  </data>
  <data>
      <_fname>Bert</_fname>
      <_lname>Limbert</_lname>
      <email>example@example.com</email>
      <phones>+79817076617</phones>
  </data>
</xml>
```

Request example – “matching”:“custom”

JSON

```
{
    "workflow_id": 12,
    "matching": "custom",
    "field_name": "New_field_7",
    "data": [
        {
            "data": {
                "_fname": "Kristina",
                "_lname": "Lambert",
                "email": "example@example.com",
                "phones": "+79568150395",
                "New_field_7": "00015"
            }
        },
        {
            "data": {
                "_fname": "Christopher",
                "_lname": "Lambert",
                "email": "example@example.com",
                "phones": "+79568150396",
                "New_field_7": "00016"
            }
        }
    ]
}
```

XML

```
<xml>
    <workflow_id>12</workflow_id>
    <matching>custom</matching>
    <field_name>CRM_ID</field_name>
    <data>
        <_fname>Naomy</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+798144890880</phones>
        <CRM_ID>7830</CRM_ID>
    </data>
    <data>
        <_fname>Nataly</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+73755334351</phones>
        <CRM_ID>0934</CRM_ID>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "task_id": "30421978-7f6d-4f39-913f-fb6cad8e2477"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <task_id>675231ce-63b4-4bca-a153-10d87b938d26</task_id>
</xml>
```

Example of a response when a request restriction is triggered

JSON

```
{
  "error": 400,
  "error_debug": "http: request body too large",
  "error_text": "Invalid request"
}
```

XML

```
<xml>
  <error>400</error>
  <error_debug>http: request body too large</error_debug>
  <error_text>Invalid request</error_text>
</xml>
```

Checking the status of the task

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON

```
{
  "task_id": "task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5"
}
```

XML

```
<xml>
  <task_id>task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "error": 0,
            "error_text": "",
            "result": [
                {
                    "profile_id": "61e947333f62fc26d384452c"
                },
                {
                    "profile_id": "61e947333f62fc26d384452e"
                }
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <error>0</error>
            <error_text></error_text>
            <result>
                <profile_id>61e947333f62fc26d384452c</profile_id>
            </result>
            <result>
                <profile_id>61e947333f62fc26d384452e</profile_id>
            </result>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get scenarios list

Description

Retrieves all created scenarios list.

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	`1`	**No**	**The first ID in the response** **By default –** `1`
limit	**int**	`1`	**No**	**Resources list limit** **By default –** `1000`
groups	**[]string**	`[ "group1", "group2" ]`	**No**	**List of access groups** **By default –** `[]`
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access groups IDs** **By default –** `[]`
ui_tags	**[]string**	`["tag1", "tag2"]`	**No**	**List of tags** **By default –** `[]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`	**No**	**Attributes assigned to the ma**

Request example

JSON

```
{
    "from_id": 10,
    "limit": 1,
    "ui_tags": [
        "B2C"
    ]
}
```

XML

```
<xml>
    <from_id>10</from_id>
    <limit>1</limit>
    <ui_tags array='true'>B2C</ui_tags>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 10,
            "update_time": "2025-05-21T09:17:53.362Z",
            "name": "Government tip",
            "list_id": 1,
            "is_active": true,
            "high_priority": true,
            "ui_tags": [
                "B2C"
            ],
            "mode": "skip",
            "groups": [
                {
                    "ID": 2,
                    "title": "Sales and promotion"
                }
            ],
            "nodes": [
                {
                    "name": "DEFAULT_START_NODE_TITLE",
                    "type": "start",
                    "node_id": 0
                },
                {
                    "name": "ANY_IMPORT_START_NODE_TITLE",
                    "type": "start",
                    "node_id": 2
                },
                {
                    "templates": [
                        1
                    ],
                    "name": "Welcome letter",
                    "type": "mailing",
                    "node_id": 3,
                    "campaign_id": 17
                },
                {
                    "name": "1 d.0 h.0 m.",
                    "type": "delay",
                    "node_id": 4
                },
                {
                    "name": "Is the subscription confirmed?",
                    "type": "condition",
                    "node_id": 5
                },
                {
                    "name": "Got the SMS reminder?",
                    "type": "condition",
                    "node_id": 6
                },
                {
                    "templates": [
                        5
                    ],
                    "name": "SMS message",
                    "type": "mailing",
                    "node_id": 7,
                    "campaign_id": 30
                },
                {
                    "name": "1 d.0 h.0 m.",
                    "type": "delay",
                    "node_id": 8
                },
                {
                    "name": "Application to a call center or other product",
                    "type": "feedback",
                    "node_id": 9
                },
                {
                    "title": "Guidelines",
                    "type": "feedback",
                    "node_id": 10
                },
                {
                    "name": "Specify code",
```

```
                    "type": "promo_attach",
                    "node_id": 11
                },
                {
                    "templates": [
                        22
                    ],
                    "name": "Offer",
                    "type": "mailing",
                    "node_id": 12
                    "campaign_id": 39
                }
                {
                    "name": "Go to scenario #29",
                    "type": "workflow_jump",
                    "node_id": 13,
                    "next_workflow_id": 29,
                    "next_node_id": 0
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 11,
    "total_count": 20
}
```

XML

```
<xml>
    <data>
        <groups>
            <id>2</id>
            <update_time>2025-05-21T09:17:53.362Z</update_time>
            <name>Sales and promotion</name>
        </groups>
        <id>10</id>
        <is_active>true</is_active>
        <high_priority>true</high_priority>
        <list_id>1</list_id>
        <mode>skip</mode>
        <name>Administrative chain</name>
        <nodes>
            <name>DEFAULT_START_NODE_TITLE</name>
            <node_id>0</node_id>
            <type>start</type>
        </nodes>
        <nodes>
            <name>ANY_IMPORT_START_NODE_TITLE</name>
            <node_id>2</node_id>
            <type>start</type>
        </nodes>
        <nodes>
            <campaign_id>17</campaign_id>
            <name>Welcome letter</name>
            <node_id>3</node_id>
            <templates>1</templates>
            <type>mailing</type>
        </nodes>
        <nodes>
            <name>1 д.0 ч.0 м.</name>
            <node_id>4</node_id>
            <type>delay</type>
        </nodes>
        <nodes>
            <name>Is the subscription confirmed?</name>
            <node_id>5</node_id>
            <type>condition</type>
        </nodes>
        <nodes>
            <name>Got the SMS reminder?</name>
            <node_id>6</node_id>
            <type>condition</type>
        </nodes>
        <nodes>
            <campaign_id>30</campaign_id>
            <name>SMS message</name>
            <node_id>7</node_id>
            <templates>5</templates>
            <type>mailing</type>
        </nodes>
        <nodes>
            <name>1 д.0 ч.0 м.</name>
            <node_id>8</node_id>
            <type>delay</type>
        </nodes>
        <nodes>
            <name>Application to a call center
            or other product</name>
            <node_id>9</node_id>
            <type>feedback</type>
        </nodes>
        <nodes>
            <name>Guidelines</name>
            <node_id>10</node_id>
            <type>feedback</type>
        </nodes>
        <nodes>
            <name>Specify code</name>
            <node_id>11</node_id>
            <type>promocode_attach</type>
        </nodes>
        <nodes>
            <campaign_id>39</campaign_id>
            <name>Offer</name>
            <node_id>12</node_id>
            <templates>22</templates>
            <type>mailing</type>
        </nodes>
```

```
        <nodes>
            <name>Go to scenario #29</name>
            <next_node_id>0</next_node_id>
            <next_workflow_id>29</next_workflow_id>
            <node_id>13</node_id>
            <type>workflow_jump</type>
        </nodes>
        <ui_tags>B2C</ui_tags>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>11</next_from_id>
    <total_count>20</total_count>
</xml>
```

Response parameters

Parameter	Type	description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Resource ID**
name	**string**	**Resource name**
list_id	**int**	**Database ID**
is_active	**bool**	**Active or inactive scenario**
high_priority	**bool**	**Whether the scenario is high-priority. Tasks of high-priority scenarios are processed by dedicated workers. See High-prio**
mode	**string**	**Mode if the profile is recaptured by the scenario**
ui_tags	**array**	**List of tags**
groups	**array of objects**	**List of groups**
nodes	**array of objects**	**List of nodes**
total_count	**int**	**Total resources count**
next_from_id	**int**	**Identifier of the following scenario**
node_id	**int**	**Node Identifier**
campaign_id	**int**	**Mailing Identifier (for node “Channels”)**
templates	**int**	**Template Identifier (for node “Channels”)**
next_workflow_id	**int**	**Identifier of the next scenario (for the scenario transition node)**
next_node_id	**int**	**Identifier of the next node (for the scenario transition element)**
update_time	**string**	**When the scenario was updated**
attributes	**object**	**Attributes assigned to the mailing and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Activate scenario

Description

The request is used to activate a scenario.

URL

Method: POST

`https://example.com/api/v1.1/workflows/activate`

Request parameters

Parameter	Type	Example	Required	Des
id	**int**	**1**	**Yes**	**Scenario ID**
skip_profiles	**bool**	**true**	**No**	**Clear the scenario before reactivating**

Request example

JSON

```
{
    "id": 1,
    "skip_profiles": true
}
```

XML

```
<xml>
    <id>1</id>
    <skip_profiles>true</skip_profiles>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Deactivate scenario

Description

The request is used to deactivate a scenario.

URL

Method: POST

`https://example.com/api/v1.1/workflows/deactivate`

Request parameters

Parameter	Type	Example	Required	Des
id	**int**	**1**	**Yes**	**Scenario ID**
skip_profiles	**bool**	**true**	**No**	**Clear the scenario before reactivating**

Request example

JSON

```
{
    "id": 1,
    "skip_profiles": true
}
```

XML

```
<xml>
    <id>1</id>
    <skip_profiles>true</skip_profiles>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get scenario information

Description

Get detailed information about an automation scenario.

Request URL

Method: GET, POST

`https://example.com/api/v1.1/workflows/get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Scenario identifier**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 10,
        "update_time": "2025-05-21T09:17:53.362Z",
        "name": "Welcome chain",
        "list_id": 1,
        "is_active": true,
        "high_priority": true,
        "ui_tags": [
            "B2C"
        ],
        "mode": "skip",
        "groups": [
            {
                "id": 2,
                "name": "Sales and promotion"
            }
        ],
        "attributes": {
            "season": "spring_2024",
            "branch": "marketing"
        },
        "nodes": [
            {
                "name": "DEFAULT_START_NODE_TITLE",
                "type": "start",
                "node_id": 0
            },
            {
                "templates": [
                    1
                ],
                "name": "Welcome letter",
                "type": "campaign",
                "node_id": 3,
                "campaign_id": 17
            },
            {
                "name": "1 d.0 h.0 m.",
                "type": "delay",
                "node_id": 4
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <id>10</id>
        <update_time>2025-05-21T09:17:53.362Z</update_time>
        <name>Welcome chain</name>
        <list_id>1</list_id>
        <is_active>true</is_active>
        <high_priority>true</high_priority>
        <ui_tags>B2C</ui_tags>
        <mode>skip</mode>
        <groups>
            <id>2</id>
            <name>Sales and promotion</name>
        </groups>
        <attributes>
            <season>spring_2024</season>
            <branch>marketing</branch>
        </attributes>
        <nodes>
            <name>DEFAULT_START_NODE_TITLE</name>
            <type>start</type>
            <node_id>0</node_id>
        </nodes>
        <nodes>
            <name>Welcome letter</name>
            <type>campaign</type>
            <node_id>3</node_id>
            <campaign_id>17</campaign_id>
            <templates>1</templates>
        </nodes>
        <nodes>
            <name>1 d.0 h.0 m.</name>
            <type>delay</type>
            <node_id>4</node_id>
        </nodes>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
data object		
id	**int**	**Scenario identifier**
name	**string**	**Scenario name**
list_id	**int**	**Profile database identifier**
is_active	**bool**	**Whether the scenario is active**
high_priority	**bool**	**Whether the scenario is high-priority. Tasks of high-priority scenarios are processed by dedicated workers. See High-priority s**
mode	**string**	**Mode when a profile enters the scenario more than once**
ui_tags	**array**	**List of tags**
groups	**array of objects**	**List of groups**
attributes	**object**	**Attributes assigned to the scenario and their values**
nodes	**array of objects**	**List of scenario elements**
update_time	**string**	**When the scenario was updated**

The `nodes` object fields are described in the Get scenarios list article.

Change scenario priority

Description

The request enables or disables high priority for a scenario. Tasks of high-priority scenarios are proces
High-priority scenarios.
The change affects new scenario tasks. Tasks that are already in the queue are processed with the old

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/priority`

Request parameters

Parameter	Type	Example	Required	De
id	**int**	**1**	**Yes**	**Scenario identifier**
high_priority	**bool**	**true**	**Yes**	**Scenario high priority:** `true` **or** `false`

Request example

JSON

```
{
    "id": 1,
    "high_priority": true
}
```

XML

```
<xml>
    <id>1</id>
    <high_priority>true</high_priority>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get profile tier in a loyalty program

Description

Retrieves the current tier of a loyalty program member within the requested tiers group.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/tiers`

Request parameters

Parameter	Type	Example	Required	De
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
tiers_group_id	**int**	**10**	**Yes**	**Tiers group ID (for example, 10)**
matching	**string**	**“email”**	**Yes**	**Profile lookup method. Possible values:** `"email"`, `"pho`
field_name	**string**	**“user_id”**	**Yes, if** `matching = custom`	**Custom field name used for lookup**
field_value	**string / int**	**123456**	**Yes, if** `matching = custom`	**Custom field value**
email	**string**	**“user@example.com”**	**Yes, if** `matching = email`	**Email used to find the profile**
phone	**string**	**“+1234567890”**	**Yes, if** `matching = phone`	**Phone number in international format**
profile_id	**string**	**“profile123”**	**Yes, if** `matching = profile_id`	**Profile ID**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "tiers_group_id": 10,
  "matching": "email",
  "email": "user@example.com",
  "phone": "+1234567890",
  "profile_id": "profile123"
}
```

XML

```
<xml>
    <loyalty_program_id>123</loyalty_program_id>
    <tiers_group_id>10</tiers_group_id>
    <matching>email</matching>
    <email>user@example.com</email>
    <phone>+1234567890</phone>
    <profile_id>profile123</profile_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "tiers_group_type": "cumulative",
      "tier_eid": "silver1-2",
      "tier_id": 13,
      "tier_order": 1,
      "reached_at": "2024-06-15T09:12:34Z"
    },
    {
      "tiers_group_type": "cumulative",
      "tier_eid": "gold1-3",
      "tier_id": 14,
      "tier_order": 2,
      "reached_at": "2024-09-01T12:34:56Z"
    }
  ]
}
```

XML

```
<xml>
    <data>
        <tiers_group_type>cumulative</tiers_group_type>
        <tier_eid>silver1-2</tier_eid>
        <tier_id>13</tier_id>
        <tier_order>1</tier_order>
        <reached_at>2024-06-15T09:12:34Z</reached_at>
    </data>
    <data>
        <tiers_group_type>cumulative</tiers_group_type>
        <tier_eid>gold1-3</tier_eid>
        <tier_id>14</tier_id>
        <tier_order>2</tier_order>
        <reached_at>2024-09-01T12:34:56Z</reached_at>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
tiers_group_type	**string**	**Tiers group type**
tier_eid	**string**	**External tier ID**
tier_id	**int**	**Tier ID**
tier_order	**int**	**Tier order number (starting from 0)**
reached_at	**string**	**Tier reached date in RFC3339 format**

Export points transactions

Description

**Export a report of transactions for the selected points currency over a specified period to a CSV file. T
the task status and download the resulting file.**

Step 1. Create an export task

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/transactions_export`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
points_currency_eid	**string**	**“bonus_points”**	**Yes, if `points_currency_iid` is not provided**	**External points currency ID**
points_currency_iid	**int**	**1**	**Yes, if `points_currency_eid` is not provided**	**Internal points currency ID**
from_id	**string**	**“a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8”**	**No**	**Transaction ID (UUID) for pagination; if**
date_from	**string (RFC3339)**	**“2023-09-01T00:00:00Z”**	**Yes**	**Period start date (inclusive)**
date_to	**string (RFC3339)**	**“2023-09-30T23:59:59Z”**	**Yes**	**Period end date (inclusive)**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "date_from": "2023-09-01T00:00:00Z",
  "date_to": "2023-09-30T23:59:59Z",
  "from_id": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8"
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <date_from>2023-09-01T00:00:00Z</date_from>
  <date_to>2023-09-30T23:59:59Z</date_to>
  <from_id>a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8</from_id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "task_id": "task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
task_id	**string (UUID)**	**Export task ID**

Step 2. Check export status

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	Example	
task_id	**string**	**“task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5”**	**Yes**

Request example

JSON

```
{
  "task_id": "task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5"
}
```

XML

```
<xml>
  <task_id>task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5</task_id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "status": "completed",
    "progress": 100,
    "result": {
      "hashed_file": "jvIXPQX2T9MxHZuKnxGHCz1ZxdvYKGjcyy_Fy31ByPCo6wNtU1lyrCMTfagIIURdpL8jbRysirZ0gJcmVNCXhcX5qK_trhiw/Transactions_loyalty
59Z.zip",
      "is_api": true,
      "loyalty_id": 1,
      "next_from_id": "1148b0b0-d33d-4d27-b582-8058ce7fc740",
      "points_currency_iid": 0,
      "total": 1000
    }
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <status>completed</status>
    <progress>100</progress>
    <result>
      <hashed_file>jvIXPQX2T9MxHZuKnxGHCz1ZxdvYKGjcyy_Fy31ByPCo6wNtU1lyrCMTfagIIURdpL8jbRysirZ0gJcmVNCXhcX5qK_trhiw/Transactions_loyalty_1
59Z.zip</hashed_file>
      <is_api>true</is_api>
      <loyalty_id>1</loyalty_id>
      <next_from_id>1148b0b0-d33d-4d27-b582-8058ce7fc740</next_from_id>
      <points_currency_iid>0</points_currency_iid>
      <total>1000</total>
    </result>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
status	**string**	**Task status**
progress	**int**	**Execution progress**
hashed_file	**string**	**Hashed file name for download**
is_api	**bool**	**Export via API**
loyalty_id	**int**	**Loyalty program ID**

Parameter	Type	Description
next_from_id	**string**	**Next transaction ID for pagination; if not empty, not all data has been exported and you can pass this UUID as `from_id` in a new export r**
points_currency_iid	**int**	**Internal points currency ID**
total	**int**	**Total number of exported transactions**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 3. Download the result

The file download link looks like this:

`https://example.com/api/download/jvIXPQX2T9MxHZuKnxGHCz1ZxdvYKGjcyy_Fy31ByPCo6wNtU1lyrCMTfagIIURdpL8jbRysirZ0gJcmVNCXh`
`09-01T00%2000%2000Z_2026-09-30T23%2059%2059Z.zip`

- `https://example.com` — **platform domain**
- `api/download/` — **path to the download endpoint**
- `jvIXPQX2T9MxHZuKnxGHCz1ZxdvYKGjcyy_Fy31ByPCo6wNtU1lyrCMTfagIIURdpL8jbRysirZ0gJcmVNCXhcX5qK_trhiw/Transactions_loya`
  `30T23%2059%2059Z.zip` — **value of the `hashed_file` parameter from the previous response**

To download the file, send a GET request to the specified link with the authorization header `Authorizati`

Expiring points for a period

Description

Returns the number of points that are subject to expiration during the specified period for the selected

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/expiring_points`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
points_currency_eid	**string**	**“bonus_points”**	**Yes, if `points_currency_iid` is not provided**	**External points currency ID**
points_currency_iid	**int**	**1**	**Yes, if `points_currency_eid` is not provided**	**Internal points currency ID**
date_from	**string (RFC3339)**	**“2023-09-01T00:00:00Z”**	**Yes**	**Period start date (inclusive)**
date_to	**string (RFC3339)**	**“2023-09-30T23:59:59Z”**	**Yes**	**Period end date (inclusive)**
matching	**string**	**“email”**	**Yes**	**Profile lookup method. Possible value**
field_name	**string**	**“user_id”**	**Yes, if `matching = custom`**	**Custom field name**
field_value	**string / int**	**“12345”**	**Yes, if `matching = custom`**	**Custom field value**
email	**string**	**“user@example.com”**	**Yes, if `matching = email`**	**Member email**
phone	**string**	**“+1234567890”**	**Yes, if `matching = phone`**	**Member phone number in internationa**
profile_id	**string**	**“profile_123”**	**Yes, if `matching = profile_id`**	**Profile ID**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "matching": "email",
  "email": "user@example.com",
  "date_from": "2023-09-01T00:00:00Z",
  "date_to": "2023-09-30T23:59:59Z"
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
  <date_from>2023-09-01T00:00:00Z</date_from>
  <date_to>2023-09-30T23:59:59Z</date_to>
</xml>
```

Response example

JSON

```
{
  "data": {
    "expiring_points": "150.24"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <expiring_points>150.24</expiring_points>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
expiring_points	**string**	**Number of points that will expire during the specified period**

Get profile account transactions

Description

Returns the transaction history for a loyalty program member’s points account. Supports pagination an

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/transactions`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
points_currency_eid	**string**	**“bonus_points”**	**Yes, if `points_currency_iid` is not provided**	**External points currency ID**
points_currency_iid	**int**	**1**	**Yes, if `points_currency_eid` is not provided**	**Internal points currency ID**
from_id	**string**	**“a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8”**	**No**	**UUID of the last transaction for**
limit	**int**	**20**	**No**	**Number of records per page (1–1**
matching	**string**	**“email”**	**Yes**	**Profile lookup method. Possible**
field_name	**string**	**“user_id”**	**Yes, if `matching = custom`**	**Custom field name**
field_value	**string / int**	**“12345”**	**Yes, if `matching = custom`**	**Custom field value**
email	**string**	**“user@example.com”**	**Yes, if `matching = email`**	**Member email**
phone	**string**	**“+1234567890”**	**Yes, if `matching = phone`**	**Member phone number in interna**
profile_id	**string**	**“profile_123”**	**Yes, if `matching = profile_id`**	**Profile ID**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "matching": "email",
  "email": "user@example.com",
  "limit": 20
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
  <limit>20</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "transaction_id": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8",
      "transaction_eid": "trans_12345",
      "created_at": "2023-09-01T12:34:56Z",
      "transaction_type": "accrual",
      "amount": "100.1",
      "balance_before": "500.5",
      "balance_after": "600.6",
      "promo_iid": 42,
      "promo_eid": "promo_2023",
      "description": "Points accrual for a purchase",
      "activation_date": "2023-09-01T12:34:56Z",
      "expiration_date": "2023-12-31T23:59:59Z",
      "source_type": "order",
      "source_iid": 789,
      "source_sid": "order_123",
      "source_subiid": 1,
      "is_temp": false,
      "custom_json": {"key": "value"},
      "product_eid": "prod_123",
      "sku_eid": "sku_456",
      "canceled": false
    }
  ],
  "next_from_id": "b2c3d4e5-f6g7-8901-h2i3-j4k5l6m7n8o9",
  "total_count": 150,
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <transaction_id>a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8</transaction_id>
    <transaction_eid>trans_12345</transaction_eid>
    <created_at>2023-09-01T12:34:56Z</created_at>
    <transaction_type>accrual</transaction_type>
    <amount>100.1</amount>
    <balance_before>500.5</balance_before>
    <balance_after>600.6</balance_after>
    <promo_iid>42</promo_iid>
    <promo_eid>promo_2023</promo_eid>
    <description>Points accrual for a purchase</description>
    <activation_date>2023-09-01T12:34:56Z</activation_date>
    <expiration_date>2023-12-31T23:59:59Z</expiration_date>
    <source_type>order</source_type>
    <source_iid>789</source_iid>
    <source_sid>order_123</source_sid>
    <source_subiid>1</source_subiid>
    <is_temp>false</is_temp>
    <custom_json>{"key": "value"}</custom_json>
    <product_eid>prod_123</product_eid>
    <sku_eid>sku_456</sku_eid>
    <canceled>false</canceled>
  </data>
  <next_from_id>b2c3d4e5-f6g7-8901-h2i3-j4k5l6m7n8o9</next_from_id>
  <total_count>150</total_count>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string (UUID)**	**Unique transaction ID**
transaction_eid	**string**	**External transaction ID**
created_at	**string (RFC3339)**	**Transaction creation date and time**
transaction_type	**string**	**Transaction type (`accrual`, `redeem`, `expire`, etc.)**
amount	**string**	**Number of accrued or redeemed points**
balance_before	**string**	**Balance before the operation**
balance_after	**string**	**Balance after the operation**

Parameter	Type	Description
promo_iid	**int**	**Internal promotion ID**
promo_eid	**string**	**External promotion ID**
description	**string**	**Transaction description**
activation_date	**string (RFC3339)**	**Points activation date**
expiration_date	**string (RFC3339)**	**Points expiration date**
source_type	**string**	**Transaction source (** `api` , `manual` , `system` , `order` , `workflow` , `tier_group` **)**
product_eid	**string**	**External product ID**
sku_eid	**string**	**External SKU ID**
canceled	**boolean**	**Transaction canceled flag**
next_from_id	**string (UUID)**	**UUID for the next page**
total_count	**int**	**Total number of transactions**

Get trigger promotions list

Description

Returns a list of loyalty program trigger promotions. Pagination and filtering by active status are suppo

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/trigger_promotions`

Request parameters

Parameter	Type	Example	Required	Description
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
from_id	**int**	**0**	**No**	**Promotion ID for pagination (records with ID > `from_id`)**
limit	**int**	**20**	**No**	**Number of records (1–100), default 20**
is_active	**boolean**	**true**	**No**	**Active status filter: `true` — active, `false` — inactive, omitted — all**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "limit": 20,
  "is_active": true
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <limit>20</limit>
  <is_active>true</is_active>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 5,
      "name": "Trigger promotion: Bonus for purchase",
      "description": "Bonus accrual for each purchase",
      "external_id": "trigger_promo_2023_q3",
      "start_date": "2023-09-01T00:00:00Z",
      "end_date": "2023-09-30T23:59:59Z",
      "is_active": true,
      "redeem_rules": [
        {
          "points_currency_iid": 1,
          "points_currency_eid": "curr1",
          "static_amount": "100.16",
          "min_amount": "50",
          "max_amount": "500"
        }
      ],
      "accrual_rules": [
        {
          "points_currency_iid": 1,
          "points_currency_eid": "curr1",
          "static_amount": "100",
          "min_amount": "50",
          "max_amount": "500"
        }
      ]
    }
  ],
  "next_from_id": 8,
  "total_count": 134,
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <id>5</id>
    <name>Trigger promotion: Bonus for purchase</name>
    <description>Bonus accrual for each purchase</description>
    <external_id>trigger_promo_2023_q3</external_id>
    <start_date>2023-09-01T00:00:00Z</start_date>
    <end_date>2023-09-30T23:59:59Z</end_date>
    <is_active>true</is_active>
  </data>
  <next_from_id>8</next_from_id>
  <total_count>134</total_count>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Unique promotion ID**
name	**string**	**Promotion name**
description	**string**	**Promotion description**
external_id	**string**	**External promotion ID**
start_date	**string (RFC3339)**	**Start date**
end_date	**string (RFC3339)**	**End date**
is_active	**boolean**	**Active flag**
redeem_rules	**array**	**Redemption rules (`points currency`, values)**
accrual_rules	**array**	**Accrual rules (`points currency`, values)**
next_from_id	**int**	**ID for the next page**
total_count	**int**	**Total number of promotions**

Accrue points to a member

Description

Accrues points to a loyalty program member’s balance using a trigger-based promotion.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/accrue_points`

Request parameters

Parameter	Type	Example	Required	Des
loyalty_program_id	int	123	Yes	Loyalty program ID
promotion_id	int	25	No	Trigger promotion ID
points_currency_iid	int	1	No	Internal points currency ID
points_currency_eid	string	“stars”	No	External points currency ID
amount	string	“100”	Yes	Amount of points to accrue
is_temp	boolean	false	No	Temporary transaction flag. Can be rolled back by time
temp_timeout_sec	int	123	No	Timeout in seconds after which a temporary transactio
created_at	string	“2024-09-01T12:34:56Z”	No	Transaction creation date and time (current time if omi
activation_date	string	“2024-09-01T12:34:56Z”	No	Points activation date (points are active immediately if
expiration_date	string	“2024-12-31T23:59:59Z”	No	Points expiration date (taken from currency settings if
transaction_eid	string	“uiuiuiiu-ouiihfs-sdfsdf”	No	External transaction ID
external_data	object	`{"order_id": "123"}`	No	Arbitrary additional transaction data (JSON object)
matching	string	“email”	Yes	Profile lookup method: `email`, `phone`, `profile_id`, `custom`
field_name	string	“user_id”	Yes, if `matching = custom`	Custom field name for lookup
field_value	string / int	123456	Yes, if `matching = custom`	Custom field value
email	string	“user@example.com”	Yes, if `matching = email`	Member email
phone	string	“+1234567890”	Yes, if `matching = phone`	Member phone number in international format
profile_id	string	“profile123”	Yes, if `matching = profile_id`	Profile ID

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "promotion_id": "PROMO123",
  "points_currency_eid": "stars",
  "amount": "100",
  "matching": "email",
  "email": "user@example.com",
  "is_temp": false,
  "transaction_eid": "uiuiuiiu-ouiihfs-sdfsdf",
  "external_data": {
    "order_id": "ORDER-123"
  }
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <promotion_id>PROMO123</promotion_id>
  <points_currency_eid>stars</points_currency_eid>
  <amount>100</amount>
  <matching>email</matching>
  <email>user@example.com</email>
  <is_temp>false</is_temp>
  <transaction_eid>uiuiuiiu-ouiihfs-sdfsdf</transaction_eid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance_before": "500.00",
    "balance_after": "600.50"
  }
}
```


XML


```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance_before>500.00</balance_before>
    <balance_after>600.50</balance_after>
  </data>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Points accrual transaction ID**
balance_before	**string**	**Points balance before accrual**
balance_after	**string**	**Points balance after accrual**

Redeem member points

Description

Redeems points from a loyalty program member's account using a trigger-based promotion.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/redeem_points`

Request parameters

Parameter	Type	Example	Required	Des
loyalty_program_id	int	123	Yes	Loyalty program ID
promotion_id	int	123	No	Trigger promotion ID
points_currency_iid	int	1	No	Internal points currency ID
points_currency_eid	string	"stars"	No	External points currency ID (for example, `"stars"`)
amount	string	"100"	Yes	Amount of points to redeem
is_temp	boolean	false	No	Temporary transaction flag. Can be rolled back by time
temp_timeout_sec	int	123	No	Timeout in seconds after which a temporary transactio
created_at	string	"2024-09-01T12:34:56Z"	No	Transaction creation date and time (current time if omi
transaction_eid	string	"uiuiuiiu-ouiihfs-sdfsdf"	No	External transaction ID
external_data	object	`{"order_id": "123"}`	No	Arbitrary additional transaction data (JSON object)
matching	string	"email"	Yes	Profile lookup method: `email`, `phone`, `profile_id`, `custom`
field_name	string	"user_id"	Yes, if `matching = custom`	Custom field name for lookup
field_value	string / int	123456	Yes, if `matching = custom`	Custom field value
email	string	"user@example.com"	Yes, if `matching = email`	Member email
phone	string	"+1234567890"	Yes, if `matching = phone`	Member phone number in international format
profile_id	string	"profile123"	Yes, if `matching = profile_id`	Profile ID

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "promotion_id": "PROMO123",
  "points_currency_eid": "stars",
  "amount": "100",
  "matching": "email",
  "email": "user@example.com",
  "is_temp": false,
  "transaction_eid": "uiuiuiiu-ouiihfs-sdfsdf",
  "external_data": {
    "order_id": "ORDER-123"
  }
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <promotion_id>PROMO123</promotion_id>
  <points_currency_eid>stars</points_currency_eid>
  <amount>100</amount>
  <matching>email</matching>
  <email>user@example.com</email>
  <is_temp>false</is_temp>
  <transaction_eid>uiuiuiiu-ouiihfs-sdfsdf</transaction_eid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance_before": "500.00",
    "balance_after": "399.50"
  }
}
```


XML


```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance_before>500.00</balance_before>
    <balance_after>399.50</balance_after>
  </data>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Points redemption transaction ID**
balance_before	**string**	**Points balance before redemption**
balance_after	**string**	**Points balance after redemption**

Commit temporary transaction

Description

Commits a temporary transaction created with the `is_temp` flag. If a temporary points accrual transact transaction is created.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/commit_transaction`

Request parameters

Parameter	Type	Example	Required	
transaction_id	**string**	**“TRANS12345”**	**No**	**Temporary transaction ID**
transaction_eid	**string**	**“TRANS12345”**	**No**	**External temporary transaction ID**

(At least one identifier must be provided.)

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Committed transaction ID**
balance	**string**	**Current active points balance of the member**

Preliminary Order Calculation

Description

The `preorder` method allows you to perform a preliminary order calculation considering promotions and
The request result can be used to display final amounts, discounts, and write-offs before creating the f

> **TIP**
>
> **This method does not create an order in the system; it returns the calculated state of the order with a**
> **To create the calculated order after a successful preorder, use the `preorder_commit` method.**

URL

Method: POST

`https://example.com/api/v1.1/market/orders/preorder`

Request Parameters

Parameter	Type	Example	Required	
external_id	**string**	**"12"**	**Yes**	**External order ID**
matching	**string**	**"email" "phone" "profile_id" "custom"**	**Yes**	**Method for searching the loyalty prog**
email	**string**	**"example@example.com"**	**If `matching = email`**	**Member's email**
phone	**string**	**"+79000000000"**	**If `matching = phone`**	**Member's phone number**
profile_id	**string**	**"696e304f547840a7286619e0"**	**If `matching = profile_id`**	**Profile ID**
data	**JSON object**	\{ "endpoint": "site", "status": "new", "currency": "RUB", "total_price": "139999.00", "lines": [...] \}	**Yes**	**The [`data`](#data-object-preorder) ob**

> **TIP**
>
> **The `lines` field is required and must contain at least one order item.**

data object

Parameter	Type	Example	Required	
endpoint	**string**	**"bot", "website"**	****Yes****	**Order source (**
status	**string**	**"new", "confirmed", "paid"**	****Yes****	**Order status. U**
currency	**string**	**"RUB", "USD", "EUR"**	****Yes****	**Order currency**
total_price	**string**	**"2500.00"**	****Yes****	**Total order cos**
final_total_price	**string**	**"2150.00"**	****Yes****	**Final order cos**
delivery_cost	**string**	**"300.00"**	**No**	**Delivery cost**
create_time	**string (ISO 8601)**	**"2026-02-18T13:01:05.141Z"**	**No**	**Order creation used**
region	**string**	**"region_moscow"**	**No**	**Region identifi**
lines	**array**	**See "lines array" section below**	****Yes****	**Array of order**
promocodes	**array**	**See "promocodes array" section below**	**No**	**Array of promo**
applied_promotions	**array**	**See "applied_promotions array" section below**	**No**	**Promotions ap**
redeem_points	**array**	**See "redeem_points array" section below**	****Yes**, if `type="redeem_points"` is specified in `applied_promotions`**	**Array of point**

custom_fields	**object**	{
		"field1": "value1",
		"field2": "value2"
		}
			No	**Order custom f**

tags	**array**	["summer_sale", "first_order"]	**No**	**Tags for labeli**

lines array

The array contains information about the order items. Must contain at least 1 item.
Each item is a separate object within the array with the following parameters:

Parameter	Type	Example	Required	
external_id	**string**	**"line_1"**	****Yes****	**External line ID in th**
product	**string**	**"iphone17pmb256"**	****Yes*****	**Product identifier**
sku	**string**	**"IPHONE17-256-BLACK"**	****Yes*****	**SKU identifier**
line_number	**integer**	**1**	**No**	**Sequential line numb**

count	**integer**	**1**	****Yes****	**Quantity of the prod**
base_price_per_item	**string**	**"139999.00"**	****Yes****	**Base price per unit**
min_price_per_item	**string**	**"130000.00"**	**No**	**Minimum allowable p**
final_price_per_item	**string**	**"139500.00"**	****Yes****	**Final price per unit a**
status	**string**	**"new"**	**No**	**Order line status**
custom_fields	**object**	**\{"color": "black", "size": "XL"\}**	**No**	**Line custom fields**
applied_promotions	**array**	**See "applied promotions array" section below**	**No**	**Promotions applied t**

*** — one of the parameters is required: `product` or `sku`**

promocodes array

The array contains objects with information about promo codes.
A full description of the promo code structure can be found in this article.

Parameter	Type	Example	Required	
code	**string**	**"SUMMER20"**	****Yes****	**Promo code**
pool_id	**integer**	**42**	****Yes****	**Promo code pool identifier**

redeem_points array

An array with information about loyalty point redemption. Each object describes a separate redemptio
A full description of the point redemption structure can be found in this article.

Parameter	Type	Example	Required	
points_currency_iid	**integer**	**1**	****Yes*****	**Internal identifier of the point**
points_currency_eid	**string**	**"bonus1"**	****Yes*****	**External identifier of the poin**
name	**string**	**"Bonus points"**	**No**	**Name of the point type**
amount	**integer**	**499**	****Yes****	**Number of points to redeem**

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

applied_promotions array

> **TIP**
> **The `applied_promotions` array can be contained both in the `data` object and in the `lines` array within tha array will contain promotions applied to the order, in the second — promotions applied to the product**

An array of promotions applied to the order. Each object in the array describes a separate promotion.

Parameter	Type	Example	Required
type	**string**	**"discount", "accrue_points", "redeem_points", "code_issue", "message"**	****Yes****
promotion	**object**	`"promotion": { "external_id": "promo_1", "name": "Summer Sale" }`	****Yes****
amount	**string**	**"150.00"**	****Yes,** if `type` is `discount`, `accrue_points`, or `redeem_points`**
points_info	**object**	`"points_info": { "points_currency_iid": 1, "points_currency_eid": "bonus1", "name": "Bonus points", "amount": "50" }`	****Yes,** if `type` is `accrue_points` or `redeem_points`**

The structure of the `points_info` object is as follows:

Parameter	Type	Example	Required	
points_currency_iid	**integer**	**1**	****Yes*****	**Internal ide**
points_currency_eid	**string**	**"bonus1"**	****Yes*****	**External ide**
name	**string**	**"Bonus points"**	**No**	**Name of the**
amount	**string**	**"6999"**	****Yes****	**Number of p**
expiration_date	**string (ISO 8601)**	**"2027-02-18T00:00:00.000Z"**	**No**	**Expiration d**

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

Request Example

JSON

```
{
  "external_id": "12",
  "matching": "email",
  "email": "example@example.com",
  "data": {
    "endpoint": "site",
    "status": "new",
    "delivery_cost": "0.00",
    "total_price": "139999.00",
    "currency": "RUB",
    "redeem_points": [
      {
        "points_currency_iid": 1,
        "amount": 499
      }
    ],
    "lines": [
      {
        "external_id": "line___ASTGRf",
        "product": "iphone17pmb256",
        "base_price_per_item": "139999.00",
        "min_price_per_item": "0.00",
        "final_price_per_item": "139999.00",
        "count": 1,
        "custom_fields": {},
        "currency": "RUB"
      }
    ]
  }
}
```


XML


```
<xml>
  <external_id>12</external_id>
  <matching>email</matching>
  <email>example@example.com</email>
  <data>
    <endpoint>site</endpoint>
    <status>new</status>
    <delivery_cost>0.00</delivery_cost>
    <total_price>139999.00</total_price>
    <currency>RUB</currency>
    <redeem_points>
      <points_currency_iid>1</points_currency_iid>
      <amount>499</amount>
    </redeem_points>
    <lines>
      <external_id>line___ASTGRf</external_id>
      <product>iphone17pmb256</product>
      <base_price_per_item>139999.00</base_price_per_item>
      <min_price_per_item>0.00</min_price_per_item>
      <final_price_per_item>139999.00</final_price_per_item>
      <count>1</count>
      <currency>RUB</currency>
    </lines>
  </data>
</xml>
```


Response Example

JSON

```
{
    "data": {
        "profile": {
            "profile_id": "696e304f547840a7286619e0"
        },
        "order": {
            "processing_target": "calculated",
            "total_price": "139999.00",
            "final_total_price": "139500.00",
            "delivery_cost": "0.00",
            "currency": "RUB",
            "lines": [
                {
                    "external_id": "line___ASTGRf",
                    "line_number": 0,
                    "count": 1,
                    "product": "iphone17pmb256",
                    "base_price_per_item": "139999.00",
                    "min_price_per_item": "0.00",
                    "final_price_per_item": "139500.00",
                    "custom_fields": {},
                    "applied_promotions": [
                        {
                            "type": "accrue_points",
                            "promotion": {
                                "external_id": "bonus",
                                "name": "Point Accrual"
                            },
                            "points_info": {
                                "points_currency_iid": 1,
                                "points_currency_eid": "bonus1",
                                "name": "Main Point",
                                "amount": "6999"
                            }
                        },
                        {
                            "type": "redeem_points",
                            "promotion": {
                                "external_id": "bonus",
                                "name": "Point Redemption"
                            },
                            "points_info": {
                                "points_currency_iid": 1,
                                "points_currency_eid": "bonus1",
                                "name": "Main Point",
                                "amount": "499"
                            }
                        }
                    ]
                }
            ],
            "custom_fields": {},
            "applied_promotions": [],
            "points_info": [
                {
                    "points_currency_iid": 1,
                    "points_currency_eid": "bonus1",
                    "name": "Main Point",
                    "status": "success",
                    "active_amount_for_current_order": "20999",
                    "accrue_points_amount_for_current_order": "6999",
                    "redeemed_points_amount_for_current_order": "499"
                }
            ],
            "promocodes_info": []
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
    <profile>
      <profile_id>696e304f547840a7286619e0</profile_id>
    </profile>
    <order>
      <processing_target>calculated</processing_target>
      <total_price>139999.00</total_price>
      <final_total_price>139500.00</final_total_price>
      <delivery_cost>0.00</delivery_cost>
      <currency>RUB</currency>
      <lines>
        <external_id>line___ASTGRf</external_id>
        <line_number>0</line_number>
        <count>1</count>
        <product>iphone17pmb256</product>
        <base_price_per_item>139999.00</base_price_per_item>
        <min_price_per_item>0.00</min_price_per_item>
        <final_price_per_item>139500.00</final_price_per_item>
        <applied_promotions>
          <type>accrue_points</type>
          <promotion>
            <external_id>bonus</external_id>
            <name>Point Accrual</name>
          </promotion>
          <points_info>
            <points_currency_iid>1</points_currency_iid>
            <points_currency_eid>bonus1</points_currency_eid>
            <name>Main Point</name>
            <amount>6999</amount>
          </points_info>
        </applied_promotions>
        <applied_promotions>
          <type>redeem_points</type>
          <promotion>
            <external_id>bonus</external_id>
            <name>Point Redemption</name>
          </promotion>
          <points_info>
            <points_currency_iid>1</points_currency_iid>
            <points_currency_eid>bonus1</points_currency_eid>
            <name>Main Point</name>
            <amount>499</amount>
          </points_info>
        </applied_promotions>
      </lines>
      <points_info>
        <points_currency_iid>1</points_currency_iid>
        <points_currency_eid>bonus1</points_currency_eid>
        <name>Main Point</name>
        <status>success</status>
        <active_amount_for_current_order>20999</active_amount_for_current_order>
        <accrue_points_amount_for_current_order>6999</accrue_points_amount_for_current_order>
        <redeemed_points_amount_for_current_order>499</redeemed_points_amount_for_current_order>
      </points_info>
    </order>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response Parameters

Parameter	Type	Descriptio
data	**object**	**Calculation result data**
error	**int**	**Error code**
error_text	**string**	**Error description**

The `data` object has the following structure.

Parameter	Type	Description
profile	**object**	**Member profile. Contains the profile identifier**
order	**object**	**Calculated order. More details**

Order Confirmation

Description

**The `preorder_commit` method performs the final order confirmation after a successful preliminary calcula
method creates the order in the system, applies all calculated discounts, accrues and redeems points,**

URL

Method: POST

`https://example.com/api/v1.1/market/orders/preorder_commit`

Request Parameters

Parameter	Type	Example	Required	Des
external_id	**string**	**"order_12345"**	****Yes****	**External order ID in the client's system**
profile_id	**string**	**"696e304f547840a7286619e0"**	****Yes*****	**Profile identifier. Field used for matching**
email	**string**	**"user@example.com"**	****Yes*****	**Member's email. Field used for matching**
phone	**string**	**"+79991234567"**	****Yes*****	**Member's phone number. Field used for matching**
skip_triggers	**boolean**	**false**	**No**	**Disable trigger execution when creating the order. Defa**

data	**object**	<pre>"data": {
		"endpoint": "bot",
		"status": "confirmed",
		"currency": "RUB",
		"total_price": "2500.00",
		"final_total_price": "2150.00",
		"delivery_cost": "300.00",
		"lines": [...],
		"promocodes": [...],
		"redeem_points": [...],
		"region": "region_moscow",
		"custom_fields": ,
		"tags": ["tag1", "tag2"]
		}</pre>
			****Yes****	**Object with order data**

*** — to identify the profile, one of the parameters must be provided: `profile_id`, `email`, or `phone`**

data object

Parameter	Type	Example	Required	
endpoint	**string**	**"bot", "website"**	****Yes****	**Order source (touchpoint).**
status	**string**	**"new", "confirmed", "paid"**	****Yes****	**Order status. Uses statuse**
currency	**string**	**"RUB", "USD", "EUR"**	****Yes****	**Order currency (three-lett**
total_price	**string**	**"2500.00"**	****Yes****	**Total order cost before dis**
final_total_price	**string**	**"2150.00"**	****Yes****	**Final order cost after appl**
delivery_cost	**string**	**"300.00"**	**No**	**Delivery cost**
create_time	**string (ISO 8601)**	**"2026-02-18T13:01:05.141Z"**	**No**	**Order creation date and ti**
region	**string**	**"region_moscow"**	**No**	**Region identifier for applyi**
lines	**array**	**See "lines array" section below**	****Yes****	**Array of order items. Must**
promocodes	**array**	**See "promocodes array" section below**	**No**	**Array of promo codes appl**
requested_promotions	**array**	**See "requested_promotions array" section below**	**No**	**Promotions applied to this This field should contain t `/preorder` method respons**
redeem_points	**array**	**See "redeem_points array" section below**	****Yes**, if `type="redeem_points"` is specified in `requested_promotions`**	**Array of point redemption**

custom_fields	**object**	<pre>{
		"field1": "value1",
		"field2": "value2"
		}</pre>
			No	**Order custom fields**

tags	**array**	<pre>[
		"summer_sale",
		"first_order"
		]</pre>
			No	**Tags for labeling the order**

lines array

**The array contains information about the order items. Must contain at least 1 item.
Each item is a separate object within the array with the following parameters:**

Parameter	Type	Example	Required	Description
external_id	**string**	**"line_1"**	****Yes****	**External line ID in the client's system**
product	**string**	**"iphone17pmb256"**	****Yes*****	**Product identifier**
sku	**string**	**"IPHONE17-256-BLACK"**	****Yes*****	**SKU identifier**
line_number	**integer**	**1**	**No**	**Sequential line number in the order**
count	**integer**	**1**	****Yes****	**Quantity of the product**
base_price_per_item	**string**	**"139999.00"**	****Yes****	**Base price per unit**
min_price_per_item	**string**	**"130000.00"**	**No**	**Minimum allowable price per unit**
final_price_per_item	**string**	**"139500.00"**	****Yes****	**Final price per unit after applying all discounts**

status	**string**	**"new"**	**No**	**Order line status**
custom_fields	**object**	**\{"color": "black", "size": "XL"\}**	**No**	**Line custom fields**
requested_promotions	**array**	**See "requested_promotions array" section below**	**No**	**Promotions applied to this line. This field should contain the structure from the `applied_promotions` field of t order line.**

*** — one of the parameters is required: `product` or `sku`**

promocodes array

The array contains objects with information about promo codes.
A full description of the promo code structure can be found in this article.

Parameter	Type	Example	Required	
code	**string**	**"SUMMER20"**	****Yes****	**Promo code**
pool_id	**integer**	**42**	****Yes****	**Promo code pool identifier**

redeem_points array

An array with information about loyalty point redemption. Each object describes a separate redemptio
A full description of the point redemption structure can be found in this article.

Parameter	Type	Example	Required	
points_currency_iid	**integer**	**1**	****Yes*****	**Internal identifier of the point**
points_currency_eid	**string**	**"bonus1"**	****Yes*****	**External identifier of the poin**
name	**string**	**"Bonus points"**	**No**	**Name of the point type**
amount	**integer**	**499**	****Yes****	**Number of points to redeem**

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

requested_promotions array

> **TIP**
> **The `requested_promotions` array can be contained both in the `data` object and in the `lines` array within t array will contain promotions applied to the order, in the second — promotions applied to the product**

An array of promotions applied to the order. Each object in the array describes a separate promotion.

Parameter	Type	Example	Required
type	**string**	**"discount", "accrue_points", "redeem_points", "code_issue", "message"**	****Yes****
promotion	**object**	`"promotion": { "external_id": "promo_1", "name": "Summer Sale" }`	****Yes****
amount	**string**	**"150.00"**	****Yes,** if `type` is `discount`, `accrue_points`, or `redeem_points`**
points_info	**object**	`"points_info": { "points_currency_iid": 1, "points_currency_eid": "bonus1", "name": "Bonus points", "amount": "50" }`	****Yes,** if `type` is `accrue_points` or `redeem_points`**

The structure of the `points_info` object is as follows:

Parameter	Type	Example	Required	
points_currency_iid	**integer**	**1**	****Yes*****	**Internal ide**
points_currency_eid	**string**	**"bonus1"**	****Yes*****	**External ide**
name	**string**	**"Bonus points"**	**No**	**Name of the**
amount	**string**	**"6999"**	****Yes****	**Number of p**
expiration_date	**string (ISO 8601)**	**"2027-02-18T00:00:00.000Z"**	**No**	**Expiration d**

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

Request Example

JSON

```
{
  "external_id": "order_12345",
  "profile_id": "696e304f547840a7286619e0",
  "skip_triggers": false,
  "data": {
    "endpoint": "bot",
    "status": "confirmed",
    "currency": "RUB",
    "total_price": "2500.00",
    "final_total_price": "2150.00",
    "delivery_cost": "300.00",
    "create_time": "2026-02-18T13:01:05.141Z",
    "region": "region_moscow",
    "lines": [
      {
        "external_id": "line_1",
        "product": "popit",
        "base_price_per_item": "1500.00",
        "final_price_per_item": "1350.00",
        "count": 1,
        "requested_promotions": [
          {
            "type": "discount",
            "promotion": {
              "external_id": "promo_1",
              "name": "Summer Sale"
            },
            "amount": "150.00"
          },
          {
            "type": "accrue_points",
            "promotion": {
              "external_id": "bonus_promo",
              "name": "Point Accrual"
            },
            "points_info": {
              "points_currency_iid": 1,
              "points_currency_eid": "bonus1",
              "name": "Bonus points",
              "amount": "75"
            }
          }
        ]
      },
      {
        "external_id": "line_2",
        "sku": "skuf",
        "base_price_per_item": "700.00",
        "min_price_per_item": "650.00",
        "final_price_per_item": "650.00",
        "count": 2,
        "requested_promotions": [
          {
            "type": "redeem_points",
            "promotion": {
              "external_id": "redeem_promo",
              "name": "Point Redemption"
            },
            "points_info": {
              "points_currency_iid": 1,
              "points_currency_eid": "bonus1",
              "name": "Bonus points",
              "amount": "50"
            }
          }
        ]
      }
    ],
    "promocodes": [
      {
        "code": "HYDO-4728-DZGU",
        "pool_id": 42
      }
    ],
    "redeem_points": [
      {
        "points_currency_eid": "bonus1",
        "amount": 100
      }
    ],
```

```
    "requested_promotions": [
      {
        "type": "code_issue",
        "promotion": {
          "external_id": "welcome_promo",
          "name": "Welcome Gift"
        }
      }
    ],
    "custom_fields": {
      "source": "telegram_bot",
      "campaign": "spring_sale"
    },
    "tags": ["premium", "first_order"]
  }
}
```

XML

```xml
<xml>
  <external_id>order_12345</external_id>
  <profile_id>696e304f547840a7286619e0</profile_id>
  <skip_triggers>false</skip_triggers>
  <data>
    <endpoint>bot</endpoint>
    <status>confirmed</status>
    <currency>RUB</currency>
    <total_price>2500.00</total_price>
    <final_total_price>2150.00</final_total_price>
    <delivery_cost>300.00</delivery_cost>
    <create_time>2026-02-18T13:01:05.141Z</create_time>
    <region>region_moscow</region>
    <lines>
      <external_id>line_1</external_id>
      <product>popit</product>
      <base_price_per_item>1500.00</base_price_per_item>
      <final_price_per_item>1350.00</final_price_per_item>
      <count>1</count>
      <requested_promotions>
        <type>discount</type>
        <promotion>
          <external_id>promo_1</external_id>
          <name>Summer Sale</name>
        </promotion>
        <amount>150.00</amount>
      </requested_promotions>
      <requested_promotions>
        <type>accrue_points</type>
        <promotion>
          <external_id>bonus_promo</external_id>
          <name>Point Accrual</name>
        </promotion>
        <points_info>
          <points_currency_iid>1</points_currency_iid>
          <points_currency_eid>bonus1</points_currency_eid>
          <name>Bonus points</name>
          <amount>75</amount>
        </points_info>
      </requested_promotions>
    </lines>
    <lines>
      <external_id>line_2</external_id>
      <sku>skuf</sku>
      <base_price_per_item>700.00</base_price_per_item>
      <min_price_per_item>650.00</min_price_per_item>
      <final_price_per_item>650.00</final_price_per_item>
      <count>2</count>
      <requested_promotions>
        <type>redeem_points</type>
        <promotion>
          <external_id>redeem_promo</external_id>
          <name>Point Redemption</name>
        </promotion>
        <points_info>
          <points_currency_iid>1</points_currency_iid>
          <points_currency_eid>bonus1</points_currency_eid>
          <name>Bonus points</name>
          <amount>50</amount>
        </points_info>
      </requested_promotions>
    </lines>
    <promocodes>
      <code>HYDO-4728-DZGU</code>
      <pool_id>42</pool_id>
    </promocodes>
    <redeem_points>
      <points_currency_eid>bonus1</points_currency_eid>
      <amount>100</amount>
    </redeem_points>
    <requested_promotions>
      <type>code_issue</type>
      <promotion>
        <external_id>welcome_promo</external_id>
        <name>Welcome Gift</name>
      </promotion>
    </requested_promotions>
  </data>
</xml>
```

Response Example

JSON


```
{
  "data": {
    "status": "success",
    "processing_target": "immediate",
    "external_id": "order_12345",
    "issued_promocodes": [
      {
        "codes": ["HYDO-4728-DZGU"],
        "pool": {
          "pool_id": 100,
          "name": "Welcome Promocodes"
        },
        "promotion": {
          "external_id": "welcome_promo",
          "name": "Welcome Gift"
        }
      }
    ],
    "points_changes": [
      {
        "points_info": {
          "points_currency_iid": 1,
          "points_currency_eid": "bonus1",
          "name": "Bonus points",
          "amount": "21500"
        },
        "accrue_points": 75,
        "redeem_points": 150
      }
    ],
    "contents": [
      {
        "promotion": {
          "external_id": "message_promo",
          "name": "Information Message"
        },
        "contents": [
          {
            "type": "text",
            "message": "Thank you for your purchase! You have received 75 bonus points."
          }
        ]
      }
    ]
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
    <status>success</status>
    <processing_target>immediate</processing_target>
    <external_id>order_12345</external_id>
    <issued_promocodes>
      <codes>HYDO-4728-DZGU</codes>
      <pool>
        <pool_id>100</pool_id>
        <name>Welcome Promocodes</name>
      </pool>
      <promotion>
        <external_id>welcome_promo</external_id>
        <name>Welcome Gift</name>
      </promotion>
    </issued_promocodes>
    <points_changes>
      <points_info>
        <points_currency_iid>1</points_currency_iid>
        <points_currency_eid>bonus1</points_currency_eid>
        <name>Bonus points</name>
        <amount>21500</amount>
      </points_info>
      <accrue_points>75</accrue_points>
      <redeem_points>150</redeem_points>
    </points_changes>
    <contents>
      <promotion>
        <external_id>message_promo</external_id>
        <name>Information Message</name>
      </promotion>
      <contents>
        <type>text</type>
        <message>Thank you for your purchase! You have received 75 bonus points.</message>
      </contents>
    </contents>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Description
data	**object**	**Confirmation result data**
error	**int**	**Error code (0 on success)**
error_text	**string**	**Error description**

Parameters returned in the `data` object:

Parameter	Type	Description
status	**string**	**Processing status:** • `success` — **order successfully created** • `outdated` — **price has changed since the preliminary calculation (/preorder must be repeated)**
processing_target	**string**	**Only for `success` status. Value: `immediate` or `processed`**
external_id	**string**	**External order ID (from the request)**
issued_promocodes	**array**	**Array of issued promo codes (for `code_issue` type promotions)**
points_changes	**array**	**Array of point balance changes by point currency types**
contents	**array**	**Array of content for display (text messages from promotions)**

The `issued_promocodes` array consists of objects with information about applied promo codes:

Parameter	Type	Description
codes	**array**	**Array of issued promo codes**
pool	**object**	**Information about the promo code pool**
pool.pool_id	**integer**	**Pool identifier**
pool.name	**string**	**Pool name**
promotion	**object**	**Object with promotion information**
promotion.external_id	**string**	**External promotion identifier**
promotion.name	**string**	**Promotion name**

The `points_changes` array consists of objects with information about point balance changes:

Parameter	Type	
points_info	**object**	**Object with point type information**
points_info.points_currency_iid	**integer**	**Internal identifier of the point type**
points_info.points_currency_eid	**string**	**External identifier of the point type**
points_info.name	**string**	**Name of the point type**
points_info.amount	**string**	**Current balance after the operation**

Parameter	Type	
accrue_points	**integer**	**Number of points accrued**
redeem_points	**integer**	**Number of points redeemed**

Roll back temporary transaction

Description

Rolls back a temporary transaction created with the `is_temp` flag. After rollback, the active points bala

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/rollback_transaction`

Request parameters

Parameter	Type	Example	Required	
transaction_id	**string**	**“TRANS12345”**	**No**	**Temporary transaction ID**
transaction_eid	**string**	**“TRANS12345”**	**No**	**External temporary transaction ID**

At least one of the identifiers must be provided (`transaction_id` or `transaction_eid`).

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Rolled-back transaction ID**
balance	**string**	**Current active points balance of the member after rollback**

Cancel points transaction

Description

Cancels any points transaction — temporary or permanent. When canceled, the current active points b

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/cancel_transaction`

Request parameters

Parameter	Type	Example	Required	
transaction_id	**string**	**“TRANS12345”**	**No**	**Transaction ID**
transaction_eid	**string**	**“TRANS12345”**	**No**	**External transa**

At least one identifier must be provided.

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Canceled transaction ID**
balance	**string**	**Active points balance after cancellation**

Get points account balance

Description

Returns the current points account balance of a loyalty program member: the total number of points a the selected points currency.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/balance`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
points_currency_iid	**int**	**1**	**No**	**Internal points currency ID**
points_currency_eid	**string**	**“starts”**	**No**	**External points currency ID (for**
matching	**string**	**“email”**	**Yes**	**Profile lookup method: `email`, `p`**
field_name	**string**	**“user_id”**	**Yes, if `matching = custom`**	**Custom field name for lookup**
field_value	**string / int**	**123456**	**Yes, if `matching = custom`**	**Custom field value**
email	**string**	**“user@example.com”**	**Yes, if `matching = email`**	**Member email**
phone	**string**	**“+1234567890”**	**Yes, if `matching = phone`**	**Member phone number in intern**
profile_id	**string**	**“profile123”**	**Yes, if `matching = profile_id`**	**Profile ID**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "points_currency_eid": "starts",
  "matching": "email",
  "email": "user@example.com"
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>starts</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "total_balance": "1000.00",
    "active_balance": "800.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <total_balance>1000.00</total_balance>
    <active_balance>800.50</active_balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
total_balance	**string**	**Total number of points in the points account (string format)**
active_balance	**string**	**Number of active points in the points account (string format)**

Register member in a loyalty program

Description

**Registers a member in a loyalty program. Before registration, the profile is searched in the database u
(email, phone number, profile ID, or a custom field).**

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
matching	**string**	**“email”**	**Yes**	**Profile lookup method:** `email` , `phone` , `profile_id`
registration_date	**string**	**“2024-09-01T12:34:56Z”**	**No**	**Registration date in RFC3339 format; if omitte**
email	**string**	**“user@example.com”**	**Yes, if** `matching = email`	**Member email used to find the profile**
phone	**string**	**“+1234567890”**	**Yes, if** `matching = phone`	**Member phone number in international format**
profile_id	**string**	**“profile123”**	**Yes, if** `matching = profile_id`	**Profile ID**
field_name	**string**	**“user_id”**	**Yes, if** `matching = custom`	**Custom field name for lookup**
field_value	**string / int**	**“123456”**	**Yes, if** `matching = custom`	**Custom field value**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "matching": "email",
  "email": "user@example.com",
  "registration_date": "2024-09-01T12:34:56Z"
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <email>user@example.com</email>
  <registration_date>2024-09-01T12:34:56Z</registration_date>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "123456"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>123456</profile_id>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**ID of the profile registered in the loyalty program**

Batch adding participants to loyalty program

Description

Adds multiple participants to a loyalty program in a single request. Before adding, each profile is looke method: email, phone, profile ID, or custom field.

> **CAUTION**
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible to transfer more data using multiple streams.**

URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member_batch`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program identifier**
matching	**string**	**“email”**	**Yes**	**Profile lookup method:** `email` , `phone` , `prof`
data	**array**	**[{“email”: “user@example.com”}]**	**Yes**	**Array of participants to add**

`data` array

Parameter	Type	Example	Required	
email	**string**	**“user@example.com”**	**Yes, if** `matching = email`	**Email for pro**
phone	**string**	**“+1234567890”**	**Yes, if** `matching = phone`	**Phone numb**
profile_id	**string**	**“profile123”**	**Yes, if** `matching = profile_id`	**Profile ident**
field_name	**string**	**“user_id”**	**Yes, if** `matching = custom`	**Custom field**
field_value	**string / int**	**“123456”**	**Yes, if** `matching = custom`	**Custom field**
registration_date	**string**	**“2024-09-01T12:34:56Z”**	**No**	**Registration**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "matching": "email",
  "data": [
    {
      "email": "user1@example.com",
      "registration_date": "2024-09-01T12:34:56Z"
    },
    {
      "email": "user2@example.com"
    }
  ]
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <data>
    <item>
      <email>user1@example.com</email>
      <registration_date>2024-09-01T12:34:56Z</registration_date>
    </item>
    <item>
      <email>user2@example.com</email>
    </item>
  </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "663a138cee44bb50a67115cd"
    },
    {
      "error": 409,
      "error_text": "Member already exists",
      "profile_id": "69df4fd5ff232800e2d1e2d6"
    },
    {
      "error": 404,
      "error_text": "Profile not found",
      "profile_id": null
    }
  ]
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <item>
      <error>0</error>
      <error_text>Successful operation</error_text>
      <profile_id>663a138cee44bb50a67115cd</profile_id>
    </item>
    <item>
      <error>409</error>
      <error_text>Member already exists</error_text>
      <profile_id>69df4fd5ff232800e2d1e2d6</profile_id>
    </item>
    <item>
      <error>404</error>
      <error_text>Profile not found</error_text>
    </item>
  </result>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code (0 — successful operation)**
error_text	**string**	**Error description**
result	**array**	**Array of results for each participant**

`result` element parameters

Parameter	Type	Description
error	**int**	**Error code for the specific participant**
error_text	**string**	**Error description for the specific participant**
profile_id	**string**	**Profile identifier (null on error)**

Error codes in `result`

Code	Description
0	**Successfully added**
404	**Profile not found**
409	**Member already exists in the loyalty program**

Task for batch add participants to loyalty program

Description

Starts an asynchronous task for bulk adding participants to a loyalty program. Returns a `task_id` that c retrieve results.

> **CAUTION**
>
> **A single request should not:**
>
> - **Exceed 10,000 profiles;**
> - **Exceed 10 MB of data. Larger volumes can be sent in multiple streams.**

URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member_batch_task`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program**
matching	**string**	**“email”**	**Yes**	**Profile lookup me**
data	**array**	**[{“email”: “user@example.com”}]**	**Yes**	**Array of particip**
task_id	**string**	**“199b9c75-c540-455e-9e38-3d54e51e1d7a”**	**No (required for subsequent batches)**	**Existing task ide**
next_key	**string**	**“abc123”**	**No**	**Pagination key fo**

`data` array

Parameter	Type	Example	Required	
email	**string**	**“user@example.com”**	**Yes, if `matching = email`**	**Email for pro**
phone	**string**	**“+1234567890”**	**Yes, if `matching = phone`**	**Phone numb**
profile_id	**string**	**“profile123”**	**Yes, if `matching = profile_id`**	**Profile ident**
field_name	**string**	**“user_id”**	**Yes, if `matching = custom`**	**Custom field**
field_value	**string / int**	**“123456”**	**Yes, if `matching = custom`**	**Custom field**
registration_date	**string**	**“2024-09-01T12:34:56Z”**	**No**	**Registration**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "matching": "email",
  "data": [
    {
      "email": "user1@example.com",
      "registration_date": "2024-09-01T12:34:56Z"
    },
    {
      "email": "user2@example.com"
    }
  ]
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <data>
    <item>
      <email>user1@example.com</email>
      <registration_date>2024-09-01T12:34:56Z</registration_date>
    </item>
    <item>
      <email>user2@example.com</email>
    </item>
  </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "task_id": "199b9c75-c540-455e-9e38-3d54e51e1d7a"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <task_id>199b9c75-c540-455e-9e38-3d54e51e1d7a</task_id>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code (0 — successful operation)**
error_text	**string**	**Error description**
task_id	**string**	**Asynchronous task identifier**

Remove member from loyalty program

Description

Removes a member from a loyalty program. Before removal, the profile is searched using the selected number, profile ID, or a custom field.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/delete_member`

Request parameters

Parameter	Type	Example	Required	
loyalty_program_id	**int**	**123**	**Yes**	**Loyalty program ID**
matching	**string**	**“email”**	**Yes**	**Profile lookup method:** `email` , `ph`
email	**string**	**“user@example.com”**	**Yes, if** `matching = email`	**Email used to find the profile**
phone	**string**	**“+1234567890”**	**Yes, if** `matching = phone`	**Phone number in international fo**
profile_id	**string**	**“profile123”**	**Yes, if** `matching = profile_id`	**Profile ID**
field_name	**string**	**“user_id”**	**Yes, if** `matching = custom`	**Custom field name**
field_value	**string / int**	**“123456”**	**Yes, if** `matching = custom`	**Custom field value**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "matching": "email",
  "email": "user@example.com"
}
```

XML

```
<xml>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <email>user@example.com</email>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "123456"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>123456</profile_id>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**ID of the removed profile**

Get form information

Description

Allows you to retrieve information about a form: field structure, page settings, transition logic, styling,

URL

Method: POST

`https://example.com/api/v1.1/forms/get`

Request Parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes, if `shortname` is not specified**	**Form ID.**
shortname	**string**	**“form_6Ven2v7O”**	**Yes, if `id` is not specified**	**Form short name.**
form	**boolean**	**true**	**No**	**Whether to export form structure info. De**
actions	**boolean**	**true**	**No**	**Whether to export configured actions info**

Request Example

JSON

```
{
  "id": 1,
  "form": true,
  "actions": true
}
```

XML

```
<xml>
  <id>1</id>
  <form>true</form>
  <actions>true</actions>
</xml>
```

Response Example

JSON

```
{
    "data": {
        "id": 1,
        "groups": [
            {"id": 2, "name": "example_group_1"},
            {"id": 3, "name": "example_group_2"}
        ],
        "is_delete": false,
        "createtime": "2026-02-20T10:43:51.91Z",
        "updatetime": "2026-04-23T11:00:46.242Z",
        "name": "Example Form",
        "shortname": "form_6Ven2v7O",
        "descr": "",
        "type": "basic",
        "ui_tags": [],
        "is_nps": false,
        "is_published": true,
        "published_time": "2026-04-23T11:00:46.234Z",
        "is_temporary": false,
        "tz": "",
        "actions": [
            {
                "action_id": "NrqKjgfX",
                "type": "profile_import",
                "profile_import": {
                    "db_id": 3,
                    "matching_mode": "email",
                    "mode_add": true,
                    "mode_update": true,
                    "is_update_utm": true,
                    "match_mapping": [
                        {
                            "ext_name": "field_4",
                            "ext_num": 0,
                            "int_name": "email",
                            "ignore_empty": true,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        }
                    ],
                    "column_mapping": [
                        {
                            "ext_name": "field_1",
                            "ext_num": 0,
                            "int_name": "_fname",
                            "ignore_empty": false,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        },
                        {
                            "ext_name": "field_4",
                            "ext_num": 0,
                            "int_name": "phones",
                            "ignore_empty": false,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        }
                    ],
                    "auto_subscribe": []
                }
            }
        ],
        "form": {
            "form_title": "Example Form",
            "last_page_id": 2,
            "language": "EN",
            "basic_form_pages": [
                {
                    "id": 1,
                    "title": "Page 1",
                    "shortname": "page_1",
                    "fields": [
                        {
                            "type": "line",
```

```
                "name": "field_1",
                "comment": "",
                "hidden": false,
                "title": "Name",
                "index_custom_option": -1,
                "condition": {
                    "condition": "",
                    "value": null
                },
                "count_range": 5,
                "min_range": 0,
                "max_range": 20,
                "options": [],
                "default_value": "",
                "validation_rules": [],
                "right_answer": "",
                "right_answer_score": 0,
                "nps_field": null
            },
            {
                "type": "email",
                "name": "field_4",
                "comment": "",
                "hidden": false,
                "title": "Email",
                "index_custom_option": -1,
                "condition": {
                    "condition": "",
                    "value": null
                },
                "count_range": 5,
                "min_range": 0,
                "max_range": 20,
                "options": [],
                "default_value": "",
                "validation_rules": [
                    {
                        "type": "email",
                        "message": ""
                    }
                ],
                "right_answer": "",
                "right_answer_score": 0,
                "nps_field": null
            },
            {
                "type": "rating_stars",
                "name": "field_5",
                "comment": "",
                "hidden": false,
                "title": "Rating",
                "index_custom_option": -1,
                "condition": {
                    "condition": "",
                    "value": null
                },
                "count_range": 5,
                "min_range": 0,
                "max_range": 20,
                "options": [],
                "default_value": 0,
                "validation_rules": [],
                "right_answer": "",
                "right_answer_score": 0,
                "nps_field": null
            }
        ]
    },
    {
        "id": 2,
        "title": "Page 2",
        "shortname": "page_2",
        "fields": [
            {
                "type": "line",
                "name": "field_6",
                "comment": "",
                "hidden": false,
                "title": "Additional Question",
                "index_custom_option": -1,
```

```
                "condition": {
                    "condition": "",
                    "value": null
                },
                "count_range": 5,
                "min_range": 0,
                "max_range": 20,
                "options": [],
                "default_value": "",
                "validation_rules": [],
                "right_answer": "",
                "right_answer_score": 0,
                "nps_field": null
            },
            {
                "type": "checkbox",
                "name": "field_8",
                "comment": "",
                "hidden": false,
                "title": "Checkbox",
                "index_custom_option": -1,
                "condition": {
                    "condition": "",
                    "value": null
                },
                "count_range": 5,
                "min_range": 0,
                "max_range": 20,
                "options": [],
                "default_value": "",
                "validation_rules": [],
                "right_answer": "",
                "right_answer_score": 0,
                "nps_field": null
            }
        ]
    }
],
"final_page": {
    "title": "Thank you!",
    "message": "Your form has been submitted successfully.",
    "display_score": false,
    "required_score": 0,
    "redirect_pass_url": "",
    "redirect_fail_url": ""
},
"style": {
    "preset": "default",
    "background_color": "#6590e7",
    "background_image": "",
    "buttons_color": "#000000",
    "buttons_background_color": "#3678E2",
    "font": "Fira Sans",
    "text_align": "initial",
    "font_color": "#0D0F12",
    "font_size": 18,
    "border_radius": 12,
    "rating_stars_color": "#ffea78",
    "rating_stars_background_color": "#f1f3f9",
    "rate_star": {
        "figure_selected_color": "",
        "figure_unselected_color": "",
        "figure_size": 0
    },
    "rate_heart": {
        "figure_selected_color": "",
        "figure_unselected_color": "",
        "figure_size": 0
    },
    "rate_emoji": {
        "figure_selected_color": "",
        "figure_unselected_color": "",
        "figure_size": 0
    },
    "rate_number": {
        "buttons_selected_color": "",
        "buttons_unselected_color": "",
        "buttons_selected_background_color": "",
        "buttons_unselected_background_color": "",
        "number_size": 0
```

```
    }
},
"next_button": {
    "type": "next",
    "size": 14,
    "icon": "",
    "title": "Next Page"
},
"prev_button": {
    "type": "prev",
    "size": 14,
    "icon": "",
    "title": "Previous Page"
},
"send_button": {
    "type": "send",
    "size": 14,
    "icon": "paper plane",
    "title": "Send"
},
"final_button": {
    "type": "final",
    "size": 14,
    "icon": "",
    "title": "Go to the website"
},
"time_limit_sec": 900,
"form_check_sum": "f742b0a68fb3fa17c3d845487ed4b5ed3f6405b7",
"logic": [
    {
        "id": 1,
        "title": "",
        "page_id": 1,
        "next_node": 2,
        "type": "page",
        "condition": []
    },
    {
        "id": 2,
        "title": "",
        "page_id": 2,
        "type": "page",
        "condition": []
    }
],
"start_logic_block_id": 1,
"last_form_logic_block_id": 2,
"is_logic_enabled": true,
"flowchart_elements": [
    {
        "data": {
            "idOfLogicBlock": 1,
            "idOfPage": 1,
            "isFirstPage": true,
            "label": "Page 1"
        },
        "id": "jANe4JVI",
        "position": {
            "x": 18,
            "y": 54
        },
        "type": "page"
    },
    {
        "data": {
            "idOfLogicBlock": 2,
            "idOfPage": 2,
            "isFirstPage": false,
            "label": "Page 2"
        },
        "id": "sBFQLU8K",
        "position": {
            "x": 26,
            "y": 184
        },
        "type": "page"
    },
    {
        "id": "EmaWmrar",
        "source": "jANe4JVI",
```

```
                    "sourceHandle": null,
                    "target": "sBFQLU8K",
                    "targetHandle": null,
                    "type": "edge"
                }
            ],
            "is_nps": false,
            "restrict_refill": true,
            "resubmit_page": {
                "title": "Restriction!",
                "message": "You have already filled out this form.",
                "display_score": false,
                "required_score": 0,
                "redirect_pass_url": "",
                "redirect_fail_url": ""
            },
            "expired_days": 1
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <id>1</id>
        <groups>
            <id>2</id>
            <name>example_group_1</name>
        </groups>
        <groups>
            <id>3</id>
            <name>example_group_2</name>
        </groups>
        <is_delete>false</is_delete>
        <createtime>2026-02-20T10:43:51.91Z</createtime>
        <updatetime>2026-04-23T11:00:46.242Z</updatetime>
        <name>Example Form</name>
        <shortname>form_6Ven2v7O</shortname>
        <type>basic</type>
        <is_nps>false</is_nps>
        <is_published>true</is_published>
        <published_time>2026-04-23T11:00:46.234Z</published_time>
        <is_temporary>false</is_temporary>
        <actions>
            <action_id>NrqKjgfX</action_id>
            <type>profile_import</type>
            <profile_import>
                <db_id>3</db_id>
                <matching_mode>email</matching_mode>
                <mode_add>true</mode_add>
                <mode_update>true</mode_update>
                <is_update_utm>true</is_update_utm>
                <match_mapping>
                    <ext_name>field_4</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>email</int_name>
                    <ignore_empty>true</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </match_mapping>
                <column_mapping>
                    <ext_name>field_1</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>_fname</int_name>
                    <ignore_empty>false</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </column_mapping>
                <column_mapping>
                    <ext_name>field_4</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>phones</int_name>
                    <ignore_empty>false</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </column_mapping>
            </profile_import>
        </actions>
        <form>
            <form_title>Example Form</form_title>
            <last_page_id>2</last_page_id>
            <language>EN</language>
            <basic_form_pages>
                <id>1</id>
                <title>Page 1</title>
                <shortname>page_1</shortname>
                <fields>
                    <type>line</type>
                    <name>field_1</name>
                    <hidden>false</hidden>
                    <title>Name</title>
                    <index_custom_option>-1</index_custom_option>
                    <condition>
                    </condition>
                    <count_range>5</count_range>
                    <min_range>0</min_range>
                    <max_range>20</max_range>
                    <right_answer_score>0</right_answer_score>
                </fields>
                <fields>
                    <type>email</type>
                    <name>field_4</name>
```

```
            <hidden>false</hidden>
            <title>Email</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <validation_rules>
                <type>email</type>
            </validation_rules>
            <right_answer_score>0</right_answer_score>
        </fields>
        <fields>
            <type>rating_stars</type>
            <name>field_5</name>
            <hidden>false</hidden>
            <title>Rating</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <default_value>0</default_value>
            <right_answer_score>0</right_answer_score>
        </fields>
    </basic_form_pages>
    <basic_form_pages>
        <id>2</id>
        <title>Page 2</title>
        <shortname>page_2</shortname>
        <fields>
            <type>line</type>
            <name>field_6</name>
            <hidden>false</hidden>
            <title>Additional Question</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <right_answer_score>0</right_answer_score>
        </fields>
        <fields>
            <type>checkbox</type>
            <name>field_8</name>
            <hidden>false</hidden>
            <title>Checkbox</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <right_answer_score>0</right_answer_score>
        </fields>
    </basic_form_pages>
    <final_page>
        <title>Thank you!</title>
        <message>Your form has been submitted successfully.</message>
        <display_score>false</display_score>
        <required_score>0</required_score>
        <final_redirect>false</final_redirect>
    </final_page>
    <style>
        <preset>default</preset>
        <background_color>#6590e7</background_color>
        <buttons_color>#000000</buttons_color>
        <buttons_background_color>#3678E2</buttons_background_color>
        <font>Fira Sans</font>
        <text_align>initial</text_align>
        <font_color>#0D0F12</font_color>
        <font_size>18</font_size>
        <border_radius>12</border_radius>
        <rating_stars_color>#ffea78</rating_stars_color>
        <rating_stars_background_color>#f1f3f9</rating_stars_background_color>
        <rate_star>
            <figure_size>0</figure_size>
```

```
        </rate_star>
        <rate_heart>
            <figure_size>0</figure_size>
        </rate_heart>
        <rate_emoji>
            <figure_size>0</figure_size>
        </rate_emoji>
        <rate_number>
            <number_size>0</number_size>
        </rate_number>
    </style>
    <next_button>
        <type>next</type>
        <size>14</size>
        <title>Next Page</title>
    </next_button>
    <prev_button>
        <type>prev</type>
        <size>14</size>
        <title>Previous Page</title>
    </prev_button>
    <send_button>
        <type>send</type>
        <size>14</size>
        <icon>paper plane</icon>
        <title>Send</title>
    </send_button>
    <final_button>
        <type>final</type>
        <size>14</size>
        <title>Go to the website</title>
    </final_button>
    <time_limit_sec>900</time_limit_sec>
    <form_check_sum>f742b0a68fb3fa17c3d845487ed4b5ed3f6405b7</form_check_sum>
    <logic>
        <id>1</id>
        <page_id>1</page_id>
        <next_node>2</next_node>
        <type>page</type>
    </logic>
    <logic>
        <id>2</id>
        <page_id>2</page_id>
        <type>page</type>
    </logic>
    <start_logic_block_id>1</start_logic_block_id>
    <last_form_logic_block_id>2</last_form_logic_block_id>
    <is_logic_enabled>true</is_logic_enabled>
    <flowchart_elements>
        <data>
            <idOfLogicBlock>1</idOfLogicBlock>
            <idOfPage>1</idOfPage>
            <isFirstPage>true</isFirstPage>
            <label>Page 1</label>
        </data>
        <id>jANe4JVI</id>
        <position>
            <x>18</x>
            <y>54</y>
        </position>
        <type>page</type>
    </flowchart_elements>
    <flowchart_elements>
        <data>
            <idOfLogicBlock>2</idOfLogicBlock>
            <idOfPage>2</idOfPage>
            <isFirstPage>false</isFirstPage>
            <label>Page 2</label>
        </data>
        <id>sBFQLU8K</id>
        <position>
            <x>26</x>
            <y>184</y>
        </position>
        <type>page</type>
    </flowchart_elements>
    <flowchart_elements>
        <id>EmaWmrar</id>
        <source>jANe4JVI</source>
        <sourceHandle>null</sourceHandle>
```

```
                <target>sBFQLU8K</target>
                <targetHandle>null</targetHandle>
                <type>edge</type>
            </flowchart_elements>
            <is_nps>false</is_nps>
            <restrict_refill>true</restrict_refill>
            <resubmit_page>
                <title>Restriction!</title>
                <message>You have already filled out this form.</message>
                <display_score>false</display_score>
                <required_score>0</required_score>
            </resubmit_page>
            <expired_days>1</expired_days>
        </form>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Description
error	**int**	**Error code.**
error_text	**string**	**Error text.**
data	**object**	**Form information. See data.**

`data` Object

Parameter	Type	Description
id	**int**	**System form ID.**
groups	**array[object]**	**List of groups the form belongs to. Each element has `id` (int) and `name` (string).**
is_delete	**boolean**	**Deletion flag.**
is_published	**boolean**	**Publication flag.**
name	**string**	**Form name.**
descr	**string**	**Form description.**
createtime	**string**	**Creation date and time.**
updatetime	**string**	**Last update date and time.**
published_time	**string**	**Publication date and time.**
shortname	**string**	**Internal form ID.**
tz	**string**	**Time zone.**
ui_tags	**array[string]**	**Array of user tags.**
is_nps	**boolean**	**NPS survey flag.**
actions	**array**	**Array of actions (present if actions=true in request). Each element is an action object.**
form	**object**	**Form settings object (present if form=true in request). See form.**
type	**string**	**Form type. Not used, always “basic”.**
is_temporary	**boolean**	**Unused parameter (always false).**

`action` Object (element of actions array)

Parameter	Type	Description
action_id	**string**	**Unique action ID.**
type	**string**	**Action type (e.g., “profile_import”).**
profile_import	**object**	**Profile import settings. See profile_import.**

`profile_import` Object

Parameter	Type	Description
db_id	**int**	**Database ID.**
matching_mode	**string**	**Matching mode (e.g., “email”). For more details about matching, see this article.**
mode_add	**boolean**	**Allow adding new profiles.**
mode_update	**boolean**	**Allow updating existing profiles.**
is_update_utm	**boolean**	**Update UTM tags.**
skip_triggers	**boolean**	**Skip triggers.**
use_geo	**boolean**	**Use geodata.**
auto_subscribe	**array**	**Auto-subscription settings array.**
match_mapping	**array**	**Record matching mapping array. Each element is a mapping object.**
column_mapping	**array**	**Form field to database field mapping array. Each element is a mapping object.**

`mapping` Object (element of match_mapping and column_mapping arrays)

Parameter	Type	Description
ext_name	**string**	**Form field name.**
int_name	**string**	**Field name in the target system.**
ignore_empty	**boolean**	**Ignore empty values.**
are_nulls_empty	**boolean**	**Whether to treat NULL as empty.**
action	**string**	**Action (usually “set”).**
submatch	**string**	**Substitution expression. Used for custom fields in the profile database of type “Object”.**
ext_num	**int**	**External field number. Not used (default is “0”).**
normalizers	**array**	**Normalizers array. Not used, always an empty array.**

`form` Object

Parameter	Type	Description
form_title	**string**	**Form title.**
language	**string**	**Language code (e.g., “EN”).**
is_nps	**boolean**	**NPS survey flag (duplicated at the top level).**
restrict_refill	**boolean**	**Prevent repeat filling.**
last_page_id	**int**	**Last page ID.**

Parameter	Type	Description
basic_form_pages	**array**	**Array of form pages. Each page is described in page.**
final_page	**object**	**Final page. See final_page.**
resubmit_page	**object**	**Page shown when attempting to refill. See resubmit_page.**
style	**object**	**Style settings. See style.**
next_button	**object**	**“Next” button. See button.**
prev_button	**object**	**“Previous” button. See button.**
send_button	**object**	**Submit button. See button.**
final_button	**object**	**“Go to website” button. See button.**
time_limit_sec	**int**	**Fill time limit in seconds.**
form_check_sum	**string**	**Form checksum.**
logic	**array**	**Page transition logic. See logic.**
start_logic_block_id	**int**	**Starting logic block ID.**
last_form_logic_block_id	**int**	**Last logic block ID.**
is_logic_enabled	**boolean**	**Whether transition logic is enabled.**
flowchart_elements	**array**	**Flowchart elements (for the visual editor).**
expired_days	**int**	**Days until the form becomes unavailable (1 — after one day).**
start_date	**string**	**Form activation start date and time (UTC). If not set — absent.**
end_date	**string**	**Form activation end date and time (UTC). After this, the form stops accepting responses.**

`page` Object (element of basic_form_pages array)

Parameter	Type	Description
id	**int**	**Page ID.**
title	**string**	**Page title.**
shortname	**string**	**Page short name.**
fields	**array**	**Array of fields on the page. Each field is described in field.**

`field` Object (element of fields array)

Parameter	Type	Description
type	**string**	**Field type (line, email, rating_stars, checkbox, etc.).**
name	**string**	**System field name.**
title	**string**	**Displayed field title.**
comment	**string**	**Field comment.**
hidden	**boolean**	**Hidden field.**
default_value	**string / number**	**Default value.**
validation_rules	**array**	**Validation rules (e.g., `{"type":"email","message":""}`).**
condition	**object**	**Field display condition (usually empty).**
options	**array**	**Options for dropdowns, radio buttons, etc.**
right_answer	**string**	**Correct answer (for quizzes).**
right_answer_score	**number**	**Score for correct answer.**
nps_field	**string/null**	**Link to NPS field.**

`final_page` Object

Parameter	Type	Descriptic
title	**string**	**Final page title.**
message	**string**	**Successful submission message.**
display_score	**boolean**	**Whether to display the score.**
required_score	**int**	**Minimum score to pass.**
redirect_pass_url	**string**	**Redirect URL on success.**
redirect_fail_url	**string**	**Redirect URL on failure.**

`resubmit_page` Object

`resubmit_page` has a similar structure (title “Restriction!”, message about already filled).

`button` Object (for next_button, prev_button, send_button, final_button)

Parameter	Type	Description
type	**string**	**Button type (next, prev, send, final).**
size	**int**	**Font size (px).**
icon	**string**	**Icon name (e.g., “paper plane”).**
title	**string**	**Button text.**

`style` Object

Parameter	Type	
preset	**string**	**Preset style name (“default”).**
background_color	**string**	**Background color (HEX).**
background_image	**string**	**Background image URL.**
font	**string**	**Font name.**
font_color	**string**	**Font color.**
font_size	**int**	**Font size (px).**
buttons_color	**string**	**Button text color.**
buttons_background_color	**string**	**Button background color.**
border_radius	**int**	**Border radius (px).**
text_align	**string**	**Text alignment (initial, center, etc.).**
rating_stars_color	**string**	**Rating stars color.**
rating_stars_background_color	**string**	**Rating stars background color.**
rate_star, rate_heart, rate_emoji, rate_number	**object**	**Additional settings for different rating types.**

`logic` Object (element of logic array)

Parameter	Type	Description
id	**int**	**Logic block ID.**
title	**string**	**Title (may be empty).**
page_id	**int**	**ID of the page this block belongs to.**
next_node	**int**	**ID of the next node (only if there is a transition).**
next_node_true	**int**	**ID of the next node when condition is met (for type condition).**
next_node_false	**int**	**ID of the next node when condition is not met (for type condition).**
condition_type	**string**	**Logical operator for a group of conditions (and / or, for type condition).**
type	**string**	**Block type (page or condition).**
condition	**array**	**Transition conditions.**

For more details about form logic, see the article Conditional Page-by-Page Logic in Forms and Surveys

Get form list

Description

The method returns a list of forms with the ability to filter by tags, groups, publication status, and NPS

URL

Method: POST

`https://example.com/api/v1.1/forms/list`

Request Parameters

Parameter	Type	Example	Required	
from_id	**int**	**1**	**No**	**ID of the first form in the res**
limit	**int**	**50**	**No**	**Limit on the number of retur**
ui_tags	**array[string]**	**[“main”, “feedback”]**	**No**	**List of tags to search**
groups	**array[string]**	**[“clients”, “partners”]**	**No**	**List of groups to search**
groups_id	**array[int]**	**[5, 12]**	**No**	**List of group IDs to search**
is_nps	**boolean**	**true**	**No**	**Filter by NPS forms**
is_published	**boolean**	**true**	**No**	**Filter by published forms**

Request Example

JSON

```
{
    "id": 1,
    "limit": 1,
    "is_nps": true
}
```

XML

```
<xml>
  <id>1</id>
  <limit>2</limit>
  <is_nps>true</is_nps>
</xml>
```

Response Example

JSON

```
{
  "data": [
    {
      "id": 1,
      "groups": [
        {"id": 2, "name": "example_group_1"},
        {"id": 3, "name": "example_group_2"}
      ],
      "is_delete": false,
      "createtime": "2026-02-20T10:43:51.91Z",
      "updatetime": "2026-03-26T13:19:23.961Z",
      "name": "Customer Survey",
      "shortname": "form_6Ven2v7O",
      "descr": "",
      "type": "basic",
      "ui_tags": [],
      "is_nps": false,
      "is_published": false,
      "is_temporary": false
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "total_count": 1
}
```

XML

```
<xml>
  <data>
    <id>1</id>
    <groups>
      <id>2</id>
      <name>example_group_1</name>
    </groups>
    <groups>
      <id>3</id>
      <name>example_group_2</name>
    </groups>
    <is_delete>false</is_delete>
    <createtime>2026-02-20T10:43:51.91Z</createtime>
    <updatetime>2026-03-26T13:19:23.961Z</updatetime>
    <name>Customer Survey</name>
    <shortname>form_6Ven2v7O</shortname>
    <descr></descr>
    <type>basic</type>
    <is_nps>false</is_nps>
    <is_published>false</is_published>
    <is_temporary>false</is_temporary>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>1</total_count>
</xml>
```


Response Parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
total_count	**int**	**Total number of forms**
data	**array[object]**	**Array of form objects**

`data` Array

Parameter	Type	Description
id	**int**	**System form ID**
groups	**array[object]**	**List of groups. Each element has `id` (int) and `name` (string)**
is_delete	**boolean**	**Deletion flag**
createtime	**string (date-time)**	**Creation date and time**
updatetime	**string (date-time)**	**Update date and time**
name	**string**	**Form name**
shortname	**string**	**Form short name**
descr	**string**	**Form description**
type	**string**	**Form type**
ui_tags	**array[string]**	**List of UI tags**
is_nps	**boolean**	**NPS form flag**
is_published	**boolean**	**Publication status**
is_temporary	**boolean**	**Temporary form flag**

Export form fill data by user

Description

**Creates a task to export form fill data submitted by a specific user. The user is identified via matching
The method returns a `task_id`. To get the result, you need to check the task status and download the r**

Request URL

Method: `POST`

`https://example.com/api/v1.1/forms/fill`

Request Parameters

Parameter	Type	Example	Required	De
id	**int**	**1**	**Yes, if `shortname` is not specified**	**Form ID**
shortname	**string**	**“form_6Ven2v7O”**	**Yes, if `id` is not specified**	**Form short name**
db_id	**int**	**3**	**Yes**	**Database ID for matching**
file_format	**string**	**“csv”**	**No**	**File format: `csv` or `json`**
matching	**string**	**“email”**	**No**	**Matching mode. If not specified, determined automati**
profile_id	**string**	**“507f1f77bcf86cd799439011”**	**No**	**Profile ID**
email	**string**	**“user@example.com”**	**No**	**User email**
phone	**string**	**“+79001234567”**	**No**	**User phone**
field_name	**string**	**“custom_field”**	**No**	**Custom field name for matching**
field_value	**string**	**“value”**	**No**	**Custom field value**

INFO

At least one matching field (`profile_id`, `email`, `phone`, `field_name` + `field_value`) must be specified.

Matching Modes

Value	Description
`email`	**Match by email**
`phone`	**Match by phone**
`profile_id`	**Match by profile ID**
`custom`	**Match by custom field (`field_name` + `field_value`)**

Request Example

JSON

```
{
  "id": 1,
  "db_id": 3,
  "email": "user@example.com",
  "matching": "email",
  "file_format": "csv"
}
```

XML

```
<xml>
  <id>1</id>
  <db_id>3</db_id>
  <email>user@example.com</email>
  <matching>email</matching>
  <file_format>csv</file_format>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "task_id": "550e8400-e29b-41d4-a716-446655440000"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>550e8400-e29b-41d4-a716-446655440000</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Descripti
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**object**	**Response data object**

`data` **Object**

Parameter	Type	Description
task_id	**string**	**Async task ID for retrieving fill data**

Step 2. Checking task status and getting download link

Request URL

Method: `POST`

`https://example.com/api/v1.1/tasks/status`

Request Parameters

Parameter	Type	Example	Required
task_id	**string**	**“550e8400-e29b-41d4-a716-446655440000”**	**Yes**

Request Example

JSON

```
{
  "task_id": "550e8400-e29b-41d4-a716-446655440000"
}
```

XML

```
<xml>
  <task_id>550e8400-e29b-41d4-a716-446655440000</task_id>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "status": "completed",
    "progress": 100,
    "result": {
      "form_id": 1,
      "total": 1,
      "url": "abCDefghIklMnop57498257/1_20002_form_export.zip"
    }
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <status>completed</status>
    <progress>100</progress>
    <result>
      <form_id>1</form_id>
      <total>1</total>
      <url>abCDefghIklMnop57498257/1_20002_form_export.zip</url>
    </result>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Description
status	**string**	**Task status (`completed` — task is done)**
progress	**int**	**Task progress in percent**
result	**object**	**Export result object. See result**
error	**int**	**Error code**
error_text	**string**	**Error text**

`result` Object

Parameter	Type	Description
form_id	**int**	**Form ID**
total	**int**	**Total number of fill records**
url	**string**	**Encrypted path to the data file**

Step 3. Downloading the file

To download the file, send a GET request to a URL composed of the platform's base URL and the `url` v
`https://example.com/api/download/` + value of `url`
For example:
`https://example.com/api/download/abCDefghIklMnop57498257/1_20002_form_export.zip`
where:

- `https://example.com` **— platform domain**
- `api/download/` **— fixed prefix for file download**
- `abCDefghIklMnop57498257/1_20002_form_export.zip` **— value of the `url` parameter from the step 2 response**

The request must include the authorization header `Authorization: Bearer <token>`.

Export form fill data

Description

Exports form fill data to a file. To get the result, you need to check the task status and download the re

Step 1. Creating an export task

Request URL

Method: `POST`

`https://example.com/api/v1.1/forms/fills`

Request Parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes, if `shortname` is not specified**	**Form ID**
shortname	**string**	**“form_6Ven2v7O”**	**Yes, if `id` is not specified**	**Form short name**
date_from	**string**	**“2026-04-01”**	**Yes**	**Start date of the period. Format:**
date_to	**string**	**“2026-04-30”**	**Yes**	**End date of the period. Format: Y**
file_format	**string**	**“csv”**	**No**	**File format: `csv` or `json`**
limit	**int**	**1000**	**No**	**Maximum number of records. De**

Request Example

JSON

```
{
  "id": 1,
  "date_from": "2026-04-01",
  "date_to": "2026-04-30",
  "file_format": "csv"
}
```

XML

```
<xml>
  <id>1</id>
  <date_from>2026-04-01</date_from>
  <date_to>2026-04-30</date_to>
  <file_format>csv</file_format>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "task_id": "550e8400-e29b-41d4-a716-446655440000"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>550e8400-e29b-41d4-a716-446655440000</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Descripti
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**object**	**Response data object**

`data` Object

Parameter	Type	Description
task_id	**string**	**Async task ID for retrieving fill data**

Step 2. Checking task status and getting download link

Request URL

Method: `POST`

`https://example.com/api/v1.1/tasks/status`

Request Parameters

Parameter	Type	Example	Required
task_id	**string**	**“550e8400-e29b-41d4-a716-446655440000”**	**Yes**

Request Example

JSON

```
{
  "task_id": "550e8400-e29b-41d4-a716-446655440000"
}
```

XML

```
<xml>
  <task_id>550e8400-e29b-41d4-a716-446655440000</task_id>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "status": "completed",
    "progress": 100,
    "result": {
      "form_id": 1,
      "total": 42,
      "url": "abCDefghIklMnop57498257/1_20002_form_export.zip"
    }
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <status>completed</status>
    <progress>100</progress>
    <result>
      <form_id>1</form_id>
      <total>42</total>
      <url>abCDefghIklMnop57498257/1_20002_form_export.zip</url>
    </result>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Description
status	**string**	**Task status (`completed` — task is done)**
progress	**int**	**Task progress in percent**
result	**object**	**Export result object. See result**
error	**int**	**Error code**
error_text	**string**	**Error text**

`result` Object

Parameter	Type	Description
form_id	**int**	**Form ID**
total	**int**	**Total number of fill records**
url	**string**	**Encrypted path to the data file**

Step 3. Downloading the file

To download the file, send a GET request to a URL composed of the platform's base URL and the `url` v
`https://example.com/api/download/` + value of `url`

For example:

`https://example.com/api/download/abCDefghIklMnop57498257/1_20002_form_export.zip`

where:

- `https://example.com` **— platform domain**
- `api/download/` **— fixed prefix for file download**
- `abCDefghIklMnop57498257/1_20002_form_export.zip` **— value of the `url` parameter from the step 2 response**

The request must include the authorization header `Authorization: Bearer <token>`.

Publish form

Description

The method allows you to forcibly publish a form at the time of the request execution (run_now). While you to set a publication date, this method publishes the form immediately.

URL

Method: POST

`https://example.com/api/v1.1/forms/activate`

Request Parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes, if `shortname` is not specified**
shortname	**string**	**“form_6Ven2v7O”**	**Yes, if `id` is not specified**

Request Example

JSON

```
{
  "id": 1
}
```

XML

```
<xml>
<id>1</id>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "url": "https://forms.example.com/form?token=a1b2c3d4e5f67890abcdef1234567890"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
<data>
<url>https://forms.example.com/form?token=a1b2c3d4e5f67890abcdef1234567890</url>
</data>
<error>0</error>
<error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	Descripti
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**object**	**Response data object**

`data` Object

Parameter	Type	Description
url	**string**	**Link to the published form**

Unpublish form

Description

The method unpublishes a form. After this, the form becomes unavailable for users to fill out.

URL

Method: `POST`

`https://example.com/api/v1.1/forms/deactivate`

Request Parameters

Request body (application/json):

Parameter	Type	Example	Required
id	**integer**	**1**	**Yes, if `shortname` is not specified**
shortname	**string**	**form_6Ven2v7O**	**Yes, if `id` is not specified**

Request Example

JSON

```
{
  "id": 1
}
```

XML

```
<xml>
  <id>1</id>
</xml>
```

Response Example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	
error	**integer**	**Error code**
error_text	**string**	**Error text**

Delete form

Description

**Permanently deletes the form without moving it to the trash. If necessary, you can delete only the coll
itself.**

URL

Method: `POST`

`https://example.com/api/v1.1/forms/delete`

Request Parameters

Request body (application/json):

Parameter	Type	Example	Required	Description
id	**int**	**1**	**Yes, if `shortname` is not specified**	**Form ID**
shortname	**string**	**“form_6Ven2v7O”**	**Yes, if `id` is not specified**	**Form short name**
data_only	**boolean**	**false**	**No**	**If `true` — only form fill data (`forms_data`) is deleted, the form itself is preserved. Defaul**

Request Example

JSON

```
{
  "id": 1,
  "data_only": false
}
```

XML

```
<xml>
  <id>1</id>
  <data_only>false</data_only>
</xml>
```

Response Example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type	
error	**integer**	**Error code**
error_text	**string**	**Error text**

Import promo codes

Description

Imports one or multiple promo codes into a loyalty program.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/import`

Request example

Parameter	Type	Example	Required	
codes	**JSON array**	["CODE-1", "CODE-2"]	**Yes**	**Pron**
loyalty_id	**int**	**1**	**Yes**	**Loya**

Request example

JSON

```
{
    "codes": [
        "CODE-1",
        "CODE-2"
    ],
    "loyalty_id": 1
}
```

XML

```
<xml>
    <codes>CODE-1</codes>
    <codes>CODE-2</codes>
    <loyalty_id>6</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "code": "CODE-1",
            "error": 0,
            "error_text": ""
        },
        {
            "code": "CODE-2",
            "error": 0,
            "error_text": ""
        }
    ],
    "discarded_by_limit": 0,
    "duplicated": 0,
    "error": 0,
    "error_text": "Successful operation",
    "inserted": 2
}
```

XML

```
<xml>
    <data>
        <code>CODE-1</code>
        <error>0</error>
    </data>
    <data>
        <code>CODE-2</code>
        <error>0</error>
    </data>
    <discarded_by_limit>0</discarded_by_limit>
    <duplicated>0</duplicated>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <inserted>2</inserted>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
code	**string**	**Promo code**
discarded_by_limit	**int**	**Import error due to loyalty program limitations**
duplicated	**int**	**Import error due to duplication**
inserted	**int**	**Number of successfully imported promo codes**

Get promo code information

Description

Retrieves a loyalty program promotional code information.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/get`

Request parameters

Parameter	Type	Example	Required	
format	**int**	**“json”**	**No**	**Response data form By default – json**
code	**string**	**“CODE-12345”**	**Yes**	**Promo code**
loyalty_id	**int**	**“1”**	**Yes**	**Loyalty program ID**

Request example

JSON

```
{
    "code": "CODE-1",
    "loyalty_id": 5
}
```

XML

```
<xml>
    <code>CODE-1</code>
    <loyalty_id>5</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "created": "2024-04-01T12:39:34Z",
        "is_attached": false,
        "is_activated": false,
        "is_expired": false,
        "attached": "",
        "activated": "",
        "expired": "",
        "expire_date": "",
        "loyalty_id": 5,
        "code": "CODE-1",
        "database_id": 1,
        "profile_id": "6a4bb37590746e76af4cc02e"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <code>CODE-1</code>
        <created>2024-04-01T12:39:34Z</created>
        <database_id>1</database_id>
        <is_activated>false</is_activated>
        <is_attached>false</is_attached>
        <is_expired>false</is_expired>
        <loyalty_id>5</loyalty_id>
        <profile_id>6a4bb37590746e76af4cc02e</profile_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Параметр	Тип	Описание
error	**int**	**Error code**
error_text	**string**	**Error text**
activated	**datetime**	**Promo code activation time**
attached	**datetime**	**Promo code attach time**
code	**string**	**Promo code**
created	**datetime**	**Promo code creation time**
expire_date	**datetime**	**Promo code expire date**
expired	**datetime**	**Promo code expire date**
is_activated	**bool**	**Activation flag**
is_attached	**bool**	**Attached to a profile flag**
is_expired	**bool**	**Expired flag**
loyalty_id	**int**	**Loyalty program ID**
database_id	**int**	**Database ID, if promocode attached to profile**
profile_id	**string**	**Profile ID, if promocode attached to profile**
fields	**array**	**An array of additional promo code fields. Returned only when the promo code has additional fields. Each element contains:** `fieldname` **(string) — fi — field value**

Activate promo code

Description

Activates a promo code.

> **CAUTION**
>
> **You can only activate a promotional code when it is attached to a profile.**

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/activate`

Request parameters

Parameter	Type	Example	Required	
code	**string**	**“CODE-12345”**	**Yes**	**Promo code**
format	**int**	**“json”**	**No**	**Response data forma By default– json**
loyalty_id	**int**	**“1”**	**Yes**	**Loyalty program ID**

Request example

JSON

```
{
    "code": "Code-1",
    "loyalty_id": 5
}
```

XML

```
<xml>
    <code>Code-1</code>
    <loyalty_id>5</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Update promo code

Description

Updates promo code expiry date, activates or deactivates a promo code.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/update`

Request parameters

Parameter	Type	Example	Required	
format	**int**	**“json”**	**No**	**Response data format. By default – json**
activated	**bool**	**true**	**No**	**Promo code activation or deactivation**
code	**string**	**“CODE-12345”**	**Yes**	**Promotional code**
expire_date	**string**	**“2018-08-25T08:00:00Z”**	**No**	**Promo code expire date in RFC 3339 form**
loyalty_id	**int**	**1**	**Yes**	**Loyalty program ID**

CAUTION

At least one of request options is expected for a successful operation, expiry date change or activatio

Request example

JSON

```
{
    "activated": true,
    "code": "CODE-12345",
    "expire_date": "2018-08-25T08:00:00Z",
    "loyalty_id": 1
}
```

XML

```
<xml>
    <activated>true</activated>
    <code>Code-1</code>
    <expire_date>2024-08-10T08:00:00Z</expire_date>
    <loyalty_id>6</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Attach promo code

Description

Attaches a promotion code to a profile.

If a `"code"` parameter is specified, it will be used for promo code matching.

- **If the code is already attached and its multiple usage is restricted, an error will occur.**
- **If a code is found unattached — it will be attached.**
- **If a code is not found — it will be imported and attached.**

With no `"code"` parameter loyalty program ID will be used for matching.

- **If there are promo codes available in a loyalty program — one will be attached.**

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/attach`

Request parameters

Parameter	Type	Example	Required	
code	**string**	**“CODE-12345”**	**No**	**Promo code**
db_id	**int**	**1**	**Yes**	**Database ID**
email	**string**	**“john@example.com”**	**No**	**Email address**
field_name	**string**		**No**	**Custom database field for profile mat**
field_value	**string**		**No**	**Custom field value**
matching	**string**	**“profile_id” “email” “phone” “custom”**	**No**	**Profiles matching mode**
loyalty_id	**int**	**1**	**Yes**	**Loyalty program ID**
phone	**string**	**“+79000000000”**	**No**	**Phone number**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**Profile ID**

Request example

JSON

```
{
    "db_id": 1,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "loyalty_id": 5,
    "code": "CODE-1"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <loyalty_id>5</loyalty_id>
    <code>CODE-1</code>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Detach promo code

Description

Detaches a promotion code from a profile.

Use one of these parameters for profile matching:

- `email`
- `field_name` , `field_value`
- `phone`
- `profile_id`

If no `"code"` parameter is specified or if a specified code is attached to multiple profiles — an error will

Request URL

method: POST

`https://example.com/api/v1.1/promocodes/detach`

Request parameters

Parameter	Type	Example	Required	
code	**string**	**“CODE-12345”**	**No**	**Promotion code**
db_id	**int**	**1**	**Yes**	**Database ID**
email	**string**	**“john@example.com”**	**No**	**Email address**
field_name	**string**		**No**	**Custom profile matching**
field_value	**string**		**No**	**Custom field value**
matching	**string**	**“profile_id” “email” “phone” “custom”**	**No**	**Profiles matching mode**
loyalty_id	**int**	**1**	**Yes**	**Loyalty program ID**

Request example

JSON

```
{
    "code": "Code-1",
    "db_id": 1,
    "loyalty_id": 4,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <code>Code-1</code>
    <db_id>1</db_id>
    <loyalty_id>4</loyalty_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get all promo codes

Description

Retrieves all promo codes from a loyalty program.

URL

Method: POST

`https://example.com/api/v1.1/promocodes/export_all`

Request parameters

Parameter	Type	Example	Required	
format	**int**	**“json”**	**No**	**Response data format. By default – json.**
loyalty_id	**int**	**1**	**Yes**	**Loyalty program ID**

Request example

JSON

```
{
    "loyalty_id": 6
}
```

XML

```
<xml>
    <loyalty_id>6</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "code": "Code-1"
        },
        {
            "code": "Code-2",
            "attached": "2024-04-01T14:02:05Z",
            "activated": "2024-04-01T14:04:12Z"
        },
        {
            "code": "Code-3"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <code>Code-1</code>
    </data>
    <data>
        <activated>2024-04-01T14:04:12Z</activated>
        <attached>2024-04-01T14:02:05Z</attached>
        <code>Code-2</code>
    </data>
    <data>
        <code>Code-3</code>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
data	**Array**	**Promo codes list**
code	**string**	**Promo code**
attached	**string**	**Attachment date. No property if not attached**
activated	**string**	**Activation date. No property if not activated**
expire_date	**string**	**Promo code expiry date. Optional parameter**
expired	**string**	**Expiry date in case a promo code is expired**
error	**int**	**Error code**
error_text	**string**	**Error text**

Goals and goal values registration

Description

Records a goal achievement and optionally value for a customer profile. As part of event registration, y
profile information.

You can use event data in the following mechanics:

- **segment profiles based on activity history;**
- **trigger a mailing or a scenario;**
- **add dynamic content into messages.**

Request URL

Method: POST

`https://example.com/api/v1.1/goals/register`

Request parameters

Main parameters

Parameter	Type	Example	Required	
db_id	**int**	**1**	**Yes**	**Database ID**
pixel_id	**int**	**1**	**Yes**	**Pixel ID**
goal	**string**	**"card"**	**No**	**Goal name**
goals	**JSON array**	`[{"goal": "card", "value": 10.0}]`	**No**	**Goals list**
value	**float**	**-25.5**	**No**	**Goal value**
date	**string**	**"2006-01-25T15:04:05+07:00"**	**No**	**Event date in RF**

Matching parameters

These parameters are used to search for a profile in the database. Read more.

Parameter	Type	Example	Required	Description
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**matching:"email"**	**Email address**
phone	**string**	**"+79000000000"**	**matching:"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**matching:"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**matching:"custom"**	**Custom profile data field name**
field_value	**int/string**	**"123420"**	**matching:"custom"**	**Custom profile data field value. Searching by combination of tags is available for tag type profile fields. Values**

Advanced parameters

Parameter	Type	Example	Required	Descri
pixel_data	**JSON object**	`{ "key":"value", "array":[1,2,3] }`	**No**	**A JSON object containing additional target event data. The object must not contain nested objects or arrays. The Dates in this object are specified in RFC 3339 format (`2006**
skip_triggers	**bool**	**false**	**No**	**Disables triggers and scenarios set up for an event. By default — false.**
enrich_profile	**bool**	**true**	**No**	**The event updates profile data. By default — true.**
is_test	**bool**	**false**	**No**	**Test event flag. By default — false.**
vendor	**string**	**"altcraft"**	**No**	**Profile field `_vendor` at the time of the event**
external_type	**string**	**"event_pixel_test"**	**No**	**External identifier type**
external_event_id	**string**	**"qwerty123"**	**No**	**External identifier**
altcraft_client_id	**string**	**"9f11d32d-2935-43d0-b8df-f4c3568095b2"**	**No**	**Altcraft unique end-to-end identifier**

HTTP-parameters

Parameter	Type	Example	Required	Description
referer	**string**	**"https://example.com/page"**	**No**	**Original request referer. Must contain the full URL of the page where**
user_agent	**string**	**"Mozilla/5.0 (Linux; Android 6.0.1; SM-G935S Build/MMB29K; wv)"**	**No**	**Original request browser**
accept_language	**string**	**"fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5"**	**No**	**Original request language**
client_ip	**string**	**"10.0.0.0"**	**No**	**Original request IP address**
client_ip_v4	**string**	**"198.16.74.224"**	**No**	**Client IPv4 address**

Parameter	Type	Example	Required	Description
client_ip_v6	string	"fe80::5215:5556:d75f:806b"	No	Client IPv6 address
browser	string	"Chrome"	No	Client browser You can obtain a list of valid values through the request [`/v1.1/dictior guide/miscellaneous/dictionary)
os	string	"Ubuntu"	No	Client operating system You can obtain a list of valid values through the request [`/v1.1/dictior guide/miscellaneous/dictionary)
device	string	"Desktop"	No	Client device type You can obtain a list of valid values through the request [`/v1.1/dictior guide/miscellaneous/dictionary)

TIP

The browser, os and device fields are automatically populated by the platform based on the contents

If you pass both user_agent and separate values for browser, os and device in the same request, the d precedence and overwrites the specified values.

If you pass browser, os and device fields without user_agent, you must specify `"enrich_profile": true` to platform will not update the profile.

Geolocation parameters

The platform will automatically determine the values if `client_ip` is passed.

Parameter	Type	Example	Required	
lat	float	55.244075	Yes, if lon is passed	Client latitude
lon	float	55.244075	Yes, if lat is passed	Client longitude
country	string	"RU"	No	Client country (ISO code or full n
region	string	"Ryazan"	No	Client region
city	string	"Ryazan"	No	Client city
address	string	"Pochtovaya street"	No	Client address
zip	string	"390000"	No	Client zip code
time_zone	string	"Europe/Moscow"	No	Client timezone

TIP

The platform automatically determines the values of the country, region, city, zip, and time_zone field in the client_ip, client_ip_v4, or client_ip_v6 parameters. If both the IP address and individual values fo request, the IP-defined data will take precedence and overwrite the manually passed values.

To update a profile using values from the country, region, city, zip and time_zone fields, the `"enrich_pro`

UTM-parameters

Parameter	Type	Example	Required
utm_content	string	"utm_content"	No
utm_medium	string	"utm_medium"	No
utm_source	string	"utm_source"	No
utm_campaign	string	"utm_campaign"	No
utm_term	string	"utm_term"	No

Network parameters

Parameter	Type	Example	Required	
net_ssid	string	"10.200.0.0"	No	Client network nar
net_mac_addr	string	"28:87:ba:c1:0d:5e"	No	Client MAC addres
net_gateway_ipv4	string	"10.200.0.1"	No	Client gateway IPv
net_gateway_ipv6	string	"::ff"	No	Client gateway IPv

Mailing parameters

Parameter	Type	Example	Required	Description
send_message_id	string	"w4Wq877djSG1_2_a1_6t_2p_U_2_g. 2UCSmxz7zFnuCfCMK_5KmKH9"	No	ID of the sent message in the platform. Use the parameter to allow you to associate

Loyalty programs parameters

Parameter	Type	Example	Required	
loyalty_id	int	5	No	Loyalty p
promo_id	string	"6285183c3f62fcd4ac5ed36d"	No	Attached

Market parameters

Parameter	Type	Example	Required	Description
endpoint_eid	**string**	**"example_endpoint"**	**No**	**External contact point identifier.** **If a specific profile base is selected for a pixel, then this base must match the**
order_eid	**string**	**"12345"**	**No**	**External order ID**
product_eid	**string**	**"example_product"**	**Yes,** **if sku_eid is passed**	**External product ID**
sku_eid	**string**	**"example_sku"**	**No**	**External SKU ID. When this parameter is used, product_eid must also be passe**
region_eid	**string**	**"region_eYFe2AE6s"**	**No**	**External region ID**
count_items	**int**	**1**	**No**	**Quantity of items in the order item**

categories	**array**	`[`
		`  "category_1",`
		`  "category_2"`
		`]`
			No	**Product categories or SKUs**

INFO

The maximum length of any string field is 128 characters.

Request examples

1 goal, no original request header data used:

JSON

```
{
    "db_id": 1,
    "goal": "card",
    "pixel_id": 1,
    "referer": "https://example.com/page",
    "value": -25.5,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <db_id>1</db_id>
    <goal>card</goal>
    <pixel_id>1</pixel_id>
    <referer>https://example.com/page</referer>
    <value>-25.5</value>
    <matching>profile_id</matching>
    <profile_id>65d49bc74a6f1407ac656547</profile_id>
</xml>
```

2 goals, IP and referer from the original request head are used:

JSON

```
{
    "client_ip": "10.20.30.40",
    "db_id": 24,
    "goals": [
        {
            "goal": "cart",
            "value": 10.99
        },
        {
            "goal": "two",
            "value": -12.34
        }
    ],
    "pixel_id": 7,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "referer": "https://example.com/page"
}
```

XML

```
<xml>
    <client_ip>10.20.30.40</client_ip>
    <db_id>24</db_id>
    <goals>
        <goal>cart</goal>
        <value>10.99</value>
    </goals>
    <goals>
        <goal>two</goal>
        <value>-12.34</value>
    </goals>
    <pixel_id>7</pixel_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <referer>https://example.com/page</referer>
</xml>
```

Example of a request to send additional information about an event:

JSON

```
{
    "db_id": 24,
    "goals": [
        {
            "goal": "cart",
            "value": 10.99
        },
        {
            "goal": "two",
            "value": -12.34
        }
    ],
    "client_ip": "192.168.0.0",
    "pixel_id": 7,
    "referer": "https://example.com/page",
    "accept_language": "fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36",
    "matching": "profile_id",
    "profile_id": "651516181a5c52f94532fb1d",
    "pixel_data": {
        "hello": "hi",
        "arr": [
            1,
            2,
            3
        ],
        "int_val": 56893,
        "bool_val": true,
        "date_val": "2023-08-12"
    },
    "skip_triggers": false,
    "enrich_profile": true,
    "is_test": true,
    "vendor": "altcraft",
    "external_type": "event_pixel_test",
    "external_event_id": "qwerty123",
    "altcraft_client_id": "9f11d32d-2935-43d0-b8df-f4c3568095b2",
    "browser": "Chrome",
    "os": "Ubuntu",
    "device": "Desktop",
    "utm_content": "utm_content",
    "utm_medium": "utm_medium",
    "utm_source": "utm_source",
    "utm_campaign": "utm_campaign",
    "utm_term": "utm_term",
    "client_ip_v4": "198.16.74.224",
    "client_ip_v6": "fe80::5215:5556:d75f:806b",
    "net_ssid": "10.200.0.0",
    "net_mac_addr": "28:87:ba:c1:0d:5e",
    "net_gateway_ipv4": "10.200.0.1",
    "net_gateway_ipv6": "::ff",
    "loyalty_id": 5,
    "promo_id": "6285183c3f62fcd4ac5ed36d",
    "endpoint_eid": "online_store",
    "order_eid": "12345",
    "product_eid": "hd_450_bt_earphones",
    "sku_eid": "hd_450bt_earphones_black",
    "region_eid": "region_eYFe2AE6s",
    "count_items": 1,
    "categories": [
        "category_KBZ1qurFP"
    ],
    "lat": 55.244075,
    "lon": 19.379932,
    "country": "RU",
    "region": "Ryazan",
    "city": "Ryazan",
    "address": "Pochtovaya street",
    "zip": "390000",
    "time_zone": "Europe/Moscow",
    "send_message_id": "w4Wq877djSG1_2_a1_6t_2p_U_2_g.2UCSmxz7zFnuCfCMK_5KmKH9"
}
```


XML

```
<xml>
    <db_id>24</db_id>
    <goals>
        <goal>cart</goal>
        <value>10.99</value>
    </goals>
    <goals>
        <goal>two</goal>
        <value>-12.34</value>
    </goals>
    <client_ip>192.168.0.0</client_ip>
    <pixel_id>7</pixel_id>
    <referer>https://example.com/page</referer>
    <accept_language>fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5</accept_language>
    <user_agent>Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36</user_agent>
    <matching>profile_id</matching>
    <profile_id>651516181a5c52f94532fb1d</profile_id>
    <pixel_data>
        <hello>hi</hello>
        <arr>1</arr>
        <arr>2</arr>
        <arr>3</arr>
        <int_val>56893</int_val>
        <bool_val>true</bool_val>
        <date_val>2023-08-12</date_val>
    </pixel_data>
    <skip_triggers>false</skip_triggers>
    <enrich_profile>true</enrich_profile>
    <is_test>true</is_test>
    <vendor>altcraft</vendor>
    <external_type>event_pixel_test</external_type>
    <external_event_id>qwerty123</external_event_id>
    <altcraft_client_id>9f11d32d-2935-43d0-b8df-f4c3568095b2</altcraft_client_id>
    <browser>Chrome</browser>
    <os>Ubuntu</os>
    <device>Desktop</device>
    <utm_content>utm_content</utm_content>
    <utm_medium>utm_medium</utm_medium>
    <utm_source>utm_source</utm_source>
    <utm_campaign>utm_campaign</utm_campaign>
    <utm_term>utm_term</utm_term>
    <client_ip_v4>198.16.74.224</client_ip_v4>
    <client_ip_v6>fe80::5215:5556:d75f:806b</client_ip_v6>
    <net_ssid>10.200.0.0</net_ssid>
    <net_mac_addr>28:87:ba:c1:0d:5e</net_mac_addr>
    <net_gateway_ipv4>10.200.0.1</net_gateway_ipv4>
    <net_gateway_ipv6>::ff</net_gateway_ipv6>
    <loyalty_id>5</loyalty_id>
    <promo_id>6285183c3f62fcd4ac5ed36d</promo_id>
    <endpoint_eid>online_store</endpoint_eid>
    <order_eid>12345</order_eid>
    <product_eid>hd_450_bt_earphones</product_eid>
    <sku_eid>hd_450bt_earphones_black</sku_eid>
    <region_eid>region_eYFe2AE6s</region_eid>
    <count_items>1</count_items>
    <categories>category_KBZ1qurFP</categories>
    <lat>55.244075</lat>
    <lon>19.379932</lon>
    <country>RU</country>
    <region>Ryazan</region>
    <city>Ryazan</city>
    <address>Pochtovaya street</address>
    <zip>390000</zip>
    <time_zone>Europe/Moscow</time_zone>
    <send_message_id>w4Wq877djSG1_2_a1_6t_2p_U_2_g.2UCSmxz7zFnuCfCMK_5KmKH9</send_message_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Processing and adding a subscription

The API request is used to transfer the push token from the mobile application to the Altcraft Platform
With this request, you can:

- **Add new subscriptions.**
- **Change the status of existing subscriptions.**

The profile search mode can be set using the `matching` field in the request (see specification below). If will be created.
Moreover, if several databases are specified in the resource, then all of them will be processed separa will create and update profiles in several databases at once. An example of such a case can be seen b If there are no databases in the resource, then an error will be returned.

Request URL

Method: POST

`https://<cookiesaver_host>/push_subscribe`

Request parameters

Parameter	Type	Example	Required	Description
id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**Profile ID**
resource_token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**Unique resource ID.**
db_ids	**[ ] int**	**[2,3]**	**No**	**Resource database filter.**
matching	**string**	**“custom”**	**No**	**Profile search mode. If the parameter is not passed or is empty, profile search by ID will be used. If ID is al For the `push_subscribe` method, this parameter can only accept the following values: • custom • profile_id**
subscriptions	**JSON array SubscriptionObject**	`[{ “provider” : “android-firebase”, “subscription_id” : “%subscriptionid%” }]`	**Yes**	**Push subscriptions that should be added to profile or should be updated. “subscription_id” is unique for each device, Firebase provides it.**

SubscriptionObject

Parameter	Type	Example	Required	
provider	**string**	**“android-firebase”**	**Yes**	**Altcraft Platform Provider ID. The available values: • android-firebase • ios-firebase • ios-apns • yandex-Appmetrica-device-id-ios • yandex-Appmetrica-device-id-android • yandex-Appmetrica-ios-ifa • yandex-Appmetrica-google-aid • yandex-Appmetrica-android-token • yandex-Appmetrica-ios-token • android_huawei • ios_huawei Each token type in AppMetrica has its own provi**
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Yes**	**Subscription push token**
unsubscribe	**bool**	**false**	**No**	**Unsubscribe flag. Dismiss subscription if its valu**

Response parameters

Parameter	Type	Description
data	**JSON array DataObject**	**Information about the created subscriptions. The array can contain several objects in the following cases: • Several SubscriptionObject were passed in the request. • ID field was not used in the request, and the Altcraft Platform resource uses several databases. In this case, a new profile with subscripti**
error	**int**	**Error code**
error_text	**string**	**Error text**

DataObject

Parameter	Type	Example	Description
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Subscription push token**
unsubscribe	**bool**	**false**	**Flag whether the subscription was added or canceled**
profile_id	**string**	**“5fb62e815ae47a4aa04dfadd”**	**ID of the profile (transferred or newly created) for which the subscription**

Parameter	Type	Example	Description
provider	**string**	**“android-firebase”**	**Altcraft Platform Provider ID.** **The available values:** • **android-firebase** • **ios-firebase** • **ios-apns** • **yandex-Appmetrica-device-id-ios** • **yandex-Appmetrica-device-id-android** • **yandex-Appmetrica-ios-ifa** • **yandex-Appmetrica-google-aid** • **yandex-Appmetrica-android-token** • **yandex-Appmetrica-ios-token** • **android_huawei** • **ios_huawei** **Each token type in AppMetrica has its own provider ID.**
db_id	**int**	**2**	**Altcraft Platform Database ID. The profile is stored in this database.**
resource_id	**int**	**1**	**Altcraft Platform resource ID.**
subscription_hash_id	**string**	**“92807e89”**	**Subscription hash**
status	**string**	**“error”**	**Operation status** • **ok** • **error**
description	**string**	**“Duplicate push subscription”**	**Error description**

Examples

Request with matching by system profile ID

Request

```
{
  "id": "59db33d269d426509e9c4bf9",
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bf9",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request with matching by a custom profile field

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfc",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request with matching by push subscriptions

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfa",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request to a resource with several databases

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response

```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "provider": "android-firebase",
      "resource_id": 22,
      "status": "ok",
      "subscription_hash_id": "340ca49c",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    },
    {
      "db_id": 2,
      "description": "matching error: unknown field",
      "resource_id": 22,
      "status": "error"
    }
  ]
}
```


Request with matching by a custom profile field in a certain database

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "db_ids": [1],
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 1,
      "description": "",
      "profile_id": "637f1a284fda7c5c08be62f",
      "provider": "android-firebase",
      "resource_id": 31,
      "status": "ok",
      "subscription_hash_id": "f8af057de",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    }
  ]
}
```


Request did not update or add any profile subscriptions

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "resource_id": 22,
      "status": "ok"
    }
  ]
}
```

Add app push events

Description

Adds application push events.

Request URL

Method: POST

`https://example.com/api/v1.1/pushes/post_events`

Request parameters

Parameter	Type	Example	Required	
events	**int**	**1 - delivered 2 - opened 3 - delivered and opened**	**Yes**	**Events**
launch_id	**int**	**1**	**Yes**	**Is sent alongside push from Altcraft to Data payl**
provider	**string**	**“android-firebase”**	**Yes**	**Push provider**
subscription_id	**string**	**“subscriptionid”**	**Yes**	**subscription ID**
uid	**string**	**“uid”**	**Yes**	**Is sent alongside push from Altcraft to Data payl**

Request example

JSON

```
{
  "events": 1,
  "launch_id": "launch_id",
  "provider": "android-firebase",
  "subscription_id": "subscriptionid",
  "uid": "uid"
}
```

XML

```
<xml>
  <events>1</events>
  <launch_id>launch_id</launch_id>
  <provider>android-firebase</provider>
  <subscription_id>subscriptionid</subscription_id>
  <uid>uid</uid>
</xml>
```

Request example with uid in special format

JSON

```
{
  "events": 1,
  "uid": "e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns"
}
```

XML

```
<xml>
    <events>1</events>
    <uid>e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns</uid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Order data object

Description

Order data object is used in order import requests.

Parameter	Type	Example	Required	Description
external_id	string	“123456”	Yes	Order additional ID
matching	string	“email”	No	Subscriber search mode. By default – email For each type of matching, certain fields must be passed in the request body. You can learn more about tl
endpoint	string	“website”	Yes	Touchpoint ID
tags	array	[“one”, “two”]	No	List of tags
status	string	“new” “delivered” “paid” “canceled” “part_refunded”	Yes	Order status ID
delivery_cost	string	“200”	No	Cost of delivery
total_price	string	“3000”	Yes	Total cost of the order. It consists of the cost of each item in the order and the cost of delivery.
currency	string	“USD”	Yes	The currency that is used to indicate the cost of the order. Currencies are configured in the platform interface: Market —> Settings —> Other. If the request passes a occur.
create_time	string	“2006-01-25” “2006-01-25 15:04:05”	No	Order creation date (RFC 3339 format)
update_time	string	“2006-01-25” “2006-01-25 15:04:05”	No	Order update date (RFC 3339 format)

custom_fields	object	{
		“order_field_id_1”: “data”,
		“order_field_id_2”: 123
		}
			No	Custom order fields
				`"[field ID]": "[field value]"`

region	string	“region_external_id”	No	Region additional ID. You can create a region and get its identifier in the platform interface: Market —> Settings.

lines	array	[
		{
		“external_id”: “7777”,
		“product”:
		“product_external_id”,
		“base_price_per_item”: “600”,
		“min_price_per_item”: “300”,
		“final_price_per_item”: “500”,
		“count”: 5
		}
		]
			Yes	Order item data. At least one position.
				
				See below.

Object structure

JSON

```
[
  {
    "external_id": "123456",
    "matching": "email",
    "email": "johndoe@example.com",
    "endpoint": "endpoint_external_id",
    "tags": [
      "one",
      "two"
    ],
    "status": "new",
    "delivery_cost": "100",
    "total_price": "1000",
    "currency": "USD",
    "custom_fields": {
      "order_field_id_1": "data",
      "order_field_id_2": 123
    },
    "region": "region_external_id",
    "lines": [
      {
        "external_id": "1234000",
        "status": "new",
        "sku": "sku_external_id",
        "base_price_per_item": "900",
        "min_price_per_item": "900",
        "final_price_per_item": "900",
        "count": 5,
        "custom_fields": {
          "line_field_id_2": "data",
          "line_field_id_1": 123
        }
      }
    ]
  }
]
```

XML

```
<xml>
    <row>
        <external_id>123456</external_id>
        <matching>email</matching>
        <email>johndoe@example.com</email>
        <endpoint>endpoint_external_id</endpoint>
        <tags>one</tags>
        <tags>two</tags>
        <status>new</status>
        <delivery_cost>100</delivery_cost>
        <total_price>1000</total_price>
        <currency>USD</currency>
        <custom_fields>
            <order_field_id_1>data</order_field_id_1>
            <order_field_id_2>123</order_field_id_2>
        </custom_fields>
        <region>region_external_id</region>
        <lines>
            <external_id>1234000</external_id>
            <status>0001</status>
            <sku>sku_external_id</sku>
            <base_price_per_item>900</base_price_per_item>
            <min_price_per_item>900</min_price_per_item>
            <final_price_per_item>900</final_price_per_item>
            <count>5</count>
            <custom_fields>
                <line_field_id_2>data</line_field_id_2>
                <line_field_id_1>123</line_field_id_1>
            </custom_fields>
        </lines>
    </row>
</xml>
```

Lines object

Description

`lines` **is the array of objects. It contains objects with data about order items. One object stores one or**

Parameter	Type	Example	Required	Descript
external_id	**string**	**“1235678”**	**Yes**	**Item additional ID**
status	**string**	**“new”**	**No**	**Order item status ID:** • `delivering` • `processing` • `new` • `refunded` • `canceled` • `delivered` • **or custom status id**
sku	**string**	**IPhone_Red_128Gb**	**Do not pass it in the request if you pass the `product` parameter**	**SKU additional ID. If the product is represented as a product - s variety - specify only the SKU ID.**
product	**string**	**MP3_player_BassPro**	**Do not pass it in the request if you pass the `sku` parameter**	**Product additional ID**
base_price_per_item	**string**	**“600”**	**Yes**	**Base price**
min_price_per_item	**string**	**“300”**	**Yes**	**Minimum price**
final_price_per_item	**string**	**“500”**	**Yes**	**Final price**
count	**int**	**5**	**Yes**	**Quantity of products or SKUs in the order item**
custom fields	**object**	`{ “line_field_id_2”: 123, “line_field_id_2”: “data” }`	**No**	**Value of custom fields for the order item**

Object stucture

JSON

```
[
  {
    "external_id": "12345",
    "status": "new",
    "product": "MP3_player_BassPro",
    "base_price_per_item": "600",
    "min_price_per_item": "300",
    "final_price_per_item": "500",
    "count": 5,
    "custom_fields": {
      "lot_number": 123,
      "imported": true
    }
  }
]
```

Example of SKU:

```
[
  {
    "external_id": "76543",
    "status": "processing",
    "sku": "IPhone_Red_128Gb",
    "base_price_per_item": "1000",
    "min_price_per_item": "700",
    "final_price_per_item": "850",
    "count": 13,
    "custom_fields": {
      "lot_number": 93,
      "localization": "RU"
    }
  }
]
```

XML

```
<xml>
<external_id>1235678</external_id>
  <status>0001</status>
  <sku>sku_external_id</sku>
  <base_price_per_item>600</base_price_per_item>
  <min_price_per_item>300</min_price_per_item>
  <final_price_per_item>500</final_price_per_item>
  <count>5</count>
  <custom_fields>
    <line_field_id_2>data</line_field_id_2>
  </custom_fields>
</xml>
```

Product data object

Description

Product data object used in product import requests.

Parameter	Type	Example	Required	Description
name	**string**	**“Canon EOS Digital Camera”**	**Yes**	**Product’s name.**
external_id	**string**	**“12345”**	**Yes**	**Product ID that is used when adding an item to an order.**
price	**string**	**“490”**	**No**	**Current price of the product.**
old_price	**string**	**“520”**	**No**	**Old price of the product.**
currency	**string**	**“USD”**	**No**	**The currency that is used when specifying the product price.** **Currencies are configured in the platform interface: Market —> Settings —> Other. If the reques will occur.**
tags	**array**	[“camera”, “Canon”]	**No**	**List of tags.**
manufacturer_name	**string**	**“Canon”**	**No**	**Manufacturer’s name.**
description	**string**	**“Test description”**	**No**	**Product description.** **The length of the text is no more than 1000 characters.**
url	**string**	**“https://example.com”**	**No**	**URL of the product page on the store site.**
pictures	**array**	[“https://example_1.com”, “https://example_2.com”]	**No**	**List of URL links to the product images.**
count	**int**	**60**	**No**	**The number of products available for sale.**
expiration_time	**string**	**“730”**	**No**	**Shelf life or service life (in days).**
is_available	**bool**	**true**	**No**	**Whether the product is in stock or not.**
delivery	**bool**	**true**	**No**	**Possibility of courier delivery (for all regions to which the store delivers).**
barcode	**array**	[5235235, 347347, 335578769]	**No**	**List of product barcodes from the manufacturer.**
manufacturer_warranty	**bool**	**true**	**No**	**Manufacturer’s official warranty.**

custom_fields	**object**	{
		“color”: “black”,
		“weight”: 701,
		“dimensions”: “105*141*77”
		}
			No	**Custom product fields.**
				Learn more

regional_data	**array**	[
		{
		“external_id”:
		“region_YnCrJj8Y1”,
		“price”: “36000”,
		“old_price”: “38500”,
		“currency”: “RUB”,
		“custom_fields”:{
		“color”: “black”,
		“weight”: 701
		},
		“count”: 30,
		“is_available”: true,
		“delivery”: true
		}
		]
			No	**Product data that varies by region.**
				
				`external_id` **is the region ID. You can create a region and get its ID in the platform interface: Mar**

categories	**array**	[“category_1”, “category_2”]	**No**	**Product category ID.** **Learn more**

sku	**array**	[
		{
		“name”: “Canon EOS 90D Body”,
		“external_id”: “12345678”,
		}
		]
			No	**SKU data for this product.**
				Learn more

JSON

JSON

```
[
    {
        "name": "iPhone",
        "external_id": "123456",
        "price": "500",
        "old_price": "400",
        "currency": "USD",
        "tags": [
            "one",
            "two"
        ],
        "manufacturer_name": "Apple",
        "description": "Test description",
        "url": "https://example.com",
        "pictures": [
            "https://example.com",
            "https://example.com"
        ],
        "count": 60,
        "expiration_time": "730",
        "is_available": true,
        "delivery": true,
        "barcode": [
            5235235,
            347347,
            335578769
        ],
        "manufacturer_warranty": false,
        "custom_fields": {
            "product_field_id_1": 123,
            "product_field_id_2": "data"
        },
        "regional_data": [
            {
                "external_id": "region_YnCrJj8Y1",
                "price": "864825",
                "old_price": "400",
                "currency": "USD",
                "custom_fields": {
                    "product_field_id_1": 123,
                    "product_field_id_2": "data"
                },
                "count": 30,
                "is_available": true,
                "delivery": true
            }
        ],
        "categories": [
            "category_1",
            "category_2"
        ],
        "sku": [
            {
                "name": "iPhone",
                "external_id": "12345678",
            }
        ]
    }
]
```


XML

```xml
<xml>
  <row>
    <name>iPhone</name>
    <external_id>123456</external_id>
    <price>500</price>
    <old_price>400</old_price>
    <currency>USD</currency>
    <tags>one</tags>
    <tags>two</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <description>Test description</description>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>60</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>5235235</barcode>
    <barcode>347347</barcode>
    <barcode>335578769</barcode>
    <manufacturer_warranty>false</manufacturer_warranty>
    <custom_fields>
      <product_field_id_1>123</product_field_id_1>
      <product_field_id_2>data</product_field_id_2>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>864825</price>
      <old_price>400</old_price>
      <currency>USD</currency>
      <custom_fields>
        <product_field_id_1>123</product_field_id_1>
        <product_field_id_2>data</product_field_id_2>
      </custom_fields>
      <count>30</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
    <categories>category_1</categories>
    <categories>category_2</categories>
    <sku>
      <name>iPhone</name>
      <external_id>12345678</external_id>
    </sku>
  </row>
</xml>
```

SKU data object

Description

SKU data object used in SKU import requests.

Parameter	Type	Example	Required	Description
name	**string**	**“Canon EOS 90D Body”**	**Yes**	**SKU’s name.**
external_id	**string**	**“12345678”**	**Yes**	**SKU ID that is used when adding an item to an order.**
price	**string**	**“510”**	**No**	**Current price of the SKU.**
old_price	**string**	**“600”**	**No**	**Old price of the SKU.**
currency	**string**	**“USD”**	**No**	**The currency that is used when specifying the SKU price.** **Currencies are configured in the platform interface:** **Market —> Settings —> Other.** **If the request passes a currency that is not set in the platform, an error will o**
tags	**array**	[“camera”, “Canon”]	**No**	**List of tags.**
manufacturer_name	**string**	**“Canon”**	**No**	**Manufacturer’s name.**
description	**string**	**“test description”**	**No**	**SKU description.** **The length of the text is no more than 1000 characters.**
product_external_id	**string**	**“12345”**	**Yes**	**The ID of the product to which the SKU is linked.** **Must be the same for all SKUs of the same product.**
url	**string**	**“https://example.com”**	**No**	**URL of the SKU page on the store site.**
pictures	**array**	[“https://example_1.com”, “https://example_2.com”]	**No**	**List of URL links to the SKU images.**
count	**int**	**10**	**No**	**The number of SKU available for sale.**
expiration_time	**string**	**“730”**	**No**	**Shelf life or service life (in days).**
is_available	**bool**	**true**	**No**	**Whether the SKU is in stock or not.**
delivery	**bool**	**false**	**No**	**Possibility of courier delivery (for all regions to which the store delivers).**
barcode	**array**	[5235235, 347347, 335578769]	**No**	**List of SKU barcodes from the manufacturer.**
manufacturer_warranty	**bool**	**true**	**No**	**Manufacturer’s official warranty.**

custom_fields	**object**	{
		“color”: “black”,
		“weight”: 730,
		“dimensions”: “105*139*79”
		}
			No	**Custom SKU fields.**
				Learn more

regional_data	**array**	[
		{
		“external_id”: “region_YnCrJj8Y1”,
		“price”: “36000”,
		“old_price”: “38500”,
		“currency”: “RUB”,
		“custom_fields”:{
		“color”: “black”,
		“weight”: 701
		},
		“count”: 10,
		“is_available”: true,
		“delivery”: true
		}
		]
			No	**SKU data that varies by region.**
				
				`external_id` **is the region ID. You can create a region and get its ID in the platfo**

categories	**array**	[“category_1”, “category_2”]	**No**	**SKU category ID.** **Learn more**

JSON

JSON

```
[
  {
    "name": "iPhone",
    "external_id": "12345",
    "price": "235",
    "old_price": "400",
    "currency": "USD",
    "description": "",
    "product_external_id": "12345678",
    "categories": ["1234", "12345"],
    "tags": ["one", "two"],
    "manufacturer_name": "Apple",
    "url": "https://example.com",
    "pictures": ["https://example.com", "https://example.com"],
    "count": 50,
    "expiration_time": "730",
    "is_available": true,
    "delivery": true,
    "barcode": [23346346, 23546362, 76834566],
    "manufacturer_warranty": true,
    "custom_fields": {
      "sku_custom_field_1": 123,
      "sku_custom_field_2": "data"
    },
    "regional_data": [
      {
        "external_id": "region_YnCrJj8Y1",
        "price": "864825",
        "old_price": "400",
        "currency": "USD",
        "custom_fields": {
          "sku_custom_field_1": 123,
          "sku_custom_field_2": "data"
        },
        "count": 30,
        "is_available": true,
        "delivery": true
      }
    ]
  }
]
```


XML

```xml
<xml>
  <row>
    <name>iPhone</name>
    <external_id>12345</external_id>
    <price>235</price>
    <old_price>400</old_price>
    <currency>USD</currency>
    <description></description>
    <product_external_id>12345678</product_external_id>
    <categories>1234</categories>
    <categories>12345</categories>
    <tags>one</tags>
    <tags>two</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>50</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <sku_custom_field_1>123</sku_custom_field_1>
      <sku_custom_field_2>data</sku_custom_field_2>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>864825</price>
      <old_price>400</old_price>
      <currency>USD</currency>
      <custom_fields>
        <sku_custom_field_1>123</sku_custom_field_1>
        <sku_custom_field_2>data</sku_custom_field_2>
      </custom_fields>
      <count>30</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </row>
</xml>
```

Categories array

The `categories` array is used to create categories and subcategories in the market. Each object creates following requests:

- **Import products, SKUs and categories**
- **Import SKUs and categories**

Request parameters

Parameter	Type	Example	Required	
external_id	**string**	**“1234”**	**Yes**	**Category ID**
name	**string**	**“Cell phones and accessories”**	**Yes**	**Field name**
parent_external_id	**string**	**“123”**	**No**	**Main category ID. Passed if you create a s**

JSON

JSON

```
[
  {
        "external_id": "123",
        "name": "Test category",
  },
  {
        "external_id": "12345",
        "name": "Test subcategory",
        "parent_external_id": "123"
  }
]
```

XML

```
<xml>
<external_id>123</external_id>
    <name>Test category</name>
  </row>
  <row>
    <external_id>12345</external_id>
    <name>Test subcategory</name>
    <parent_external_id>123</parent_external_id>
</xml>
```

Custom fields array

The `custom_fields` array is used to create custom fields for various market entities (order, order item, pr each of which creates one custom field.

The array is passed in the following API requests:

- **Import products, SKUs and categories**
- **Import SKUs and categories**
- **Import order and item statuses**

Request parameters

Parameter	Type	Example	Required	D
external_id	**string**	**“1234”**	**Yes**	**Custom field ID**
name	**string**	**“color”**	**Yes**	**Custom field name**
entity_type	**string**	**“product”** **“sku”** **“order”** **“line_order”**	**Yes**	**Entity type**
required	**bool**	**false**	**Yes**	**Whether the field is required or not**
value_type	**string**	**“unicode”** **“int”** **“float”** **“bool”** **“enum”** **“date”**	**Yes**	**The type of data in the field**
default_value	**string**	**“black”**	**Yes**	**Default value of the custom field**
enum_values	**object**	`{"one": 1, "two": 2}`	**Yes,** **if value_type == enum**	**The value of each property in a custom field with ty**

JSON

JSON

```
[
  {
        "external_id": "12",
        "name": "product_custom_field",
        "entity_type": "product",
        "required": true,
        "value_type": "unicode",
        "default_value": "data"
  },
  {
        "external_id": "123",
        "name": "sku_custom_field",
        "entity_type": "sku",
        "required": false,
        "value_type": "int",
        "default_value": 100
   },
   {
        "external_id": "1234",
        "name": "order_custom_field",
        "entity_type": "order",
        "required": false,
        "value_type": "enum",
        "default_value": {"one": 1, "two": 2}
   },
   {
        "external_id": "1234",
        "name": "order_custom_field",
        "entity_type": "line_order",
        "required": false,
        "value_type": "bool",
        "default_value": false
   }
]
```

XML

```
<xml>
  <row>
    <external_id>12</external_id>
    <name>product_custom_field</name>
    <entity_type>product</entity_type>
    <required>true</required>
    <value_type>unicode</value_type>
    <default_value>data</default_value>
  </row>
  <row>
    <external_id>123</external_id>
    <name>sku_custom_field</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>int</value_type>
    <default_value>100</default_value>
  </row>
  <row>
    <external_id>1234</external_id>
    <name>order_custom_field</name>
    <entity_type>order</entity_type>
    <required>false</required>
    <value_type>enum</value_type>
    <default_value>
      <one>1</one>
      <two>2</two>
    </default_value>
  </row>
  <row>
    <external_id>1234</external_id>
    <name>order_custom_field</name>
    <entity_type>line_order</entity_type>
    <required>false</required>
    <value_type>bool</value_type>
    <default_value>false</default_value>
  </row>
</xml>
```

Import order and item statuses

Description

Imports an order, payment method and statuses of order items into the market.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/import`

Request parameters

Parameter	Type	Example	Required	Description
custom_fields	array of objects	[{ “external_id”: “1234”, “name”: “1234”, “entity_type”: “line_order”, “required”: false, “value_type”: “unicode”, “default_value”: “data” }]	No	Create custom fields for the following entities: • product • SKU • order • order item Learn more
custom_line_statuses	array of objects	[{ “external_id”: “1234”, “name”: “Arrived at the pickup point”, “type”: “in_process” }]	No	Custom status of the order item. It is necessary to pass the status id (`external_id`), its name (`name`) a • `delivered` • `canceled` • `refunded` • `in_process` Fields must be created before they can be used in an order.
skip_triggers	bool	true	No	Skip trigger events. By default – false
data	array of objects	[{ “external_id”: “123456”, “matching”:“email”, “email”:“example@example.com” “endpoint”: “endpoint_external_id”, “status”: “new”, “total_price”: “3000”, “currency”: “USD”, “create_time”: “2022-01-02T15:04:05.141Z”, “lines”: [{ “external_id”: “7777”, “status”: “new”, “product”: “product_external_id”, “base_price_per_item”: “600”, “min_price_per_item”: “300”, “final_price_per_item”: “500”, “count”: 5 }] }]	Yes	Order data. In this block, you can specify values for previously creat

TIP

If you need to set a value for an existing additional field, pass an object in `custom_fields` where the key and the value is the required data

Request example

JSON

```
{
  "custom_fields": [
{
"external_id": "order_field_id_123",
"name": "Additional order field",
"entity_type": "order",
"required": false,
      "value_type": "unicode",
      "default_value": "value of the test field"
    },
    {
"external_id": "line_field_id_123",
"name": "Additional field of the order item",
"entity_type": "line_order",
"required": false,
      "value_type": "unicode",
      "default_value": "value of the test field"
    }
  ],
  "custom_line_statuses": [
{
"external_id": "status_id_123",
"name": "Delivered to pick-up point",
"type": "in_process"
    }
  ],
  "skip_triggers": true,
  "data": [
    {
      "external_id": "1234000",
      "matching": "email",
      "email": "johndoe@example.com",
      "endpoint": "website",
      "tags": ["one", "two"],
      "status": "delivered",
      "delivery_cost": "100",
      "total_price": "1000",
      "currency": "USD",
      "create_time": "2022-01-02T15:04:05.141Z",
      "custom_fields": {
        "order_field_id_123": "data"
      },
      "region": "region_YnCrJj8Y1",
      "lines": [
        {
          "external_id": "line_01",
          "status": "status_id_123",
          "product": "iphone_test_1",
          "base_price_per_item": "900",
          "min_price_per_item": "900",
          "final_price_per_item": "900",
          "count": 1,
          "custom_fields": {
            "line_field_id_123": "data"
          }
        }
      ]
    }
  ]
}
```

XML

```
<xml>
  <custom_fields>
    <external_id>order_field_id_123</external_id>
    <name>Additional order field</name>
    <entity_type>order</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>value of the test field</default_value>
  </custom_fields>
  <custom_fields>
    <external_id>line_field_id_123</external_id>
    <name>Additional field of the order item</name>
    <entity_type>line_order</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>value of the test field</default_value>
  </custom_fields>
  <custom_line_statuses>
    <external_id>status_id_123</external_id>
    <name>Delivered to the pick-up point</name>
    <type>in_process</type>
  </custom_line_statuses>
  <skip_triggers>true</skip_triggers>
  <data>
    <external_id>1234000</external_id>
    <matching>email</matching>
    <email>johndoe@example.com</email>
    <endpoint>website</endpoint>
    <tags>one</tags>
    <tags>two</tags>
    <status>delivered</status>
    <delivery_cost>100</delivery_cost>
    <total_price>1000</total_price>
    <currency>USD</currency>
    <create_time>2022-01-02T15:04:05.141Z</create_time>
    <custom_fields>
      <order_field_id_123>data</order_field_id_123>
    </custom_fields>
    <region>region_YnCrJj8Y1</region>
    <lines>
      <external_id>line_01</external_id>
      <status>status_id_123</status>
      <product>iphone_test_1</product>
      <base_price_per_item>900</base_price_per_item>
      <min_price_per_item>900</min_price_per_item>
      <final_price_per_item>900</final_price_per_item>
      <count>1</count>
      <custom_fields>
        <line_field_id_123>data</line_field_id_123>
      </custom_fields>
    </lines>
  </data>
</xml>
```

INFO

`custom_fields` at the root of the request creates new fields (an array), while in `data` it fills them with va `"field_id": "value"`)

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Product IDs that were successfully imported**

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "result": [
    {
      "error": 0,
      "external_id": "1234000"
    }
  ]
}
```


XML


```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>1234000</external_id>
    </result>
</xml>
```

Get orders list

Description

Retrieves a list of all orders in an account or orders for a specific touchpoint.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/list`

Request parameters

Parameter	Type	Example	Required	Descrption
from_id	**string**	**“2oDantiaiPtY-b7b41276360564d4”**	**No**	**The pointer to the next page. It is taken from the previous request, or passed em**
limit	**int**	**3**	**No**	**Order list limit**
endpoint	**string**	**“website”**	**No**	**Touchpoint. Passed if it is necessary to get a list of orders for a specific touchpo**

Request example

JSON

```
{
  "from_id": "",
  "limit": 1,
  "endpoint": "qwerty"
}
```

XML

```
<xml>
  <from_id></from_id>
  <limit>1</limit>
  <endpoint>qwerty</endpoint>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "next_from_id": "2p2FXnb7zzoh-26167c2af5162ca4",
    "result": [
        {
            "profile": {
                "profile_id": "61addfc13f62fc8440i8b013",
                "xxh": "636gcaf5614536asf"
            },
            "external_id": "12345678-0001",
            "endpoint": "website",
            "status": "new",
            "delivery_cost": "50000",
            "total_price": "4549000",
            "currency": "RUB",
            "custom_fields": {},
            "lines": [
                {
                    "external_id": "line_DwWuNkAYV",
                    "product": "xiaomi_11T",
                    "base_price_per_item": "4499000",
                    "min_price_per_item": "0",
                    "final_price_per_item": "4499000",
                    "count": 1,
                    "custom_fields": {},
                    "currency": "RUB"
                }
            ],
            "create_time": "2021-12-27T09:38:39Z",
            "update_time": "2021-12-27T09:38:39Z"

        }
    ]
}
```


XML


```
<xml>
    <next_from_id>2p2FXnb7zzoh-26167c2af5162ca4</next_from_id>
    <result>
        <profile>
            <profile_id>61addfc13f62fc8440i8b013</profile_id>
            <xxh>636gcaf5614536asf</xxh>
        </profile>
        <external_id>12345678-0001</external_id>
        <endpoint>website</endpoint>
        <status>new</status>
        <delivery_cost>50000</delivery_cost>
        <total_price>4549000</total_price>
        <currency>RUB</currency>
        <lines>
            <external_id>line_DwWuNkAYV</external_id>
            <product>xiaomi_11T</product>
            <base_price_per_item>4499000</base_price_per_item>
            <min_price_per_item>0</min_price_per_item>
            <final_price_per_item>4499000</final_price_per_item>
            <count>1</count>
            <currency>RUB</currency>
        </lines>
        <create_time>2021-12-27T09:38:39Z</create_time>
        <update_time>2021-12-27T09:38:39Z</update_time>
    </result>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	Descrption
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the result contains 0 lines of history.**
result	**array**	**Array of received data on orders**

Delete order

Description

Delete an order in the market, including from the profile history (the request completely deletes the or
In the request, you need to use a token that is assigned a role with the appropriate access rules.

URL

Method: POST

`https://example.com/api/v1.1/market/orders/delete`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Not if external_id is passed**
external_id	**string**	**“A598t32D”**	**Not if id is passed**

Request example

JSON

```
{
        "id": 1
}

{
        "external_id": "A598t32D"
}
```

XML

```
<xml>
  <id>1</id>
</xml>

<xml>
  <external_id>"A598t32D"</external_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Get order status

Description

Get information of order status.
In the request, you need to use a token that is assigned a role with the appropriate access rules.

URL

Method: POST

`https://example.com/api/v1.1/market/orders/get_status`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Not if external_id is passed**
external_id	**string**	**“A598t32D”**	**Not if id is passed**

Request example JSON

JSON

```
{
        "id": 1
}

{
        "external_id": "A598t32D"
}
```

XML

```
<xml>
  <id>1</id>
</xml>

<xml>
  <external_id>"A598t32D"</external_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": {
        "endpoint": "marketplace",
        "status": "delivered"
    }
}
```

XML

```
<xml>
   <error>0</error>
   <error_text></error_text>
   <result>
      <endpoint>"marketplace"</endpoint>
      <status>"delivered"</status>
   </result>
</xml>
```

Response parameters

Parameter	Type	Description
result	**array**	**Array of received data on orders**
error	**int**	**Error code**
error_text	**string**	**Error text**

Update order line status

Description

The request updates the status of an order line in the Market.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/update_order_line_status`

Request parameters

Parameter	Type	Example	Required	
order_external_id	**string**	**“order_1”**	**Yes**	C
order_line_external_id	**string**	**“line_DBSrmlVYD”**	**Yes**	C
status_external_id	**string**	• `delivering` • `processing` • `new` • `refunded` • `canceled` • `delivered` • `<custom_status_id>`	**Yes**	C

Request example

JSON

```
{
    "format":"json",
    "order_external_id":"order_1",
    "order_line_external_id":"line_DBSrmlVYD",
    "status_external_id":"delivered"
}
```

XML

```
<xml>
  <format>json</format>
  <order_external_id>order_1</order_external_id>
  <order_line_external_id>line_DBSrmlVYD</order_line_external_id>
  <status_external_id>delivered</status_external_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "order_external_id": "order_1",
        "order_line_external_id": "line_DBSrmlVYD",
        "status_external_id": "delivered"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <order_external_id>order_1</order_external_id>
        <order_line_external_id>line_DBSrmlVYD</order_line_external_id>
        <status_external_id>delivered</status_external_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descrption
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**object**	**Updated order item data**

Import products, SKUs and categories

Description

Allows you to import 3 types of entities at the same time: products, SKUs and categories.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/import`

Request parameters

Parameter	Type	Example	Required	

custom_fields	array of objects	[
		{
		“external_id”: “1234”,
		“name”: “product_custom_field”,
		“entity_type”: “product”,
		“required”: false,
		“value_type”: “unicode”,
		“default_value”: “data”
		}
		]
			No	Creating custom fields for th
				• product
				• SKU
				• order
				• order item
				
				Learn more

categories	array of objects	[
		{
		“external_id”: “1234”,
		“name”: “Camera & Photo”,
		“parent_external_id”: “123”
		}
		]
			No	Creating categories and sub
				Learn more

data	array of objects	[
		{
		“name”: “iPhone”,
		“product_external_id”: “12345678”,
		“external_id”: “123456”
		}
		]
			Yes	Data about products and the

Request example

JSON

```
{
  "custom_fields": [
{
"external_id": "test_field_string",
"name": "String field",
"entity_type": "sku",
"required": false,
      "value_type": "unicode",
      "default_value": "Default value of the sku test field"
    }
  ],
  "categories": [
{
"external_id": "test_category_2",
"name": "Test category 2"
}
],
  "data": [
    {
      "name": "Apple iPhone 13 256GB",
      "external_id": "sku_test_1",
      "price": "1000",
      "old_price": "1100",
      "currency": "USD",
"description": "SKU description",
"product_external_id": "iphone_test_1",
"categories": ["test_category_2"],
      "tags": ["phone", "Apple"],
      "manufacturer_name": "Apple",
      "url": "https://example.com",
      "pictures": ["https://example.com", "https://example.com"],
      "count": 10,
      "expiration_time": "730",
      "is_available": true,
      "delivery": true,
      "barcode": [23346346, 23546362, 76834566],
      "manufacturer_warranty": true,
      "custom_fields": {
"test_field_string": "Test field"
      },
      "regional_data": [
{
"external_id": "region_YnCrJj8Y1",
"price": "75000",
"old_price": "80000",
"currency": "RUB",
"custom_fields": {
"test_field_string": "Test field"
          },
          "count": 8,
          "is_available": true,
          "delivery": true
        }
      ]
    }
  ]
}
```


XML

```
<xml>
  <custom_fields>
    <external_id>test_field_string</external_id>
    <name>String field</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>Default value of the sku test field</default_value>
  </custom_fields>
  <categories>
    <external_id>test_category_2</external_id>
    <name>Test category 2</name>
  </categories>
  <data>
    <name>Apple iPhone 13 256GB</name>
    <external_id>sku_test_1</external_id>
    <price>1000</price>
    <old_price>1100</old_price>
    <currency>USD</currency>
    <description>SKU description</description>
    <product_external_id>iphone_test_1</product_external_id>
    <categories>test_category_2</categories>
    <tags>phone</tags>
    <tags>Apple</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>10</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <test_field_string>Test field</test_field_string>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>75000</price>
      <old_price>80000</old_price>
      <currency>RUB</currency>
      <custom_fields>
        <test_field_string>Test field</test_field_string>
      </custom_fields>
      <count>8</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": [
        {
            "error": 0,
            "external_id": "sku_test_1"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>sku_test_1</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Product IDs that were successfully imported**

Import SKUs and categories

Description

Imports 2 types of entities at the same time: SKUs and categories.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/import`

Request parameters

Parameter	Type	Example	Required	

custom fields	array of objects	[
		{
		“external_id”: “1234”,
		“name”: “1234”,
		“entity_type”: “sku”,
		“required”: false,
		“value_type”: “unicode”,
		“default_value”: “data”
		}
		]
			No	Creating custom fields for the fol
				• product
				• SKU
				• order
				• order item
				Learn more

categories	array of objects	[
		{
		“external_id”: “1234”,
		“name”: “Camera & Photo”,
		“parent_external_id”: “123”
		}
		]
			No	Creating categories and subcate
				Learn more

data	array of objects	[
		{
		“name”: “iPhone”,
		“external_id”: “123456”
		}
		]
			Yes	SKU data
				Learn more

Request example

JSON

```
{
    "custom_fields": [
        {
          "external_id": "test_field_string",
          "name": "Custom field for SKU",
          "entity_type": "sku",
          "required": false,
          "value_type": "unicode",
          "default_value": "Default value of the SKU custom field"
        }
      ],
      "categories": [
        {
          "external_id": "test_category_2",
          "name": "Electronics"
        }
      ],
    "data": [
      {
        "name": "Apple iPhone 13 256GB",
        "external_id": "sku_test_1",
        "price": "1000",
        "old_price": "1100",
        "currency": "USD",
        "description":"SKU description",
        "product_external_id": "iphone_test_1",
        "categories": [
          "test_category_2"
        ],
        "tags": [
          "phone",
          "Apple"
        ],
        "manufacturer_name": "Apple",
        "url": "https://example.com",
        "pictures": [
          "https://example.com",
          "https://example.com"
        ],
        "count": 10,
        "expiration_time": "730",
        "is_available": true,
        "delivery": true,
        "barcode": [
          23346346,
          23546362,
          76834566
        ],
        "manufacturer_warranty": true,
        "custom_fields": {
          "test_field_string": "Test field value",
        },
        "regional_data": [
          {
            "external_id": "region_YnCrJj8Y1",
            "price": "75000",
            "old_price": "80000",
            "currency": "RUB",
            "custom_fields": {
                "test_field_string": "Test field value for this region",
            },
            "count": 8,
            "is_available": true,
            "delivery": true
          }
        ]
      }
    ]
  }
```


XML

```
<xml>
  <custom_fields>
    <external_id>test_field_string</external_id>
    <name>Custom field for SKU</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>Default value of the SKU custom field</default_value>
  </custom_fields>
  <categories>
    <external_id>test_category_2</external_id>
    <name>Electronics</name>
  </categories>
  <data>
    <name>Apple iPhone 13 256GB</name>
    <external_id>sku_test_1</external_id>
    <price>1000</price>
    <old_price>1100</old_price>
    <currency>USD</currency>
    <description>SKU description</description>
    <product_external_id>iphone_test_1</product_external_id>
    <categories>test_category_2</categories>
    <tags>phone</tags>
    <tags>Apple</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>10</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <test_field_string>Test field value</test_field_string>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>75000</price>
      <old_price>80000</old_price>
      <currency>RUB</currency>
      <custom_fields>
        <test_field_string>Test field value for this region</test_field_string>
      </custom_fields>
      <count>8</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "result": [
        {
          "error": 0,
          "external_id": "sku_test_1"
        }
  ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>sku_test_1</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**IDs of SKUs that were successfully imported**

Get products list

Description

Retrieves a list of products.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/list`

Request parameters

Parameter	Type	Example	Required	Description
from_id	string	“2oDantiaiPtY-b7b41276360564d4”	No	The pointer to the next page. It is taken from the previous request, or passed em
limit	int	3	No	Product list limit

Request example

JSON

```
{
  "from_id": "0014a2feb331ace2.20200124134127",
  "limit": 3
}
```

XML

```
<xml>
  <from_id>0014a2feb331ace2.20200124134127</from_id>
  <limit>3</limit>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": [
        {
            "name": "Xiaomi 11T",
            "external_id": "xiaomi_11T",
            "sku": [
                {
                    "name": "Xiaomi 11T 256ГБ",
                    "external_id": "xiaomi_11t_256gb",
                },
                {
                    "name": "Xiaomi 11T 128 ГБ",
                    "external_id": "xiaomi_11t_128gb",
                }
            ]
        },
        {
            "name": "Apple iPhone 13",
            "external_id": "iphone_13",
            "sku": [
                {
                    "name": "Apple iPhone 13 256GB",
                    "external_id": "iphone_13_256gb",
                },
                {
                    "name": "Apple iPhone 13 128GB",
                    "external_id": "iphone_13_128gb",
                }
            ]
        },
                  {
            "name": "Wireless headphones Xiaomi Mi True Pro",
            "external_id": "xiaomi_mi_true_pro",
        }
    ]
}
```


XML


```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <name>Xiaomi 11T</name>
        <external_id>xiaomi_11T</external_id>
        <sku>
            <name>Xiaomi 11T 256ГБ</name>
            <external_id>xiaomi_11t_256gb</external_id>
        </sku>
        <sku>
            <name>Xiaomi 11T 128 ГБ</name>
            <external_id>xiaomi_11t_128gb</external_id>
        </sku>
    </result>
    <result>
        <name>Apple iPhone 13</name>
        <external_id>iphone_13</external_id>
        <sku>
            <name>Apple iPhone 13 256GB</name>
            <external_id>iphone_13_256gb</external_id>
        </sku>
        <sku>
            <name>Apple iPhone 13 128GB</name>
            <external_id>iphone_13_128gb</external_id>
        </sku>
    </result>
    <result>
        <name>Wireless headphones Xiaomi Mi True Pro</name>
        <external_id>xiaomi_mi_true_pro</external_id>
    </result>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**

Parameter	Type	Description
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the result contains 0 lines of history.**
result	**array**	**Array of received data on products and their SKUs**

Get SKUs list

Description

Retrieves a list of SKUs.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/list`

Request parameters

Parameter	Type	Example	Required	Description
from_id	**string**	**“2oDantiaiPtY-b7b41276360564d4”**	**No**	**The pointer to the next page. It is taken from the previous request, or passed em**
limit	**int**	**3**	**No**	**SKU list limit**

Request example

JSON

```
{
  "from_id": "",
  "limit": 3
}
```

XML

```
<xml>
    <from_id></from_id>
    <limit>3</limit>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "next_from_id": "L8PSdcsMmLq-6021b5621680598b",
    "result": [
        {
            "name": "Xiaomi 11T 256GB",
            "external_id": "xiaomi_11t_256gb"
        },
        {
            "name": "Xiaomi 11T 128GB",
            "external_id": "xiaomi_11t_128gb"
        },
        {
            "name": "Apple iPhone 13 256GB",
            "external_id": "iphone_13_256gb"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <next_from_id>L8PSdcsMmLq-6021b5621680598b</next_from_id>
    <result>
        <name>Xiaomi 11T 256GB</name>
        <external_id>xiaomi_11t_256gb</external_id>
    </result>
    <result>
        <name>Xiaomi 11T 128 GB</name>
        <external_id>xiaomi_11t_128gb</external_id>
    </result>
    <result>
        <name>Apple iPhone 13 256GB</name>
        <external_id>iphone_13_256gb</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the result contains 0 lines of history.**
result	**array**	**An array of received data on SKUs**

Delete products

Description

Delete products in the marketplace. The method supports both single deletion and batch deletion of m
To delete, you must use a token for which a role with the appropriate access rights is assigned.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/delete`

Request parameters

Parameter	Type	Example	Required	
ids	**array of integers**	**[1, 2, 3]**	**No, if `external_ids` is provided**	**Array o**
external_ids	**array of strings**	**[“product_1”, “product_2”]**	**No, if `ids` is provided**	**Array o**

Request example

Single JSON

Identification by parameter: `"ids"`:

```
{
        "ids": [1]
}
```

Identification by parameter: `"external_ids"`:

```
{
        "external_ids": ["product_123"]
}
```

Batch JSON

```
{
        "ids": [1, 2],
        "external_ids": ["product_3"]
}
```

Single XML

Identification by parameter: `"ids"`:

```
<xml>
  <ids>1</ids>
</xml>
```

Identification by parameter: `"external_ids"`:

```
<xml>
  <external_ids>product_123</external_ids>
</xml>
```

Batch XML

```
<xml>
  <ids>1</ids>
  <ids>2</ids>
  <external_ids>product_3</external_ids>
</xml>
```

Response example

JSON

```
{
    "deleted_count": 2,
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <deleted_count>2</deleted_count>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter	Type	Description
deleted_count	**int**	**Number of actually deleted products**
error	**int**	**Error code**
error_text	**string**	**Error text**

Delete SKU

Description

Delete SKUs (stock keeping units) in the marketplace. The method supports both single deletion and b one request.

To delete, you must use a token for which a role with the appropriate access rights is assigned.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/delete`

Request parameters

Parameter	Type	Example	Required	
ids	**array of integers**	**[1, 2, 3]**	**No, if `external_ids` is provided**	**Array o**
external_ids	**array of strings**	**[“sku_1”, “sku_2”]**	**No, if `ids` is provided**	**Array o**

Request example

Single JSON

Identification by parameter: `"ids"`:

```
{
        "ids": [1]
}
```

Identification by parameter: `"external_ids"`:

```
{
        "external_ids": ["sku_123"]
}
```

Batch JSON

```
{
        "ids": [1, 2],
        "external_ids": ["sku_3"]
}
```

Single XML

Identification by parameter: `"ids"`:

```
<xml>
  <ids>1</ids>
</xml>
```

Identification by parameter: `"external_ids"`:

```
<xml>
  <external_ids>sku_123</external_ids>
</xml>
```

Batch XML

```
<xml>
  <ids>1</ids>
  <ids>2</ids>
  <external_ids>sku_3</external_ids>
</xml>
```

Response example

JSON

```
{
    "deleted_count": 2,
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <deleted_count>2</deleted_count>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter	Type	Description
deleted_count	**int**	**Number of actually deleted SKUs**
error	**int**	**Error code**
error_text	**string**	**Error text**

Get summary report

Description

Retrieves a summary report.

Request URL

Method: POST

`https://example.com/api/v1.1/reports/summary`

Request parameters

Parameter	Type	Example	Required	Descript
resource_id	**int**	**1**	**No**	**resource ID**
date_from	**string**	**“2018-08-10”**	**Yes**	**Time period start date**
date_to	**string**	**“2018-08-15”**	**Yes**	**Time period end date**
data_type	**string**	**“segment”**	**Yes**	**Audience source type**
data_id	*	**1**	**No**	**Audience source ID**
data_group	**string**	**“day”**	**No**	**Data grouping parameter (“day” , “week” or “month”)**
show_deleted	**bool**	**false**	**No**	**Show deleted entries data**
is_test_data	**bool**	**false**	**No**	**Show test and watchers data**

Report sources:

`data_type`	Description	`data_id` parameter available	
campaign	**Mailing**	**Yes**	**int**
list	**Database**	**Yes**	**int**
segment	**Segment**	**Yes**	**int**
message	**Message template**	**Yes**	**int**
sender	**Sender**	**Yes**	**int**
domaingroup	**Email domain group**	**Yes**	**string**
domain	**Email domain**	**No**	**-**
vendor	**Vendor**	**No**	**-**
day	**Daily report**	**No**	**-**
week	**Weekly report**	**No**	**-**
month	**Monthly report**	**No**	**-**
environment	**Mobile or desktop traffic**	**No**	**-**

Request example

JSON

```
{
    "data_group": "vendor",
    "data_id": 7,
    "data_type": "list",
    "date_from": "2024-04-04",
    "date_to": "2024-05-05",
    "format": "json"
}
```

XML

```
<xml>
  <data_group>vendor</data_group>
  <data_id>7</data_id>
  <data_type>list</data_type>
  <date_from>2024-04-04</date_from>
  <date_to>2024-05-05</date_to>
  <format>xml</format>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "clicks": 22,
            "complains": 2,
            "confirms": 7,
            "delivs": 22,
            "hbounces": 3,
            "id": "Vendor #2",
            "opens": 44,
            "reads": 2,
            "sbounces": 15,
            "sends": 41,
            "suppresses": 1,
            "uclicks": 15,
            "unsubs": 3,
            "uopens": 22
        },
        {
            "clicks": 34,
            "complains": 4,
            "confirms": 13,
            "delivs": 24,
            "hbounces": 7,
            "id": "Vendor #1",
            "opens": 48,
            "reads": 3,
            "sbounces": 21,
            "sends": 31,
            "suppresses": 3,
            "uclicks": 21,
            "unsubs": 6,
            "uopens": 24
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <clicks>22</clicks>
    <complains>2</complains>
    <confirms>7</confirms>
    <confirmsubs>0</confirmsubs>
    <delivs>22</delivs>
    <hbounces>3</hbounces>
    <id>Vendor #2</id>
    <opens>44</opens>
    <reads>2</reads>
    <sbounces>15</sbounces>
    <sends>41</sends>
    <suppresses>1</suppresses>
    <uclicks>15</uclicks>
    <unsubs>3</unsubs>
    <uopens>22</uopens>
  </data>
  <data>
    <clicks>34</clicks>
    <complains>4</complains>
    <confirms>13</confirms>
    <confirmsubs>0</confirmsubs>
    <delivs>24</delivs>
    <hbounces>7</hbounces>
    <id>Vendor #1</id>
    <opens>48</opens>
    <reads>3</reads>
    <sbounces>21</sbounces>
    <sends>31</sends>
    <suppresses>3</suppresses>
    <uclicks>21</uclicks>
    <unsubs>6</unsubs>
    <uopens>24</uopens>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Descriptic
error	int	Error code
error_text	string	Error text
id	string	Report data source ID
clicks	int	Email clicks
complains	int	FBL complaints
opens	int	Opened emails
reads	int	Emails read
replies	int	Email replies
sends	int	Email sends
confirms	int	Confirmed emails
delivs	int	Email deliveries
undelivs	int	Undelivered emails
unsubs	int	Opt-outs
uopens	int	Email unique opens
ureads	int	Email unique reads
offence	int	Communication policy offences
glanceds	int	Messages glanced
hbounces	int	Email hard bounces
sbounces	int	Email soft bounces
suppresses	int	Suppressed messages
uclicks	int	Email unique clicks
ulclicks	int	Unique email opt-out clicks
sends_push	int	Push sends
clicks_push	int	Push clicks
uclicks_push	int	Push unique clicks
delivs_push	int	Push deliveries
ureads_push	int	Push unique reads
opens_push	int	Push opens
uopens_push	int	Push unique opens
push_ctr	int	Push CTR
sbounce_push	int	Soft bounce push
hbounce_push	int	Hard bounce push
sends_sms	int	SMS sends
clicks_sms	int	SMS clicks
uclicks_sms	int	SMS unique clicks
delivs_sms	int	SMS deliveries
undelivs_sms	int	Undelivered SMS
sms_ctr	int	SMS CTR
unsubs_sms	int	SMS opt-out rate
open_rate	int	Opens rate
push_deliv_rate	int	Push deliveries rate
u_read_rate	int	Unique read rate
u_open_click_rate	int	Unique opens and clicks rate
sb_rate	int	Soft bounce rate
email_read_rate	int	Emails read rate
delivery_rate	int	Deliveries rate
email_unsub_rate	int	Email opt-out rate
email_u_read_rate	int	Email unique rate
open_click_rate	int	CTOR
suppresses_rate	int	Suppressed messages rate
unsub_rate	int	Opt-out rate
u_click_rate	int	Unique clicks rate
click_rate	int	Clicks rate
sms_deliv_rate	int	SMS deliveries rate
hb_rate	int	Hard bounce rate
push_open_rate	int	Push open rate
undelivs_push	int	Undelivered push messages
complain_rate	int	FBL complains rate
u_open_rate	int	Unique opens rate
email_reply_rate	int	Reply rate im emails
sends_total	int	Total sends
delivs_total	int	Total delivered
undelivs_total	int	Total undelivered
reads_total	int	Total reads

Parameter	Type	Descriptic
clicks_total	**int**	**Total clicks**
uclicks_total	**int**	**Total unique clicks**
opens_total	**int**	**Total unique opens**
ureads_total	**int**	**Total unique reads**
uopens_total	**int**	**Total unique opens**

Get soft bounces report

Description

Retrieves email soft bounces report.
Email providers return error codes from 400 to 499. This marks a soft bounce: a message cannot be te

Request URL

Method: POST

`https://example.com/api/v1.1/reports/bounces`

Request parameters

Parameter	Type	Example	Required	
date_from	**string**	**“2018-08-10”**	**Yes**	**Report st**
date_to	**string**	**“2018-08-15”**	**Yes**	**Report en**
campaign_id	**int**	**1**	**No**	**Mailing ID**
group_by	**string**	**“from_domain”**	**No**	**Group res**
top	**int**	**2**	**No**	**Top X sof**

`group_by` **parameter possible values:**

- `ip` **- group by sender IP address**
- `from_domain` **– group by from domain**

Request example

JSON

```
{
    "campaign_id": 116,
    "date_from": "2024-04-26",
    "date_to": "2024-05-01",
    "format": "json",
    "top": 2
}
```

XML

```
<xml>
    <campaign_id>116</campaign_id>
    <date_from>2024-04-26</date_from>
    <date_to>2024-05-01</date_to>
    <format>xml</format>
    <top>2</top>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "Your message was not delivered because the other user mailbox was not available",
            "to_domain": "example.com"
        },
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather becaus
            "to_domain": "jetwire.name"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>Your message was not delivered because the other user mailbox was not available</bounce_text>
        <to_domain>example.com</to_domain>
    </data>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a t
        <to_domain>jetwire.name</to_domain>
    </data>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
bounce_text	**string**	**Email provider response**
count	**int**	**Email provider response count**
data_id	**string**	**Data source ID**
to_domain	**string**	**Domain**

Get undeliveries report

Description

Retrieves undelivered messages report.

Request URL

Method: POST

`https://example.com/api/v1.1/reports/undelivered`

Request parameters

Parameter	Type	Example	Required	
date_from	**string**	**“2018-08-10”**	**Yes**	**Report**
date_to	**string**	**“2018-08-15”**	**Yes**	**Report**
campaign_id	**int**	**1**	**No**	**Mailing**
group_by	**string**	**“from_domain”**	**No**	**Group d**
top	**int**	**2**	**No**	**Top X u**

`group_by` **possible values:**

- `ip` **– group by sender IP address**
- `from_domain` **– group by from domain**

Request example

JSON

```
{
    "campaign_id": 116,
    "date_from": "2024-04-26",
    "date_to": "2024-05-01",
    "format": "json",
    "top": 2
}
```

XML

```
<xml>
    <campaign_id>116</campaign_id>
    <date_from>2024-04-26</date_from>
    <date_to>2024-05-01</date_to>
    <format>json</format>
    <top>2</top>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather because
            "to_domain": "example.com"
        },
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather because
            "to_domain": "jetwire.name"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a t
        <to_domain>example.com</to_domain>
    </data>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a t
        <to_domain>jetwire.name</to_domain>
    </data>
</xml>
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
bounce_text	**string**	**Email provider response**
count	**int**	**Email provider response count**
data_id	**string**	**Data source ID**
to_domain	**string**	**Domain**

Get senders list

Description

Retrieves a list of senders available for the current account.

Request URL

Method: POST

`https://example.com/api/v1.1/senders/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	**2**	**No**	**The first sender on the list ID. By default – 1**
limit	**int**	**10**	**No**	**Senders list limit. By default – 1000**

Request example

JSON

```
{
    "limit": 2
}
```

XML

```
<xml>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "id": 1,
            "name": "AKMTA sender",
            "type": "akmta"
        },
        {
            "channel": "sms",
            "id": 2,
            "name": "Bulk SMS sender",
            "type": "bulksms_com"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 3,
    "total_count": 2
}
```

XML

```
<xml>
    <data>
        <channel>email</channel>
        <id>3</id>
        <name>Email sender</name>
        <type>akmta</type>
    </data>
    <data>
        <channel>sms</channel>
        <id>2</id>
        <name>Bulk SMS sender</name>
        <type>bulksms_com</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>5</next_from_id>
    <total_count>5</total_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result output contains 0 history lines**
id	**int**	**Sender ID**
name	**string**	**Sender name**
type	**string**	**Sender type**
channel	**string**	**Communication channel**

Get virtual senders list

Description

Retrieves a list of virtual senders.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/list`

Request parameters

Parameter	Type	Example	Required	Descr
from_id	**int**	**1**	**No**	**The first virtual sender on the list ID. By default – 1**
limit	**int**	**2**	**No**	**Virtual senders list limit. By default – 1000**

Request example

JSON

```
{
    "limit": 2
}
```

XML

```
<xml>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "email_channel": true,
            "id": 1,
            "name": "AKMTA virtual sender",
            "sms_channel": false
        },
        {
            "email_channel": false,
            "id": 2,
            "name": "Bulk SMS virtual sender",
            "sms_channel": true
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 2
}
```

XML

```
<xml>
<data>
    <email_channel>true</email_channel>
    <id>1</id>
    <name>AKMTA virtual sender</name>
    <sms_channel>false</sms_channel>
  </data>
  <data>
    <email_channel>false</email_channel>
    <id>2</id>
    <name>Bulk SMS virtual sender</name>
    <sms_channel>true</sms_channel>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>2</total_count>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
name	**string**	**Virtual sender name**
sms_channel	**bool**	**true, if SMS channel is active**
email_channel	**bool**	**true, if email channel is active**

Get virtual sender information

Description

Retrieves a virtual sender detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/get`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Virtual sender ID**

Request example

JSON

```
{
    "id": 1
}
```

XML

```
<xml>
    <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 1,
        "name": "Virtual sender",
        "email_rules": [
            {
                "type": "default",
                "value": "",
                "senders": [
                    {
                        "from_email": "",
                        "replyto_email": "",
                        "from_name": "",
                        "sender_id": 3,
                        "percent": 100
                    }
                ]
            }
        ],
        "sms_rules": [],
        "custom_channels": {}
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
    <id>1</id>
    <name>Virtual sender</name>
    <email_rules>
      <type>default</type>
      <value></value>
      <senders>
        <from_email></from_email>
        <replyto_email></replyto_email>
        <from_name></from_name>
        <sender_id>3</sender_id>
        <percent>100</percent>
      </senders>
    </email_rules>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	D
error	**int**	**Error type**
error_text	**string**	**Error text**
id	**int**	**Virtual sender ID**
name	**string**	**Virtual sender name**
email_rules	**JSON array**	**Rules for email senders**
sms_rules	**JSON array**	**Rules for SMS senders**

Clone virtual sender

Description

Creates a copy of an existing virtual sender.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/clone`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Existing virtual sender ID**
name	**string**	**“Clone of ...”**	**No**	**New virtual sender name. By default – “Clone of ...”**

Request example

JSON

```
{
    "name": "Copy of virtual sender #99",
    "id": 3
}
```

XML

```
<xml>
  <name>Copy of virtual sender #99</name>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 6
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>7</saved_id>
    </data>
</xml>
```

Response parameters

Parameter	Type	Descriptio
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New virtual sender ID**

Create virtual sender

Description

Creates a new virtual sender.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/add`

Request parameters

Parameter	Type	Example	Required	
name	**string**	**“New virtual sender”**	**Yes**	**New virtual**
email_rules	**JSON array**		**Yes**	**Rules for en**
sms_rules	**JSON array**		**Yes**	**Rules for SM**

Request example

JSON

```
{
    "name": "New virtual sender",
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ],
    "sms_rules": []
}
```

XML

```
<xml>
    <name>New virtual sender</name>
    <email_rules array ='true'>
        <senders array = 'true'>
            <percent>100</percent>
            <sender_id>3</sender_id>
        </senders>
        <type>default</type>
    </email_rules>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 8
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>9</saved_id>
    </data>
</xml>
```

Response parameters

Parameter	Type	Descriptio
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New virtual sender ID**

Update virtual sender

Description

Updates a virtual sender, changes settings.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/update`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Virtual s**
name	**string**	**“New virtual sender”**	**Yes**	**Virtual s**
email_rules	**JSON array**		**Yes**	**Rules for**
sms_rules	**JSON array**		**Yes**	**Rules for**

Request example

JSON

```
{
    "name": "Virtual sender",
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ],
    "id": 9,
    "sms_rules": []
}
```

XML

```
<xml>
  <name>Virtual sender</name>
  <email_rules array='true'>
    <senders array='true'>
      <from_email></from_email>
      <from_name></from_name>
      <percent>100</percent>
      <replyto_email></replyto_email>
      <sender_id>3</sender_id>
    </senders>
    <type>default</type>
    <value></value>
  </email_rules>
  <id>9</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 9
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Virtual sender ID**

Delete virtual sender

Description

Deletes a virtual sender entry.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/delete`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**virtual sender ID**

Request example

JSON

```
{
    "id": 12
}
```

XML

```
<xml>
  <id>11</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Add segmentation query

Description

Add segmentation query to external datatables.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/add/`

Request parameters

Parameter	Type	Example	Required	Description
name	string	"Main query"	Yes	Query name
shortname	string	"query1"	Yes	Query shortname
description	string	"query to the DB"	No	Query description
connector_id	int	2	Yes	External DB ID
column	string	"author"	Yes	Column used to search for profiles
sql	string	"SELECT * FROM table"	No	SQL query
cache_time_min	int	35	No	Result caching time
groups_id	\[\]int	\[0, 1 \]	No	List of group IDs to search for default – \[\]
parameters	array	[{ "name": "CITY", "type": "string", "title": "head", "options_type": "query" }]	No	Parameters of user variables used in SQL quer

Request example

JSON

```
{
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1,
   "column": "price"
}
```

XML

```
<xml>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
  <column>price</column>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Created query ID**

Update segmentation query

Description

Update segmentation query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/update`

Request parameters

Parameter	Type	Example	Required	Description
id	int	1	Yes	Query ID
name	string	"Main query"	Yes	Query name
shortname	string	"query1"	Yes	Query shortname
description	string	"query to the DB"	No	Query description
connector_id	int	2	Yes	External DB ID
column	string	"author"	Yes	Column used to search for profiles
sql	string	"SELECT * FROM table"	No	SQL query
cache_time_min	int	35	No	Result caching time
groups_id	\[\]int	\[0, 1 \]	No	List of group IDs to search for default – \[\
parameters	object	{ "name": "CITY", "type": "string", "title": "head", "options_type": "query" }	No	Parameters of user variables used in SQL que

Request example

JSON

```
{
   "id": 1,
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1,
   "column": "price"
}
```

XML

```
<xml>
  <id>1</id>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
  <column>price</column>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	Des
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Updated query ID**

Get segmentation query information

Description

Retrieves detailed segmentation query information.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/get`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes, if `shortname` is not specified**
shortname	**string**	**“query1”**	**Yes, if `id` is not specified`**

Request example

JSON

```
{
  "id": 2
}
```

XML

```
<xml>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1,
        "parametrs": {
           "name": "variable1",
           "title": "var",
           "default_value": "low"
        },
        "cache_time_min": 35
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <cache_time_min>35</cache_time_min>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <parametrs>
         <default_value>low</default_value>
         <name>variable1</name>
         <title>var</title>
      </parametrs>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response example

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Query ID**
name	**string**	**Query name**
shortname	**string**	**Query shortname**
description	**string**	**Query description**
connector_id	**int**	**External DB ID**
column	**string**	**Column used to search for profiles**
sql	**string**	**SQL query**
cache_time_min	**int**	**35**
groups_id	**[]int**	**List of group IDs to search for** **default – []**
parameters	**object**	**Parameters of user variables used in SQL queries**

Get segmentation queries list

Description

Retrieves segmentation queries list.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	`1`	**No**	**First mailing in the By default – 1**
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access grou**
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access grou**

Request example

JSON

```
{}
```

XML

```
<xml>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1,
        "parametrs": {
           "name": "variable1",
           "title": "var",
           "default_value": "low"
        },
        "cache_time_min": 35
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <cache_time_min>35</cache_time_min>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <parametrs>
         <default_value>low</default_value>
         <name>variable1</name>
         <title>var</title>
      </parametrs>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Query ID**
name	**string**	**Query name**
shortname	**string**	**Query shortname**
description	**string**	**Query description**
connector_id	**int**	**External DB ID**
column	**string**	**Column used to search for profiles**
sql	**string**	**SQL query**
cache_time_min	**int**	**35**
groups_id	**[]int**	**List of group IDs to search for** **default – []**
parameters	**object**	**Parameters of user variables used in SQL queries**

Delete segmentation query

Description

Delete segmentation query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/delete`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes**

Request example

JSON

```
{
  "id": 2
}
```

XML

```
<xml>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation",
}
```

XML

```
<xml>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

Add template query

Description

Add template query to external datatables.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/add/`

Request parameters

Parameter	Type	Example	Required	
name	**string**	**“Main query”**	**Yes**	**Query name**
shortname	**string**	**“query1”**	**Yes**	**Query shortname**
description	**string**	**“query to the DB”**	**No**	**Query description**
connector_id	**int**	**2**	**Yes**	**External DB ID**
sql	**string**	**“SELECT * FROM table”**	**No**	**SQL query**
groups_id	**[]int**	**[0, 1]**	**No**	**List of group IDs to se default – []**

Request example

JSON

```
{
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1
}
```

XML

```
<xml>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	De
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Created query ID**

Update template query

Description

Update template query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/update`

Request parameters

Parameter	Type	Example	Required	
id	**int**	**1**	**Yes**	**Query ID**
name	**string**	**“Main query”**	**Yes**	**Query name**
shortname	**string**	**“query1”**	**Yes**	**Query shortname**
description	**string**	**“query to the DB”**	**No**	**Query description**
connector_id	**int**	**2**	**Yes**	**External DB ID**
sql	**string**	**“SELECT * FROM table”**	**No**	**SQL query**
groups_id	**[]int**	**[0, 1]**	**No**	**List of group IDs to se default – []**

Request example

JSON

```
{
   "id": 1,
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1
}
```

XML

```
<xml>
  <id>1</id>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	Des
error	**int**	**Error code**

Parameter	Type	Des
error_text	**string**	**Error text**
saved_id	**int**	**Updated query ID**

Get template query information

Description

Retrieves detailed template query information.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/get`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes, if `shortname` is not specified**
shortname	**string**	**“query1”**	**Yes, if `id` is not specified`**

Request example

JSON

```
{
  "id": 2
}
```

XML

```
<xml>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response example

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Query ID**
name	**string**	**Query name**
shortname	**string**	**Query shortname**

Parameter	Type	Description
description	**string**	**Query description**
connector_id	**int**	**External DB ID**
sql	**string**	**SQL query**
groups_id	**[]int**	**List of group IDs to search for default – []**

Get template queries list

Description

Retrieves template queries list.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/list`

Request parameters

Parameter	Type	Example	Required	
from_id	**int**	`1`	**No**	**First mailing in the By default – 1**
groups	**[]String**	`[ "group1", "group2" ]`	**No**	**List of access grou**
groups_id	**[]int**	`[ 0, 1 ]`	**No**	**List of access grou**

Request example

JSON

```
{}
```

XML

```
<xml>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Query ID**
name	**string**	**Query name**
shortname	**string**	**Query shortname**
description	**string**	**Query description**

Parameter	Type	Description
connector_id	**int**	**External DB ID**
sql	**string**	**SQL query**
groups_id	**[]int**	**List of group IDs to search for** **default – []**

Delete template query

Description

Delete template query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/delete`

Request parameters

Parameter	Type	Example	Required
id	**int**	**1**	**Yes**

Request example

JSON

```
{
  "id": 2
}
```

XML

```
<xml>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation",
}
```

XML

```
<xml>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**

AKMTA object

JSON

JSON

```
{
    "dkim_id": 4,
    "dkim_selector": "dkim_selector",
    "ips": [
        {
            "ip": "24.24.24.34",
            "node": 10
        }
    ],
    "lock_rules_obj": {
        "common_rules": [],
        "isp_rules": [
            {
                "isps": [
                    6
                ],
                "rules": [
                    {
                        "patterns": [
                            1004,
                            1005,
                            497
                        ],
                        "rule": 1001
                    }
                ]
            }
        ]
    },
    "redirect_email": "user@example.com",
    "retry_rules_obj": {
        "common_rules": [
            {
                "patterns": [
                    1003
                ],
                "rule": 1002
            }
        ],
        "isp_rules": [
            {
                "isps": [
                    1007,
                    1008,
                    1009,
                    1010,
                    1011,
                    1012
                ],
                "rules": [
                    {
                        "patterns": [
                            1003
                        ],
                        "rule": 1002
                    }
                ]
            }
        ]
    },
    "sender_dkim_id": 8,
    "sender_dkim_selector": "sender_dkim_selector",
    "sending_rules_obj": {
        "common_rules": [
            {
                "isps": [
                    0
                ],
                "strategy": 1001
            }
        ],
        "ip_rules": [
            {
                "ips": [
                    "24.24.24.34"
                ],
                "rules": [
                    {
                        "isps": [
                            1007,
```

```
                        1009
                    ],
                    "strategy": 1002
                },
                {
                    "isps": [
                        1008,
                        1010
                    ],
                    "strategy": 1003
                }
            ]
        }
    ]
},
"users": [
    {
        "domain": "mx.localtest.ak",
        "pop3_pass": "nnr614g5",
        "redirect_email": "user1@example.com"
    },
    {
        "domain": "mx.localtest.ak",
        "pop3_pass": "nnr614g5",
        "redirect_email": "user2@example.com"
    }
]
}
```

XML

```
<xml>
    <dkim_id>4</dkim_id>
    <dkim_selector>dkim_selector</dkim_selector>
    <ips array='true'>
        <ip>24.24.24.34</ip>
        <node>10</node>
    </ips>
    <lock_rules_obj>
        <common_rules array='true'></common_rules>
        <isp_rules array='true'>
            <isps array='true'>6</isps>
            <rules array='true'>
                <patterns>1004</patterns>
                <patterns>1005</patterns>
                <patterns>497</patterns>
                <rule>1001</rule>
            </rules>
        </isp_rules>
    </lock_rules_obj>
    <redirect_email>user@example.com</redirect_email>
    <retry_rules_obj>
        <common_rules array='true'>
            <patterns array='true'>1003</patterns>
            <rule array='true'>1002</rule>
        </common_rules>
        <isp_rules array='true'>
            <isps>1007</isps>
            <isps>1008</isps>
            <isps>1009</isps>
            <isps>1010</isps>
            <isps>1011</isps>
            <isps>1012</isps>
            <rules array='true'>
                <patterns array='true'>1003</patterns>
                <rule>1002</rule>
            </rules>
        </isp_rules>
    </retry_rules_obj>
    <sender_dkim_id>8</sender_dkim_id>
    <sender_dkim_selector>sender_dkim_selector</sender_dkim_selector>
    <sending_rules_obj>
        <common_rules array='true'>
            <isps>0</isps>
            <strategy>1001</strategy>
        </common_rules>
        <ip_rules array='true'>
            <ips array='true'>24.24.24.34</ips>
            <rules array='true'>
                <isps>1007</isps>
                <isps>1009</isps>
                <strategy>1002</strategy>
            </rules>
            <rules array='true'>
                <isps>1008</isps>
                <isps>1010</isps>
                <strategy>1003</strategy>
            </rules>
        </ip_rules>
    </sending_rules_obj>
    <users>
        <domain>mx.localtest.ak</domain>
        <pop3_pass>nnr614g5</pop3_pass>
        <redirect_email>user1@example.com</redirect_email>
    </users>
    <users>
        <domain>mx.localtest.ak</domain>
        <pop3_pass>nnr614g5</pop3_pass>
        <redirect_email>user2@example.com</redirect_email>
    </users>
</xml>
```

Description

Field	
ips	**JSON array**
dkim_id	**int**

Field	
dkim_selector	**string**
sender_dkim_id	**int**
sender_dkim_selector	**string**
redirect_email	**string**
sending_rules_obj	**JSON Object**
retry_rules_obj	**JSON Object**
lock_rules_obj	**JSON Object**
users	**JSON array**

Content object

Content objects are used in mailing and scenarios queries.

JSON

JSON

```
{
  "subject": "Maybe very important message",
  "from_email": "",
  "from_name": "James",
  "replyto_email": "",
  "replyto_name": "",
  "message_id": 45
}
```

XML

```
<xml>
  <subject>Maybe very important message</subject>
  <from_email></from_email>
  <from_name>James</from_name>
  <replyto_email></replyto_email>
  <replyto_name></replyto_name>
  <message_id>45</message_id>
</xml>
```

Object parameters

Parameters	Type	Example	
message_id	**int**	**12**	**Message ID**
from_email	**string**	**“test@example.com”**	**Override from email**
from_name	**string**	**“Jenny’s List”**	**Override from name**
replyto_email	**string**	**“test@example.com”**	**Override reply to email**
replyto_name	**string**	**“Jenny”**	**Override reply to name**
subject	**string**	**“Weather alert”**	**Override message subject**

Email rule object

JSON

JSON

```
{
  "type": "from_domain",
  "value": "altcraft.com",
  "senders": [
    {
      "sender_id": 287,
      "percent": 100,
      "from_email": "no-reply@altcraft.com",
      "replyto_email": "no-reply@altcraft.com",
      "from_name": ""
    }
  ]
}
```

XML

```
<xml>
  <type>from_domain</type>
  <value>altcraft.com</value>
  <senders array='true'>
    <sender_id>287</sender_id>
    <percent>100</percent>
    <from_email>no-reply@altcraft.com</from_email>
    <replyto_email>no-reply@altcraft.com</replyto_email>
    <from_name></from_name>
  </senders>
</xml>
```

Object parameters

Parameter	Type	Example	
type	**string**	**“default”**	**Email sender rule type**
value	**string/int**	**“”**	**depending on rule type**
senders	**JSON array**	`[]`	**Senders list**

Rule types

Type	Value type	Example	
default	**string**	**“”**	**Always enabled**
group_id	**int**	**12**	**If group ID equals...**
from_domain	**string**	**“altcraft.com”**	**If From domain equals...**
to_domain	**string**	**“gmail.com”**	**If To domain equals...**

File object

JSON

JSON

```
{
    "data": "data:text/csv; base64,SEVMTE8K",
    "name": "US.txt"
}
```

XML

```
<xml>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
</xml>
```

Object parameters

Parameter	Type	Description
name	**string**	**File name**
data	**string**	**base64 encoded file**
file_key	**string**	**Previously uploaded file ID**
mimetype	**string**	**MIME information (optional)**

Profile data object

Profile data object is used in most profile import requests. It contains profile database field values and resources.
You can retrieve database fields information by making request to `api/v1.1/databases/fields_get/`.
You can also see information about databases and profile fields in User documentation.

JSON

JSON


```
{
    "data": {
        "_regcountry": "US",
        "_regcity": "New York",
        "_regip": "94.231.119.122",
        "_regurl": "www.example.com",
        "_regdate": "2019-03-14T22:00:00Z",
        "_bdate": "1990-02-22T21:00:00Z",
        "_browser": "Thunderbird",
        "_city": "Ryazan",
        "_country": "RU",
        "_fname": "Olly",
        "_ip": "94.231.119.122",
        "_lname": "Lambert",
        "_postal_code": "390000",
        "_region": "Ryazan Oblast",
        "_vendor": "form_#31",
        "_sex": 0,
        "_status": 0,
        "_tz": "Europe/Moscow",
        "email": "example@altcraft.com",
        "phones": [
            "+79000000000"
        ],
        "custom_field": "custom_value",
        "subscriptions": [
            {
                "channel": "email",
                "email": "boris.yudaev@altcraft.com",
                "priority": 0,
                "resource_id": 1,
                "status": 0
            }
        ]
    }
}
```


XML

```
<xml>
    <data>
        <regcountry>US</regcountry>
        <regcity>New York</regcity>
        <regip>94.231.119.122</regip>
        <regurl>www.example.com</regurl>
        <regdate>2019-03-14T22:00:00Z</regdate>
        <bdate>1990-02-22T21:00:00Z</bdate>
        <browser>Thunderbird</browser>
        <city>Ryazan</city>
        <country>RU</country>
        <fname>Olly</fname>
        <ip>94.231.119.122</ip>
        <lname>Lambert</lname>
        <postal_code>390000</postal_code>
        <region>Ryazan Oblast</region>
        <vendor>form_#31</vendor>
        <sex>0</sex>
        <status>0</status>
        <tz>Europe/Moscow</tz>
        <email>example@altcraft.com</email>
        <phones array='true'>+79000000000</phones>
        <custom_field>custom_value</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>boris.yudaev@altcraft.com</email>
            <priority>0</priority>
            <resource_id>1</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
</xml>
```

Description

Parameter	Type	Example	Description
_bdate	string	1990-02-22T21:00:00Z	Birth date in RFC 3339 format
_browser	string	Firefox	Web browser
_city	string	Ryazan	City
_country	string	RU	Country in TLD format
_fname	string	Olly	First name
_ip	string	94.231.119.122	IP address *
_lname	string	Lambert	Last name
_postal_code	int	390000	Postal code (ZIP)
_region	string	Ryazan Oblast	Region
_status	int	0	Global profile status. Values: • `0` — subscribed, communication possible • `1` — unsubscribed • `2` — complained about message (FBL, CFL) • `3` — invalid address • `4` — unconfirmed subscription • `5` — temporarily suspended • `6` — communication impossible for other reasons
_tz	string	Europe/Moscow	Time zone
email	string	example@altcraft.com	Email address
phones	JSON array	[“+790000000000”]	Phone numbers
subscriptions	JSON array	`["",""]`	Profile subscriptions
_regcountry	string	US	Registration country in TLD
_regcity	string	New York	Registration city
_regip	string	94.231.119.122	Registration IP address
_regurl	string	www.example.org	RegistrationURL
_regdate	string	2018-02-22T21:00:00Z	Registration date in RFC 3339 format
_sex	int	0	Gender
_vendor	string	form#33	Website form or other lead source

custom_field	*	custom_value	Custom database fields.
			
			Field functional update is possible. To do this, you must specify the *action* and *valu*
			
			An example of a request to increase the value of a field:
			“custom_field”:{
			“action”: “incr”,
			“value”: 1000
			}

INFO

*** If you send `"_ip"` parameter containing IPv4 value, Altcraft will automatically detect subscriber’s lo following system fields:**

JSON

```
{
          "_ip": "94.231.119.122", // (records IP as well)
          "_lat": 54.6197, // (lattitude)
          "_lon": 39.74, // (longitude)
          "_postal_code": "390000",
          "_region": "Ryazanskaya Oblast'",
          "_country": "RU",
          "_city": "Ryazan",
          "_tz": "Europe/Moscow"
}
```

XML

```
<xml>
    <ip>94.231.119.122</ip>// уже добавляется само по себе
    <lat>54.6197</lat>// географическая широта
    <lon>39.74</lon>// географическая долгота
    <postal_code>390000</postal_code>
    <region>Ryazanskaya Oblast</region>
    <country>RU</country>
    <city>Ryazan</city>
    <tz>Europe/Moscow</tz>
</xml>
```

Sender object

JSON

JSON

```
{
  "sender_id": 287,
  "percent": 100,
  "from_email": "no-reply@altcraft.com",
  "replyto_email": "no-reply@altcraft.com",
  "from_name": "James",
}
```

XML

```
<xml>
  <sender_id>287</sender_id>
  <percent>100</percent>
  <from_email>no-reply@altcraft.com</from_email>
  <replyto_email>no-reply@altcraft.com</replyto_email>
  <from_name>James</from_name>
</xml>
```

Object parameters

Parameter	Type	Example	Description
sender_id	**int**	`123`	**Sender ID**
percent	**int**	`100`	**Message allocation rate**
from_email	**string**	`"no-reply@altcraft.com"`	**Override from email**
replyto_email	**string**	`"no-reply@altcraft.com"`	**Override reply to email**
from_name	**string**	`"Altcraft"`	**Override from name**

Sender types

Senders are created in Altcraft administrative panel to send emails and SMS.

Type	Channel	Description
“akmta”	email	Altcraft Mail Transfer Agent
“dmta”	email	Legacy sender
“eshark”	email	EmailShark sender (legacy)
“webzo”	email	WebZo.net sender Legacy
“mandrill”	email	API MailChimp Mandrill integration
“emaildirect”	email	EmailDirect integration Legacy
“amazon_ses”	email	Amazon SES integtation
“smsc_ru”	sms	SMSc.ru API integration
“bulksms_com”	sms	BulkSMS.com API integration

SMS rule object

JSON

JSON

```
{
    "senders": [
        {
            "from_email": "",
            "from_name": "Example",
            "percent": 100,
            "replyto_email": "",
            "sender_id": 798
        }
    ],
    "type": "default",
    "value": ""
}
```

XML

```
<xml>
  <senders array='true'>
    <from_email></from_email>
    <from_name>Example</from_name>
    <percent>100</percent>
    <replyto_email></replyto_email>
    <sender_id>798</sender_id>
  </senders>
  <type>default</type>
  <value></value>
</xml>
```

Object parameters

Parameter	Type	Example	
type	**string**	`"default"`	**SMS rule type**
value	**string/int**	`""`	**Depending on type**
senders	**JSON array**	`[]`	**Senders list**

Rule types

Rule type	Value type	Example	
default	**string**	`""`	**Always enabled**
group_id	**int**	`12`	**If group ID equals...**
from_domain	**string**	`"altcraft.com"`	**If from domain equals...**

Start schedule object

JSON

JSON

```
{
    "start_hours": [
        2,
        4
    ],
    "start_mdays": [
        1
    ],
    "start_mins": [
        0,
        10,
        20,
        30,
        40,
        50
    ],
    "start_months": [
        12
    ],
    "start_wdays": [],
    "start_years": [
        2016
    ]
}
```

XML

```
<xml>
<xml>
    <start_hours>2</start_hours>
    <start_hours>4</start_hours>
    <start_mdays array='true'>1</start_mdays>
    <start_mins>0</start_mins>
    <start_mins>10</start_mins>
    <start_mins>20</start_mins>
    <start_mins>30</start_mins>
    <start_mins>40</start_mins>
    <start_mins>50</start_mins>
    <start_months array='true'>12</start_months>
    <start_wdays></start_wdays>
    <start_years array='true'>2024</start_years>
</xml>
```

Object parameters

Parameter	Type	Example	D
start_mins	**JSON array**	**[1, 31]**	**Minute launches, 0-59**
start_hours	**JSON array**	**[22]**	**Hours launches, 0-23**
start_mdays	**JSON array**	**[]**	**Month day launches, 1-31**
start_wdays	**JSON array**	**[]**	**Week day launches, Su-Sat**
start_months	**JSON array**	**[]**	**Months launches, 1-12**
start_years	**JSON array**	**[]**	**Years launches**
tz	**string**	**Europe/Moscow**	**Mailing time zone**
profile	**bool**	**true**	**Use profiles' time zones**
losers	**bool**	**true**	**Send instantly if past profiles' local time**

Subscription object

Description

The `subscription` object contains data about a profile's subscription to communication channels. The st depending on the channel type and API context

Basic Structure

Parameters	Type	Example	Description
"channel"	**string**	**"email"**	**Channel type ("email" / "sms" / "push" / "notify" / "viber" / "whatsapp" / "telegram_bot" / "max_bot")**
"db_id"	**int**	**1**	**Database identifier**
"email"	**string**	**"john@example.com"**	**Email address**
"phone"	**string**	**"+79000000000"**	**Phone number **(For SMS subscriptions)****
"provider"	**string**	**"Firefox"**	**Provider type **(For Push subscriptions)****
"subscription_id"	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription identifier **(For Push subscriptions)****
"cc_data"	**object**	\{ "id": 1 \}	**Chat ID **in Telegram bot****
		\{ "phone": "+79000000000" \}	**Profile phone in **WhatsApp*/ Viber™ / Notify****
"priority"	**int**	**1**	**Subscription priority (optional field). If the subscription priority is passed in the request, the "Increase priority for new subscriptions" option in [the work.**
"profile_id"	**string**	**"54759eb3c090d83494e2d804"**	**Profile identifier**
"resource_id"	**int**	**1**	**Resource identifier**
"custom_fields"	**object**	**"subscribed"**	**Standard and additional subscription fields (optional field). You can get information about available subscription fields in the resource via the /v1.1/resources/fields_get m**
"status"	**string**	**"subscribed"**	**Subscription status [Learn more](/en/developer-guide/objects/Subscription-status_2195807#Subscriptionstatus-Objectvalues)**
"cats"	**array**	["category_1", "category_2"]	**Resource categories to which the profile is subscribed (optional field).**

Extended structure when querying the profile through the handler `/api/v1.1/profiles/get/`

When querying the profile through the `/api/v1.1/profiles/get/` handler, the `subscription` object contains a

- **Channel data is stored in separate objects:**
  - `"email_data"` **- for Email subscriptions**
  - `"phone_data"` **- for SMS/Viber/WhatsApp**
  - `"cc_data"` **- for Telegram**
- **Technical fields**
  - `"hash_id"` **- unique subscription hash**
  - `"is_delete"` **- delete flag (soft delete)**
- **History:**
  - `"reg_info"` **- subscription registration date**

Example structure

JSON

```
{
  "resource_id": 4,
  "channel": "email",
  "email_data": {
        "email": "user@example.com",
        "domain": "example.com",
        "md5": "d922b1da12193df553aad2bee61da46b"
  },
  "status": "subscribed",
  "reg_info": {
    "date": "2025-07-23T11:27:05.487Z"
  },
  "hash_id": "7641c5c5",
  "is_delete": false
}
```

XML

```
<xml>
<subscription>
    <resource_id>4</resource_id>
    <channel>email</channel>
    <email_data>
        <email>user@example.com</email>
        <domain>example.com</domain>
        <md5>d922b1da12193df553aad2bee61da46b</md5>
    </email_data>
    <status>subscribed</status>
    <reg_info>
        <date>2025-07-23T11:27:05.487Z</date>
    </reg_info>
    <hash_id>7641c5c5</hash_id>
    <is_delete>false</is_delete>
</subscription>
</xml>
```

Object example

JSON

```
{
  "subscriptions": [
    {
      "resource_id": 1,
      "channel": "email",
      "email": "example@example.com"
    },
    {
      "resource_id": 2,
      "channel": "sms",
      "phone": "+79000000000"
    },
    {
      "resource_id": 1,
      "channel": "push",
      "subscription_id": "a81c264a938b475",
      "provider": "android-firebase",
      "custom_fields": {
        "_browser_name": "Firefox",
        "_device_type": "web",
        "custom_field_1": "test value"
      },
      "cats": [
        "category_1",
        "category_2"
      ]
    },
    {
      "resource_id": 1,
      "channel": "whatsapp",
      "cc_data": {
        "phone": "+79000000000"
      }
    },
    {
      "resource_id": 1,
      "channel": "viber",
      "cc_data": {
        "phone": "+79000000000"
      }
    },
    {
      "resource_id": 1,
      "channel": "telegram_bot",
      "cc_data": {
        "id": 12345
      }
    },
    {
      "resource_id": 1,
      "channel": "notify",
      "cc_data": {
        "phone": "+79000000000"
      }
    }
  ]
}
```

XML

```
<xml>
<subscriptions>
  <subscription>
    <resource_id>1</resource_id>
    <channel>email</channel>
    <email>example@example.com</email>
  </subscription>

  <subscription>
    <resource_id>2</resource_id>
    <channel>sms</channel>
    <phone>+79000000000</phone>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>push</channel>
    <subscription_id>a81c264a938b475</subscription_id>
    <provider>android-firebase</provider>
    <custom_fields>
      <_browser_name>Firefox</_browser_name>
      <_device_type>web</_device_type>
      <custom_field_1>test value</custom_field_1>
    </custom_fields>
    <cats>
      <category>category_1</category>
      <category>category_2</category>
    </cats>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>whatsapp</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>viber</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>telegram_bot</channel>
    <cc_data>
      <id>12345</id>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>notify</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>
</subscriptions>
```

Subscription status

Description

The status parameter in the subscription object overwrites the subscription status. It is recommended need to update the status.

Both string and numeric status representations can be used.

Word value	Description
subscribed	**Ready for messaging**
unsubscribed	**Unsubscribed**
complained	**Reported spam**
hardbounced	**Hard bounce, email probably invalid**

Word value	Description
unconfirmed	**Subscription unconfirmed**
suspended	**Temporarily unsubscribed**
invalid	**Invalid for a variety of reasons**

Meta organization, which owns Instagram, Facebook, and WhatsApp products, is recognized as extre Federation.

***** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Trigger types

For more details see the corresponding User documentation article.

Type	Description
api_call	**API call only**
api_import	**API profile import**
action	**Subscribers message actions**
update	**Profile fields update**
pixel	**Pixel activation**

Upload file

Description

Uploads a file.

Request URL

Method: POST

`https://example.com/api/v1.1/file_upload`

Request parameters

Parameter	Type	Example	Required	
header	**string**	**“emails.csv”**	**Yes**	**F**
data	**string**	**“data:text/csv; base64,YQpi...”**	**Yes**	**b**

Request example

JSON

```
{
    "data": "data:text/csv; base64,SEVMTE8K",
    "header": "US.csv"
}
```

XML

```
<xml>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <header>US.csv</header>
</xml>
```

Response example

JSON

```
{
    "data": {
        "file": "fd7006e5-5c42-4eba-aa6a-f6cb7f5cb6f2"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <file>fd7006e5-5c42-4eba-aa6a-f6cb7f5cb6f2</file>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
file	**string**	**Unique file ID**

Get message web version

Description

Retrieves an email message web version.

Request URL

Method: POST

`https://example.com/api/v1.1/webver/get`

Request parameters

Parameter	Type	Example	Req
send_message_id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**

Request example

JSON

```
{
    "send_message_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <send_message_id>abcdefghijklmnqrstuvwxyz</send_message_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "createtime": "2018-08-24T09:26:23.292Z",
        "email": {
            "from_email": "example@example.com",
            "from_name": "example.com",
            "html": "HTML content ...",
            "replyto_email": "example@example.com",
            "replyto_name": "example.com",
            "subject": "Welcome to Altcraft",
            "text": "Text version also important\n\n--\nUnsubscribe: http://trk.gmail.com/37cfe6oTGQF4LYqk6oMi8eCtZ?dummy=1"
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <createtime>2018-08-24T09:26:23.292Z</createtime>
    <email>
      <from_email>example@example.com</from_email>
      <from_name>example.com</from_name>
      <html>HTML content ...</html>
      <replyto_email>example@example.com</replyto_email>
      <replyto_name>example.com</replyto_name>
      <subject>Welcome to Altcraft</subject>
      <text>Text version also important --
Unsubscribe: http://trk.gmail.com/37cfe6oTGQF4LYqk6oMi8eCtZ?dummy=1</text>
    </email>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	D
error	**int**	**Error code**
error_text	**string**	**error text**
createtime	**date**	**Template creation time**
email	**JSON object**	**Message object**

Push providers

Push provider	Web / App	Provider name in API reques	
		ID 1	
Google Chrome (deprecated)	**Web**	**"google"**	**"Chrome"**
Mozilla Firefox	**Web**	**"mozilla"**	**"Firefox"**
Safari	**Web**	**"safari"**	**"Safari"**
Android Firebase	**App**	**"android-firebase"**	**"AndroidFi**
IOS Firebase	**App**	**"ios-firebase"**	**"IOSFireba**
Firefox Firebase	**Web**	**"firefox-firebase"**	**"FirefoxFir**
Chrome Firebase	**Web**	**"chrome-firebase"**	**"ChromeFi**
Opera Firebase	**Web**	**"opera-firebase"**	**"OperaFire**
Yandex Firebase	**Web**	**"yandex-firebase"**	**"YandexBr**
Samsung Firebase	**Web**	**"samsung-firebase"**	**"Samsungl**
IOS APNs	**App**	**"ios-apns"**	**"IOSAPNs"**
AppMetrica with iOS device id	**App**	**"yandex-Appmetrica-device-id-ios"**	**"YandexAp**
AppMetrica with Android device id	**App**	**"yandex-Appmetrica-device-id-android"**	**"YandexAp**
AppMetrica with iOS IFA	**App**	**"yandex-Appmetrica-ios-ifa"**	**"YandexAp**
AppMetrica with Google AID	**App**	**"yandex-Appmetrica-google-aid"**	**"YandexAp**
AppMetrica with Android push token	**App**	**"yandex-Appmetrica-android-token"**	**"YandexAp**
AppMetrica with iOS push token	**App**	**"yandex-Appmetrica-ios-token"**	**"YandexAp**
MailRu (устаревший)	**App**	**"mr"**	**"MR"**
Android Huawei Mobile Services	**App**	**"android-huawei"**	**"AndroidH**
IOS Huawei Mobile Services	**Web**	**"ios-huawei"**	**"IOSHuaw**
Android RuStore	**App**	**"android-rustore"**	**"AndroidRu**

Deduplication of requests

Description

Repeat requests are API requests that contain the same headers and the same data as the original req sent if the connection failed at the moment of receiving the data.

The platform will not accept a repeated request if it changes the data in order to avoid duplicate event and does not require additional configuration. By default, the platform deduplicates within 1 day from t

The platform performs deduplication according to the following algorithm:

- **Checking for the presence of Request-ID header**
- **If Request-ID is not specified, the checksum of request body and headers is checked**

It should be noted that not all requests support deduplication with checksum, and if no Request-ID is s duplication check will be performed. You can find a list of such requests in the table below.

> INFO
>
> **Different request sending clients may add their own system headers or other information to the reque deduplication.**

How do you know if the request was taken as a duplicate one?

The response to the repeated request will be the same as the response to the original one. However, th additional headers:

```
Duplicate-Key: NeOScRqBgsqj67CR5dYcqvcmPZA=
Duplicate-Request: true
```

The Duplicate-Key header consists of the Request-ID of the request and the sender token. If Request-ID store the hashed request data.

How to send a request so that it is not taken as a duplicate one?

It is recommended to mark each request with a unique identifier. In this way when you change the ider requests, even if they contain the same data.

There are 2 ways to do it:

- **Change the data in the request body or URL, for example, &request_id=12345**
- **Add the Request-ID header, for example Request-ID: a0fe37f2-b4e0-4e28-a328-4ec7461604fc**

API requests that have deduplication

All requests that modify data or activation / launch of entities are checked for repetitions:

Category	Request
Integrations	**/integrations/lpgenerator/profile_import** **/integrations/tilda/profile_import**
Suppression lists	**/v1.1/suppresses/add** **/v1.1/suppresses/update** **/v1.1/suppresses/domain_add** **/v1.1/suppresses/domains_add** **/v1.1/suppresses/domain_remove** **/v1.1/suppresses/domains_remove** **/v1.1/suppresses/email_add** **/v1.1/suppresses/emails_add** **/v1.1/suppresses/email_remove** **/v1.1/suppresses/emails_remove** **/v1.1/suppresses/phone_add** **/v1.1/suppresses/phones_add** **/v1.1/suppresses/phone_remove** **/v1.1/suppresses/phones_remove**
Subscriptions	**/v1.1/subscriptions/set** **/v1.1/subscriptions/suspend_all** **/v1.1/subscriptions/unsuspend_all** **/v1.1/subscribers/import** **/v1.1/subscribers/update**
Segments	**/v1.1/segments/add** **/v1.1/segments/update** **/v1.1/segments/refresh_counters**
Mailings	**/v1.1/campaigns/deactivate** **/v1.1/campaigns/activate** **/v1.1/campaigns/clone**
Trigger mailings	**/v1.1/campaigns/triggers/add** **/v1.1/campaigns/triggers/update** **/v1.1/campaigns/triggers/start** **/v1.1/campaigns/triggers/import_and_start** **/v1.1/campaigns/triggers/clone** **/v1.1/triggers/start_batch (only Request-ID method)** **/v1.1/triggers/start_batch_task (only Request-ID method)** **/v1.1/triggers/import_and_start_batch (only Request-ID method)** **/v1.1/triggers/import_and_start_batch_task (only Request-ID method)**

Category	Request
Broadcasts	**/v1.1/campaigns/broadcasts/add** **/v1.1/campaigns/broadcasts/update** **/v1.1/campaigns/broadcasts/start**
Regular mailings	**/v1.1/campaigns/regulars/add** **/v1.1/campaigns/regulars/update** **/v1.1/campaigns/regulars/start**
Senders	**/v1.1/admin/senders/queue/hold** **/v1.1/admin/senders/queue/unhold** **/v1.1/admin/senders/queue/purge** **/v1.1/admin/senders/update** **/v1.1/admin/senders/add_to_account** **/v1.1/admin/senders/add** **/v1.1/admin/vsenders/add** **/v1.1/admin/vsenders/update** **/v1.1/admin/vsenders/clone**
Nodes	**/v1.1/admin/nodes/mta/start** **/v1.1/admin/nodes/mta/stop**
Static segments	**/v1.1/profiles/segstatic/add** **/v1.1/profiles/segstatic/remove**
Profiles	**/v1.1/profiles/add** **/v1.1/profiles/update** **/v1.1/profiles/import** **/v1.1/profiles/add_batch** **/v1.1/profiles/update_batch** **/v1.1/profiles/import_batch**
Promocodes	**/v1.1/promocodes/attach** **/v1.1/promocodes/activate** **/v1.1/promocodes/import** **/v1.1/promocodes/detach** **/v1.1/promocodes/update**
Relations	**/v1.1/relations/attach** **/v1.1/relations/detach** **/v1.1/relations/edit** **/v1.1/relations/strengthen**
Profiles databases	**/v1.1/databases/wipe** **/v1.1/databases/refresh_counters**
Virtual senders	**/v1.1/admin/vsenders/add** **/v1.1/admin/vsenders/update** **/v1.1/admin/vsenders/clone**
Scenarios	**/v1.1/workflows/start** **/v1.1/workflows/import_and_start** **/v1.1/workflows/import_and_start_batch (only Request-ID method)** **/v1.1/workflows/import_and_start_batch_task (only Request-ID method)**
Push event registration	**/v1.1/pushes/post_events** **/v1.1/pushes/crypto_post_events**
Message templates	**/v1.1/templates/add** **/v1.1/templates/update** **/v1.1/templates/delete**
Resources	**/v1.1/resources/push_subscribe** **/v1.1/resources/refresh_counters**
Data tables	**/v1.1/datatables/query_templates/update** **/v1.1/datatables/query_templates/delete** **/v1.1/datatables/query_templates/add** **/v1.1/datatables/query_segments/update** **/v1.1/datatables/query_segments/delete** **/v1.1/datatables/query_segments/add**

How to send API request with RabbitMQ

Description

In this article, you will learn how to send API requests using RabbitMQ.

> **INFO**
>
> **A unique parameter `request_id` is sent to identify requests.**

Queues

When working with API in RabbitMQ, two queues are used:

1. **api_req — queue for incoming requests;**
2. **api_resp — the queue that receives responses to requests (only if request_id is specified).**

Structure of the sent JSON message

Parameter	Type	Example	Required
request_id	**string**	**“abcd1234”**	**Yes, if the response to the request is necessa**
request	**string**	**api/v1.1/campaigns/triggers/import_and_start_batch**	**Yes**

body	**JSON object**	“body”:{
		“data”:
		{
		“_fname”: “Fname1”,
		“_lname”: “Lname1”,
		“email”: “profile1@example.com”
		},
		“email”: “profile1@example.com”,
		“db_id”: 1,
		“detect_geo”: true
		}
			Yes

Structure of the received JSON message

Parameter	Type	Example	Re

body	**JSON обьект**	“body”:
		{
		“error”:0,
		“error_text”:“Successful operation”,
		“profile_id”:“60f039e830b8bcb28392f8eb”
		}
			Re

request_id	**string**	**“abcd1234”**	**Re**

How to send API requests in Go

You can send an API request using a script that calls RabbitMQ:

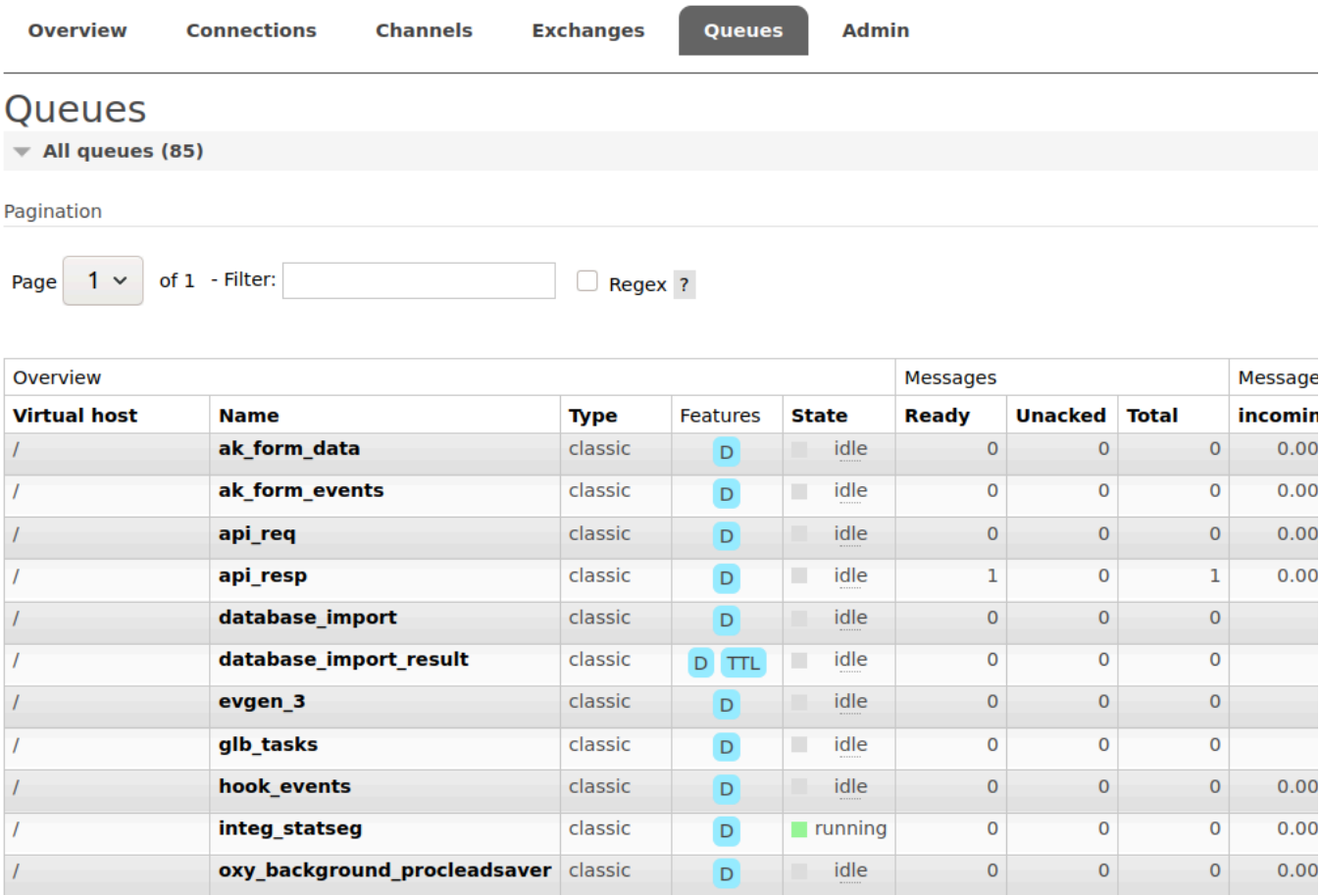
- **Write and execute a script**

Example

```
package main

        "encoding/json"
        "log"

        "github.com/streadway/amqp"
)

const accID = 1
const resourceID = 3
const dbID = 23
const msgID = 17
const segmentID = 85
const amountOfPushMsgs = 1
const emailDomain = "example.com"

const req = `{
        "account_id": 1,
        "request_id": "db1894e4-1a2c-4021-8233-9cca0b96b79e",
        "request": "api/v1.1/campaigns/triggers/import_and_start_batch",
    "body": {
                "format": "json",
                "trigger_id": 240,
                "skip_triggers": false,
                "detect_geo": true,
                "matching": "custom",
                "field_name":"CustomF",

                "custom_data":{
                        "some":"some0"
                },
                "content":{
                        "someCont":"someCont0"
                },
                "data": [
                        {
                                "data":{
                                        "_fname":"NUMBER13",
                                        "_lname":"Lambert",
                                        "phones":"790000000013",
                                        "email": "profile1@example.com",
                                        "CustomF":18
                                },
                                "custom_data":{
                                        "some":"some1"
                                },
                                "content":{
                                        "someCont":"someCont1"
                                }
                        },
                        {
                                "data":{
                                        "_fname":"NUMBER14",
                                        "_lname":"Hard",
                                        "phones":"790000000014",
                                        "email": "profile2@example.com",
                                        "CustomF":14
                                }
                        }
                ]
        }
}`

func failOnError(err error, msg string) {
        if err != nil {
                log.Fatalf("%s: %s", msg, err)
        }
}

func main() {
        var err error

        var conn *amqp.Connection
        conn, err = amqp.Dial("amqp://example:abcdefghijklmnopqrstuvwxyz127.0.0.1:5672/")
        failOnError(err, "Failed to connect to RabbitMQ")
        defer conn.Close()

        var ch *amqp.Channel
        ch, err = conn.Channel()
        failOnError(err, "Failed to open RabbitMQ channel")
```

```
	defer ch.Close()

	qUeueImport, err := ch.QueueDeclare(
		"api_req", // name
		true,      // durable
		false,     // delete when unused
		false,     // exclusive
		false,     // no-wait
		nil,       // arguments
	)
	failOnError(err, "Failed to declare RabbitMQ queue")

	for i := 0; i < amountOfPushMsgs; i++ {
		var bodyMap = make(map[string]interface{})
		err = json.Unmarshal([]byte(req), &bodyMap)
		failOnError(err, "Failed json.Unmarshal to bodyMap")

		for k, v := range bodyMap {
			log.Println(k, v)
		}

		body, err := json.Marshal(bodyMap)
		failOnError(err, "Failed to json.Marshal bodyMap")

		err = ch.Publish(
			"",               // exchange
			qUeueImport.Name, // routing key
			false,            // mandatory
			false,            // immediate
			amqp.Publishing{
				ContentType: "text/plain",
				Body:        []byte(body),
			})
		failOnError(err, "Failed to publish a message")
	}

	log.Printf("[v] Sended %d requests to %s", amountOfPushMsgs, qUeueImport.Name)
}
```

- **After executing the script, go to RabbitMQ —> Queues**

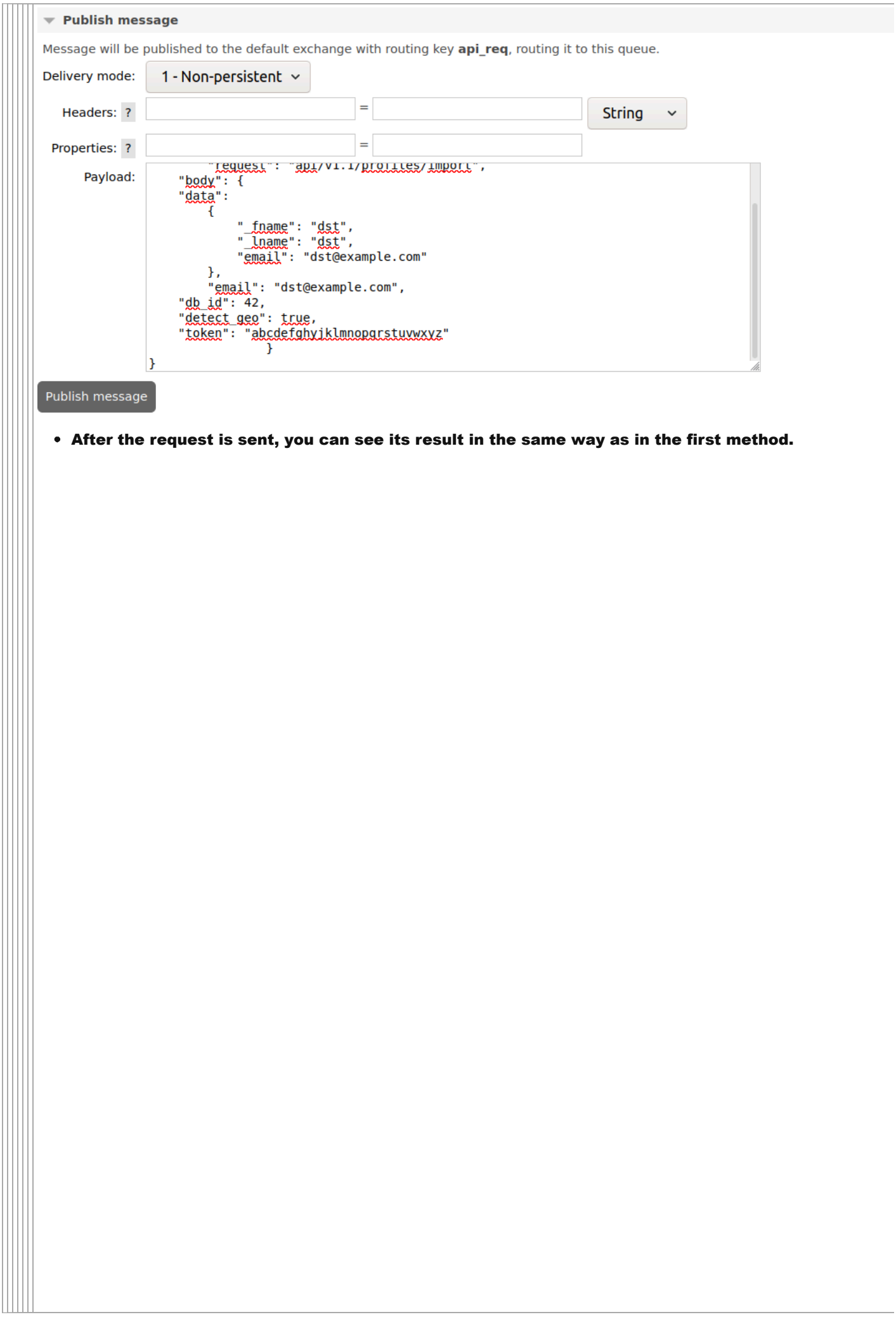


- **Choose api_resp and get a response to the API request**

How to send API requests via RabbitMQ Management Plugin

You can send an API request directly to RabbitMQ via api_req:

- **Go to RabbitMQ —> the Queues tab. Then select api_req and Publish message.**



- **After the request is sent, you can see its result in the same way as in the first method.**

List of gender identifications

In API requests, various reports and system parameters, gender can be displayed in different ways.
Male gender can be indicated on the platform by one of the parameters of the object below:

```
{
        "id": 1,
        "title": "Male",
        "values": ["male", "1", "Male", "MASCULINE",
            "M",
            "MAN",
            "♂",
            "М",
            "МУЖ",
            "МУЖСК",
            "МУЖСКОЙ",
            "МУЖЧИНА"
        ]
}
```

Any of the following code parameters is responsible for female gender:

```
{
        "id": 2,
        "title": "Female",
        "values": ["female", "2", "Female", "FEM",
            "FEMININE",
            "F",
            "WOMAN",
            "W",
            "♀",
            "Ж",
            "ЖЕН",
            "ЖЕНСК",
            "ЖЕНСКИЙ",
            "ЖЕНЩИНА"
        ]
}
```

TIP

If you need to add other designations, you can do so by adding the required values to the gender ident
about this in this article.

Obtain valid values for fields: browsers, devices, tz, oses, languages

Description

The API method allows you to retrieve valid values for the following fields:

- `browser`
- `device`
- `tz`
- `os`
- `language`

These fields are used in the following requests:

- `/v1.1/resources/fields_get` (**obtaining resource subscription fields**)
- `/api/v1.1/goals/register` (**goals and goal values registration**)

URL

Method: POST

Request URL: `https://example.com/api/v1.1/dictionary`

Request parameters

Parameter	Type	Example	Required	Description
name	**string**	**“browsers” “devices” “tz” “oses” “languages”**	**Yes**	**Specify the name of the system field to retrieve the corresponding list of valid values.**

Request example

JSON

```
{
   "name": "devices"
}
```

XML

```
<xml>
  <name>devices</name>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "title": "Desktop",
      "value": "web"
    },
    {
      "title": "Other",
      "value": "other"
    },
    {
      "title": "Mobile",
      "value": "mob"
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <value>web</value>
    <title>Desktop</title>
  </data>
  <data>
    <value>other</value>
    <title>Other</title>
  </data>
  <data>
    <value>mob</value>
    <title>Mobile</title>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

mSDK

Description

Altcraft mSDK is a mobile SDK for reliable push notification handling and interaction with the Altcraft p and iOS, supports Firebase, Huawei, RuStore and APNS, and provides extended capabilities for subscri processing, and working with multiple users on the same device.

The SDK can use JWT tokens or rToken, which enables flexible security configuration and profile matc and delivery event handling ensures stable and accurate data transmission, while the built-in WorkMan even in case of network failures.

Minimum requirements:

- **iOS: Minimum Deployment Target — 13**
- **Android: minimum API level (minSdkVersion) — 25**

Features

- **Compatible with Kotlin, Java (Android), Swift, Objective-C (iOS), JavaScript (React Native), Dart (Fl**
- **Push subscription management: subscribe, pause, unsubscribe.**
- **Automatic notification display with content configured in the platform, with UI customization optio**
- **Automatic push token refresh when it changes.**
- **Automatic delivery and open event reporting.**
- **Support for anonymous and registered users, including multiple users on the same device (with JW**
- **Mobile event registration.**
- **Automatic retry of requests via WorkManager for maximum delivery.**
- **Push provider support: Firebase, Huawei, RuStore (Android), APNS (iOS).**
- **Use of JWT tokens for security and unrestricted profile matching variants.**
- **Ability to work via rToken with restricted access tied to a push subscription identifier.**
- **Ability to override the push notification format and access all message data fields.**

GitHub

Altcraft mSDK repositories on GitHub:

- **iOS**
- **Android**
- **React Native**
- **Flutter**

Dependency Managers

Cocoapods:

```
pod 'Altcraft', '1.0.3'
```

Swift Package Manager:

```
1.0.3
```

Maven Central:

```
com.altcraft:android-sdk:1.0.6
```

npm:

```
@altcraft/react-native-sdk
```

Managing JWT and Role Token

Authorization Options

JWT Token

This authorization type uses a JWT token that the application passes to the SDK. The token is added t

JWT (JSON Web Token) is a JSON-formatted string containing claims signed to verify authenticity and i

The token is generated and signed with an encryption key on the client server side (encryption keys ar

When requested by the SDK, the application must provide the JWT token received from the server.

Advantages:

- **Increased security of API requests.**
- **Ability to search profiles using any identifiers (email, phone number, custom ID).**
- **Support for multiple users on one device.**
- **Restoring access to a profile after reinstalling the application.**
- **Identification of a specific profile across different devices.**

rToken

An alternative authorization method uses a role token (rToken) passed in the SDK configuration parame include a header containing the role token.

Characteristics:

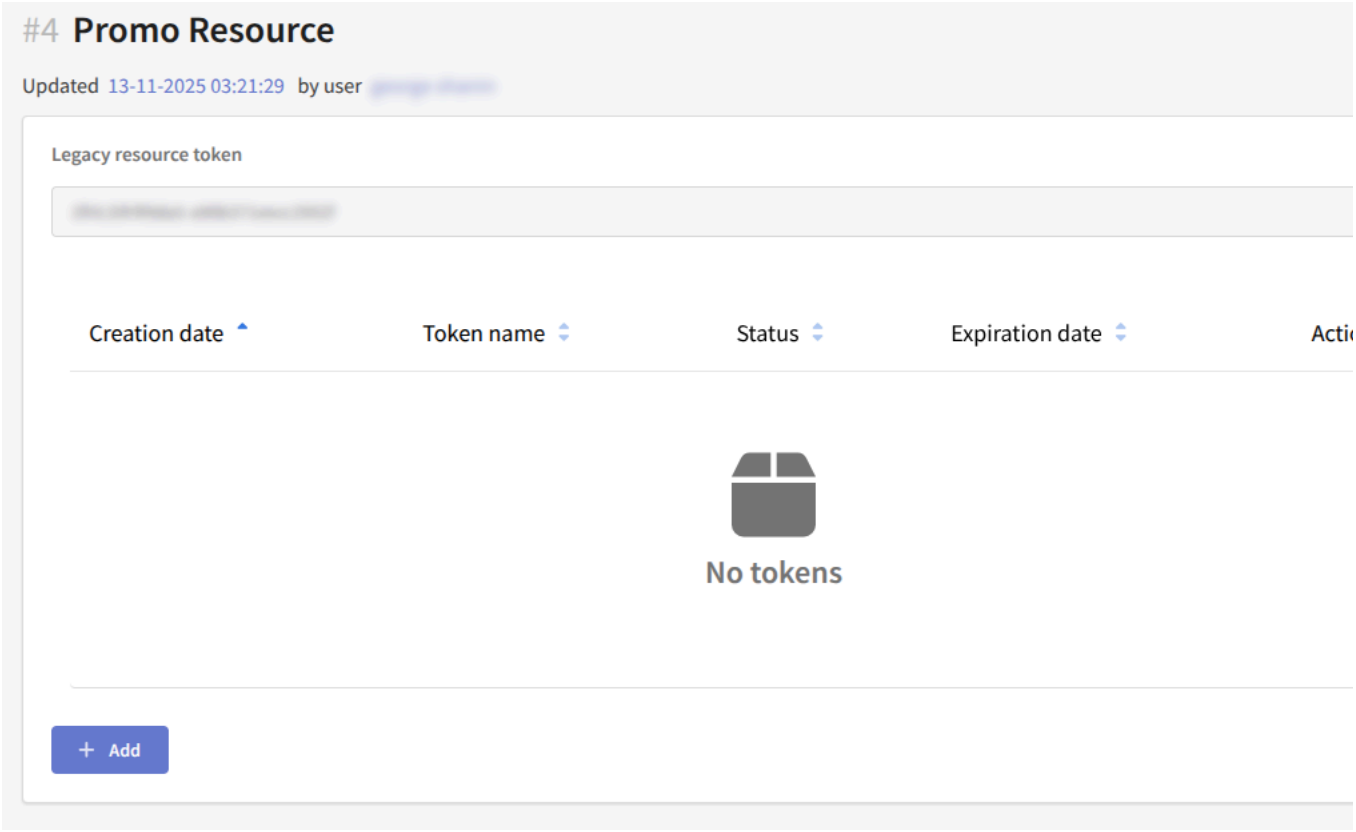
- **Profile search is only possible using the device push token (e.g. FCM).**
- **If the push token changes and is not sent to the server (for example, after app deletion and reinsta lost and a new profile is created.**

Limitations:

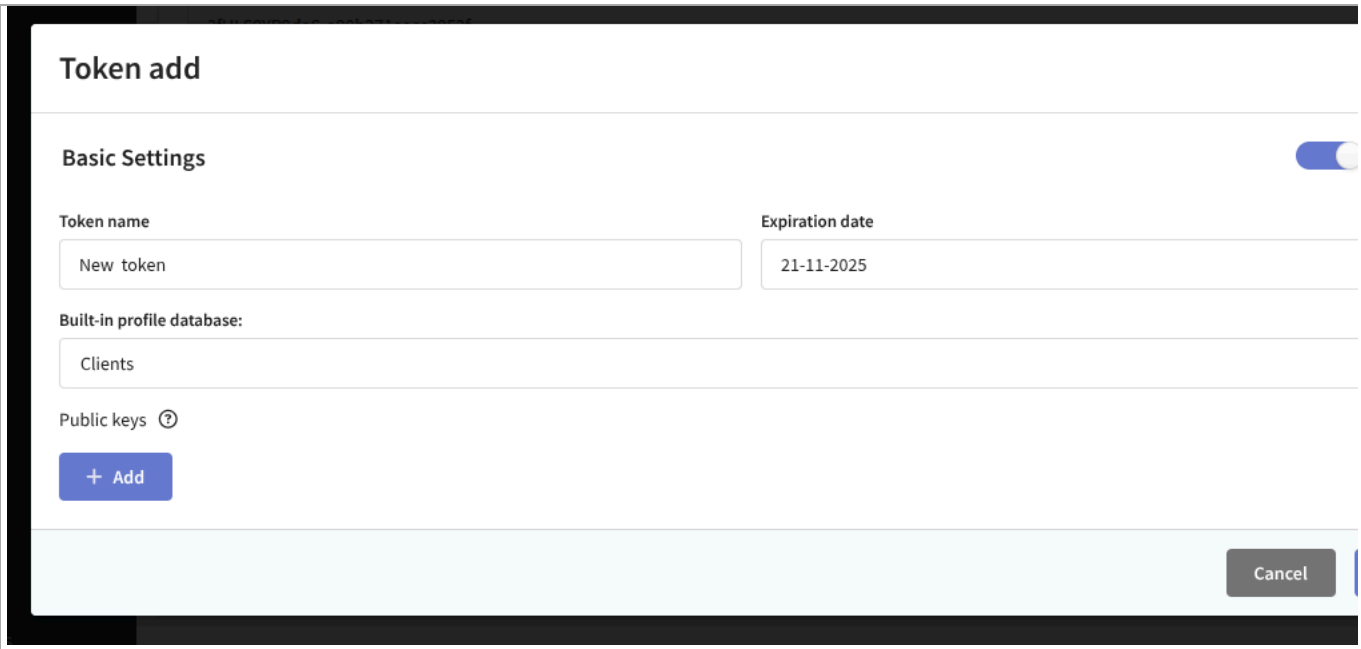
- **Loss of profile linkage if the push token changes and is not registered on the Altcraft server.**
- **No ability to use the application for different profiles on the same device.**
- **No ability to register the same user on another device.**

Configuring the Role Token and JWT Issuing Service

After creating a resource, the token management section becomes available in its settings:



To create a role token, you do not need to provide a public key. Simply specify its name, expiration dat database:



The role token acts as an access key for mSDK to the specific “resource–database” link. With this auth operations in the platform become available:

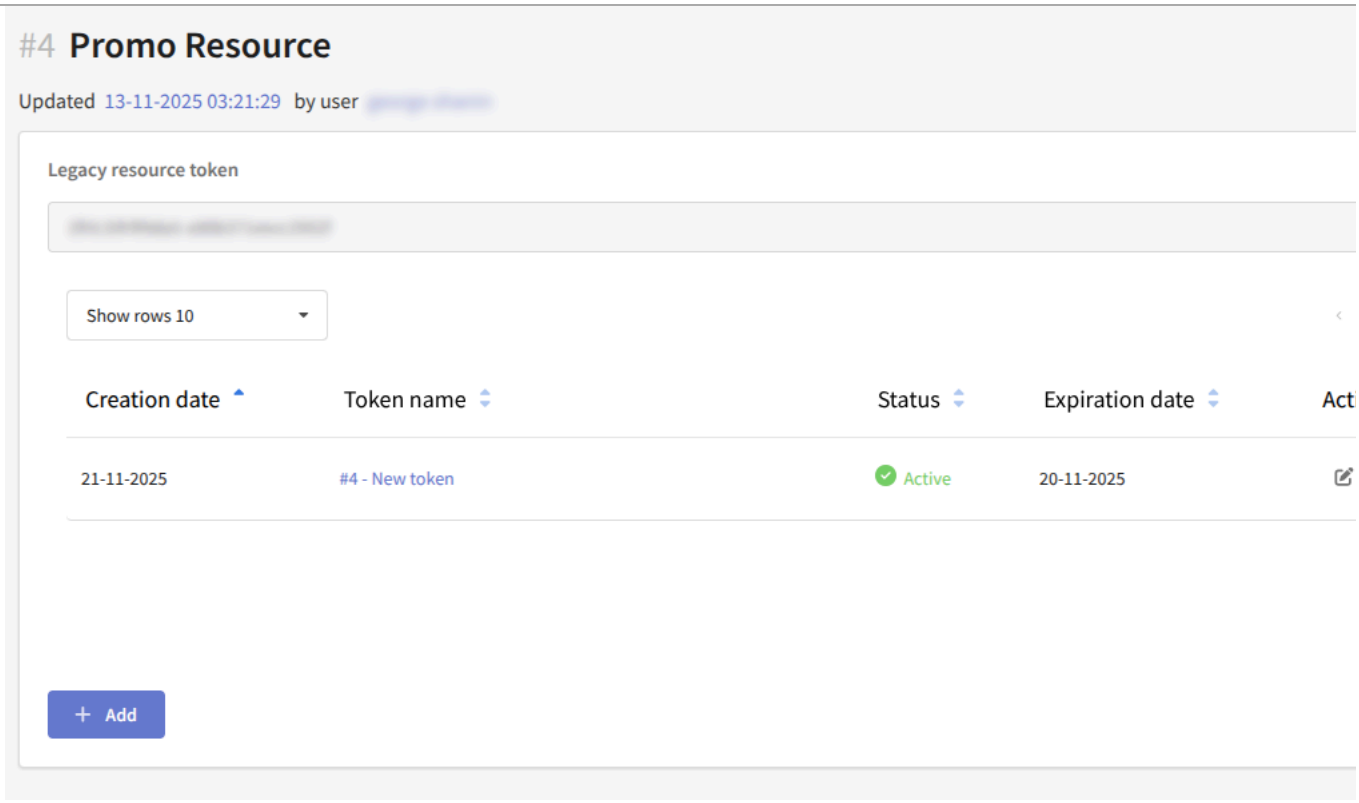
- **Event registration**
- **Profile field updates**
- **Profile import**

To create a JWT token, a public key is required. The platform supports the ES384 algorithm (ECDSA as RS256, ES256, and ES512 can also be used for compatibility with different application libraries.

TIP: Key generation

Example of generating a key using ES384: `openssl ecparam -name secp384r1 -genkey -noout -out private.ec.key out public.pem` **This generates private and public key files. The public key is used in the platform.**

After the token is created, it can be copied and used further:



Authorizing mSDK Requests with a Role Token

When used for authorization, rToken grants access to operations within a specific resource. Due to its restrictions are introduced on the type of profile matching available.

With a role token, profiles are searched using query parameters. In the current implementation, the ba parameters when using rToken:

- **provider — mobile push provider**
- **subscription_id — device token provided by the provider**

The search for a profile is done by subscription identifier within the databases linked to the resource.

Characteristics

- **Profile search is possible only by the device push token (e.g. FCM).**
- **When importing a profile via push subscription, it is marked as "temporary".**
- **When registering mobile events, linking events to a profile (writing to the event history) is not poss**
- **If the push token changes and is not sent to the server, the link to the profile is lost and a new pro**

Usage

When using the `Altcraft SDK`, the token is passed as the `rToken` parameter of the `AltcraftConfiguration` cl The library later includes it in the authorization header.

Authorizing mSDK Requests with a JWT Token

For secure authorization in the platform, it is recommended to use a JWT token. The token must be sig added during token creation.

Characteristics

- **Increased security of API requests. JWT acts as a protective wrapper for the role token, preventin token is embedded into the JWT payload, and only with the keys previously provided in the resourc mSDK actions.**
- **Ability to search profiles by any identifiers (email, phone number, custom ID), according to the mat**
- **Support for multiple users on one device.**
- **Identification of a specific profile on different devices.**
- **Restoration of access to a profile after reinstalling the application.**

Usage

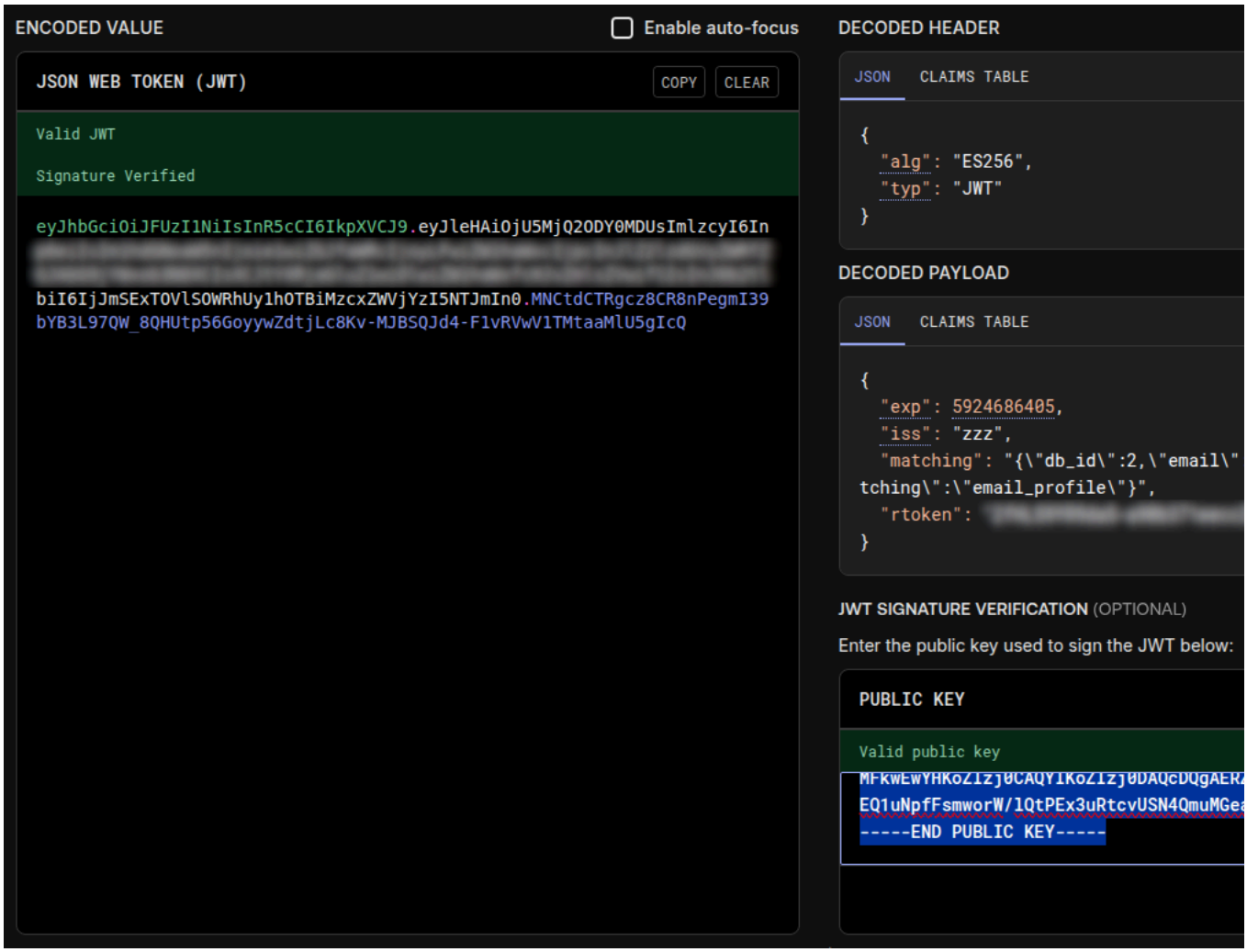
The platform expects the following JWT payload structure:

```
{
  "iss": "<App Name>",
  "exp": <UnixTimeUTC>,
  "rtoken": "<RoleToken>",
  "matching": "JSONString"
}
```

- `iss` — **issuer — unique identifier of the token issuer**
- `exp` — **expiration time — UNIX timestamp in seconds**
- `rtoken` — **the role token obtained during resource configuration**
- `matching` — `JSONString` — **a serialized object composed according to the documentation, for example** `{"db_id":2,"email":"registered_db@localhost","matching":"email_profile"}`

When using the `Altcraft SDK`, the token is passed as an implementation of the `JWTInterface`. The library l header.

Example of a final JWT:



Quick Start

This guide demonstrates how to quickly integrate Altcraft mSDK with Firebase Cloud Messaging (FCM) your app will be able to receive Altcraft push notifications.
To use Huawei Mobile Services (HMS) or RuStore, add their corresponding SDKs to your project and im same approach as for Firebase.

Step 0. Prerequisites

- **Push provider SDKs are integrated into your app project (see the corresponding integration guides**
- **The app includes a class extending `Application()`;**
- **All required entities (resource, mailing, template, profile database) are pre-created in the Altcraft p**
- **On Android 13 (Tiramisu) and later, explicit user permission is required to send push notifications.**

Step 1. Connect Altcraft mSDK to the app

Add the library dependency to your app-level `build.gradle.kts` file:

```
dependencies {
        implementation("com.altcraft:android-sdk:1.0.6")
}
```

Then sync your Gradle project.

Step 2. Implementing JWTInterface

Implement the SDK interface that provides the JWT token. Create a class in your app module that impl the `getJWT()` function:

```
class JWTProvider(private val context: Context /** add context if required */) : JWTInterface {
    override fun getJWT(): String? {
        // your code returning the current JWT token
    }
}
```

> **TIP: Note**
>
> - **`getJWT()` is a synchronous function. The SDK execution thread will be blocked until the JWT is obt**
> - **It's recommended that `getJWT()` return the token immediately from cache (`in-memory`, `SharedPreferen` speed up requests.**
> - **Ideally, prepare and cache the JWT as early as possible (e.g., during app startup) so that the SDK `null` is acceptable if no token is available.**

Example implementation using a suspend function

```
class JWTProvider(private val context: Context /** add context if required */) : JWTInterface {
    private suspend fun fetchJwt(context: Context): String? {
        // your code returning the current JWT token
        return jwt
    }

    override fun getJWT(): String? = runBlocking {
        fetchJwt(context)
    }
}
```

Step 3. Registering the JWT provider

Register the JWT provider in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(/** add context if required */))
    }
}
```

Step 4. Implementing the FCM interface

Create a class in your app module that implements `FCMInterface`:

Example implementation

```
/**
 * FCM implementation for managing Firebase Cloud Messaging tokens.
 *
 * Provides methods to retrieve and delete FCM tokens via the Firebase SDK.
 */
class FCMProvider : FCMInterface {

    /** Retrieves the current FCM token. Returns null if retrieval fails. */
    override suspend fun getToken(): String? {
        return try {
            Firebase.messaging.token.await()
        } catch (e: Exception) {
            null
        }
    }

    /** Deletes the current FCM token. */
    override suspend fun deleteToken(completion: (Boolean) -> Unit) {
        try {
            FirebaseMessaging.getInstance().deleteToken().addOnCompleteListener {
                completion(it.isSuccessful)
            }
        } catch (e: Exception) {
            completion(false)
        }
    }
}
```

Step 5. Creating a service extending `FirebaseMessagingService()`

In your app module, create a class that extends `FirebaseMessagingService` and override `onMessageReceived()` t `AltcraftSDK.PushReceiver.takePush()`:

Example service implementation

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)
        AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

Step 6. Registering the FCM service in AndroidManifest

Register your `FirebaseMessagingService` in the app's `AndroidManifest.xml`:

```
<!-- FCM service -->
<service
    android:name="<your_package_name>.FCMService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Step 7. Registering the FCM provider in `Application.onCreate()`

Register the Firebase Cloud Messaging provider in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
    }
}
```

Step 8. Initializing the SDK

In `Application.onCreate()`, create a configuration object and initialize the SDK using `initialization()`. Mak all providers (JWT and push token providers) are registered.

Initialization details

```
AltcraftSDK
 // SDK initialization and configuration setup
  └── fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null): Unit
```

`fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null)`

Example initialization:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        val config = AltcraftConfiguration.Builder(
            apiUrl = "https://pxl-your-instance.altcraft.com",
            icon = R.drawable.your_icon
        ).build()

        // set additional configuration parameters (see Android mSDK README)

        AltcraftSDK.initialization(context = this, configuration = config)
    }
}
```

Step 9. Subscribing to push notifications

Use the `pushSubscribe()` function to subscribe to push notifications at any point in your code — for exam for notifications:

```
AltcraftSDK
└──  val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    // Push notifications subscription (status = SUBSCRIBED)
    ├─ fun pushSubscribe(
         context: Context,
         sync: Boolean = true,
         profileFields: Map<String, Any?>? = null,
         customFields: Map<String, Any?>? = null,
         cats: List<DataClasses.CategoryData>? = null,
         replace: Boolean? = null,
         skipTriggers: Boolean? = null
       ): Unit
```

Example subscription call:

```
class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        NotificationPermissionHandler.isGranted(
            activity = this,
            onGranted = {
                if (isUnsubscribed()) {
                   AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(context = this)
                }
            }
        )
    }
}
```

SDK Functionality

TIP

Before proceeding, you need to configure the SDK for your application. The detailed setup guide is av

Working with subscription statuses

Changing a subscription status

```
AltcraftSDK
└─ val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
   // Subscribe (status = SUBSCRIBED)
   ├─ fun pushSubscribe(
   │     context: Context,
   │     sync: Boolean = true,
   │     profileFields: Map<String, Any?>? = null,
   │     customFields: Map<String, Any?>? = null,
     │     cats: List<CategoryData>? = null,
     │     replace: Boolean? = null,
     │     skipTriggers: Boolean? = null
     │   ): Unit
     // Suspend (status = SUSPENDED)
     ├─ fun pushSuspend(
     │     context: Context,
     │     sync: Boolean = true,
     │     profileFields: Map<String, Any?>? = null,
     │     customFields: Map<String, Any?>? = null,
     │     cats: List<CategoryData>? = null,
     │     replace: Boolean? = null,
     │     skipTriggers: Boolean? = null
     │   ): Unit
     // Unsubscribe (status = UNSUBSCRIBED)
     └─ fun pushUnSubscribe(
           context: Context,
           sync: Boolean = true,
           profileFields: Map<String, Any?>? = null,
           customFields: Map<String, Any?>? = null,
           cats: List<CategoryData>? = null,
          replace: Boolean? = null,
          skipTriggers: Boolean? = null
        ): Unit
```

Subscription status change functions:

- `fun pushSubscribe()` **— subscribes to push notifications;**
- `fun pushUnSubscribe()` **— cancels the push subscription;**
- `fun pushSuspend()` **— suspends the push subscription (no notifications are delivered, and an unsubscr user's profile);**
- `fun unSuspendPushSubscription()` **— used to implement** `LogIn` / `LogOut` **transitions.**

These functions share the same signature with the following parameters:

---

context: `Context`

Required: Yes Description: Android Context.

---

sync: `Boolean`

Default: `true` **Required: No Description: Flag that sets synchronous execution of the request.**

Successful request execution:

On success, this group of functions emits an event with code 230 containing `event.value`**, which depend**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "7f31a9c4"
           ├ provider: "android-firebase"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: {"raw":"14","ver":"[\"14.0\", \"0.0\", \"0.0\"]"}
           │  ├ _device_type: "mob"
           │  ├ _ad_track: true
           │  ├ _device_name: "Pixel"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0100"
           │  ├ _os: "Android"
           │  ├ _device_model: "Pixel 7"
           │  └ _app_ver: "1.0.0"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, `event.value["response_with_http_code"]` exposes:

- **httpCode – transport status code;**
- **ResponseWithProfile – a data class containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty if no errors)*,**
  - `profile: ProfileData?` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (SubscriptionData)**
    - **subscription categories (CategoryData)**
    - **if the request fails, only `profile = null` will be returned**

▶ **Data structures**

If `sync == false`

```
ResponseWithHttpCode<ResponseWithProfile>
  ├ httpCode: Int?
  └ response: ResponseWithProfile?
     ├ error: Int?
     ├ errorText: String?
     └ profile: ProfileData? = null
```

For asynchronous requests, `event.value["response_with_http_code"]` exposes:

- **httpCode – transport status code;**
- **ResponseWithProfile – a data class containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty if no errors)*,**
  - `profile: ProfileData?` **— profile data, always `null` for async requests.**

Failed request execution:

If a request from this group fails, an event with one of the following codes will be emitted:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only `httpCode` if the Altcraft server was unavailable;**
- **`error` and `errorText` if the server returned an error.**

Reading event values

```
AltcraftSDK.eventSDKFunctions.subscribe { event ->
    val events = setOf(
        SDKEvent.PushSubscribeRequestSuccessful,
        SDKEvent.PushSuspendRequestSuccessful,
        SDKEvent.PushUnsubscribeRequestSuccessful,
        SDKEvent.PushSubscribeRequestFailed,
        SDKEvent.PushSuspendRequestFailed,
        SDKEvent.PushUnsubscribeRequestFailed,
        SDKEvent.PushSubscribeRetryLimit,
        SDKEvent.PushEventRetryLimit,
        SDKEvent.MobileEventRetryLimit
    )

    val sdkEvent = event.event ?: return@subscribe
    if (sdkEvent !in events) return@subscribe

    (event.eventValue?.get("response_with_http_code") as? ResponseWithHttpCode<ResponseWithProfile>)?.let { responseWithHttp ->
        val httpCode = responseWithHttp.httpCode

        val response = responseWithHttp.response
        val error = response?.error
        val errorText = response?.errorText

        val profile = response?.profile
        val profileId = profile?.id
        val profileStatus = profile?.status
        val profileIsTest = profile?.isTest

        val subscription = profile?.subscription
        val subscriptionId = subscription?.subscriptionId
        val hashId = subscription?.hashId
        val provider = subscription?.provider
        val subscriptionStatus = subscription?.status

        val fields = subscription?.fields

        val cats = subscription?.cats
        val firstCat = cats?.firstOrNull()
        val catName = firstCat?.name
        val catTitle = firstCat?.title
        val catSteady = firstCat?.steady
        val catActive = firstCat?.active
    }
}
```

---

profileFields: `Map<String, Any?>?`
Default: `null`
Required: No
Description: A map containing profile fields.
The parameter may include system fields (e.g., `_fname` — first name or `_lname` — last name) as well as c in the platform UI). Allowed JSON-compatible structures:

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Array of maps:** `Array<Map<String, *>>`

If an invalid optional field is passed, the request will fail with:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `Map<String, Any?>?`
Default: `null` **Required: No Description: A map containing subscription fields.**

May include system fields (e.g., `_device_model` — device model or `_os` — operating system) as well as op in the platform UI). Allowed value types (JSON-compatible, scalars only):

- **String**
- **Boolean**
- **Int**
- **Long**
- **Float**
- **Double**
- `null`

If an invalid optional field is passed, the request will fail with:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

> TIP: Note
>
> **Most subscription system fields are collected automatically by the SDK and added to push requests. T `"_os_language"`, `"_device_type"`, `"_device_model"`, `"_device_name"`, `"_os_ver"`, `"_ad_track"`, `"_ad_id"`.**

---

cats: `listOf(CategoryData)`

Default: `null` **Required: No Description: Subscription categories.**

Category structure:

```
data class CategoryData(
    val name: String?,
    val title: String? = null,
    val steady: Boolean? = null,
    val active: Boolean?
)
```

When sending a push request with categories, use only the `name` (category name) and `active` (category are not used for request processing. `title` and `steady` are populated when retrieving subscription info.

Example:

```
val cats = listOf(
    CategoryData(name = "football", active = true),
    CategoryData(name = "hockey",   active = true)
)
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft p in the resource is used, the request will fail:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `Boolean?`

Default: `null` **Required: No Description: When enabled, all subscriptions of other profiles with the same database are set to `unsubscribed` after a successful request.**

---

skipTriggers: `Boolean?`

Default: `null` **Required: No Description: When enabled, the profile containing this subscription will be ig triggers.**

Request implementation examples

Example: subscribing to push notifications

Minimal working setup:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(context)
```

Passing all available parameters:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
    context = this,
    sync = true,
    profileFields = mapOf("_fname" to "Andrey", "_lname" to "Pogodin"),
    customFields  = mapOf("developer" to true),
    cats = listOf(CategoryData(name = "developer_news", active = true)),
    replace = false,
    skipTriggers = false
)
```

> TIP: Note

For `pushSubscribe`, `pushSuspend`, and `pushUnSubscribe`, the SDK automatically retries the request if the HTT retry occurs for codes outside this range.

Function `unSuspendPushSubscription()`

The function `fun unSuspendPushSubscription()` is intended to implement `LogIn` / `LogOut` transitions. It works a

- **searches for subscriptions with the same push token as the current one that do not belong to the p JWT token;**
- **changes the status of the found subscriptions from `subscribed` to `suspended`;**
- **changes the status of subscriptions in the profile referenced by the current JWT from `suspended` to profile exists and contains subscriptions);**
- **returns a `ResponseWithHttpCode?` data class in which `response.profile` is the current profile referenced b exist, `response.profile` will be `null`).**

Recommended `LogIn` / `LogOut` implementation

`LogIn` transition:

- **An anonymous user enters the app. The user is assigned `JWT_1`, pointing to database #1Anonymous**
- **A push subscription is created; the profile is created in #1Anonymous.**
- **The user registers and is assigned `JWT_2`, pointing to database #2Registered.**
- **Call `unSuspendPushSubscription()` — the anonymous user's subscription in #1Anonymous is suspended.**
- **Search the profile in #2Registered to restore the subscription.**
- **Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()`**
- **After receiving `null`, call `pushSubscribe()` to create a new profile in #2Registered.**

`LogOut` transition:

- **The user logs out in the app (`LogOut`).**
- **The user is assigned `JWT_1`, pointing to database #1Anonymous.**
- **Call `unSuspendPushSubscription()`, which will suspend the subscription in #2Registered and set the sub `subscribed`.**
- **The request returns `#1Anonymous != null` — the subscription already exists; no new one is required.**

Example:

```
private suspend fun unSuspend(context: Context, logIn: Boolean) {

    // Change JWT before the request
    setAuth(context, logIn)

    AltcraftSDK.pushSubscriptionFunctions
        .unSuspendPushSubscription(context)
        ?.let { result ->
            if (result.httpCode == 200 && result.response?.profile?.subscription == null) {
                AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
                    context = context
                    // Specify required params
                )
            }
        }
}

fun logIn(context: Context)  = CoroutineScope(Dispatchers.IO).launch { unSuspend(context, true) }
fun logOut(context: Context) = CoroutineScope(Dispatchers.IO).launch { unSuspend(context, false) }
```

Getting a subscription status

```
AltcraftSDK
└── val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    // Status of the profile's latest subscription
    ├── suspend fun getStatusOfLatestSubscription(
    │       context: Context
    │   ): ResponseWithHttpCode<ResponseWithProfile>?
    // Status for the current token/provider
    ├── suspend fun getStatusForCurrentSubscription(
    │       context: Context
    │   ): ResponseWithHttpCode<ResponseWithProfile>?
    // Status of the latest subscription for the given provider (if null – current provider is used)
    └── suspend fun getStatusOfLatestSubscriptionForProvider(
            context: Context,
            provider: String? = null
        ): ResponseWithHttpCode<ResponseWithProfile>?
```

Subscription status request functions:

- **`fun getStatusOfLatestSubscription()` — status of the profile's latest subscription;**
- **`fun getStatusOfLatestSubscriptionForProvider()` — status for the current token/provider;**
- **`fun getStatusForCurrentSubscription()` — status of the latest subscription by provider. If `null` is passed,**

`suspend fun getStatusOfLatestSubscription(context: Context): ResponseWithHttpCode<ResponseWithProfile>?`

Retrieves the status of the profile's latest subscription. Returns a `ResponseWithHttpCode` object containing latest subscription created in the profile. If no such subscription exists, `null` is returned.

Usage example:

```
AltcraftSDK.pushSubscriptionFunctions.getStatusOfLatestSubscription(context)
```

`suspend fun getStatusForCurrentSubscription(context: Context): ResponseWithHttpCode<ResponseWithProfile>?`

Retrieves the status for the current token/provider. Returns a `ResponseWithHttpCode` object containing `resp` subscription found by the current push token and provider. If no such subscription exists, `null` is retur

Usage example:

```
AltcraftSDK.pushSubscriptionFunctions.getStatusForCurrentSubscription(context)
```

`suspend fun getStatusOfLatestSubscriptionForProvider(context: Context, provider: String? = null): ResponseWithHttpCode`

Retrieves the latest subscription status by provider. Returns a `ResponseWithHttpCode` object containing `res` latest subscription created with the specified push provider. If the provider is not specified (`provider = n` used. If no such subscription exists, `null` is returned.

```
AltcraftSDK.pushSubscriptionFunctions.getStatusOfLatestSubscriptionForProvider(context, provider = null)
```

Below is an example of extracting profile, subscription, and category data from a status response. This fetching functions:

Data from status functions

```
CoroutineScope(Dispatchers.IO).launch {
    AltcraftSDK.pushSubscriptionFunctions
        .getStatusForCurrentSubscription(this@App)
        ?.let { it ->
            val httpCode = it.httpCode
            val response = it.response
            val error = response?.error
            val errorText = response?.errorText
            val profile = response?.profile
            val subscription = profile?.subscription
            val cats = subscription?.cats
        }
}
```

Managing provider push tokens

```
AltcraftSDK
└── val pushTokenFunctions: PublicPushTokenFunctions
    |
    // Save a token manually (onNewToken)
    ├── fun setPushToken(context: Context, provider: String?, token: String?): Unit
    |
    // Get current device token data
    ├── suspend fun getPushToken(context: Context): TokenData?
    |
    // Set Firebase Cloud Messaging provider (null — remove)
    ├── fun setFCMTokenProvider(provider: FCMInterface?): Unit
    |
    // Set Huawei Mobile Services provider (null — remove)
    ├── fun setHMSTokenProvider(provider: HMSInterface?): Unit
    |
    // Set RuStore provider (null — remove)
    ├── fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit
    |
    // Delete the push token for the specified provider
    ├── fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit
    |
    // Force token refresh (delete -> refresh)
    ├── fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit
    |
    // Change provider priority list and trigger token refresh
    └── suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit
```

SDK functions for working with a provider token:

- `fun setPushToken()` **— manually set the device push token and provider in `SharedPreferences`;**
- `fun getPushToken()` **— get the current push token;**
- `fun setFCMTokenProvider()` **— set or clear the FCM push token;**
- `fun setHMSTokenProvider()` **— set or clear the HMS push token;**
- `fun setRuStoreTokenProvider()` **— set or clear the RuStore push token;**

- `fun changePushProviderPriorityList()` — dynamically switch provider and refresh the subscription's push
- `fun deleteDeviceToken()` — delete the push token of the specified provider;
- `fun forcedTokenUpdate()` — delete the current push token and then refresh it.

`fun setPushToken(context: Context, provider: String?, token: String?): Unit`

Used to manually set the device push token and provider. Should be called in the push provider service simplified way to pass the token to the SDK without implementing provider interfaces.

Using this function to pass the token is not recommended. The recommended approach is to implemen `APNSInterface`.

Usage example:

```
AltcraftSDK.pushTokenFunctions.setPushToken(context, provider, token)
```

Example of passing a token in `FCMService.onNewToken()`:

```
class FCMService : FirebaseMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)

        // Pass the new token manually
        AltcraftSDK.pushTokenFunctions.setPushToken(this, FCM_PROVIDER, token)
    }

    override fun onDeletedMessages() {}

    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)
        AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

`suspend fun getPushToken(context: Context): TokenData?`

The function returns the current device push-token data and its provider as the data class `TokenData(val` the token is unavailable, `null` will be returned.

Usage example:

```
AltcraftSDK.pushTokenFunctions.getPushToken(context)
```

Example of a push token data request:

```
CoroutineScope(Dispatchers.IO).launch {
    AltcraftSDK.pushTokenFunctions.getPushToken(context).let {
        val provider = it?.provider
        val token = it?.token
    }
}
```

`fun setFCMTokenProvider(provider: FCMInterface?): Unit`

Sets or removes the Firebase Cloud Messaging push-token provider. To disable the provider, pass `null`.

Usage example:

```
AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
```

> TIP: Important
>
> **Call `setFCMTokenProvider()` in `Application.onCreate()` before calling `AltcraftSDK.initialization()`. This guarant app process, regardless of the lifecycle state and whether the app is running in the foreground or bac**

`fun setHMSTokenProvider(provider: HMSInterface?): Unit`

Sets or removes the Huawei Mobile Services push-token provider. To disable the provider, pass `null`.

Usage example:

```
AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
```

> TIP: Important
>
> **Call `setHMSTokenProvider()` in `Application.onCreate()` before calling `AltcraftSDK.initialization()`. This guarant app process, regardless of the lifecycle state and whether the app is running in the foreground or bac**

`fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit`

Sets or removes the RuStore push-token provider. To disable the provider, pass `null`.

Usage example:

```
AltcraftSDK.pushTokenFunctions.setRuStoreTokenProvider(RuStoreProvider())
```

> TIP: Important

Call `setRuStoreTokenProvider()` in `Application.onCreate()` before calling `AltcraftSDK.initialization()`. This guar the app process, regardless of the lifecycle state and whether the app is running in the foreground or

---

`suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit`

Allows dynamic switching of the push provider with subscription token refresh. Pass a new list with a example: `listOf(HMS_PROVIDER, RUSTORE_PROVIDER, FCM_PROVIDER)`.

```
AltcraftSDK.pushTokenFunctions.changePushProviderPriorityList(context, listOf(HMS_PROVIDER, RUSTORE_PROVIDER, FCM_PROVIDER))
```

`fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit`

Deletes the push token for the specified provider. The token is invalidated and removed from the device provider's server. After deletion, a new token can be requested.

```
AltcraftSDK.pushTokenFunctions.deleteDeviceToken(context, provider) {
    // callback after deletion
}
```

`fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit`

Deletes the current push token and then refreshes it.

Usage example:

```
AltcraftSDK.pushTokenFunctions.forcedTokenUpdate(context) {
    // callback after update
}
```

`suspend fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit`

Deletes the push token for the specified provider. The token is invalidated and removed from the device provider's server. After deletion, a new token can be requested.

```
// Delete the push token for the given provider (invalidate locally and on the server)
AltcraftSDK.pushTokenFunctions.deleteDeviceToken(context, provider) {
    // callback after deletion
}
```

`fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit`

Deletes the current push token and then refreshes it.

Usage example:

```
// Force token refresh (delete -> update)
AltcraftSDK.pushTokenFunctions.forcedTokenUpdate(context) {
    // callback after update
}
```

Example of provider registration

We do not recommend using the `setPushToken` function to set a push token manually. Instead, configure for each provider you use. Below is an example implementation of this approach:

Recommended provider registration method

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        // initialize RuStore provider
        RuStorePushClient.init(this, "rustore project id")

        // set JWT provider
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))

        // set FCM provider
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        // set HMS provider
        AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())

        // set RuStore provider
        AltcraftSDK.pushTokenFunctions.setRuStoreTokenProvider(RuStoreProvider())

        // create AltcraftConfiguration
        val config = AltcraftConfiguration.Builder(
            apiUrl = "your api url",
            R.drawable.ic_altcraft_label
        ).build()

        // initialize SDK
        AltcraftSDK.initialization(context = this@App, configuration = config)
    }
}
```

Passing push notifications to the SDK

`PushReceiver` **is a public class that contains the function that accepts notifications.**

```
AltcraftSDK
└── open class PushReceiver
    // Incoming push handling
    ├── open suspend fun pushHandler(
    │       context: Context,
    │       message: Map<String, String>
    │   ): Unit
    // SDK entry point for push delivery
    └── companion object
        └── fun takePush(
                context: Context,
                message: Map<String, String>
            ): Unit
```

Functions for passing push notifications to the SDK:

- `fun takePush()` **— accepts notifications in the push providers' service for further processing on the S**
- `fun pushHandler()` **— launches the standard Altcraft push-notification handling mechanism in the SDK**

Receiving a notification

`fun takePush(context: Context, message: Map<String, String>): Unit`

Accepts push notifications for further processing on the SDK side.

Usage example:

```
   override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)

       AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
    // message.data – message payload
```

Handling a notification

`open suspend fun pushHandler(context: Context, message: Map<String, String>): Unit`

Starts the standard Altcraft push-notification handling mechanism in the SDK.

Receiving notifications in any app package (optional)

To receive notifications in any application package, do the following:

Step 1. Create an `AltcraftPushReceiver` **class extending** `PushReceiver`**. Override the** `pushHandler()` **function in**

```
@Keep
class AltcraftPushReceiver : AltcraftSDK.PushReceiver() {
    override suspend fun pushHandler(context: Context, message: Map<String, String>) {
        // standard processing of push messages and displaying notifications
        super.pushHandler(context, message)
    }
}
```

TIP: Note

The class must be named AltcraftPushReceiver. If you name it differently, the SDK will not be able to

Step 2. Depending on your business goals, configure the logic of the `AltcraftPushReceiver` class:

- **If you want the SDK to handle and display notifications:**
  - **use `super.pushHandler(context, message)`.**
- **If you want to handle notifications yourself:**
  - **do not call `super.pushHandler()`;**
  - **manually register open events using the `openEvent()` function; otherwise, it will not be registere**

"deliveryEvent" delivery events are registered automatically. Creating `AltcraftPushReceiver` classes does event.

TIP: Note

If you created multiple `AltcraftPushReceiver` classes, calling `super.pushHandler()` in each of them will displ avoid duplicate notifications, call `super.pushHandler()` in only one class.

Step 3. Add the package names that contain `AltcraftPushReceiver` classes to the `pushReceiverModules` config SDK will automatically detect the presence of `AltcraftPushReceiver` classes in the specified packages via If the application code will be obfuscated, the class must be annotated with `@Keep` or added to the `R8/P` will not be able to discover it.

Example of adding a package to the `pushReceiverModules` parameter:

```
pushReceiverModules = listOf(
   context.packageName, // application package
   "com.altcraft.altcraftmobile.test"
)
```

Mobile Events

```
// Function of the PublicMobileEventFunctions object

AltcraftSDK
└── val mobileEventFunctions: PublicMobileEventFunctions

    // Send a mobile event to the server
    └── fun mobileEvent(
        context: Context,
        sid: String,
        eventName: String,
        sendMessageId: String? = null,
        payload: Map<String, Any?>? = null,
        matching: Map<String, Any?>? = null,
        matchingType: String? = null,
        profileFields: Map<String, Any?>? = null,
        subscription: Subscription? = null,
        utm: UTM? = null
    ): Unit
```

Case: Transferring advertising campaign information

Information about the app advertising campaign that led to the installation can be sent to the platform `customFields` parameters of the `pushSubscribe` function, as well as the `utm` or `payload` parameters of the `m` receiving a UTM tag in the app as a string, you need to call the `mobileEvent()` function:

Transfer via payload

```
AltcraftSDK.mobileEventFunctions.mobileEvent(
    this,
    sid = "your sid",
    eventName = "app_install",
    payload = mapOf("utm" to "your utm tag")
)
```

696a08e91275353a39ee3991

пт, 16 янв. 2026 г. 12:46:17

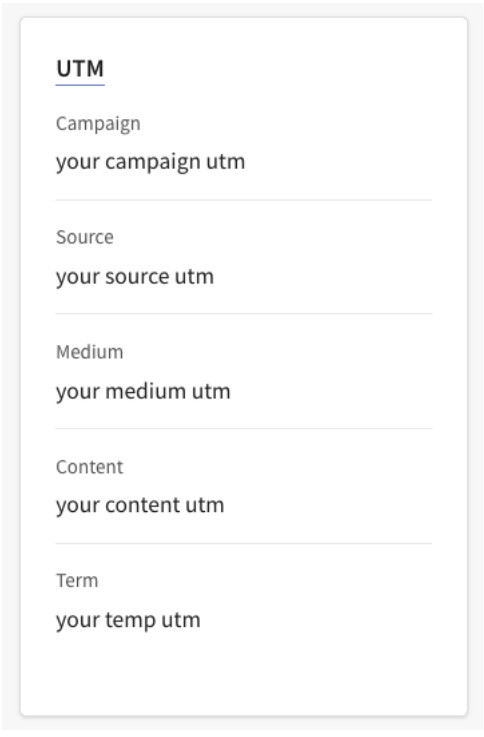
Открыт Pixel

Достигнута цель

- **Цели:** android_app_install
- **Пиксель:** #1 pixel
- **UTM Content:** your content utm
- **UTM Source:** your source utm
- **UTM Campaign:** your campaign utm
- **UTM Term:** your temp utm

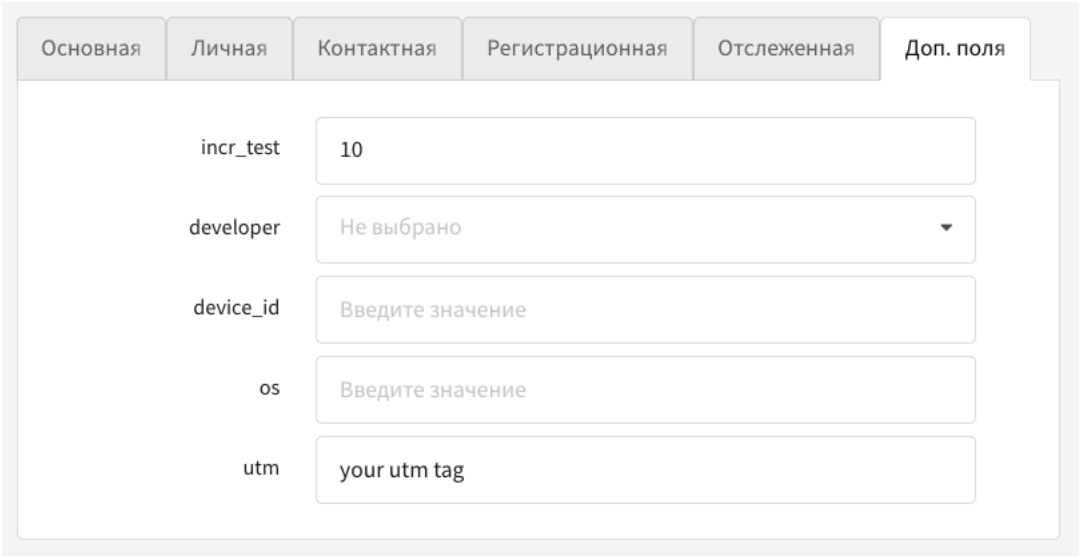
Transfer to UTM fields

```
AltcraftSDK.mobileEventFunctions.mobileEvent(
    this,
    sid = "your sid",
    eventName = "app_install",
    utm = DataClasses.UTM(
        campaign = "your campaign utm",
        content = "your content utm",
        keyword = "your keyword utm",
        medium = "your medium utm",
        source = "your source utm",
        temp = "your temp utm"
    )
)
```



Transfer to a custom profile field

```
// transfer a UTM tag to a profile field
// profile fields must be created in the platform in advance
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: ["utm": "your utm tag"]
)
```



Data sent using any of these methods can be used to create a segment in the platform.
To register a mobile event, use the `mobileEvent()` function. It has the following parameters:

---

context: `Context`
Required: Yes
Description: Android Context.

---

sid: `String`
Required: Yes
Description: String identifier of the pixel to which mobile events are linked.

---

eventName: `String`
Required: Yes
Description: The name of the mobile event.

---

sendMessageId: `String?`
Required: No
Description: SMID identifier of the sent message (if the event is associated with a specific mailing).

---

payload: `String`
Required: No
Description: Event data — a map with string keys and values, where only scalar data types are allowed

- **String**
- **Boolean**
- **Int**
- **Long**
- **Float**
- **Double**
- `null`

---

matching: `Map<String, Any?>?`
Required: No
Description: A map for passing values with matching types and identifiers.

matchingType: `String?`
Required: No
Description: The type of matching.

---

profileFields: `Map<String, Any?>?`
Required: No
Description: Profile fields — a map with string keys and JSON-compatible values:

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Arrays of maps:** `Array<Map<String, *>>`

> TIP: Note
>
> **This parameter is used only when working with JWT authorization.**

---

utm: `UTM?`
Required: No
Description: UTM tags. Added using the `UTM` **data class, where each UTM type is a separate parameter.**

```
data class UTM (
    val campaign: String? = null,
    val content: String? = null,
    val keyword: String? = null,
    val medium: String? = null,
    val source: String? = null,
    val temp: String? = null
)
```

---

subscription: `Subscription?`
Required: No
Description: Parameter for adding a subscription for a selected channel.
Possible values are implementations (subtypes) of the sealed interface `Subscription`**:**

- `EmailSubscription` **— email subscription**
- `SmsSubscription` **— SMS subscription**
- `PushSubscription` **— push subscription**
- `CcDataSubscription` **— subscription in Telegram, WhatsApp, Viber, or Notify.**

> TIP: Note
>
> **Used only when working with JWT authorization.**

Implementations of the `Subscription` interface

General model: `Subscription`
Purpose — base (sealed) interface for all types of subscriptions. Serialization is polymorphic, with the
Common fields (for all implementations):

Field	Type	Required	Description
`resource_id`	`Int`	Yes	Identifier of the subscription resource/source
`status`	`String?`	No	Subscription status (e.g., active/suspended)
`priority`	`Int?`	No	Delivery priority for the subscription
`custom_fields`	`Map<String, Any?>?`	No	Custom fields (key–value) for extended segmentation
`cats`	`List<String>?`	No	Subscription categories
`channel`	`String`	Yes	Channel type; defined by the implementation

Subscription types:

EmailSubscription (`channel = "email"`)

Required fields

Field	Type	Required	
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`email`	`String`	**Yes**	**Recipient email address**

Optional fields

Field	Type	Required	Des
`status`	`String`	**No**	**Subscription status**
`priority`	`Int`	**No**	**Subscription priority**
`customFields`	`Map<String, Any?>`	**No**	**Standard and custom subscription fields**
`cats`	`List<String>`	**No**	**Subscription categories**

SmsSubscription (`channel = "sms"`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`phone`	`String`	**Yes**	**Phone number in international format**

Optional fields

Field	Type	Required	Des
`status`	`String`	**No**	**Subscription status**
`priority`	`Int`	**No**	**Subscription priority**
`customFields`	`Map<String, Any?>`	**No**	**Standard and custom subscription fields**
`cats`	`List<String>`	**No**	**Subscription categories**

PushSubscription (`channel = "push"`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`provider`	`String`	**Yes**	**Provider (e.g., `"android-firebase"`)**
`subscriptionId`	`String`	**Yes**	**Unique subscription identifier from the provider**

Optional fields

Field	Type	Required	Des
`status`	`String`	**No**	**Subscription status**
`priority`	`Int`	**No**	**Subscription priority**
`customFields`	`Map<String, Any?>`	**No**	**Standard and custom subscription fields**
`cats`	`List<String>`	**No**	**Subscription categories**

CcDataSubscription (`channel ∈ {"telegram_bot","whatsapp","viber","notify"}`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`channel`	`String`	**Yes**	**One of: `"telegram_bot"` , `"whatsapp"` , `"viber"` , `"notify"`**
`ccData`	`JsonObject`	**Yes**	**Channel-specific data (e.g., chat ID, phone number, tokens)**

Optional fields

Field	Type	Required	Des
`status`	`String`	**No**	**Subscription status**
`priority`	`Int`	**No**	**Subscription priority**
`customFields`	`Map<String, Any?>`	**No**	**Standard and custom subscription fields**
`cats`	`List<String>`	**No**	**Subscription categories**

Profile Fields Update

> **INFO: Important**
>
> **Updating profile fields with the `updateProfileFields()` function works only when using JWT authorizatio**

```
AltcraftSDK
└── val profileFunctions: PublicProfileFunctions
    // Updates Altcraft profile fields
    └── fun updateProfileFields(
        context: Context,
        profileFields: Map<String, Any?>? = null,
        skipTriggers: Boolean? = null
    ): Unit
```

To update profile fields, use the `updateProfileFields()` function. It has the following parameters:

---

context: `Context`
Required: Yes
Description: Android Context.

---

profileFields: `Map<String, Any?>?`
Required: No
Description: Profile fields — a map containing the values for the fields that need to be updated (JSON-c

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Array of maps:** `Array<Map<String, *>>`

> INFO: Note
> **For the update to succeed, the fields must have been previously added to the subscriber's profile.**

---

skipTriggers: `Boolean?`
Default: `null`
Required: No
Description: When enabled, the profile containing this subscription will be ignored in mailing triggers a

Receiving SDK Events in the Application

```
// Functions of the Events object

AltcraftSDK
└── val eventSDKFunctions: Events
    // Subscribe to SDK events (replaces the existing subscriber)
    ├── fun subscribe(
    │       newSubscriber: (Event) -> Unit
    │   ): Unit
    // Unsubscribe from events
    └── fun unsubscribe(): Unit
```

The application can have only one active subscriber to SDK events.

Event Subscription

`fun subscribe(newSubscriber: (Event) -> Unit): Unit`

Used to subscribe and receive SDK events in the application. When an SDK event occurs, it calls the `c` the `Event` class (or one of its subclasses).

Usage example:

```
AltcraftSDK.eventSDKFunctions.subscribe { event ->
    // Handle event
}
```

All events provided by the SDK are instances of `Event` or its subclasses:

- **Event — general event (info, successful requests);**
- **Error — error event;**
- **RetryError — error event for a request that will be automatically retried by the SDK.**

Each event contains the following fields:

- **function — the name of the function that triggered the event;**
- **event — SDK event type, represented by an `SDKEvent` enum value (see "SDK Events" for the full list);**
- **eventMessage — event message;**
- **eventValue — arbitrary `[String: Any?]?` data passed as event payload;**
- **date — timestamp of the event;**
- **internal — flag indicating whether the event is internal.**

SDK Event classes

```
open class Event(
    val function: String,
    val event: SDKEvent? = null,
    val eventMessage: String? = null,
    val eventValue: Map<String, Any?>? = null,
    val date: Date = Date(),
    val internal: Boolean = false
)

open class Error(
    function: String,
    event: SDKEvent? = null,
    eventMessage: String? = null,
    eventValue: Map<String, Any?>? = null,
    date: Date = Date(),
) : Event(function, event, eventMessage, eventValue, date)

class RetryError(
    function: String,
    event: SDKEvent? = null,
    eventMessage: String? = null,
    eventValue: Map<String, Any?>? = null,
    date: Date = Date(),
) : Error(function, event, eventMessage, eventValue, date)
```

Example of a successful push subscription event

```
├─ function: processResponse
├─ event: PushSubscribeRequestSuccessful
├─ eventMessage: "successful request: push/subscribe"
├─ eventValue
│  ├─ http code: 200
│  └─ response
│     ├─ error: 0
│     ├─ errorText: ""
│     └─ profile
│        ├─ id: "your id"
│        ├─ status: "subscribed"
│        ├─ acid: "your acid"
│        ├─ isTest: false
│        └─ subscription
│           ├─ subscriptionId: "your subscriptionId"
│           ├─ hashId: "c52b28d2"
│           ├─ provider: "android-firebase"
│           ├─ status: "subscribed"
│           ├─ fields
│           │  ├─ _device_name: "Pixel 7"
│           │  ├─ _device_model: "Google Pixel 7"
│           │  ├─ _os_tz: "+0300"
│           │  ├─ _os_language: "ru"
│           │  ├─ _os_ver: {"raw":"14","ver":[14]}
│           │  ├─ _ad_track: true
│           │  ├─ _os: "Android"
│           │  └─ _device_type: "mob"
│           └─ cats
│              └─ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
└─ date: 2025-09-03 09:01:44 +0000
```

Unsubscribing from events

`fun unsubscribe(): Unit`

This function unsubscribes from SDK events. It stops event delivery while keeping the callback assigne

Usage example:

```
// Unsubscribe from SDK events
AltcraftSDK.eventSDKFunctions.unsubscribe()
```

List of all SDK events

SDKEvent enum event list

SDK events are represented by the `SDKEvent` enum class, where each value contains a string message (

General SDK events:

Value	Message
`ConfigIsSet`	**SDK configuration is installed**
`PushProviderSet`	**push provider set:**
`NotAcPush`	**received a notification unrelated to the Altcraft Platform**

Value	Message
`AcPush`	**received Altcraft push notification**
`PushIsPosted`	**push is posted**
`SdkCleared`	**SDK data has been cleared**
`ReceiverRedefined`	**receiver is redefined**
`NotAuthenticatedError`	**User is not authenticated**
`UserLogOut`	**User logged out. Anonymous session started.**

Internal errors without retry:

Value	Message
`ConfigIsNotSet`	**the configuration is not set**
`ConfigNotValid`	**invalid configuration**
`SdkInitWaitingExpired`	**SDK initialization timeout has expired**
`MobileEventPartsIsNull`	**mobile event parts is null**
`InvalidPushProvider`	**invalid provider. Available - android-firebase, android-huawei, android-rustore.**
`PushDataIsNull`	**push data is null**
`PushUidIsNull`	**uid in the push data is null or empty, it is impossible to send a push event to the server.**
`ErrorImgLoad`	**error uploading the notification image**
`NotificationErr`	**couldn’t create notification**
`ChannelNotCreated`	**the notification channel has not been created**
`SmidIsNull`	**send message ID is null**
`InvalidStaticField`	**invalid in-app static field**
`RoomMigrationError`	**Room database migration failed. The local SDK database will be recreated.**

Missing request payload (without retry):

Value	
`UnSuspendRequestDataIsNull`	**unsuspend request data is null**
`SubscriptionStatusRequestDataIsNull`	**status request data is null**
`InAppSubscriptionStatusRequestDataIsNull`	**status request data is null**
`AuthorizeRequestDataIsNull`	**authorize request data is null**

Retry limit reached:

Value	Message
`PushSubscribeRetryLimit`	**push subscribe retry limit**
`PushEventRetryLimit`	**push event retry limit**
`MobileEventRetryLimit`	**mobile event retry limit**
`ProfileUpdateRetryLimit`	**profile update retry limit**
`InAppSubscribeRetryLimit`	**in-app subscribe retry limit**
`AuthRequestRetryLimit`	**auth request retry limit**
`InAppEventRetryLimit`	**in-app event retry limit**

SDK or environment state issues:

Value	Message
`ConfigIsNull`	**config data is null**
`PushTokenIsNull`	**current push token is null**
`UserTagIsNull`	**userTag is null. It is impossible to identify the user**
`PermissionDenied`	**no permission to send notifications**
`NoInternetConnect`	**no internet connection, retry when connection is restored**
`NotUpdated`	**failed to update the push token. The subscription request was rejected**
`ResponseDataIsNull`	**response data is null**
`JwtTokenIsNull`	**JWT token is null**
`MatchingIsNull`	**matching mode is null**
`JwtTooLarge`	**JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS**
`PayloadIsMissing`	**JWT does not contain a payload**
`InternalAuthIsNull`	**internal auth data is null**
`ExternalAuthIsNull`	**external auth data is null**
`MatchingIdIsNull`	**matching claim does not contain a matching ID**
`InvalidPayload`	**invalid mobile event payload: not all values are primitives**
`InvalidCustomFields`	**invalid custom fields: not all values are primitives**

Missing request payload (with retry):

Value	Message
`PushSubscribeRequestDataIsNull`	**push subscribe request data is null**
`TokenUpdateRequestDataIsNull`	**token update request data is null**
`PushEventRequestDataIsNull`	**push event request data is null**
`InAppEventRequestDataIsNull`	**in-app event request data is null**
`MobEventRequestDataIsNull`	**mobile event request data is null**
`ProfileUpdateRequestDataIsNull`	**profile update request data is null**
`InAppSubscribeRequestDataIsNull`	**profile update request data is null**
`InAppPlacementsRequestDataIsNull`	**in-app placements request data is null**

Successful API requests:

Value	Message
`PushUnsubscribeRequestSuccessful`	**successful request: push/unsubscribe**
`PushSubscribeRequestSuccessful`	**successful request: push/subscribe**
`PushSuspendRequestSuccessful`	**successful request: push/suspend**
`InAppUnsubscribeRequestSuccessful`	**successful request: inapp/unsubscribe**
`InAppSubscribeRequestSuccessful`	**successful request: inapp/subscribe**
`InAppStatusRequestSuccessful`	**successful request: inapp/status**
`InAppEventRequestSuccessful`	**successful request: inapp/event**
`InAppPlacementsRequestSuccessful`	**successful request: in_app/placements**
`PushStatusRequestSuccessful`	**successful request: push/status**
`TokenUpdateRequestSuccessful`	**successful request: push/update**
`PushEventRequestSuccessful`	**successful request: event/push**
`MobileEventRequestSuccessful`	**successful request: event/post**
`UnsuspendRequestSuccessful`	**successful request: push/unsuspend**
`ProfileUpdateRequestSuccessful`	**successful request: profile/update**
`AuthenticateSuccessful`	**successful request: profile/authenticate**

Failed API requests:

Value	Message
`PushUnsubscribeRequestFailed`	**failed request: push/unsubscribe**
`PushSubscribeRequestFailed`	**failed request: push/subscribe**
`PushSuspendRequestFailed`	**failed request: push/suspend**
`InAppUnsubscribeRequestFailed`	**failed request: inapp/unsubscribe**
`InAppSubscribeRequestFailed`	**failed request: inapp/subscribe**
`InAppStatusRequestFailed`	**failed request: inapp/status**
`InAppEventRequestFailed`	**failed request: inapp/event**
`InAppPlacementsRequestFailed`	**failed request: in_app/placements**
`PushStatusRequestFailed`	**failed request: push/status**
`TokenUpdateRequestFailed`	**failed request: push/update**
`PushEventRequestFailed`	**failed request: event/push**
`MobileEventRequestFailed`	**failed request: event/post**
`ProfileUpdateRequestFailed`	**failed request: profile/update**
`UnsuspendRequestFailed`	**failed request: push/unsuspend**
`AuthenticateFailed`	**failed request: profile/authenticate**
`InAppStaticErrorLoad`	**error loading in-app notification static resources**
`InAppContentErrorLoad`	**error loading in-app notification content**

In-App Notifications

```
AltcraftSDK
└── val inAppFunctions: PublicInAppFunctions
    // Configure In-App notification animation
    ├── fun setAnimation(block: Animator.() -> Unit): Unit
    //
    // Set current screen marker (used for In-App notification filtering)
    ├── fun setScreen(screen: String): Unit
    //
    // Custom In-App notification trigger by name
    ├── fun inAppTrigger(context: Context, name: String): Unit
    //
    // Request available In-App placements
    ├── fun getInAppPlacements(context: Context): Unit
    //
    // Subscribe to Activity lifecycle for automatic In-App display
    ├── fun subscribeOnActivityLifecycle(context: Context): Unit
    //
    // Subscribe to In-App notifications (status = SUBSCRIBED)
    ├── fun inAppSubscribe(
    │     context: Context,
    │     sync: Boolean = true,
    │     profileFields: Map<String, Any?>? = null,
    │     customFields: Map<String, Any?>? = null,
    │     cats: List<CategoryData>? = null,
    │     replace: Boolean? = null,
    │     skipTriggers: Boolean? = null
    │   ): Unit
    //
    // Unsubscribe from In-App notifications (status = UNSUBSCRIBED)
    ├── fun inAppUnSubscribe(
    │     context: Context,
    │     sync: Boolean = true,
    │     profileFields: Map<String, Any?>? = null,
    │     customFields: Map<String, Any?>? = null,
    │     cats: List<CategoryData>? = null,
    │     replace: Boolean? = null,
    │     skipTriggers: Boolean? = null
    │   ): Unit
    //
    // In-App notification subscription status
    └── suspend fun getInAppSubscriptionStatus(context: Context): ResponseWithHttpCode<ResponseWithProfile>?
```

Functions for working with In-App notifications:

- `fun setAnimation()` — **configure In-App notification display animation;**
- `fun setScreen()` — **set current screen marker, used for In-App notification filtering;**
- `fun inAppTrigger()` — **trigger custom In-App notification by name;**
- `fun getInAppPlacements()` — **request available In-App placements from the server;**
- `fun subscribeOnActivityLifecycle()` — **subscribe to Activity lifecycle for automatic In-App notification di**
- `fun inAppSubscribe()` — **subscribe to In-App notifications;**
- `fun inAppUnSubscribe()` — **unsubscribe from In-App notifications;**
- `suspend fun getInAppSubscriptionStatus()` — **get In-App notification subscription status.**

`fun setAnimation(block: Animator.() -> Unit): Unit`

Allows configuring custom animation for In-App notification display. Accepts an extension function on parameters can be set.

Usage example:

```
AltcraftSDK.inAppFunctions.setAnimation {
    // animation configuration
}
```

`fun setScreen(screen: String): Unit`

Sets the current screen marker. The marker value is used by the Altcraft platform for In-App notificatio notifications only on specific app screens.

Call this function when navigating to a new screen so the SDK always knows which screen the user is

Usage example:

```
AltcraftSDK.inAppFunctions.setScreen("home_screen")
```

`fun inAppTrigger(context: Context, name: String): Unit`

Manually triggers the display of an In-App notification by name. Used for custom triggers when a notifi response to a user action or app event.

Usage example:

```
AltcraftSDK.inAppFunctions.inAppTrigger(context, "my_custom_trigger")
```

`fun getInAppPlacements(context: Context): Unit`

Requests available In-App placements from the Altcraft server. The request result can be retrieved via

Usage example:

```
AltcraftSDK.inAppFunctions.getInAppPlacements(context)
```

`fun subscribeOnActivityLifecycle(context: Context): Unit`

Subscribes the SDK to the Activity lifecycle for automatic In-App notification display. Upon registration lifecycle events and automatically displays In-App notifications when the app comes to the foreground.

> TIP: Important
>
> **Call `subscribeOnActivityLifecycle()` as early as possible at app startup — in `Application.onCreate()`. This is r lifecycle callback is registered before the first foreground transition. If the call is delayed, In-App noti displayed at app launch may be missed.**

Usage example:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        // Register as early as possible
        AltcraftSDK.inAppFunctions.subscribeOnActivityLifecycle(this)

        // Remaining initialization
    }
}
```

`fun inAppSubscribe(context: Context, ...): Unit`

Subscribes to In-App notifications. After a successful call, the user's profile will be subscribed to recei Altcraft platform.

Function parameters are the same as `pushSubscribe()`:

- `context` — **Android Context;**
- `sync` — **synchronous execution flag;**
- `profileFields` — **profile fields;**
- `customFields` — **subscription fields;**
- `cats` — **subscription categories;**
- `replace` — **when enabled, all subscriptions of other profiles with the same push token are set to `uns`**
- `skipTriggers` — **when enabled, the profile will be ignored by mailing and scenario triggers.**

Parameters are configured exactly the same way as in `pushSubscribe()`.

Usage example:

```
AltcraftSDK.inAppFunctions.inAppSubscribe(context)
```

`fun inAppUnSubscribe(context: Context, ...): Unit`

Unsubscribes from In-App notifications. After a successful call, the user's profile will be unsubscribed f

Function parameters are the same as `pushUnSubscribe()`.

Usage example:

```
AltcraftSDK.inAppFunctions.inAppUnSubscribe(context)
```

`suspend fun getInAppSubscriptionStatus(context: Context): ResponseWithHttpCode<ResponseWithProfile>?`

Retrieves the current In-App notification subscription status. Returns a `ResponseWithHttpCode` object conta — the current In-App subscription for the profile. If no subscription exists, `null` is returned.

Usage example:

```
CoroutineScope(Dispatchers.IO).launch {
    AltcraftSDK.inAppFunctions.getInAppSubscriptionStatus(context)?.let { responseWithHttp ->
        val httpCode = responseWithHttp.httpCode
        val response = responseWithHttp.response
        val profile = response?.profile
        val subscription = profile?.subscription
    }
}
```

Authentication

```
AltcraftSDK
└── val authFunctions: PublicAuthFunctions
    // Authenticate user
    ├── fun authenticate(context: Context): Unit
    //
    // Log out user
    ├── fun logOut(context: Context): Unit
    //
    // Check authentication status
    └── suspend fun isAuthenticated(context: Context): Boolean
```

Functions for user authentication:

- `fun authenticate()` — **authenticate the user on the Altcraft platform;**
- `fun logOut()` — **log out the user, end session;**
- `suspend fun isAuthenticated()` — **check current authentication status.**

`fun authenticate(context: Context): Unit`

Authenticates the user on the Altcraft platform. Sends a request to the server with current authenticat binds the device to the user's profile.

After successful authentication, the user gains access to personalized InApp notifications.

> TIP: Important
>
> **Call `authenticate()` only when the user actually logs into the app and is known to the client (for exampl via login form, OAuth, or another authentication mechanism in your app).**

Usage example:

```
AltcraftSDK.authFunctions.authenticate(context)
```

`fun logOut(context: Context): Unit`

Logs out the user — ends the current authentication session on the Altcraft platform. After calling, the session mode. In this mode, personalized InApp notifications are unavailable, as well as subscription m

> TIP: Important
>
> **Call `logOut()` only when the user actually logs out of the app and switches to anonymous mode.**

Usage example:

```
AltcraftSDK.authFunctions.logOut(context)
```

`suspend fun isAuthenticated(context: Context): Boolean`

Checks the current user authentication status. Returns `true` if the user is successfully authenticated otherwise.

Usage example:

```
CoroutineScope(Dispatchers.IO).launch {
    val authenticated = AltcraftSDK.authFunctions.isAuthenticated(context)
    if (authenticated) {
        // user is authenticated
    } else {
        // user is not authenticated
    }
}
```

Additional SDK Functions

Manual Push Event Registration

> TIP: Note
>
> **These functions are only used if you implement your own notification handling logic and do not pass t default, the Altcraft SDK automatically processes notifications, so these methods are only required w handling.**

```
// Functions of the PublicPushEventFunctions object

AltcraftSDK
└── val pushEventFunctions: PublicPushEventFunctions

    // Register Altcraft push delivery (triggers delivery event)
    ├── fun deliveryEvent(
    │       context: Context,
    │       message: Map<String, String>? = null,
    │       messageUID: String? = null
    │   ): Unit

    // Register Altcraft push open (triggers open event)
    └── fun openEvent(
            context: Context,
            message: Map<String, String>? = null,
            messageUID: String? = null
        ): Unit
```

Manual event registration functions:

- `fun deliveryEvent()` — **registers push delivery;**
- `fun openEvent()` — **registers push open.**

`fun deliveryEvent(context: Context, message: Map<String, String>? = null, messageUID: String? = null): Unit`

Manually registers a push delivery event on the Altcraft platform. To log the event on the server, pass r
`messageUID` **parameter.**

Usage example:

```
AltcraftSDK.pushEventFunctions.deliveryEvent(context, message, messageUID)
```

`fun openEvent(context: Context, message: Map<String, String>? = null, messageUID: String? = null): Unit`

Manually registers a push open event on the Altcraft platform. To log the event on the server, pass noti
`messageUID` **parameter.**

Usage example:

```
AltcraftSDK.pushEventFunctions.openEvent(context, message, messageUID)
```

Passing custom data to Intent extras when opening a push notification

The SDK supports passing arbitrary data from a push notification to the application using the special
must be a string containing a JSON object.

When a user taps the notification, the SDK automatically:

1. **Extracts the** `_extra` **value from the push payload.**
2. **Adds it to the** `Intent`**.**
3. **Passes it to the Activity that handles the notification open event or deep link.**

As a result, the contents of `_extra` **become a regular** `Intent` **extra and are available in the application v**

```
val extraJson = intent?.getStringExtra("_extra")
```

The `_extra` **field is intended for passing additional parameters that should not be included in the URL b**
notification open event.

The `_extra` **value can be added using Custom JSON in the push notification template.**

SDK Data Clearing

`fun clear(context: Context, onComplete: (() -> Unit)? = null)`

This function clears all SDK data and cancels all pending background tasks. It cancels `WorkManager` **jobs**
and clears `SharedPreferences`**.**

Usage example:

```
AltcraftSDK
// Full SDK data cleanup (DB, SharedPreferences, background tasks)
└── fun clear(
        context: Context,
        onComplete: (() -> Unit)? = null
    ): Unit
```

The function includes an optional `completion` **parameter, which is called after cleanup and task cancella**

Functional Profile Field Update

```
AltcraftSDK
└── val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    └── fun actionField(key: String): ActionFieldBuilder
```

`fun actionField(key: String): ActionFieldBuilder`

A helper function for simplifying the functional profile field update process. It supports the following cc

```
    .set(value)
    .unset(value)
    .incr(value)
    .add(value)
    .delete(value)
    .upsert(value)
```

Usage example:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
    context = context,
    // "_fname" — field to be changed
    // .set("Andrey") — command that sets the value "Andrey" for this field
    // actionField().set() returns Pair<String, Map<String, Any?>>, so wrap in mapOf()
    profileFields = mapOf(
        AltcraftSDK.pushSubscriptionFunctions
            .actionField("_fname").set("Andrey")
    )
)
```

Resetting retry-operation restrictions during SDK initialization

> **TIP: Note**
>
> **When the SDK is initialized, background tasks are launched to control and retry requests related to pu**
> **and push-token update checks. Execution of this function is limited to one run within a single applicat**
> **In some cases, you may need to reset this restriction. In such situations, use the function:**

`fun unlockInitialOperationsInThisSession(): Unit`

This function resets the flag that prevents background retry-control tasks from running again within th

Requesting Notification Permission

> **TIP: Note**
>
> **Starting from Android 13 (API 33, Tiramisu), apps must explicitly request the `POST_NOTIFICATIONS` permis**
> **On earlier Android versions, this call is not required — the permission is granted automatically.**

`fun requestNotificationPermission(context: Context, activity: ComponentActivity)`

Checks the current notification permission status and, if necessary, displays the system permission re
has previously denied notifications, calling this function again will re-prompt the permission dialog.

Checking Push Notification Source

`fun isAltcraftPush(message: Map<String, String>): Boolean`

A static function in the `PushReceiver` companion object that checks whether a push notification was sen

Usage example:

```
if (AltcraftSDK.PushReceiver.isAltcraftPush(message.data)) {
    AltcraftSDK.PushReceiver.takePush(this, message.data)
}
```

SDK Configuration

Prerequisites

- **The app includes a class that extends Application.**
- **The project is connected to the Maven Central repository.**
- **Add the library dependency to the app-level `build.gradle.kts` file:**

```
dependencies {
    implementation("com.altcraft:android-sdk:1.0.6")
}
```

App Preparation

Configuring the JWT interface (optional)

JWTInterface is an interface for requesting a JWT token. It provides the current JWT token from the ap
Implement this interface if you use JWT authentication for API requests. JWT confirms that user identi
application.
JWT authentication is required if you use a matching type other than push data from subscription (for e
email or a phone number).

SDK interface:

```
interface JWTInterface {
    fun getJWT(): String?
}
```

App-side implementation

```
class JWTProvider(): JWTInterface {
    override fun getJWT(): String? {
        // code that returns a JWT token
    }
}
```

Registering the provider in `Application.onCreate()`

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
    }
}
```

> **TIP: Note**
>
> **`getJWT()` is a synchronous function. The SDK execution thread will be paused until the JWT is obtaine**
> **return a value immediately from a cache (in-memory, `SharedPreferences`, or `EncryptedSharedPreferences`) to s**
> **a current JWT as early as possible (at app start) and store it in cache so the token is available to the**
> **available, returning `null` is acceptable.**

Preparing to work with push providers

To prepare the app for working with push providers, you need to:

- **Integrate the push providers' SDKs into the project.**
- **Implement the SDK interfaces for passing and deleting the push token.**
- **Extend the push providers' services.**
- **Register the push provider service in `AndroidManifest.xml`.**

Firebase Cloud Messaging

Integrating the FCM SDK into the project

The article on integrating FCM is available here.

Altcraft SDK interface implementation

FCMInterface is an interface for requesting and deleting the FCM push token:

- **Passes the FCM token to Altcraft SDK**
- **Deletes the FCM token at the request of Altcraft SDK**

SDK interface:

```
interface FCMInterface {
    suspend fun getToken(): String?
    fun deleteToken(completion: (Boolean) -> Unit)
}
```

▶ Recommended app-side implementation

Realization the provider service

Recommended `FCMService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    /**
     * Called when a new FCM token is generated.
     *
     * @param token The new FCM token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)

       AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.FCMService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Huawei Mobile Services

Integrating the HMS SDK into the project

The article on integrating HMS is available here.

Altcraft SDK interface implementation

HMSInterface is an interface for requesting and deleting the HMS push token.
SDK interface:

```
interface HMSInterface {
    suspend fun getToken(context: Context): String?
    fun deleteToken(context: Context, complete: (Boolean) -> Unit)
}
```

▶ Recommended app-side implementation

Realization the provider service

Recommended `HMSService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * HMS service for handling push tokens and incoming notifications.
 *
 * Extends [HmsMessageService] and overrides key HMS callback methods.
 */
class HMSService : HmsMessageService() {
    /**
     * Called when a new HMS token is generated.
     *
     * @param token The new HMS token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received from HMS.
     *
     * Forwards the message with additional metadata to all receivers.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        AltcraftSDK.PushReceiver.takePush(this@HMSService, message.dataOfMap)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.HMSService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

RuStore

Integrating the RuStore SDK into the project

The article on integrating RuStore is available here.

Altcraft SDK interface implementation

RustoreInterface is an interface for requesting and deleting the RuStore push token.

SDK interface:

```
interface RustoreInterface {
    suspend fun getToken(): String?
    fun deleteToken(complete: (Boolean) -> Unit)
}
```

▶ Recommended app-side implementation

Realization the provider service

Recommended `RuStoreService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * RuStore service for handling push notifications.
 *
 * Extends [RuStoreMessagingService] and overrides key callbacks.
 */
class RuStoreService : RuStoreMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with added metadata.
     *
     * @param message The received RemoteMessage.
     */
    override fun onMessageReceived(message: RemoteMessage) {
        AltcraftSDK.PushReceiver.takePush(this, message.data)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.RuStoreService"
    android:exported="true"
    tools:ignore="ExportedService">
    <intent-filter>
    </intent-filter>
</service>
```

SDK Initialization

Initialization parameters

Use the `AltcraftConfiguration` class to pass configuration parameters:

```
class AltcraftConfiguration private constructor(
    private val apiUrl: String,
    private val icon: Int? = null,
    private val rToken: String? = null,
    private val appInfo: DataClasses.AppInfo? = null,
    private val providerPriorityList: List<String>? = null,
    private val pushReceiverModules: List<String>? = null,
    private val pushChannelName: String? = null,
    private val pushChannelDescription: String? = null,
    private val enableLogging: Boolean? = null
)
```

Parameter descriptions:

apiUrl `String`
Required: Yes
Description: URL of the Altcraft API endpoint.

---

icon `Int?`
Default: `null`
Required: No
Description: Resource ID of the `drawable` **used as the notification icon.**

---

rToken `String?`
Default: `null`
Required: No
Description: Altcraft role token (identifies a resource, database, or account). Used when the only matc issued by the provider.

---

appInfo `DataClasses.AppInfo?`
Default: `null`
Required: No
Description: Basic app metadata for Firebase Analytics. To set it, use the public SDK data class `AppInfo` `com.altcraft.sdk.data.DataClasses` **):**

```
data class AppInfo(
    /** Firebase app_id */
    val appID: String,
    /** Firebase app_instance_id */
    val appIID: String,
    /** Firebase app_version */
    val appVer: String
)
```

---

enableLogging `Boolean`
Default: `null`
Required: No
Description: Enables/disables logging. Logs can be found by logCat tag `altcraft_lib` **.**

---

providerPriorityList `List<String>?`
Default: `null`
Required: No
Description: Provider priority list. Used to automatically refresh the subscription's push token if a highe unavailable. Priority is defined by index in the list: index 0 is the highest.
Usage example:

```
providerPriorityList = listOf(
    "android-firebase",
    "android-huawei",
    "android-rustore"
)
```

With this setup:

- **The SDK first requests FCM; if FCM is unavailable—HMS; if HMS is unavailable—RuStore.**
- **Works provided the corresponding provider interfaces are implemented in the app.**

For convenience, the SDK exposes public constants:

```
const val FCM_PROVIDER: String = "android-firebase"
const val HMS_PROVIDER: String = "android-huawei"
const val RUS_PROVIDER: String = "android-rustore"
```

If not set, the default provider priority is:

```
FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER
```

The value may contain a single element—in this case only that provider will be used, regardless of pus

You can omit this parameter if:

- **the project uses only one provider;**
- **the default priority meets your requirements.**

> TIP: Use cases
>
> **Case 1. Priority `RUS_PROVIDER -> FCM_PROVIDER`. A user removes RuStore—RuStore notifications become una `RustoreInterface.getToken()` implementation returning `null`, the SDK automatically switches to FCM, upd communications continue. Case 2. Parameter not set; defaults `FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER` without Google services the SDK automatically switches to HMS with no extra code.**

---

pushReceiverModules `List<String>?`

Default: `null` **Required: No Description: List of package names containing implementations of** `AltcraftP`
If set, the SDK discovers these classes and forwards incoming notifications to them via `pushHandler(cont` `String>)`**. If no classes are found in any of the packages, the notification will be displayed by the SDK.**

Example of creating an `AltcraftPushReceiver` class

```
package com.altcraft.altcraftmobile.test

@Keep
class AltcraftPushReceiver: AltcraftSDK.PushReceiver() {
    override fun pushHandler(context: Context, message: Map<String, String>) {

        // handle the notification using SDK functions
        super.pushHandler(context, message)

        // additional notification handling logic
    }
}
```

> TIP: Use case
>
> **Case: The app contains multiple modules that need Altcraft incoming push data. In each module you c `AltcraftPushReceiver: AltcraftSDK.PushReceiver()` and receive the incoming push as `message: Map<String, String`**
> **Note the class must be named exactly `AltcraftPushReceiver`.**
> **If the project has only one `AltcraftPushReceiver`, call `super.pushHandler(context, message)` inside `pushHandler` s notification correctly. If there are multiple such classes, keep the `super.pushHandler` call in only one of t will be duplicated. If you want full manual control of push handling, you can omit `super.pushHandler`.**

---

pushChannelName `String?`

Default: `null` **Required: No Description: Base name of the push-notification channel (visible in Android s sound/vibration settings of the Altcraft platform's push template, a suffix is added:**

- `allSignal` **— sound and vibration enabled**
- `soundless` **— silent channel**
- `onlySound` **— sound only (no vibration)**

Example: with `pushChannelName = "Altcraft"` **and mode** `allSignal`**, the visible channel name is** `"Altcraft_allSi` **default names:** `"allSignal"`, `"soundless"`, `"onlySound"`.

---

pushChannelDescription `String?`

Default: `null` **Required: No Description: Description of the push channel (visible in Android settings). Tl**

- `Vibration and sound enabled`

- `Vibration and sound disabled`
- `Sound enabled, vibration disabled`

Example: `"Altcraft notification channel. (vibration and sound enabled)"` **. If not set, the SDK uses these defaul**

- **icon —** `altcraft-sdk/src/main/res/drawable/icon.xml` **(if icon = null)**
- **providerPriorityList —** `FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER` **(if null)**
- **pushChannelName —** `"allSignal", "soundless", "onlySound"` **(if null)**
- **pushChannelDescription —** `"Vibration and sound enabled", "Vibration and sound disabled", "Sound enabled, vib`

Running initialization

> TIP: **Note**
>
> **Call** `AltcraftSDK.initialization(...)` **when needed, but only after registering all providers (the JWT provid**
> **Execute requests after the configuration is set.**

Use `fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? =`

```
// Initialize the SDK and set configuration

AltcraftSDK.initialization(
    context = context,
    configuration = config,
    complete = null // optional
)
```

Example of correct initialization order in `Application.onCreate()` **(after registering providers)**

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        // Initialize the RuStore SDK provider (if used)
        RuStorePushClient.init(this, "rustore-project-id-1234")

        // Register providers before initializing the SDK
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
        AltcraftSDK.pushTokenFunctions.setRuStoreTokenProvider(RuStoreProvider())

        // SDK configuration
        val config = AltcraftConfiguration.Builder(
            apiUrl = "https://pxl-example.altcraft.com",
            icon   = R.drawable.ic_notification
        ).build()

        // Initialization
        AltcraftSDK.initialization(this, config)
    }
}
```

Example of minimal working setup

```
val config = AltcraftConfiguration.Builder(
    apiUrl = "https://pxl-example.altcraft.com"
).build()

AltcraftSDK.initialization(context, config)
```

Example of full parameter configuration

```
val config = AltcraftConfiguration.Builder(
    apiUrl = "https://pxl-example.altcraft.com",
    icon   = R.drawable.ic_notification,
    rToken = null,
    appInfo = DataClasses.AppInfo(
        appID  = "com.example.app",
        appIID = "8b91f3a0-1111-2222-3333-c1a2c1a2c1a2",
        appVer = "1.0.0"
    ),
    providerPriorityList = listOf(
        FCM_PROVIDER, // "android-firebase"
        HMS_PROVIDER, // "android-huawei"
        RUS_PROVIDER  // "android-rustore"
    ),
    pushReceiverModules = listOf(
        context.packageName,
        "com.example.push_receiver",
        "com.example.feature.test"
    ),
    pushChannelName        = "Altcraft",
    pushChannelDescription = "Altcraft notifications channel"
).build()

AltcraftSDK.initialization(context, config)
```

Example of initialization with completion callback

```
AltcraftSDK.initialization(context, config) { result ->
    when {
        result.isSuccess -> {
            // actions on successful initialization
        }
        result.isFailure -> {
            // handle initialization errors
        }
    }
}
```

Public SDK API

object AltcraftSDK

```
AltcraftSDK
// SDK initialization and configuration setup
├─ fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null): Unit
// Full cleanup of SDK data (DB, SP, background tasks)
├─ fun clear(context: Context, onComplete: (() -> Unit)? = null): Unit
// Register a JWT provider
├─ fun setJWTProvider(provider: JWTInterface?): Unit
// Allow push-module reinitialization in the current session
├─ fun unlockInitialOperationsInThisSession(): Unit
// Request notification permission (Android 13+)
├─ fun requestNotificationPermission(context: Context, activity: ComponentActivity): Unit
// Base Altcraft push receiver (can be overridden)
├─ open class PushReceiver
│  // Handle incoming push message
│  ├─ open suspend fun pushHandler(context: Context, message: Map<String, String>): Unit
│  // Entry point for delivering push messages to the SDK
│  └─ companion object
│     ├─ fun takePush(context: Context, message: Map<String, String>): Unit
│     └─ fun isAltcraftPush(message: Map<String, String>): Boolean
// Public subscription functions
├─ val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
│  // Subscribe to push notifications (status = SUBSCRIBED)
│  ├─ fun pushSubscribe(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // Suspend push notifications (status = SUSPENDED)
│  ├─ fun pushSuspend(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // Unsubscribe from push notifications (status = UNSUBSCRIBED)
│  ├─ fun pushUnSubscribe(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // Change subscription status from suspended to subscribed for the one in JWT; others containing the specified push token switch from su
│  ├─ suspend fun unSuspendPushSubscription(context: Context): ResponseWithHttpCode<ResponseWithProfile>?
│  // Status of the latest profile subscription
│  ├─ suspend fun getStatusOfLatestSubscription(context: Context): ResponseWithHttpCode<ResponseWithProfile>?
│  // Status of latest subscription for a specific push provider
│  ├─ suspend fun getStatusOfLatestSubscriptionForProvider(context: Context, provider: String? = null): ResponseWithHttpCode<ResponseWithPu
│  // Status of subscription with current device token.
│  ├─ suspend fun getStatusForCurrentSubscription(context: Context): ResponseWithHttpCode<ResponseWithProfile>?
│  // Functional profile field (set/incr/…)
│  └─ fun actionField(key: String): ActionFieldBuilder
// Token management functions
├─ val pushTokenFunctions: PublicPushTokenFunctions
│  // Manually save provider token (onNewToken)
│  ├─ fun setPushToken(context: Context, provider: String?, token: String?): Unit
│  // Get current device push token
│  ├─ suspend fun getPushToken(context: Context): TokenData?
│  // Register FCM provider
│  ├─ fun setFCMTokenProvider(provider: FCMInterface?): Unit
│  // Register HMS provider
│  ├─ fun setHMSTokenProvider(provider: HMSInterface?): Unit
│  // Register RuStore provider
│  ├─ fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit
│  // Delete provider device token
│  ├─ fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit
│  // Forced token update (delete → refresh)
│  ├─ fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit
│  // Change provider priority list and update token
│  └─ suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit
// Push event tracking functions
```

```
├─ val pushEventFunctions: PublicPushEventFunctions
│  // Track delivery (delivery event)
│  ├─ fun deliveryEvent(context: Context, message: Map<String, String>? = null, messageUID: String? = null): Unit
│  // Track open (open event)
│  └─ fun openEvent(context: Context, message: Map<String, String>? = null, messageUID: String? = null): Unit
//
// Registering mobile events
├─ val mobileEventFunctions: PublicMobileEventFunctions
│  // Registers a mobile event for Altcraft
│  └─ fun mobileEvent(
│        context: Context,
│        sid: String,
│        eventName: String,
│        sendMessageId: String? = null,
│        payload: Map<String, Any?>? = null,
│        matching: Map<String, Any?>? = null,
│        matchingType: String? = null,
│        profileFields: Map<String, Any?>? = null,
│        subscription: Subscription? = null,
│        utm: UTM? = null
│     ): Unit
//
// Profile field update
├─ val profileFunctions: PublicProfileFunctions
│  // Updates profile fields for Altcraft
│  └─ fun updateProfileFields(
│        context: Context,
│        profileFields: Map<String, Any?>? = null,
│        skipTriggers: Boolean? = null
│     ): Unit
//
// Public In-App notification functions
├─ val inAppFunctions: PublicInAppFunctions
│  // Configure In-App notification animation
│  ├─ fun setAnimation(block: Animator.() -> Unit): Unit
│  // Set current screen marker (used for In-App notification filtering)
│  ├─ fun setScreen(screen: String): Unit
│  // Custom In-App notification trigger by name
│  ├─ fun inAppTrigger(context: Context, name: String): Unit
│  // Request available In-App placements
│  ├─ fun getInAppPlacements(context: Context): Unit
│  // Subscribe to Activity lifecycle for automatic In-App display
│  ├─ fun subscribeOnActivityLifecycle(context: Context): Unit
│  // Subscribe to In-App notifications (status = SUBSCRIBED)
│  ├─ fun inAppSubscribe(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // Unsubscribe from In-App notifications (status = UNSUBSCRIBED)
│  ├─ fun inAppUnSubscribe(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // In-App notification subscription status
│  └─ suspend fun getInAppSubscriptionStatus(context: Context): ResponseWithHttpCode<ResponseWithProfile>?
//
// Public authentication functions
├─ val authFunctions: PublicAuthFunctions
│  // Authenticate user
│  ├─ fun authenticate(context: Context): Unit
│  // Log out user
│  ├─ fun logOut(context: Context): Unit
│  // Check authentication status
│  └─ suspend fun isAuthenticated(context: Context): Boolean
//
// SDK events (single subscriber)
└─ val eventSDKFunctions: Events
   // Subscribe to SDK events
   ├─ fun subscribe(newSubscriber: (Event) -> Unit): Unit
   // Unsubscribe from SDK events
```

class AltcraftConfiguration

```
com.altcraft.sdk.config
└─ class AltcraftConfiguration private constructor(...)
   // Altcraft SDK configuration initialization class:
   // API URL, role token, app info, foreground service flag
   // and push notification settings (channel, receiver modules, provider priority).

   ├─ class Builder(
   │     apiUrl: String,                        // Base Altcraft API URL (required)
   │     icon: Int? = null,                     // Notification icon resource ID (optional)
   │     rToken: String? = null,                // Role token
   │     appInfo: AppInfo? = null,              // Firebase Analytics metadata (optional)
   │     enableLogging: Boolean? = null,        // Enable/disable logging (optional)
   │     providerPriorityList: List<String>? = null, // Push provider priority (optional)
   │     pushReceiverModules: List<String>? = null,  // Package modules where PushReceiver may be overridden (optional)
   │     pushChannelName: String? = null,       // Push notification channel name (optional)
   │     pushChannelDescription: String? = null  // Push notification channel description (optional)
   │   )
   │   └─ fun build(): AltcraftConfiguration     // Build valid configuration
   │
   ├─ fun getApiUrl(): String                   // Returns base Altcraft API URL
   ├─ fun getIcon(): Int?                       // Returns notification icon ID (optional)
   ├─ fun getRToken(): String?                  // Returns role token (optional)
   ├─ fun getAppInfo(): AppInfo?                // App info (ID/IID/version) (optional)
   ├─ fun getEnableLogging(): Boolean?          // Enable/disable logging (optional)
   ├─ fun getProviderPriorityList(): List<String>? // Push provider priority (optional)
   ├─ fun getPushReceiverModules(): List<String>? // Modules with PushReceiver overrides (optional)
   ├─ fun getPushChannelName(): String?         // Notification channel name (optional)
   └─ fun getPushChannelDescription(): String?  // Notification channel description (optional)
```

Event Classes

```
Event
// Base SDK event (universal telemetry).
function: String
event: SDKEvent?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date
internal: Boolean

Error
// SDK error, inherits from Event.
function: String
event: SDKEvent?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date

RetryError
// Request error with automatic retry and execution
function: String
event: SDKEvent?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date
```

SDKEvent Enum

```
// Enum class containing all SDK event types.
// Each value contains a string message.

enum class SDKEvent(val message: String) {
    // General SDK events
    ConfigIsSet("SDK configuration is installed"),
    PushProviderSet("push provider set: "),
    NotAcPush("received a notification unrelated to the Altcraft Platform"),
    AcPush("received Altcraft push notification"),
    PushIsPosted("push is posted"),
    SdkCleared("SDK data has been cleared"),
    ReceiverRedefined("receiver is redefined"),
    NotAuthenticatedError("User is not authenticated"),
    UserLogOut("User logged out. Anonymous session started."),

    // Internal errors without retry
    ConfigIsNotSet("the configuration is not set"),
    ConfigNotValid("invalid configuration"),
    SdkInitWaitingExpired("SDK initialization timeout has expired"),
    MobileEventPartsIsNull("mobile event parts is null"),
    InvalidPushProvider("invalid provider. Available - android-firebase, android-huawei, android-rustore."),
    PushDataIsNull("push data is null"),
    PushUidIsNull("uid in the push data is null or empty, it is impossible to send a push event to the server."),
    ErrorImgLoad("error uploading the notification image"),
    NotificationErr("couldn't create notification"),
    ChannelNotCreated("the notification channel has not been created"),
    SmidIsNull("send message ID is null"),
    InvalidStaticField("invalid in-app static field"),
    RoomMigrationError("Room database migration failed. The local SDK database will be recreated."),

    // Missing request payload (without retry)
    UnSuspendRequestDataIsNull("unsuspend request data is null"),
    SubscriptionStatusRequestDataIsNull("status request data is null"),
    InAppSubscriptionStatusRequestDataIsNull("status request data is null"),
    AuthorizeRequestDataIsNull("authorize request data is null"),

    // Retry limit reached
    PushSubscribeRetryLimit("push subscribe retry limit"),
    PushEventRetryLimit("push event retry limit"),
    MobileEventRetryLimit("mobile event retry limit"),
    ProfileUpdateRetryLimit("profile update retry limit"),
    InAppSubscribeRetryLimit("in-app subscribe retry limit"),
    AuthRequestRetryLimit("auth request retry limit"),
    InAppEventRetryLimit("in-app event retry limit"),

    // SDK or environment state issues
    ConfigIsNull("config data is null"),
    PushTokenIsNull("current push token is null"),
    UserTagIsNull("userTag is null. It is impossible to identify the user"),
    PermissionDenied("no permission to send notifications"),
    NoInternetConnect("no internet connection, retry when connection is restored"),
    NotUpdated("failed to update the push token. The subscription request was rejected"),
    ResponseDataIsNull("response data is null"),
    JwtTokenIsNull("JWT token is null"),
    MatchingIsNull("matching mode is null"),
    JwtTooLarge("JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS"),
    PayloadIsMissing("JWT does not contain a payload"),
    InternalAuthIsNull("internal auth data is null"),
    ExternalAuthIsNull("external auth data is null"),
    MatchingIdIsNull("matching claim does not contain a matching ID"),
    InvalidPayload("invalid mobile event payload: not all values are primitives"),
    InvalidCustomFields("invalid custom fields: not all values are primitives"),

    // Missing request payload (with retry)
    PushSubscribeRequestDataIsNull("push subscribe request data is null"),
    TokenUpdateRequestDataIsNull("token update request data is null"),
    PushEventRequestDataIsNull("push event request data is null"),
    InAppEventRequestDataIsNull("in-app event request data is null"),
    MobEventRequestDataIsNull("mobile event request data is null"),
    ProfileUpdateRequestDataIsNull("profile update request data is null"),
    InAppSubscribeRequestDataIsNull("profile update request data is null"),
    InAppPlacementsRequestDataIsNull("in-app placements request data is null"),

    // Successful API requests
    PushUnsubscribeRequestSuccessful("successful request: push/unsubscribe"),
    PushSubscribeRequestSuccessful("successful request: push/subscribe"),
    PushSuspendRequestSuccessful("successful request: push/suspend"),
    InAppUnsubscribeRequestSuccessful("successful request: inapp/unsubscribe"),
    InAppSubscribeRequestSuccessful("successful request: inapp/subscribe"),
    InAppStatusRequestSuccessful("successful request: inapp/status"),
```

```
        InAppEventRequestSuccessful("successful request: inapp/event"),
        InAppPlacementsRequestSuccessful("successful request: in_app/placements"),
        PushStatusRequestSuccessful("successful request: push/status"),
        TokenUpdateRequestSuccessful("successful request: push/update"),
        PushEventRequestSuccessful("successful request: event/push"),
        MobileEventRequestSuccessful("successful request: event/post"),
        UnsuspendRequestSuccessful("successful request: push/unsuspend"),
        ProfileUpdateRequestSuccessful("successful request: profile/update"),
        AuthenticateSuccessful("successful request: profile/authenticate"),

        // Failed API requests
        PushUnsubscribeRequestFailed("failed request: push/unsubscribe"),
        PushSubscribeRequestFailed("failed request: push/subscribe"),
        PushSuspendRequestFailed("failed request: push/suspend"),
        InAppUnsubscribeRequestFailed("failed request: inapp/unsubscribe"),
        InAppSubscribeRequestFailed("failed request: inapp/subscribe"),
        InAppStatusRequestFailed("failed request: inapp/status"),
        InAppEventRequestFailed("failed request: inapp/event"),
        InAppPlacementsRequestFailed("failed request: in_app/placements"),
        PushStatusRequestFailed("failed request: push/status"),
        TokenUpdateRequestFailed("failed request: push/update"),
        PushEventRequestFailed("failed request: event/push"),
        MobileEventRequestFailed("failed request: event/post"),
        ProfileUpdateRequestFailed("failed request: profile/update"),
        UnsuspendRequestFailed("failed request: push/unsuspend"),
        AuthenticateFailed("failed request: profile/authenticate"),
        InAppStaticErrorLoad("error loading in-app notification static resources"),
        InAppContentErrorLoad("error loading in-app notification content"),
}
```

Data Classes

```
// Device push provider token
data class TokenData(
    provider: String,
    token: String
)

// Optional data for Firebase Analytics (if used)
data class AppInfo(
    appID: String = "",
    appIID: String = "",
    appVer: String = ""
)

// API response wrapper with HTTP code
data class ResponseWithHttpCode<T : ResponseData>(
    httpCode: Int?,
    response: T?
)

// Request response (implementation of ResponseData)
data class ResponseWithProfile(
    error: Int? = null,
    @SerialName("error_text") errorText: String? = null,
    profile: ProfileData? = null
)

// User profile data
data class ProfileData(
    id: String? = null,
    status: String? = null,
    acid: String? = null,
    @SerialName("is_test") isTest: Boolean? = null,
    subscription: SubscriptionData? = null
)

// Current profile subscription
data class SubscriptionData(
    @SerialName("subscription_id") subscriptionId: String? = null,
    @SerialName("hash_id") hashId: String? = null,
    provider: String? = null,
    status: String? = null,
    fields: Map<String, JsonElement>? = null,
    cats: List<CategoryData>? = null
)

// Subscription category (name, title, flags)
data class CategoryData(
    name: String?,
    title: String? = null,
    steady: Boolean? = null,
    active: Boolean?
)

// Email channel subscription (added by mobile event)
data class EmailSubscription(
    @SerialName("resource_id") resourceId: Int,
    email: String,
    status: String? = null,
    priority: Int? = null,
    @SerialName("custom_fields") customFields: Map<String, @Contextual Any?>? = null,
    cats: List<String>? = null,
    @SerialName("channel") channel: String = "email"
)

// SMS channel subscription (added by mobile event)
data class SmsSubscription(
    @SerialName("resource_id") resourceId: Int,
    phone: String,
    status: String? = null,
    priority: Int? = null,
    @SerialName("custom_fields") customFields: Map<String, @Contextual Any?>? = null,
    cats: List<String>? = null,
    @SerialName("channel") channel: String = "sms"
)

// Push channel subscription (added by mobile event)
data class PushSubscription(
    @SerialName("resource_id") resourceId: Int,
    provider: String,
    @SerialName("subscription_id") subscriptionId: String,
```

```
    status: String? = null,
    priority: Int? = null,
    @SerialName("custom_fields") customFields: Map<String, @Contextual Any?>? = null,
    cats: List<String>? = null,
    @SerialName("channel") channel: String = "push"
)

// Subscription with cc_data (Telegram, WhatsApp, Viber, Notify)
data class CcDataSubscription(
    @SerialName("resource_id") resourceId: Int,
    @SerialName("channel") channel: String,
    @SerialName("cc_data") ccData: JsonObject,
    status: String? = null,
    priority: Int? = null,
    @SerialName("custom_fields") customFields: Map<String, @Contextual Any?>? = null,
    cats: List<String>? = null
)

// UTM parameters container for mobile event attribution (all fields optional)
data class UTM(
    campaign: String? = null,
    content: String? = null,
    keyword: String? = null,
    medium: String? = null,
    source: String? = null,
    temp: String? = null
)
```

Constants

```
com.altcraft.sdk.data
└─ object Constants
   // Push notification providers
   ├─ const val FCM_PROVIDER: String = "android-firebase"
   ├─ const val HMS_PROVIDER: String = "android-huawei"
   └─ const val RUS_PROVIDER: String = "android-rustore"
   // Subscription statuses
   ├─ const val SUBSCRIBED: String = "subscribed"
   ├─ const val UNSUBSCRIBED: String = "unsubscribed"
   └─ const val SUSPENDED: String = "suspended"
```

Interfaces

JWTInterface:

```
JWTInterface
// Access to current JWT
└─ fun getJWT(): String?
   // Returns JWT or null
```

FCMInterface:

```
FCMInterface
// FCM token operations contract
├─ suspend fun getToken(): String?
│  // Returns FCM token or null
└─ fun deleteToken(completion: (Boolean) -> Unit)
   // Deletes FCM token
```

HMSInterface:

```
HMSInterface
// HMS token operations contract
├─ suspend fun getToken(context: Context): String?
│  // Returns HMS token or null
└─ fun deleteToken(context: Context, complete: (Boolean) -> Unit)
   // Deletes HMS token
```

RuStoreInterface:

```
RustoreInterface
// RuStore token operations contract
├─ suspend fun getToken(): String?
│  // Returns RuStore token or null
└─ fun deleteToken(complete: (Boolean) -> Unit)
   // Deletes RuStore token
```

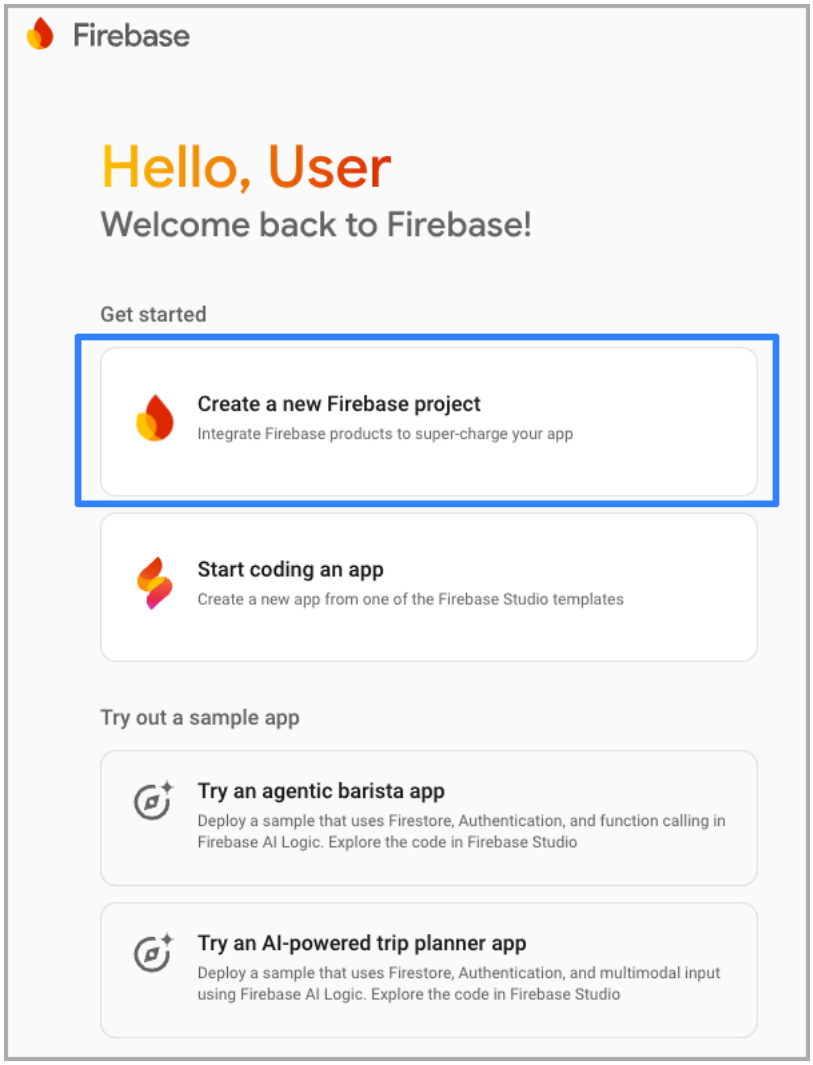
Subscription:

```
Subscription (sealed interface)
// Base subscription model contract for all channels
├─ val resourceId: Int
│  // Required Altcraft resource ID (JSON: "resource_id")
├─ val status: String?
│  // Optional subscription status
├─ val priority: Int?
│  // Optional priority
├─ val customFields: Map<String, @Contextual Any?>?
│  // Optional standard and custom fields (JSON: "custom_fields")
├─ val cats: List<String>?
│  // Optional subscription categories
└─ val channel: String
   // Required channel type (e.g., "email", "sms", "push", "telegram_bot", ...)
```

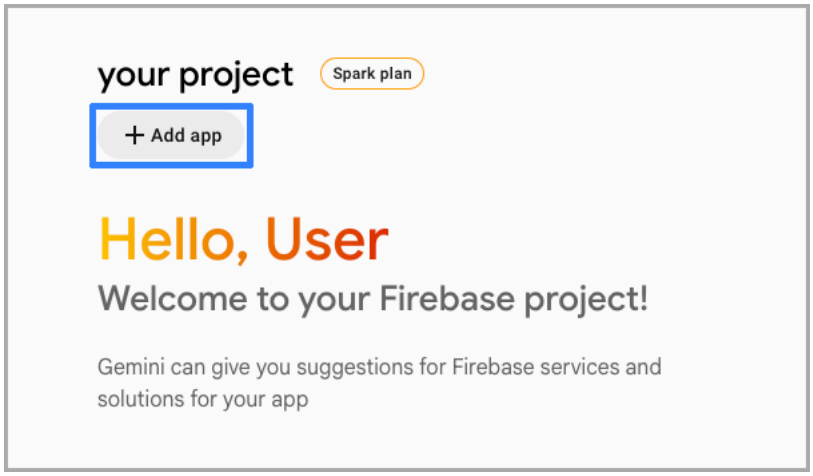
Firebase Cloud Messaging

Firebase Cloud Messaging Project Setup

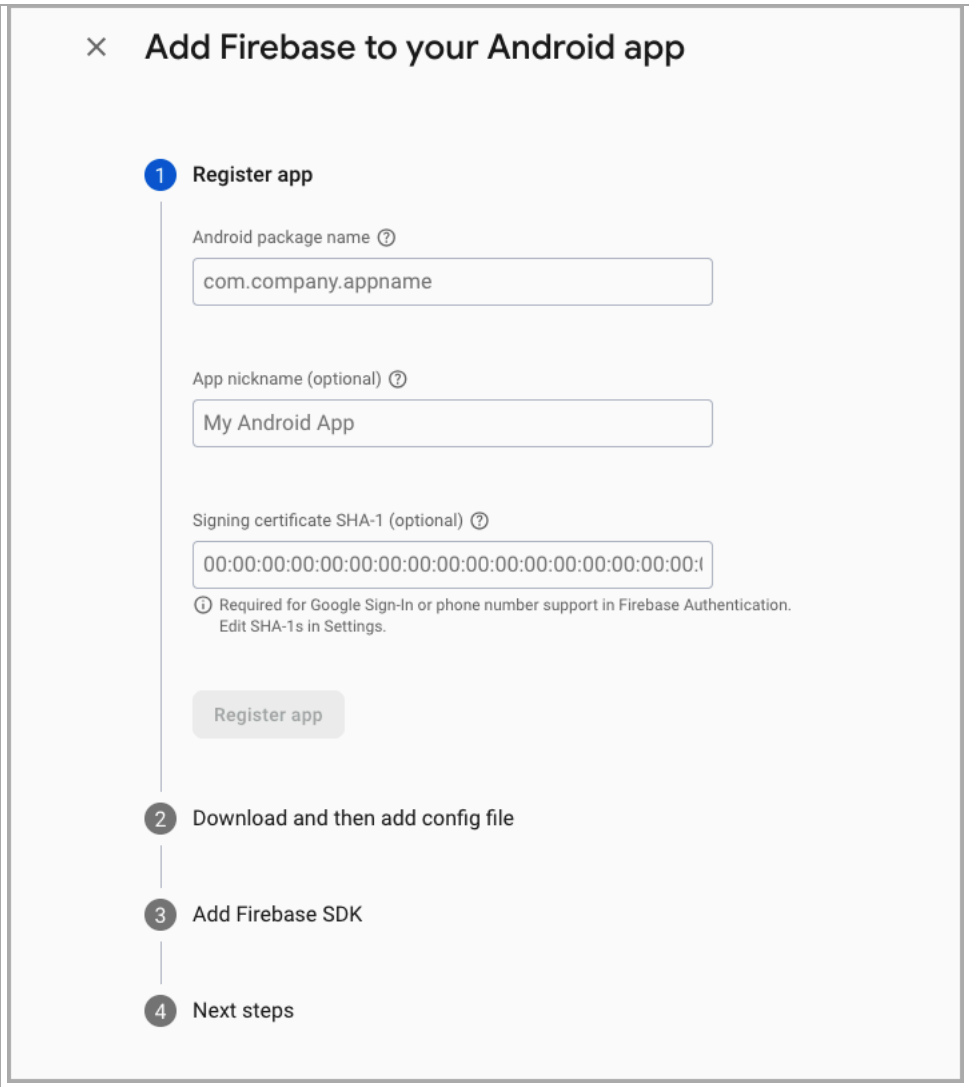
Create a [Firebase] project:



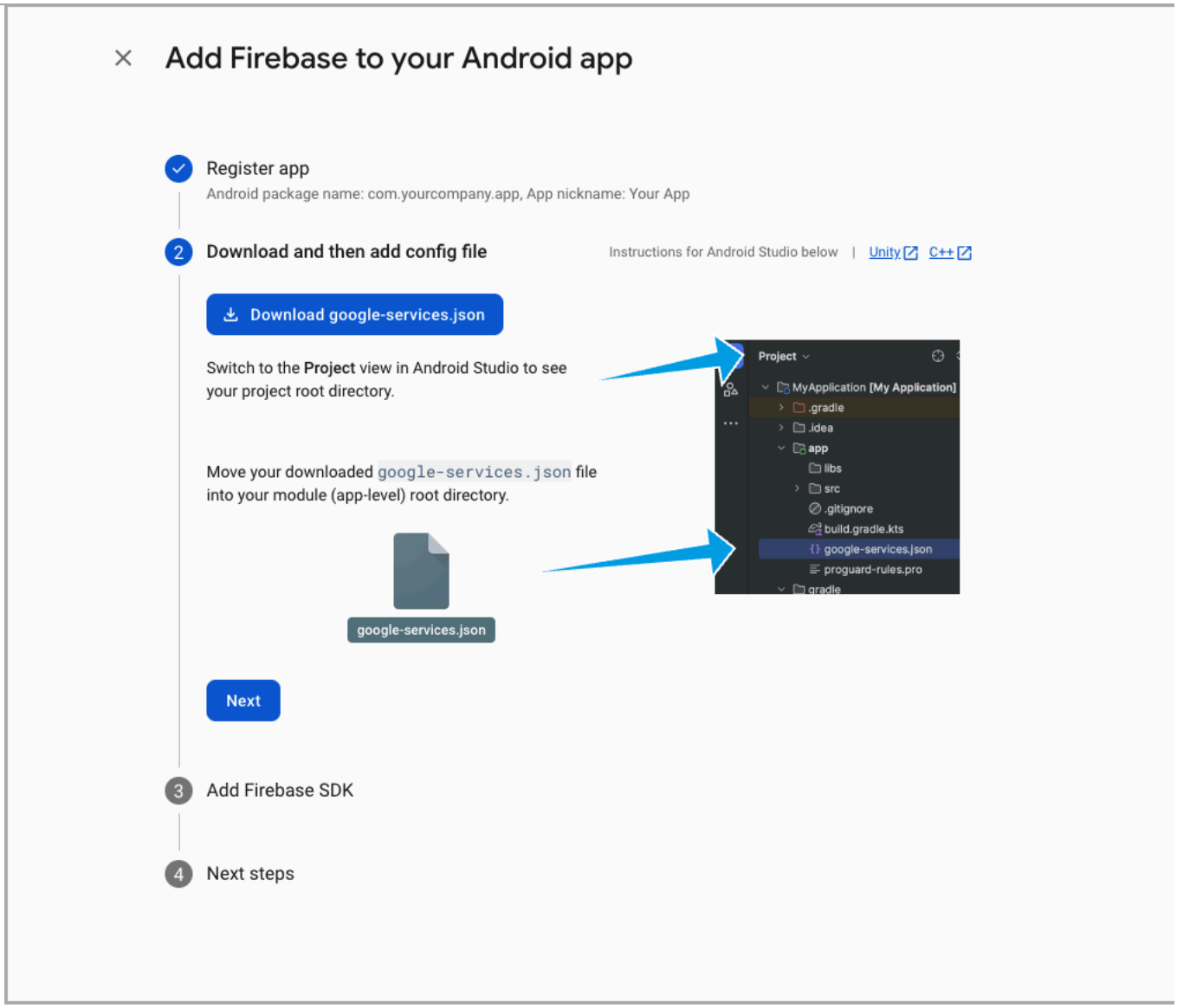
Add your app to the project:



Register the app by filling in the required details:

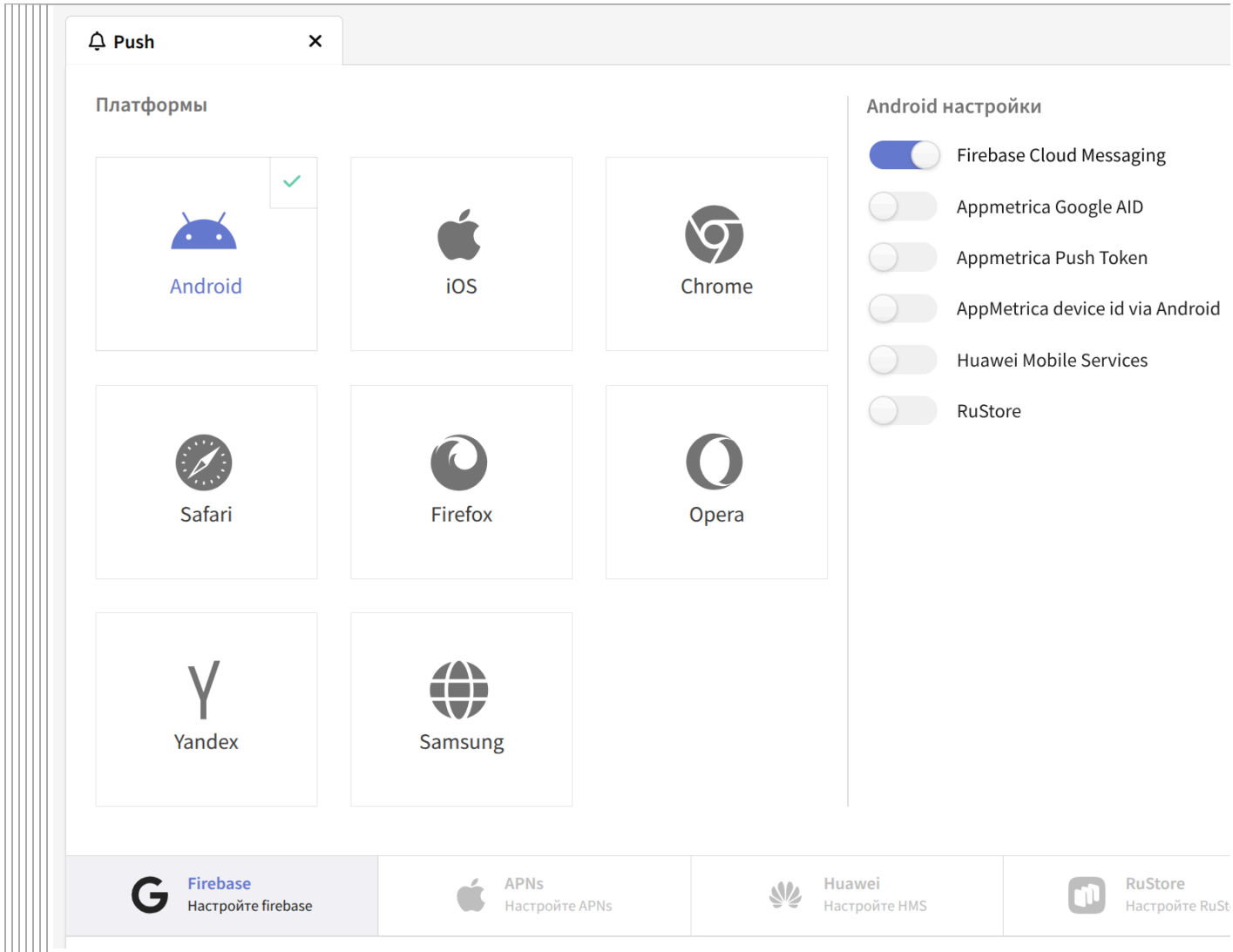


Download the `google-services.json` file:

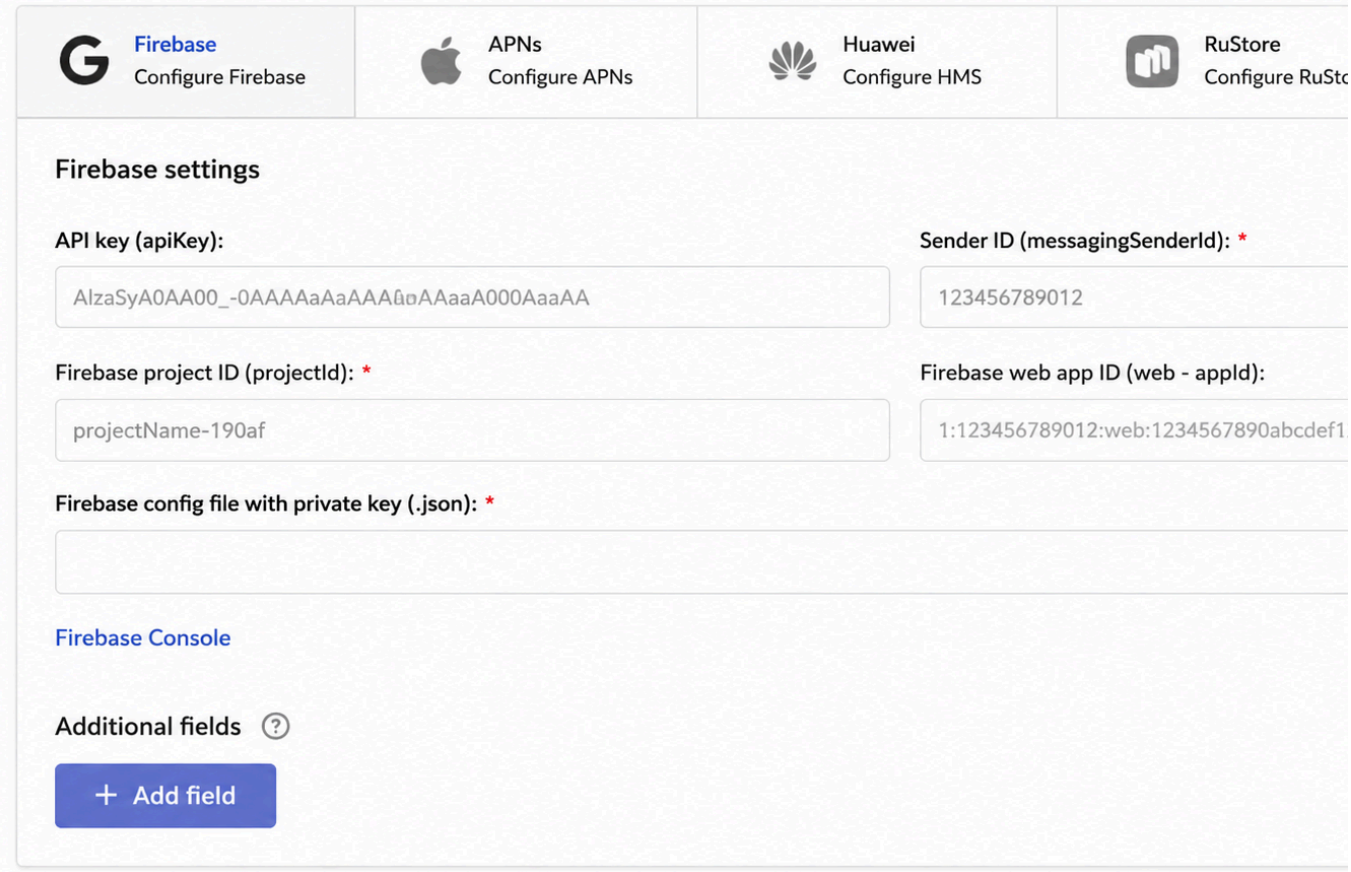


Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existi select the Android platform and enable the Firebase Cloud Messaging toggle:



Then, scroll down to the Firebase settings section and specify your Project ID (`messagingSenderId`) and Se upload the Firebase private key file (.json):

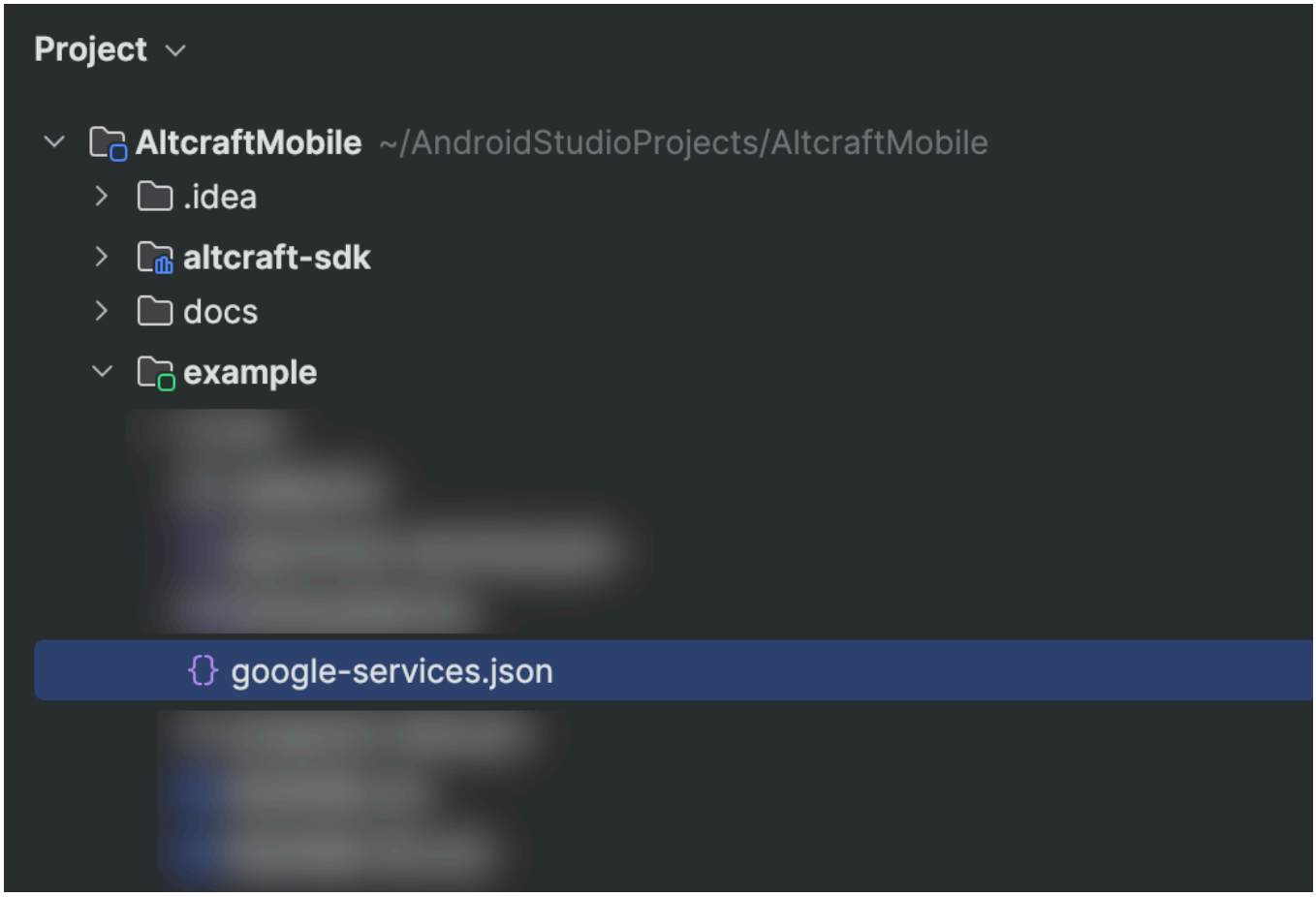


You can generate this Firebase file in Project settings —> Service account in the Firebase console. After making changes, don't forget to save the resource.

Integrating FCM into the App Project

Manual Integration

Move the downloaded `google-services.json` file to the root directory of your app module (app):

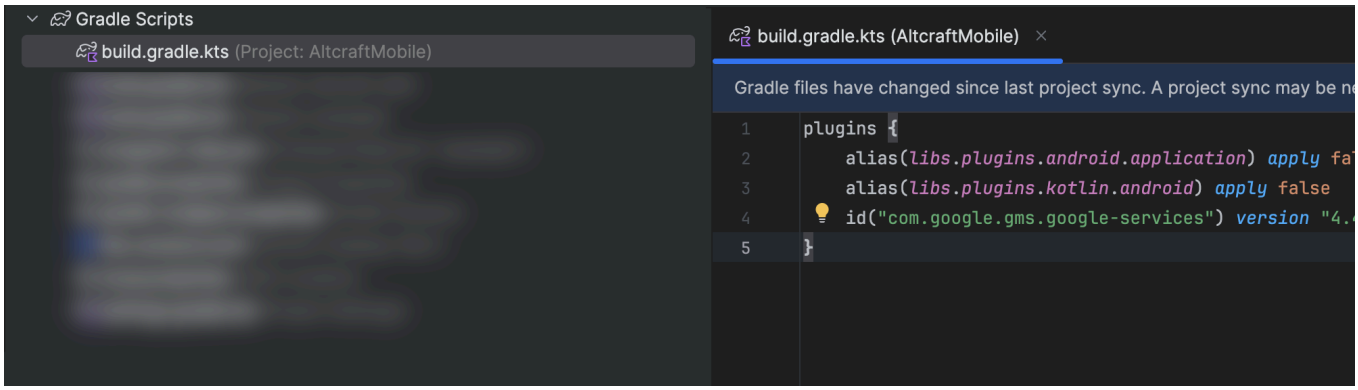


Then add the necessary plugins and dependencies to your Gradle files:

- `build.gradle.kts (project level)`

Add the `google-services` plugin to the project-level `build.gradle.kts` file:

```
plugins {
    id("com.google.gms.google-services") version "4.4.3" apply false
}
```



TIP: Note

Information about the latest plugin version is available here.

- `build.gradle.kts (app level)`

Add the `google-services` plugin to the app-level `build.gradle.kts` file:

```
plugins {
    id("com.google.gms.google-services")
}
```

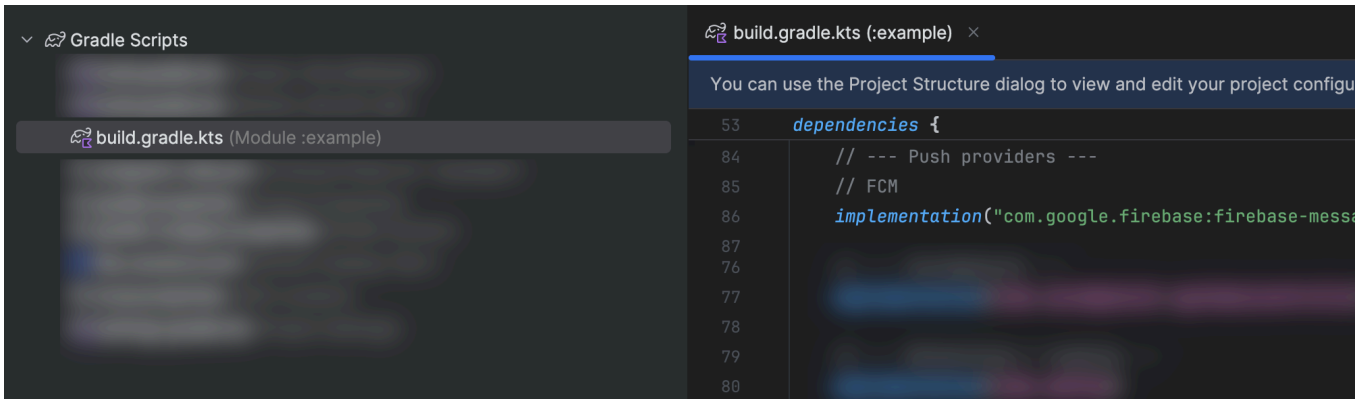
Also add dependencies:

Using firebase-bom (automatically manages Firebase library versions)

```
dependencies {
    implementation(platform("com.google.firebase:firebase-bom:34.0.0"))
    implementation("com.google.firebase:firebase-messaging")
}
```

Without firebase-bom

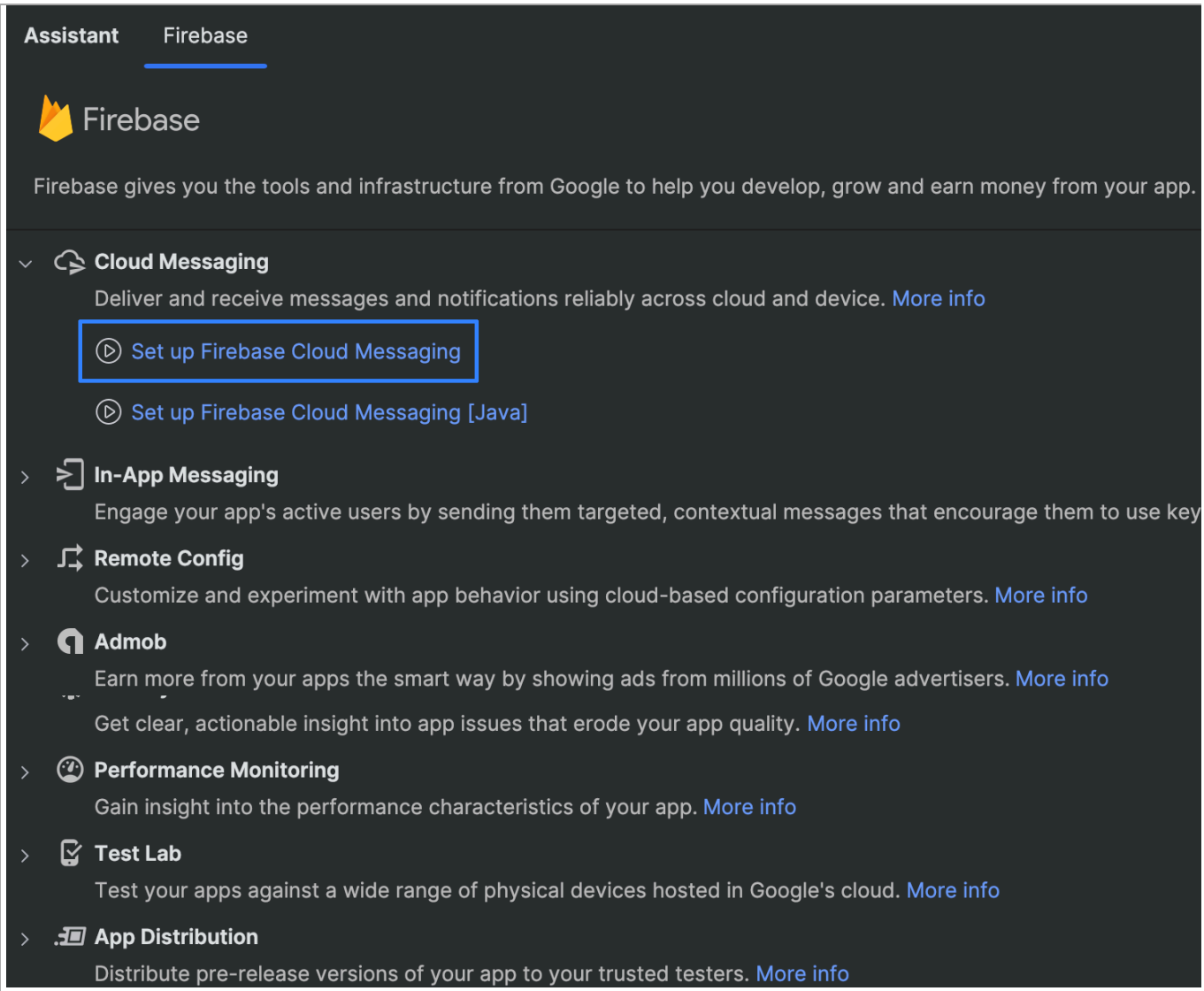
```
dependencies {
    implementation("com.google.firebase:firebase-messaging:25.0.0")
}
```



After adding the plugins and dependencies, sync your Gradle changes.

Automatic Setup in Android Studio

Navigate to Tools -> Firebase -> Cloud Messaging -> Set up Firebase Cloud Messaging:



Creating a Class Extending `FirebaseMessagingService()`

Create a class that extends `FirebaseMessagingService()`:

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    /**
     * Called when a new FCM token is generated.
     *
     * @param token The new FCM token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with additional metadata.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)

        // Message
    }
}
```

Updating the App Manifest

Add the `FCMService` declaration to your app's `AndroidManifest.xml` file:

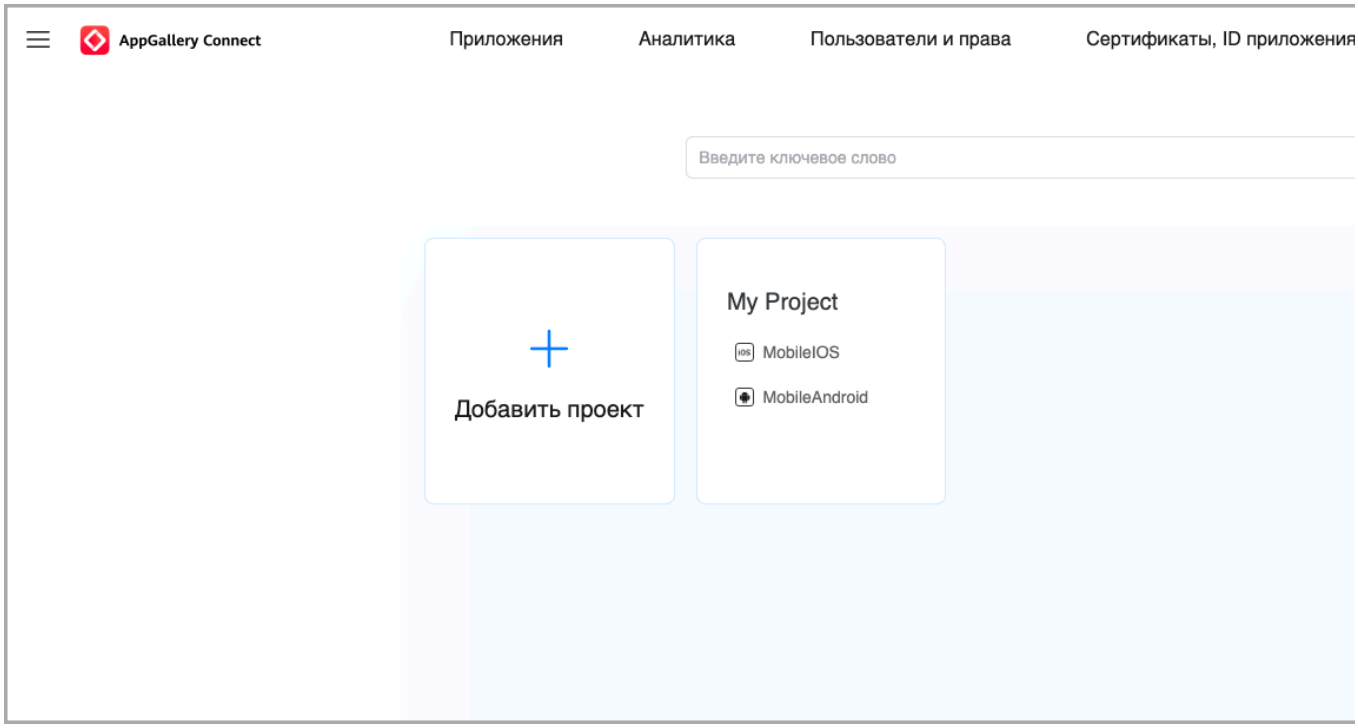
```
        <service
            android:name="com.altcraft.fcm.FCMService"
            android:exported="false">
            <intent-filter>
            </intent-filter>
        </service>
```

Replace `com.altcraft.fcm.FCMService` with your own package and class name.

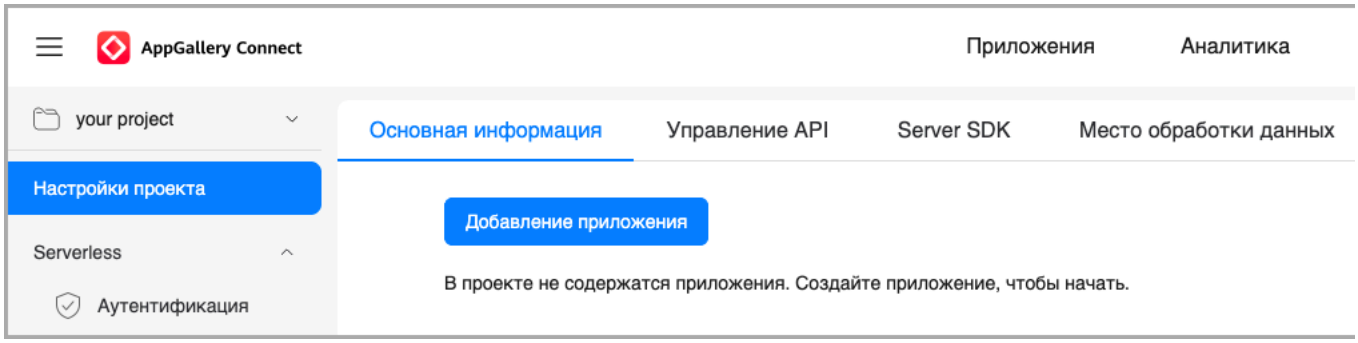
Huawei Mobile Services

Huawei Mobile Services Project Setup

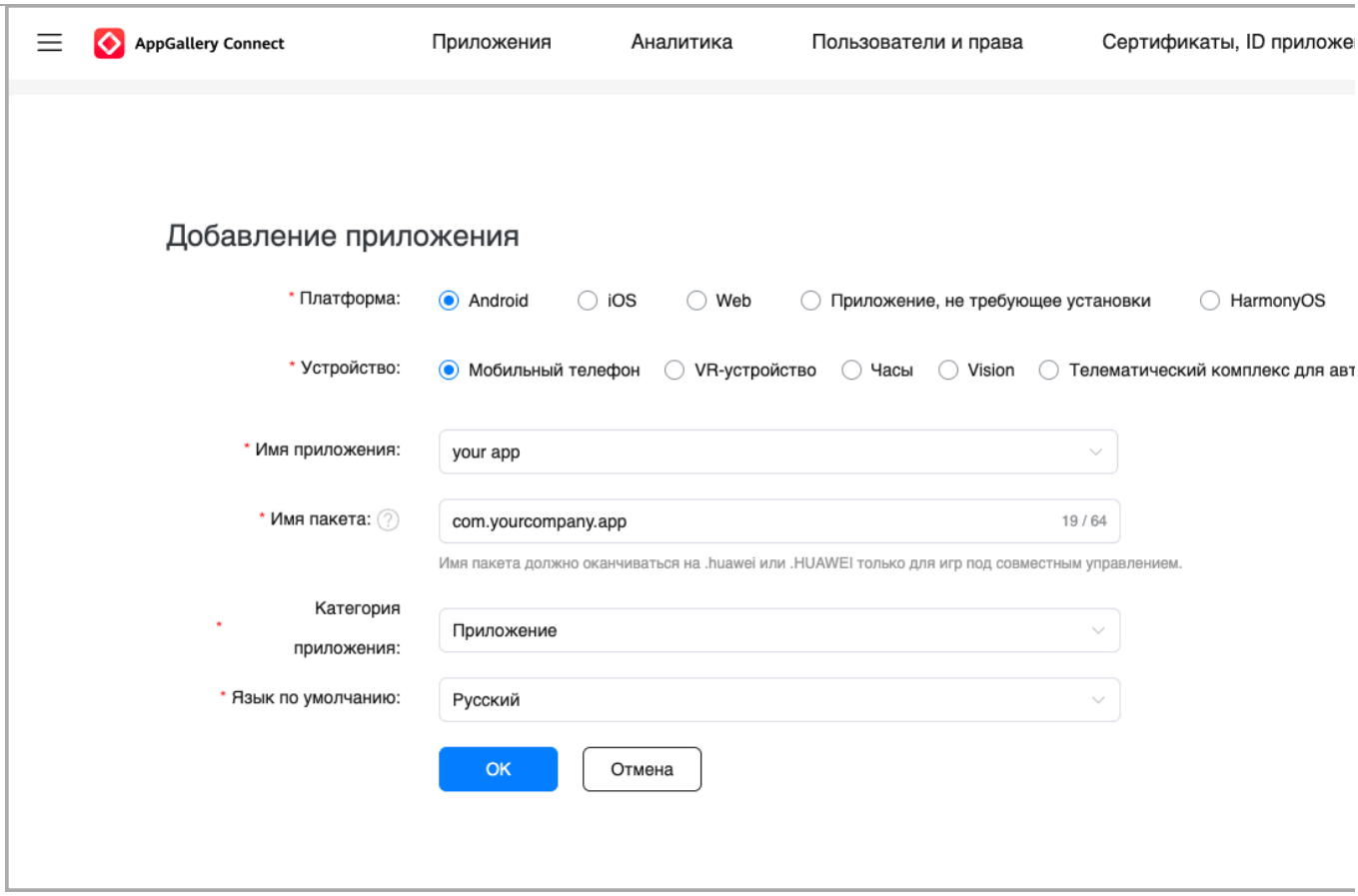
Create an [AppGallery Connect] project. Go to Development and Services —> Add project:



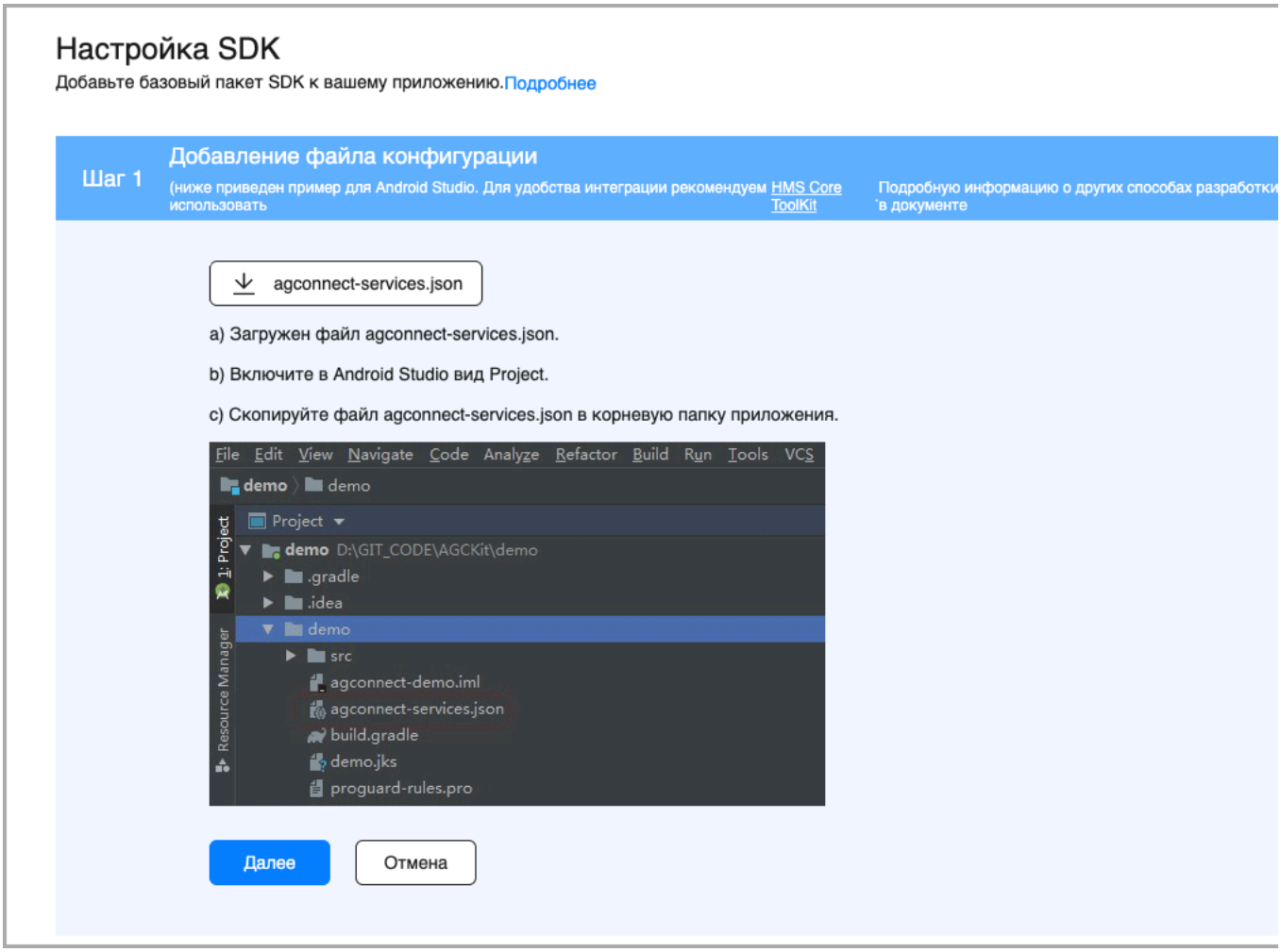
Add a new app to the project:



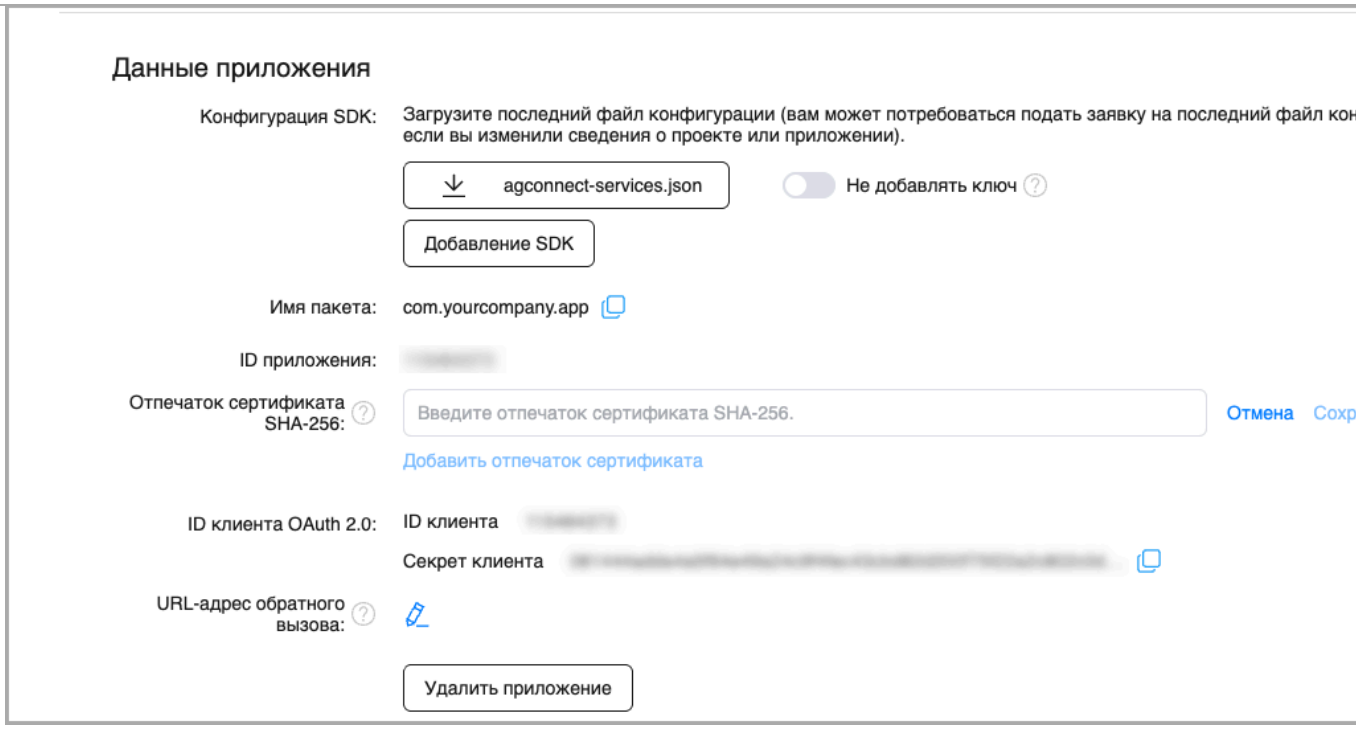
Provide your app details:



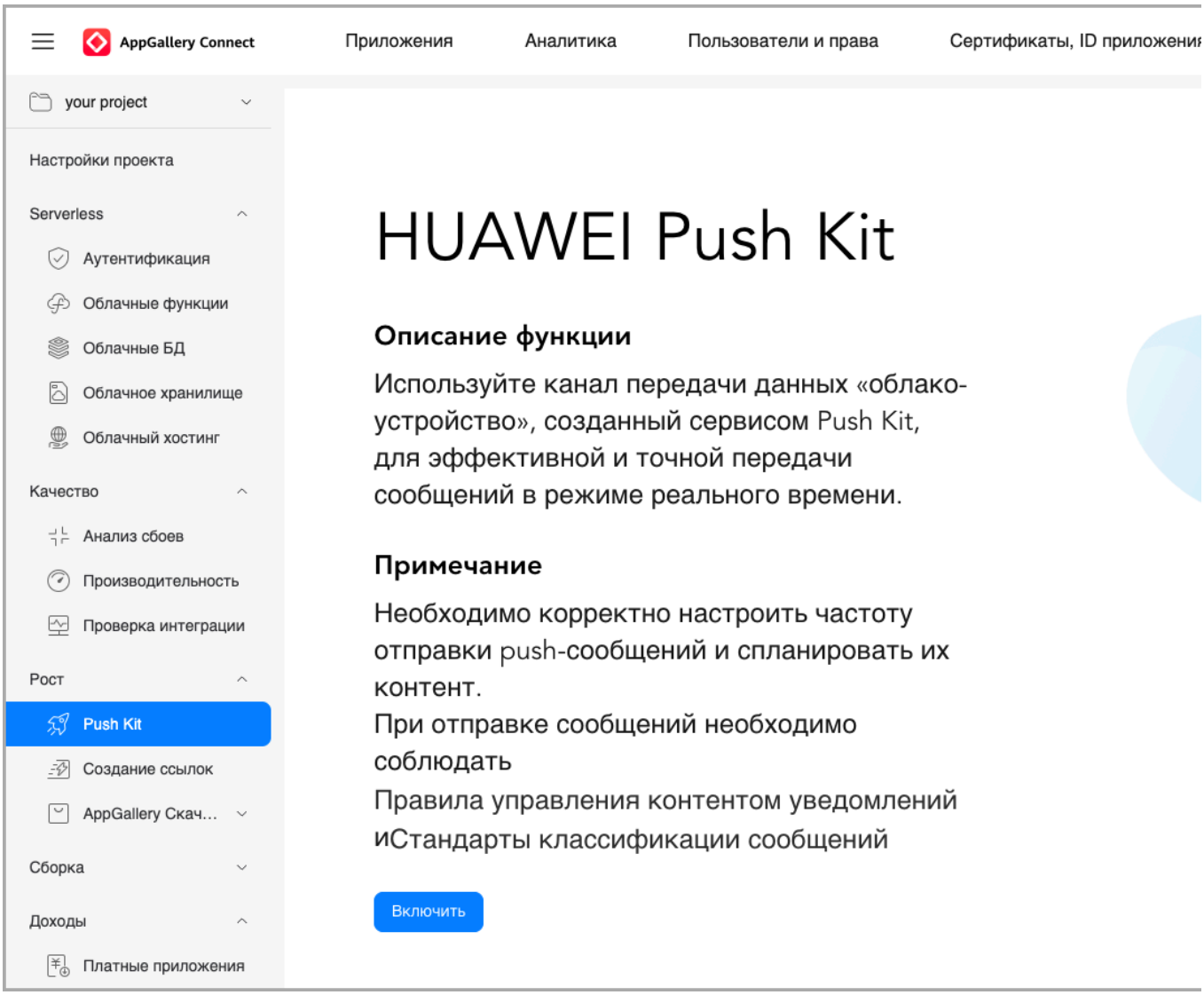
Download the `agconnect-services.json` file:



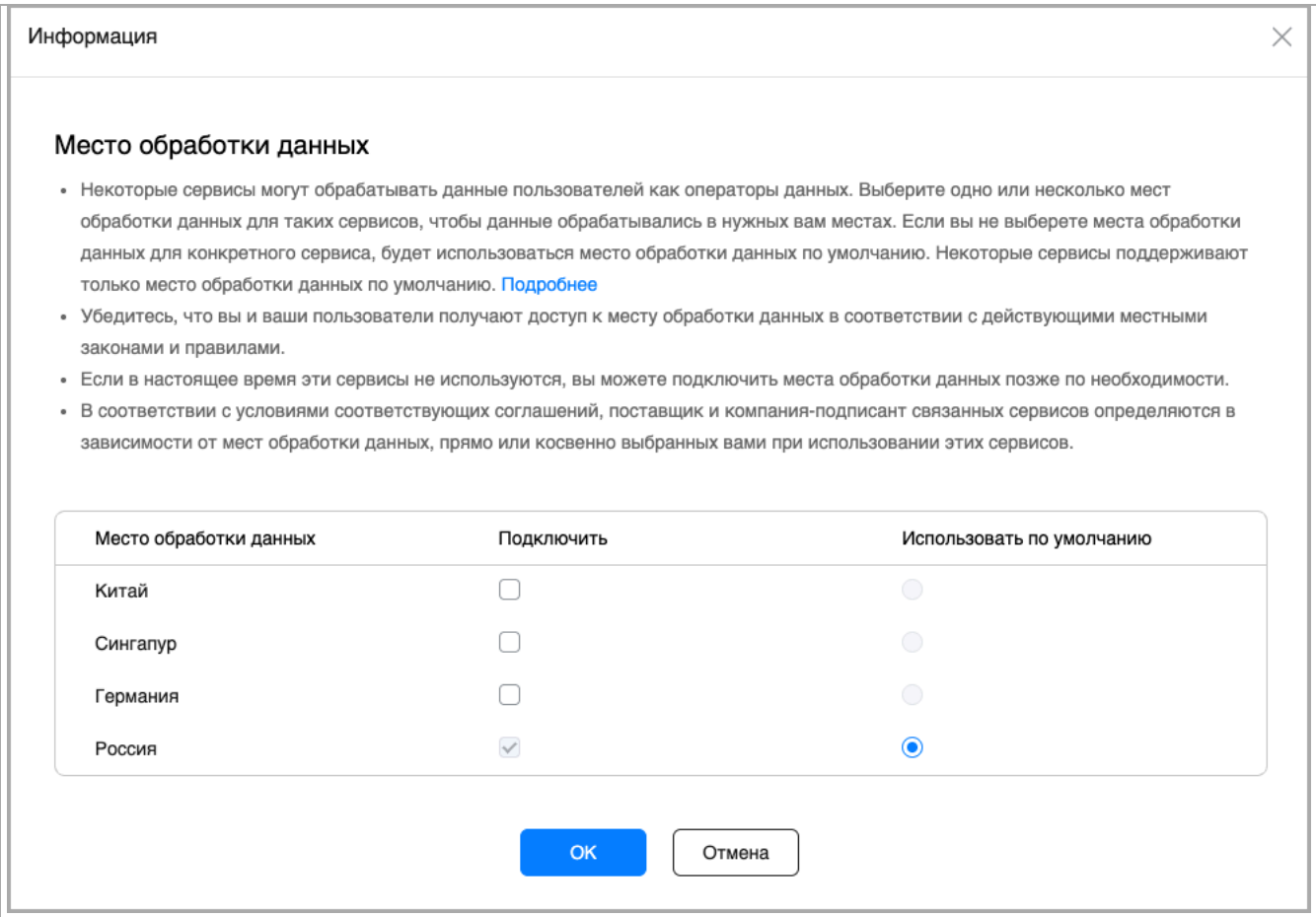
Return to the project settings and add the SHA-256 certificate fingerprint. You can obtain it by running directory of your app project:



In the left sidebar, select Push Kit under Growth:

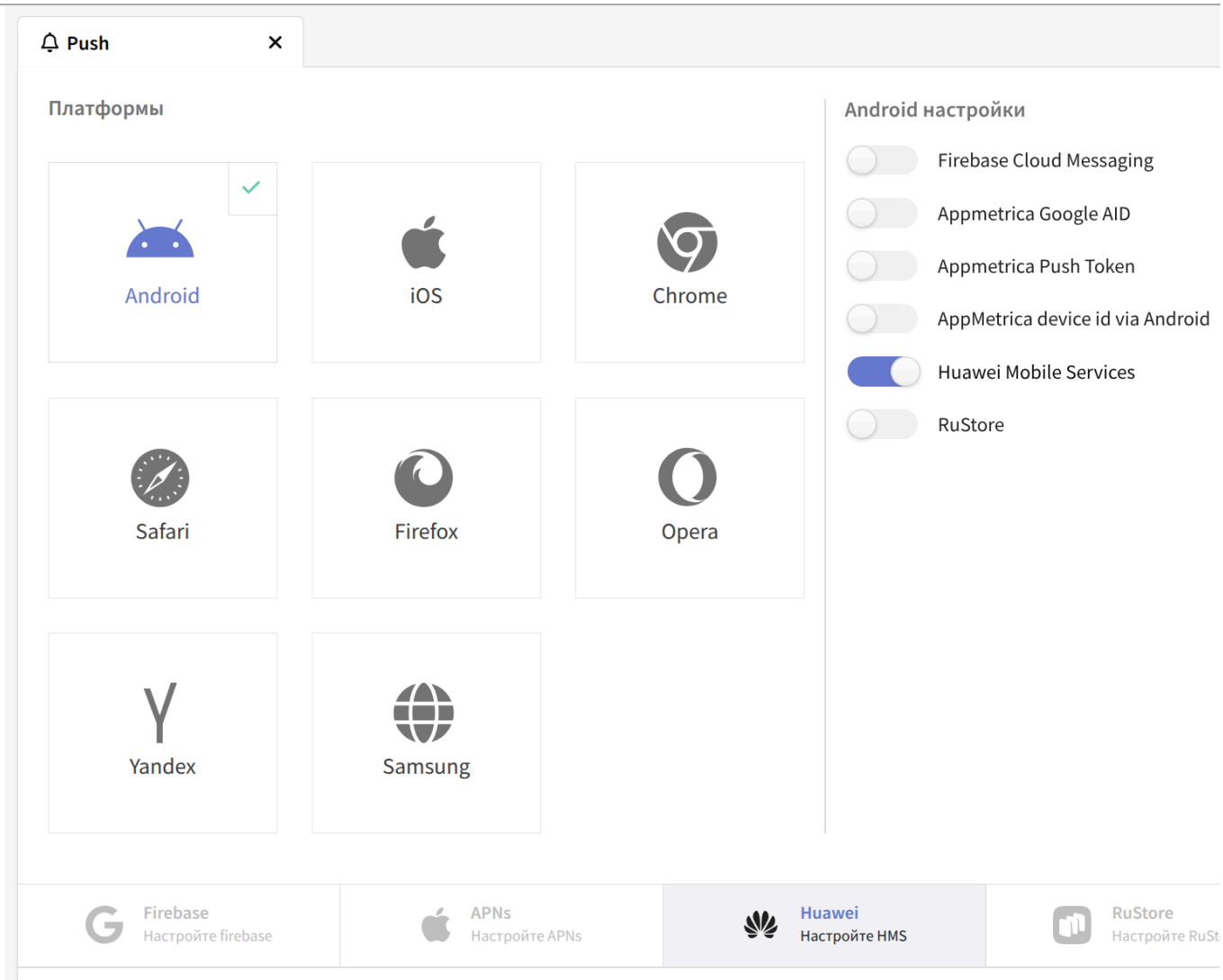


Click Enable —> Select data storage location:

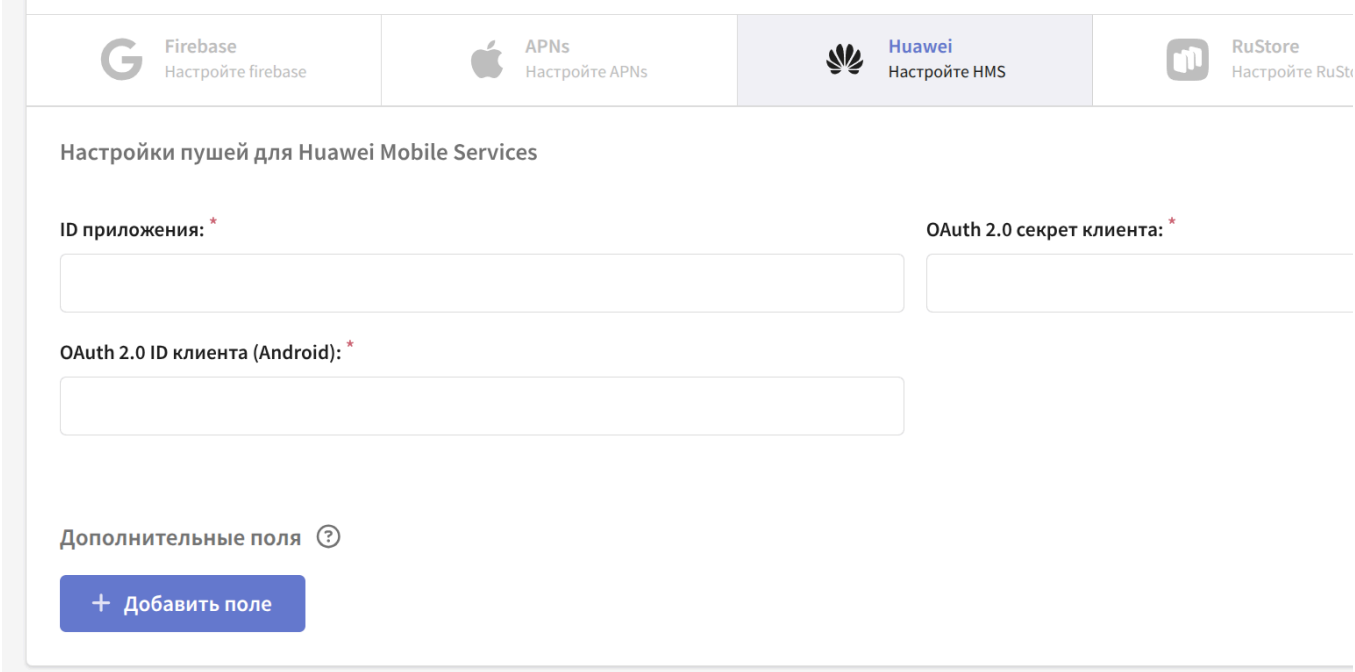


Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existi settings, select the Android platform and enable Huawei Mobile Services:



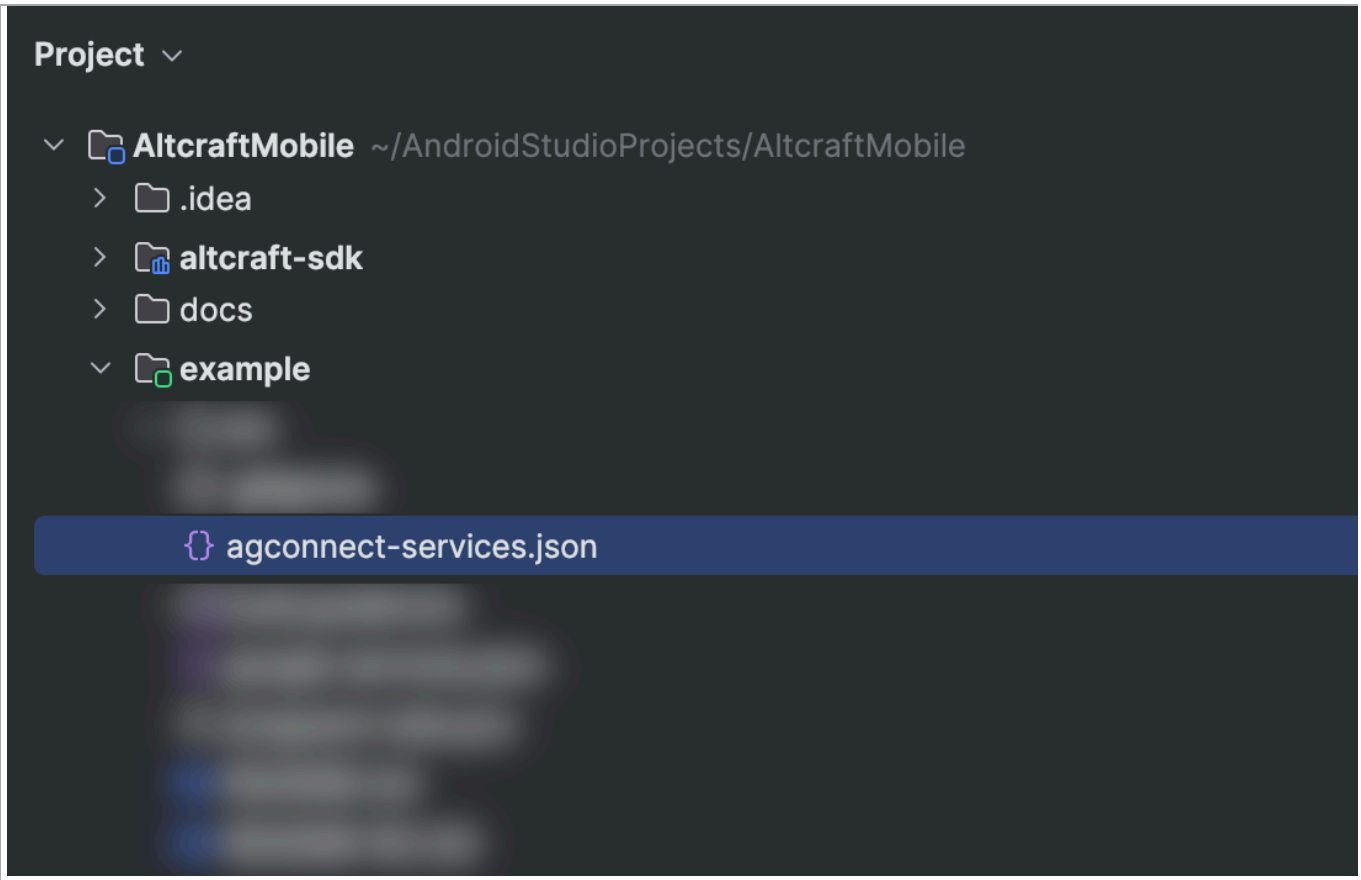
Scroll down to the Huawei settings and specify the App ID, OAuth 2.0 client ID, and OAuth 2.0 client se



After making changes, don't forget to save the resource.

Integrating Huawei Push Kit into the App Project

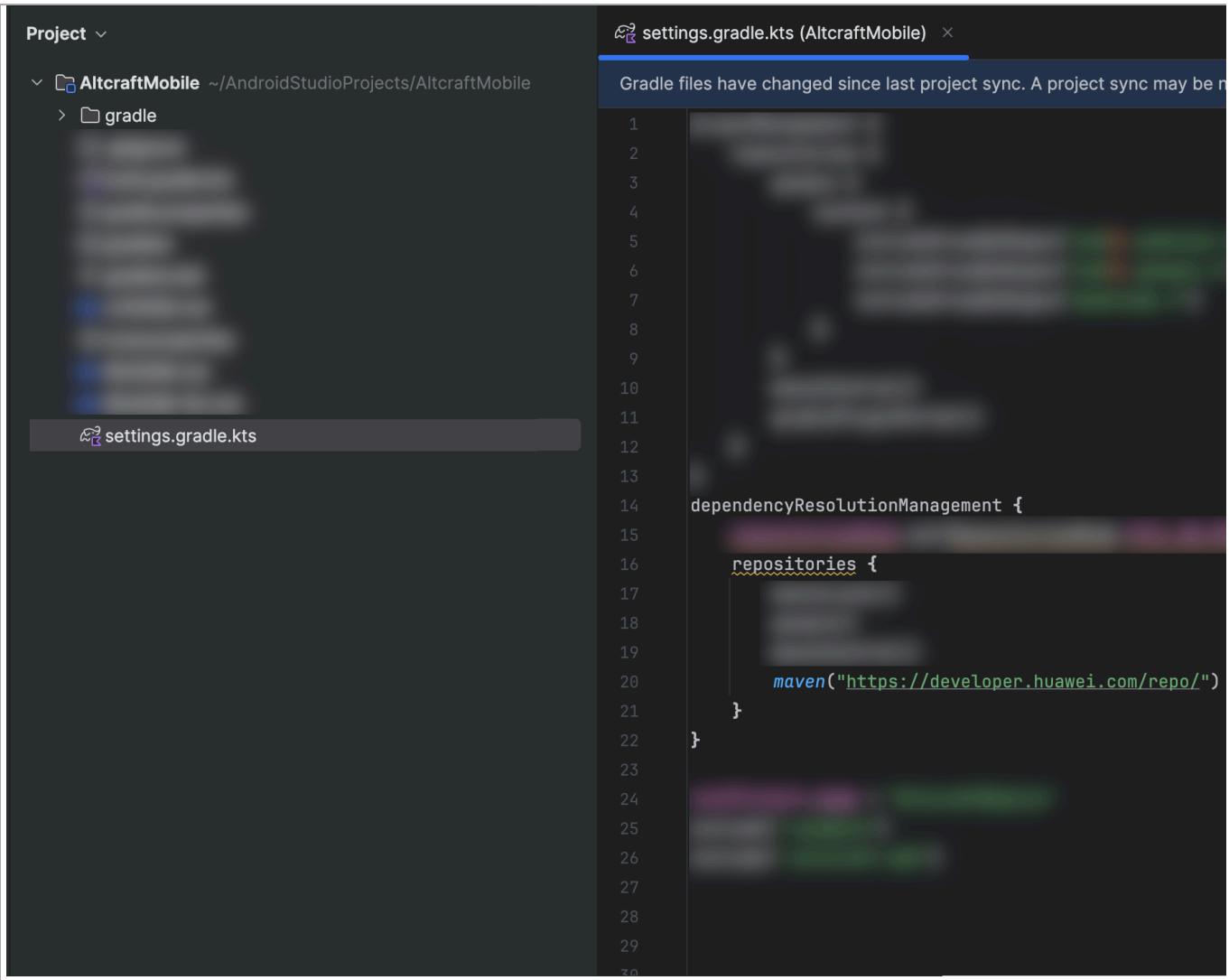
Move the downloaded `agconnect-services.json` **file to the root directory of your app module (app):**



Add repositories and dependencies to your Gradle files:

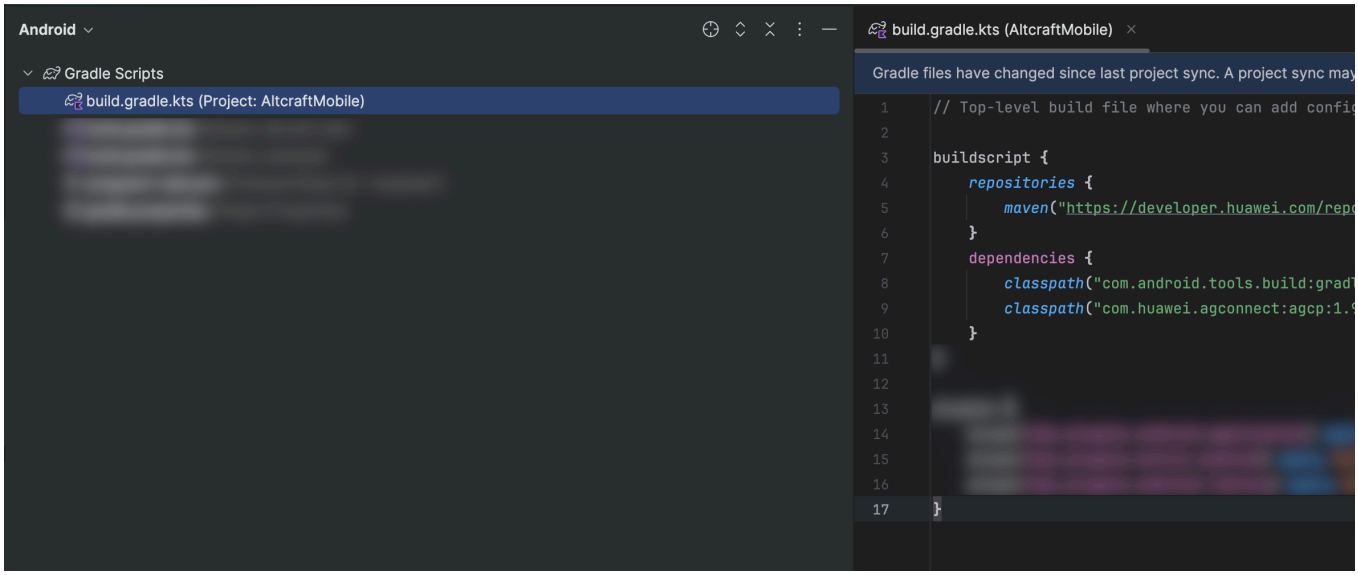
`settings.gradle.kts (project settings)`

```
dependencyResolutionManagement {
    repositories {
        // Other repositories..
        maven("https://developer.huawei.com/repo/")
    }
}
```



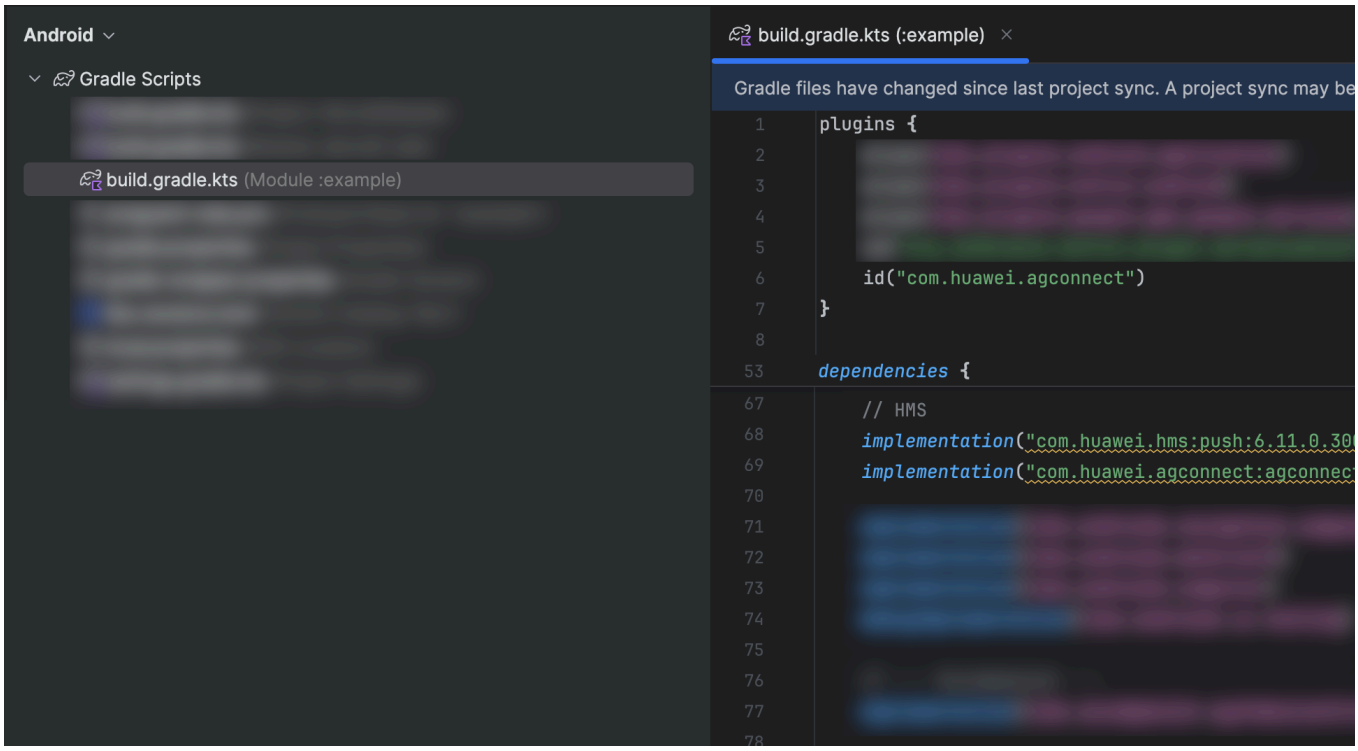
`build.gradle.kts (project level)`

```
buildscript {
    repositories {
        maven("https://developer.huawei.com/repo/")
    }
    dependencies {
        classpath("com.android.tools.build:gradle:8.10.1") // Important: use the AGP version your project uses.
        classpath("com.huawei.agconnect:agcp:1.9.1.301")
    }
}
```



`build.gradle.kts (app level)`

```
plugins {
    id("com.huawei.agconnect")
}

dependencies {
    implementation("com.huawei.hms:push:6.11.0.300")
    implementation("com.huawei.agconnect:agconnect-core:1.7.3.302")
}
```



TIP: Note

Information about the latest dependency versions is available on Huawei Developer pages:

- **AppGallery Connect**
- **HMS Push Core**

After adding the repository and dependencies, sync your Gradle changes.

Creating a Class Extending `HmsMessageService()`

Create a class that extends `HmsMessageService()`:

```
/**
 * HMS service for handling push tokens and incoming notifications.
 *
 * Extends [HmsMessageService] and overrides key HMS callback methods.
 */
class HMSService : HmsMessageService() {
    /**
     * Called when a new HMS token is generated.
     *
     * @param token The new HMS token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received from HMS.
     *
     * Forwards the message with additional metadata to all receivers.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        // Message
    }
}
```

Updating the App Manifest

Add the `HMSService` declaration to your app's `AndroidManifest.xml`:

```
<service
    android:name="com.altcraft.hms.HMSService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Replace `com.altcraft.hms.HMSService` with your own package and class name.

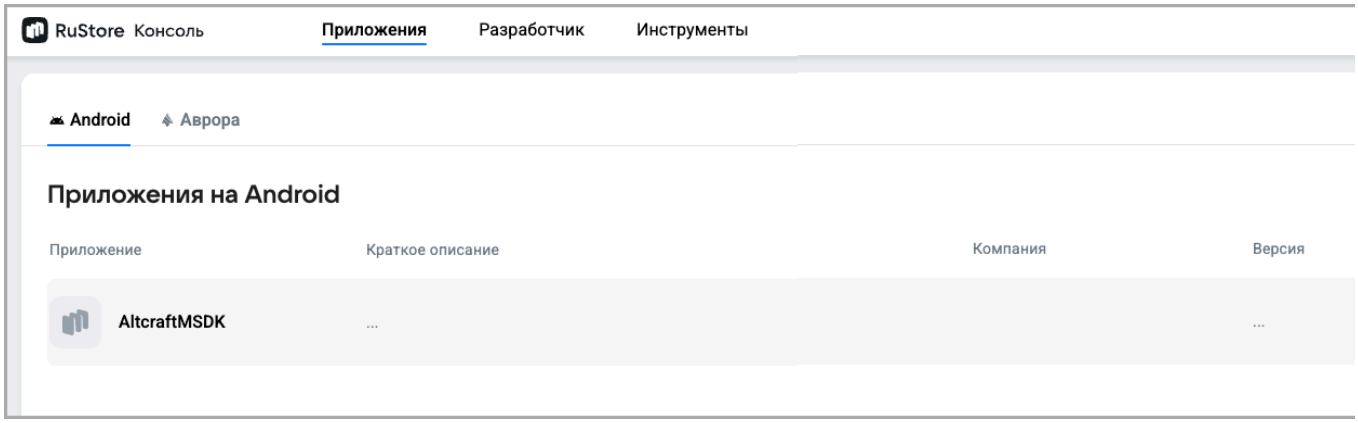
RuStore

INFO: Note

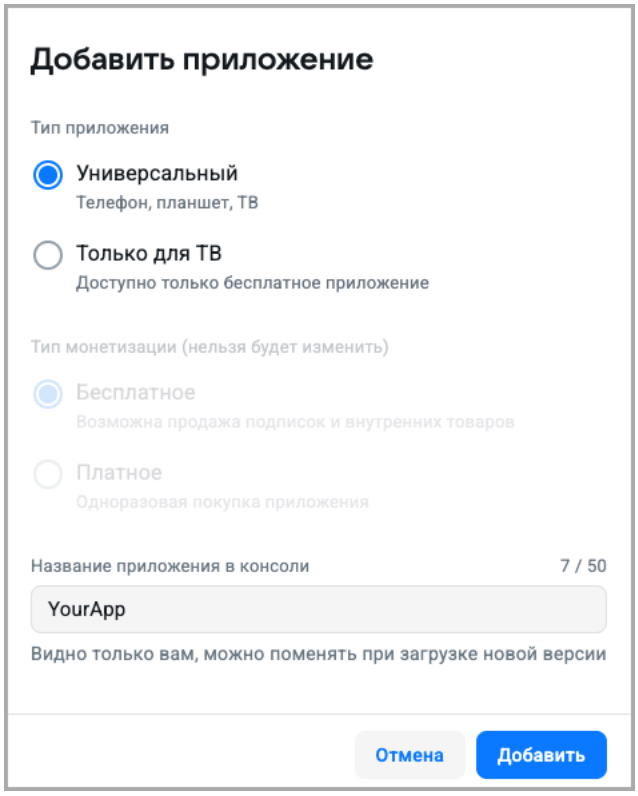
The RuStore app must be installed on the device.

RuStore Developer Console Project Setup

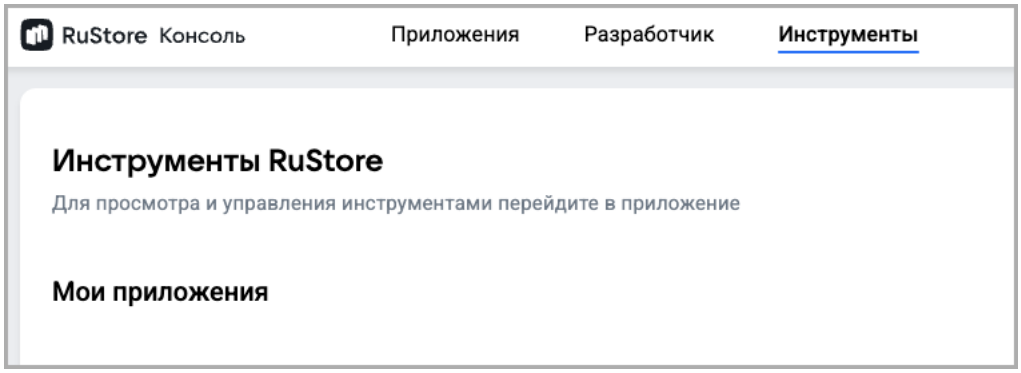
Go to the [RuStore Developer Console]:



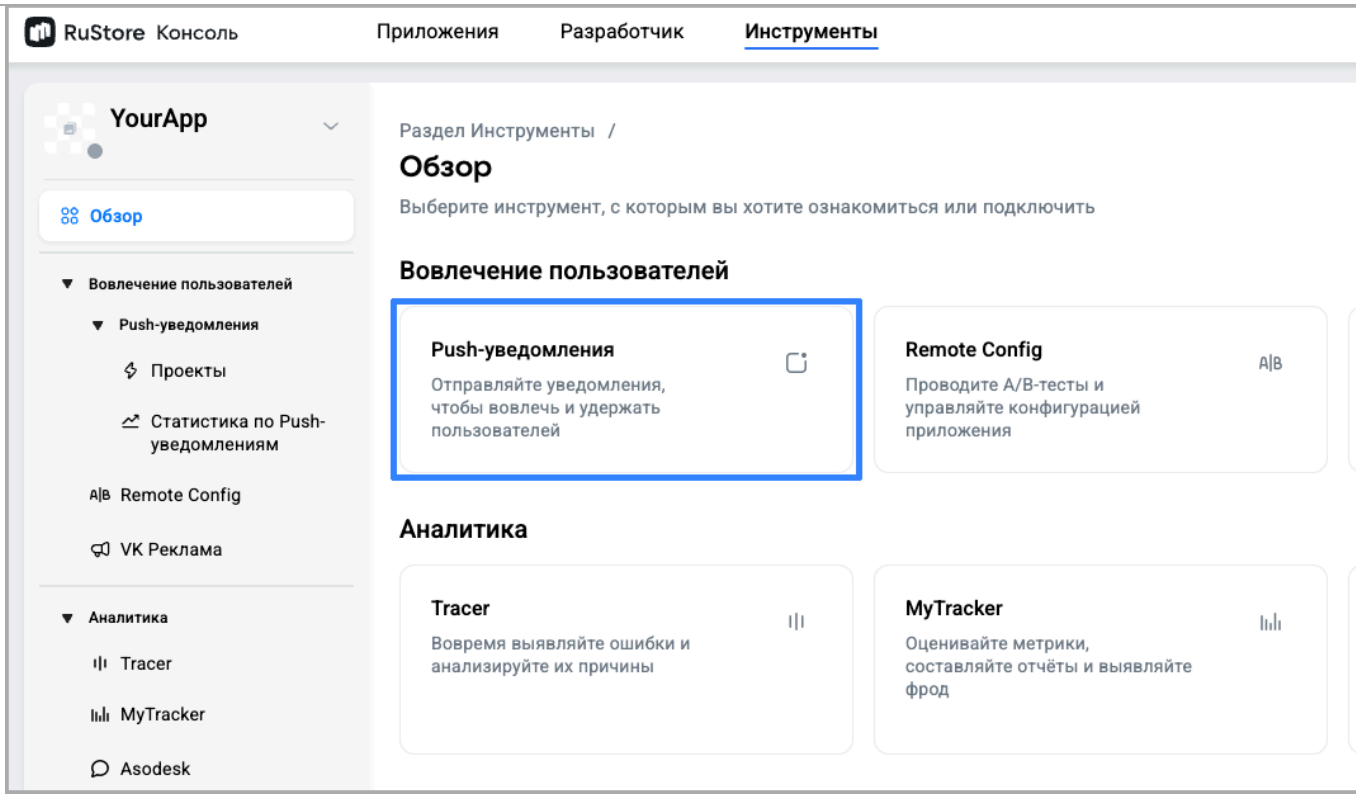
Add an app under the Application section:



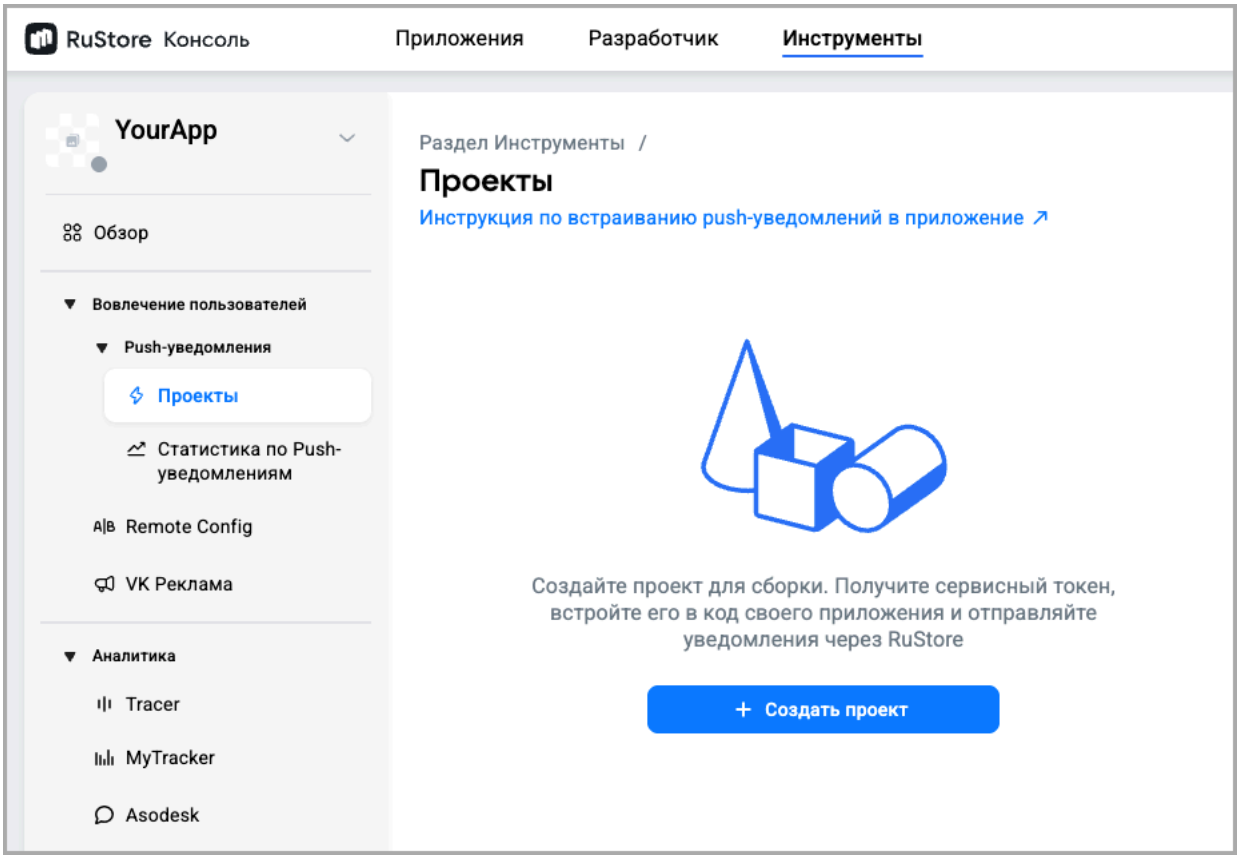
Go to the Tools section in the developer console:



Select your created app and open Push Notifications:

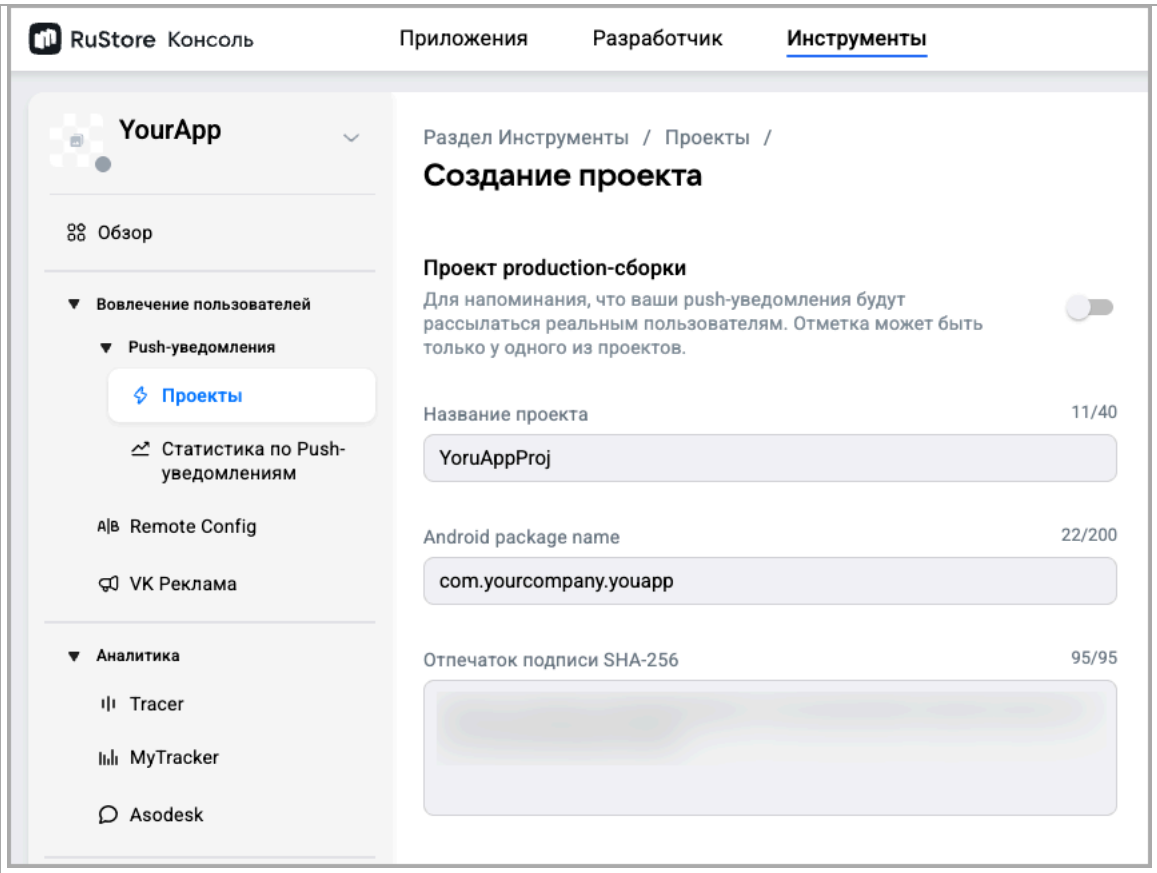


Create a new project:

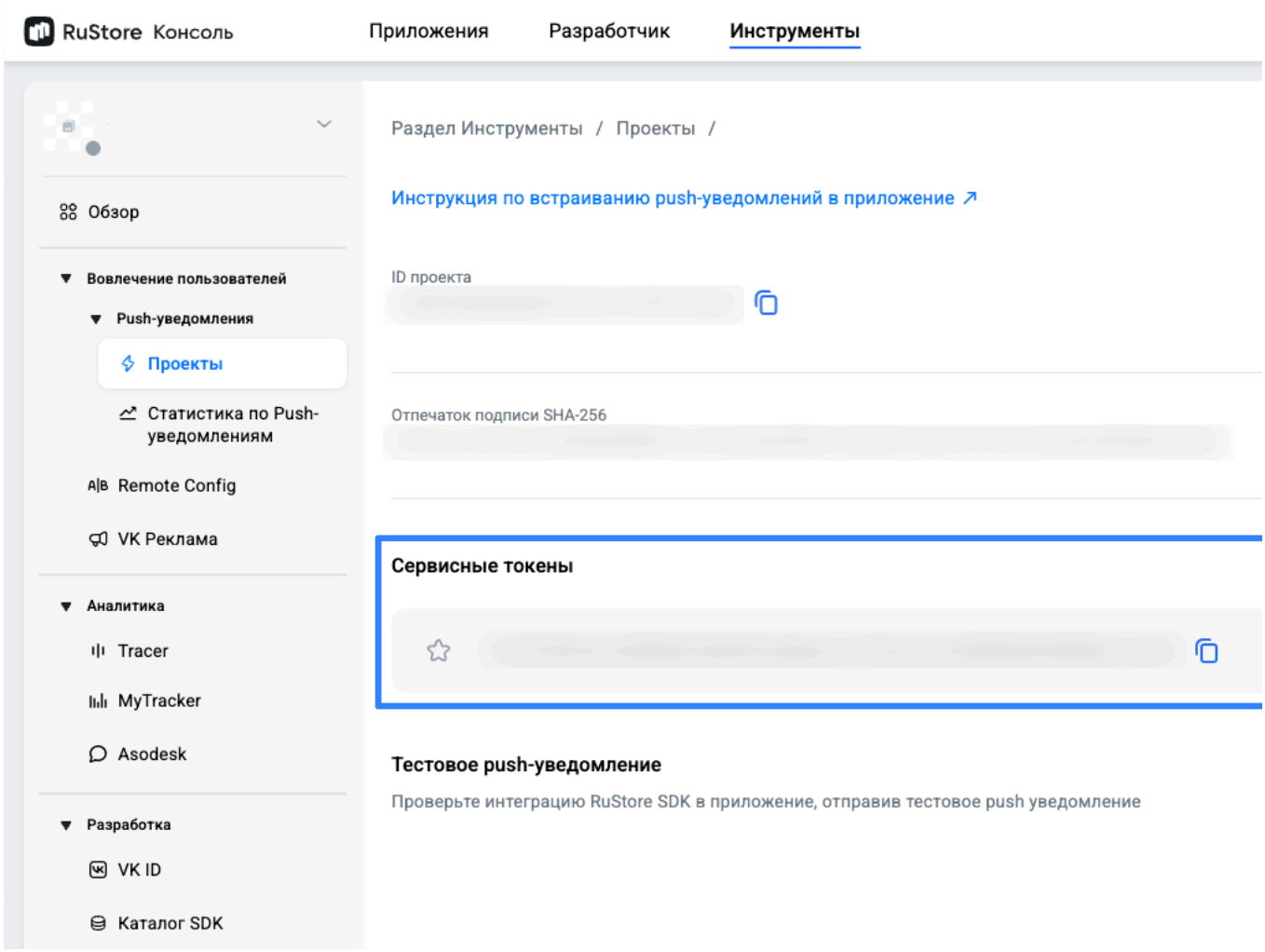


In the project setup, specify:

- **Project name**
- **App package name**
- **SHA-256 signature fingerprint (you can get it by running the `./gradlew signingReport` command in the**

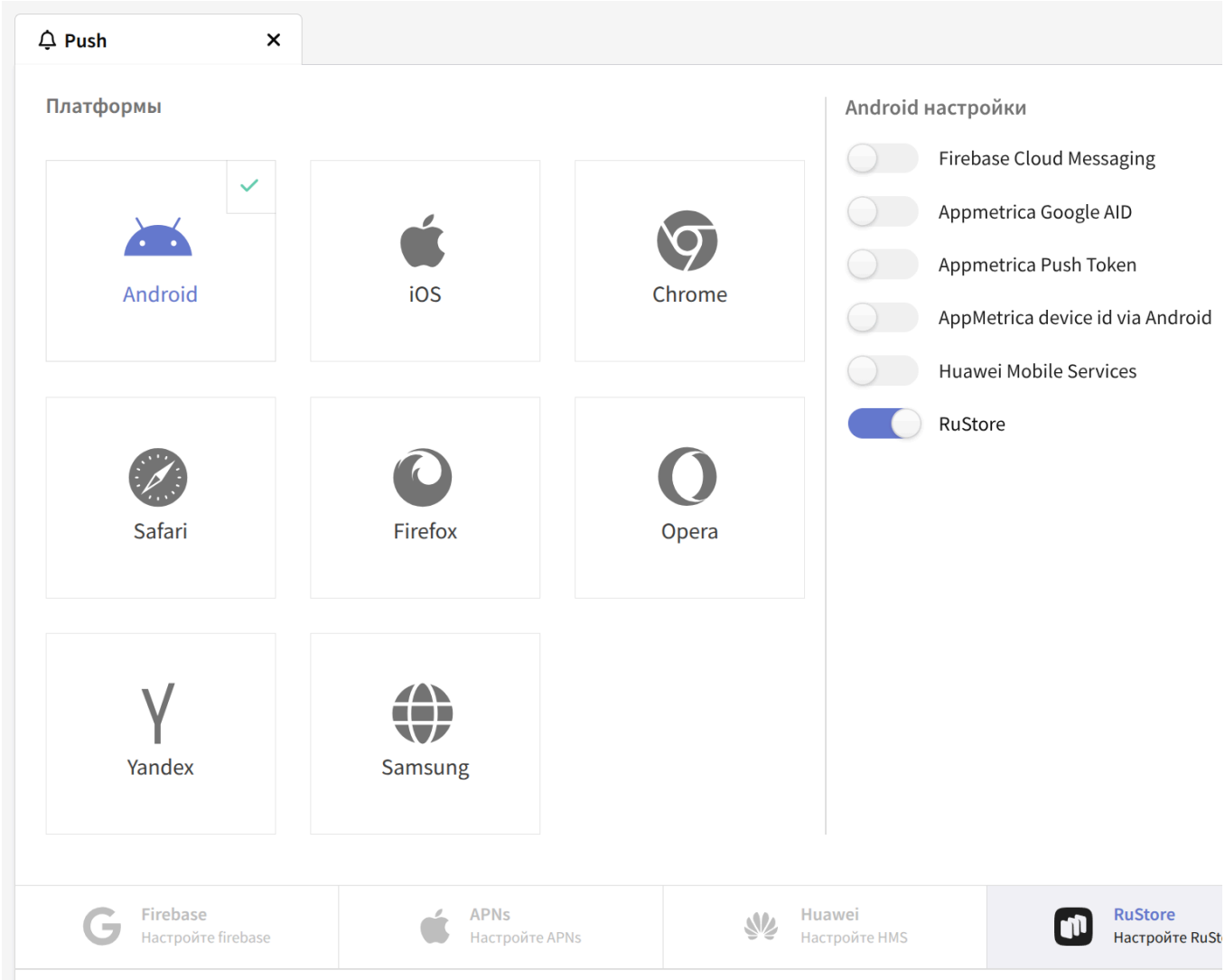


Inside the created project, generate a service token:

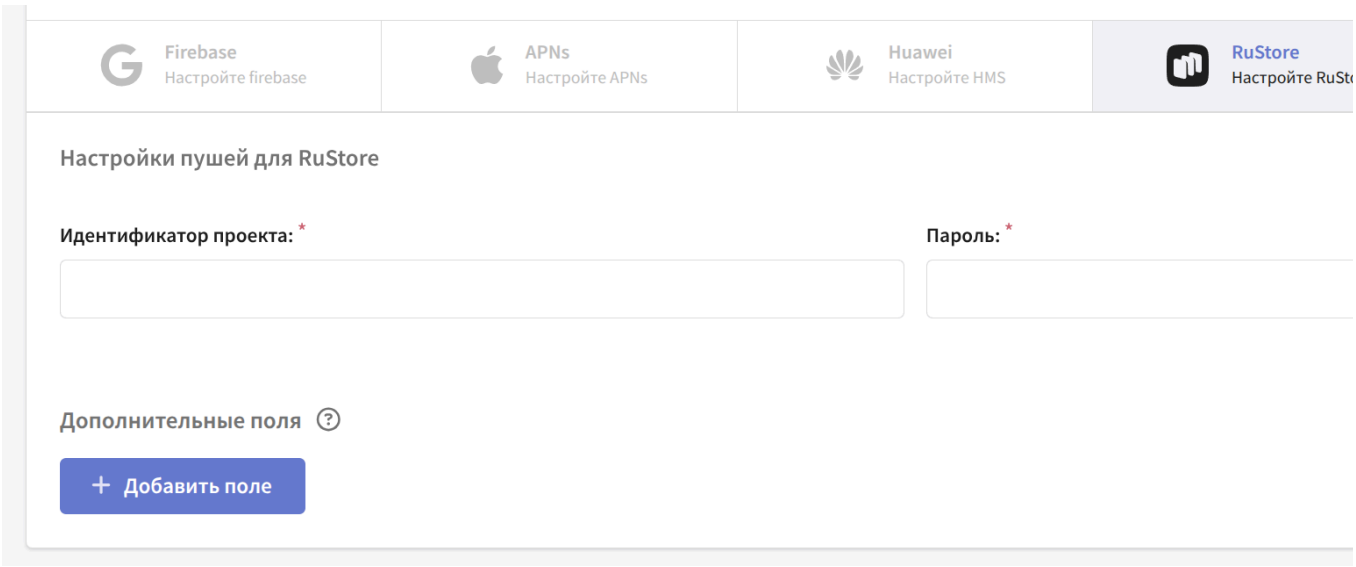


Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existi
settings, select the Android platform and enable RuStore:



Scroll down to the RuStore settings and specify the Project ID and Password:



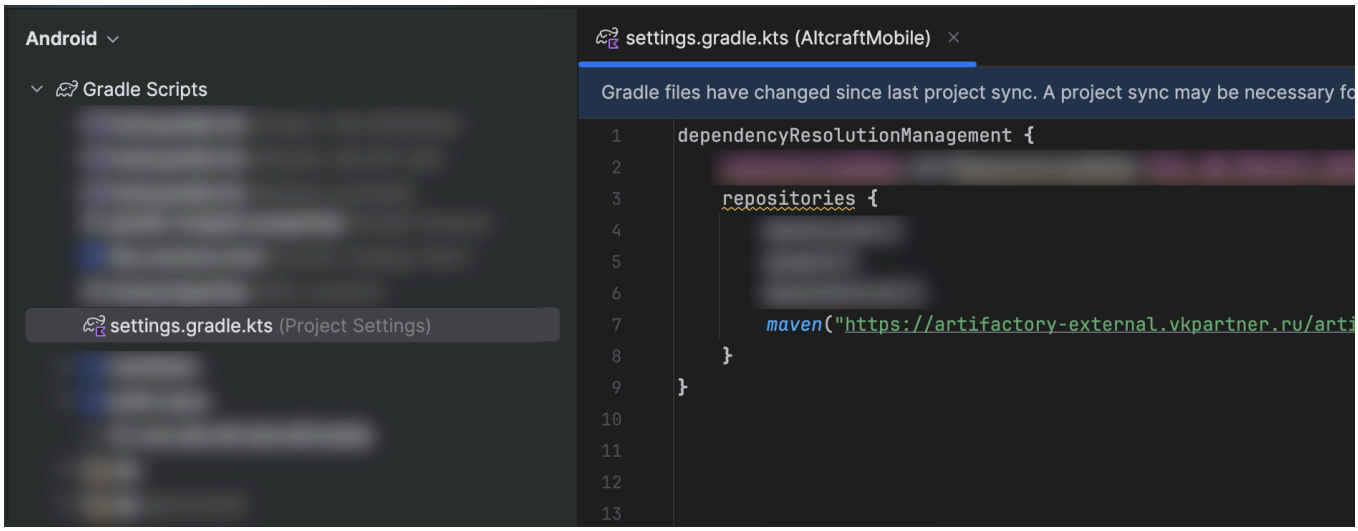
After making changes, don’t forget to save the resource.

Integrating RuStore Push Messaging into the App Project

Add repositories and dependencies to your Gradle files:

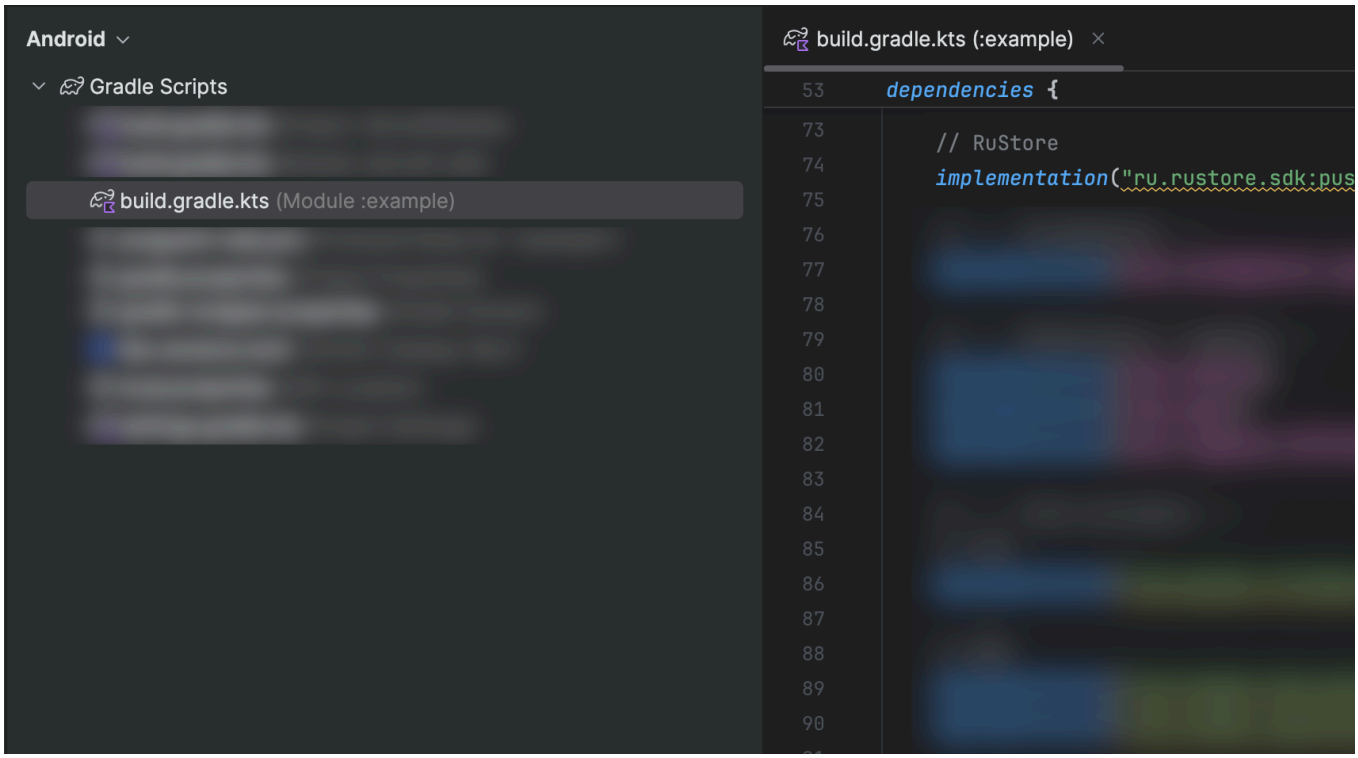
`settings.gradle.kts (project settings)`

```
dependencyResolutionManagement {
    repositories {
        // Other repositories..
        maven("https://artifactory-external.vkpartner.ru/artifactory/maven")
    }
}
```



`build.gradle.kts (app level)`

```
dependencies {
    implementation("ru.rustore.sdk:pushclient:6.10.0")
}
```



> **TIP: Note**
>
> **Information about the latest dependency versions is available here.**

After adding the repository and dependencies, sync your Gradle changes.

Initializing `RuStorePushClient`

Initialize `RuStorePushClient` in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        RuStorePushClient.init(this, "project_ID")
    }
}
```

Creating a Class Extending `RuStoreMessagingService()`

Create a class that extends `RuStoreMessagingService()`:

```
/**
 * RuStore service for handling push notifications.
 *
 * Extends [RuStoreMessagingService] and overrides key callbacks.
 */
class RuStoreService : RuStoreMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with added metadata.
     *
     * @param message The received RemoteMessage.
     */
    override fun onMessageReceived(message: RemoteMessage) {
        // Message
    }
}
```

Updating the App Manifest

Add the `RuStoreService` declaration to your app's `AndroidManifest.xml`:

```
<service
    android:name="com.altcraft.rustore.RuStoreService"
    android:exported="true"
    tools:ignore="ExportedService">
    <intent-filter>
    </intent-filter>
</service>
```

Replace `com.altcraft.rustore.RuStoreService` with your package and class name. Note that the manifest mu
`android:exported="true"`.

> **TIP: Conditions for Proper RuStore Notification Functionality**
>
> - **The RuStore app is installed on the device.**
> - **The RuStore app has notification permissions.**
> - **The RuStore app is excluded from battery optimization.**

Quick Start

This article provides a rapid integration example of Altcraft mSDK with Apple Push Notification Systen Messaging (FCM). After completing the steps, your app will be able to receive Altcraft push notificatior To use Huawei Mobile Services, add the corresponding SDKs to the project and implement their interfa shown for APNS and FCM.

Step 0. Prerequisites

- **APNS access obtained on the website;**
- **The app is configured in a Firebase project, and `GoogleServiceInfo.plist` is downloaded.**

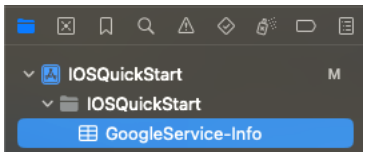
Step 1. Altcraft SDK integration into the app

Add packages from repositories using Swift Package Manager:

- **Repository Altcraft SDK;**
- **Repository Firebase.**

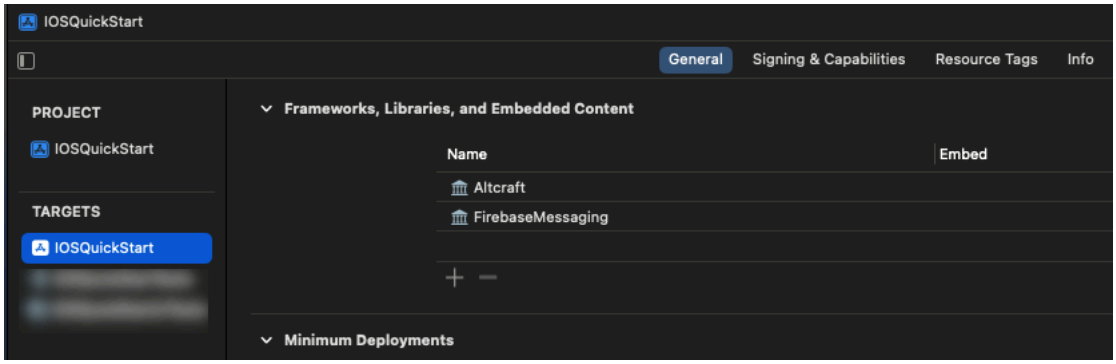
Step 2. Add `GoogleServiceInfo` to the app

Add `GoogleServiceInfo.plist` to your app folder:

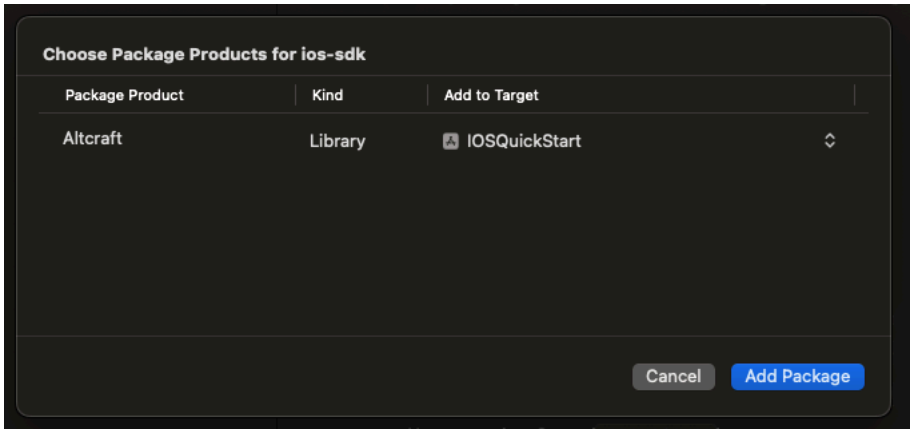


Step 3. App preparation

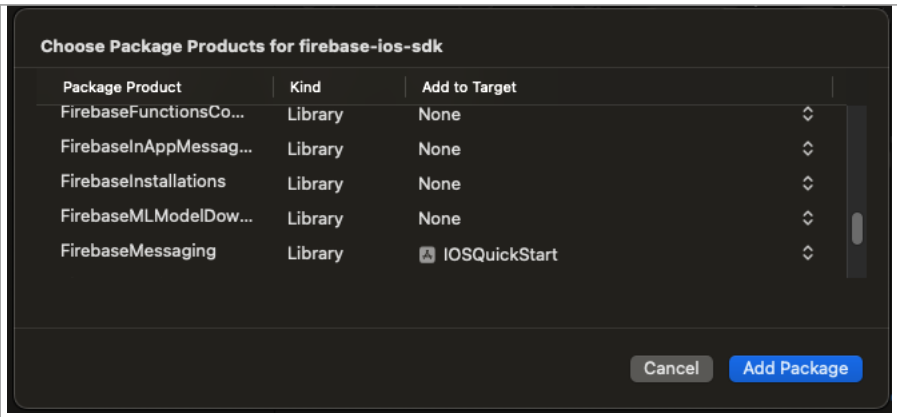
Configure the following app target settings:



- **General:**
  - **Ensure the Altcraft library is added under Frameworks, Libraries, and Embedded Content of th**



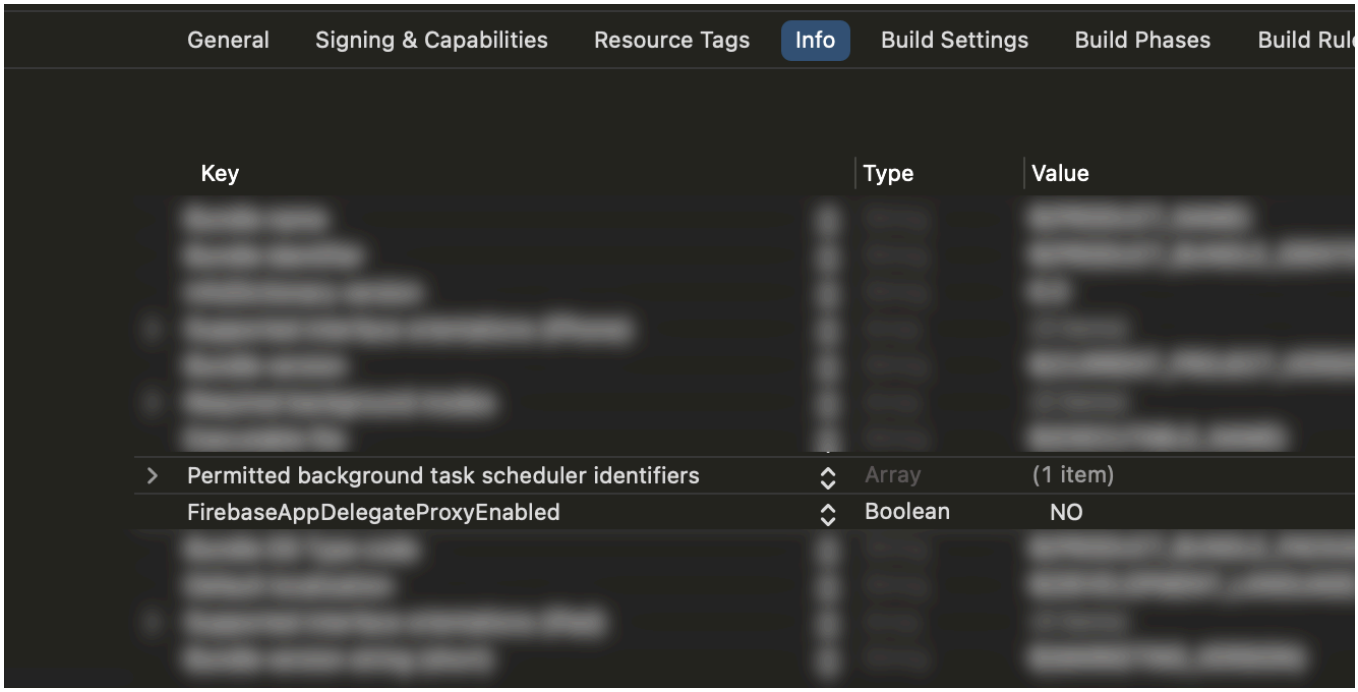

```
* Ensure the **FirebaseMessaging** library is added under **Frameworks, Libraries, and Embedded Content** of the app target:
```



- **Signing & Capabilities:**
  - **PushNotifications;**
  - **AppGroups — specify a group identifier (adding an App Group identifier is required for data exc Extension);**
  - **Background Modes — enable Background fetch, Remote notification, Background processing.**
- **Info:**
  - **Add the following keys and values:**

Key	Type	
Permitted background task scheduler identifiers	**Array**	**“lib.Altcraft.bgTask.system**
FirebaseAppDelegateProxyEnabled	**Boolean**	**NO**

The first key registers a `bgTask` that retries failed server requests in background mode. The second ke swizzling of `AppDelegate` that Firebase Messaging performs by default.



Step 4. JWTInterface realization

Implement the SDK protocol for providing a JWT token. Create a class that implements `JWTInterface` an

```
class JWTProvider: JWTInterface {
    func getToken() -> String? {
        // your code returning JWT
    }
}
```

TIP: Note

`getJWT()` is a synchronous function. The SDK execution thread will be paused until JWT is obtained. It returns immediately from cache (in-memory, `UserDefaults` or `Keychain`) to speed up requests. Ideally, pr possible (at app start) and store it in cache so the token is available without delays. Returning `nil` is available.

Step 5. FCMInterface realization

Implement the SDK protocol for providing and deleting the FCM token. Create a class that implements `deleteToken()`.

```
class FCMProvider: FCMInterface {

    /// Returns the current FCM token
    func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// set the APNs token for FCM before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// request the FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

Step 6. APNSInterface realization

Implement the SDK protocol for providing the APNS token. Create a class that implements `APNSInterface`

```
class APNSProvider: APNSInterface {
    /// Retrieves the current APNs token from local storage
    func getToken(completion: @escaping (String?) -> Void) {
        let token = getAPNsTokenFromUserDefault()  // pass the APNs token
        completion(token)
    }
}
```

Step 7. Configure Altcraft SDK and Firebase SDK in `AppDelegate`

Altcraft SDK:

```
- pass the `AppGroups` identifier to Altcraft SDK;
- register Altcraft `backgroundTasks`;
- set JWT, APNS, and FCM providers;
- register with the notification center.
```

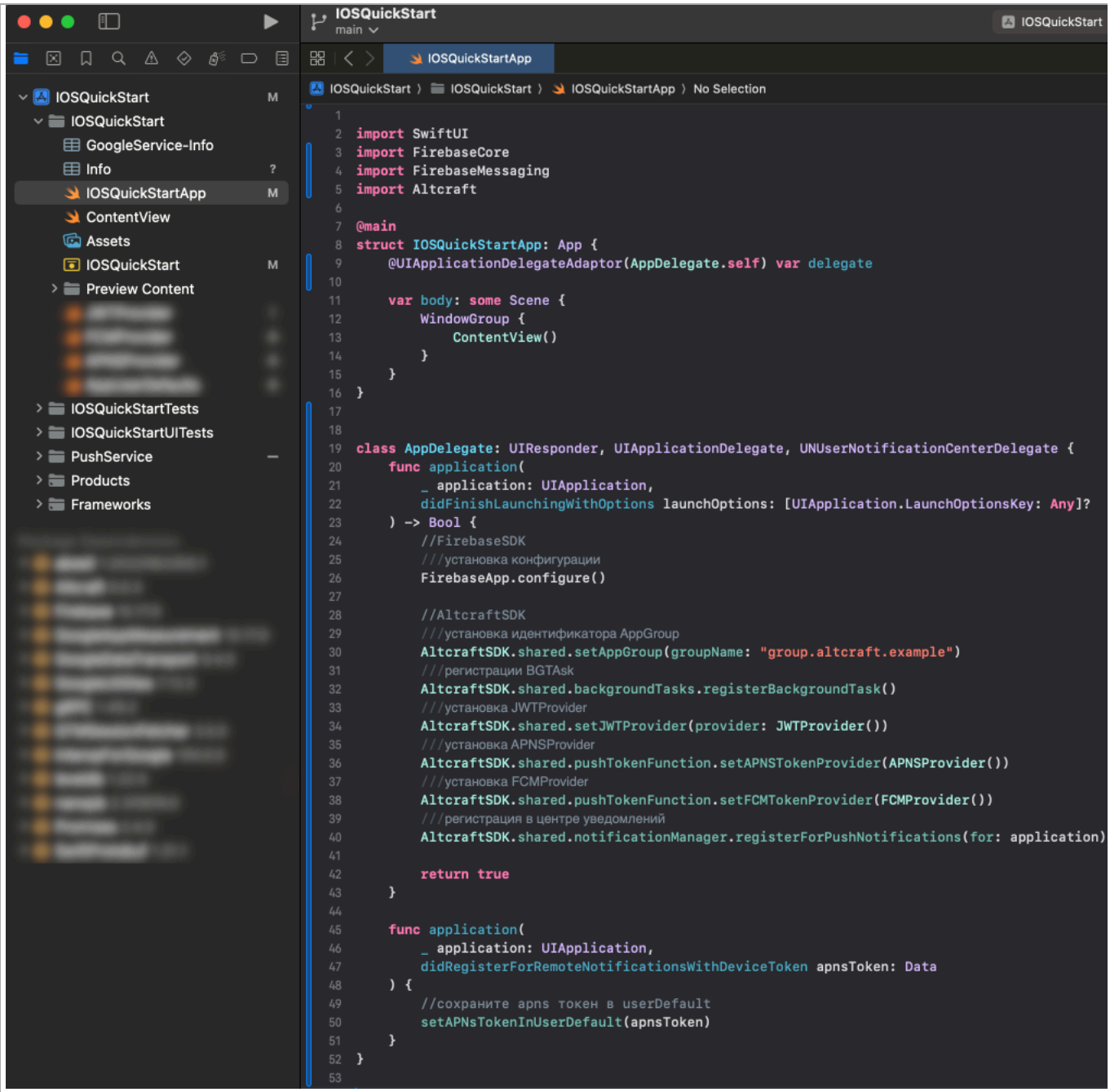
Firebase SDK:

- **apply the configuration;**

Example of correct Altcraft SDK and Firebase SDK setup in `AppDelegate`

```
class AppDelegate: UIResponder, UIApplicationDelegate,  UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        // Firebase SDK configuration
        FirebaseApp.configure()

        // Altcraft SDK
        /// set App Group identifier
        AltcraftSDK.shared.setAppGroup(groupName: "group.your_company.name")
        /// register BGTask
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        /// set JWTProvider
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())
        /// set APNSProvider
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
        /// set FCMProvider
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        /// register with notification center
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        return true
    }

    func application(
        _ application: UIApplication,
        didRegisterForRemoteNotificationsWithDeviceToken apnsToken: Data
    ) {
        // save APNs token in UserDefaults
        setAPNsTokenInUserDefault(apnsToken)
    }
}
```

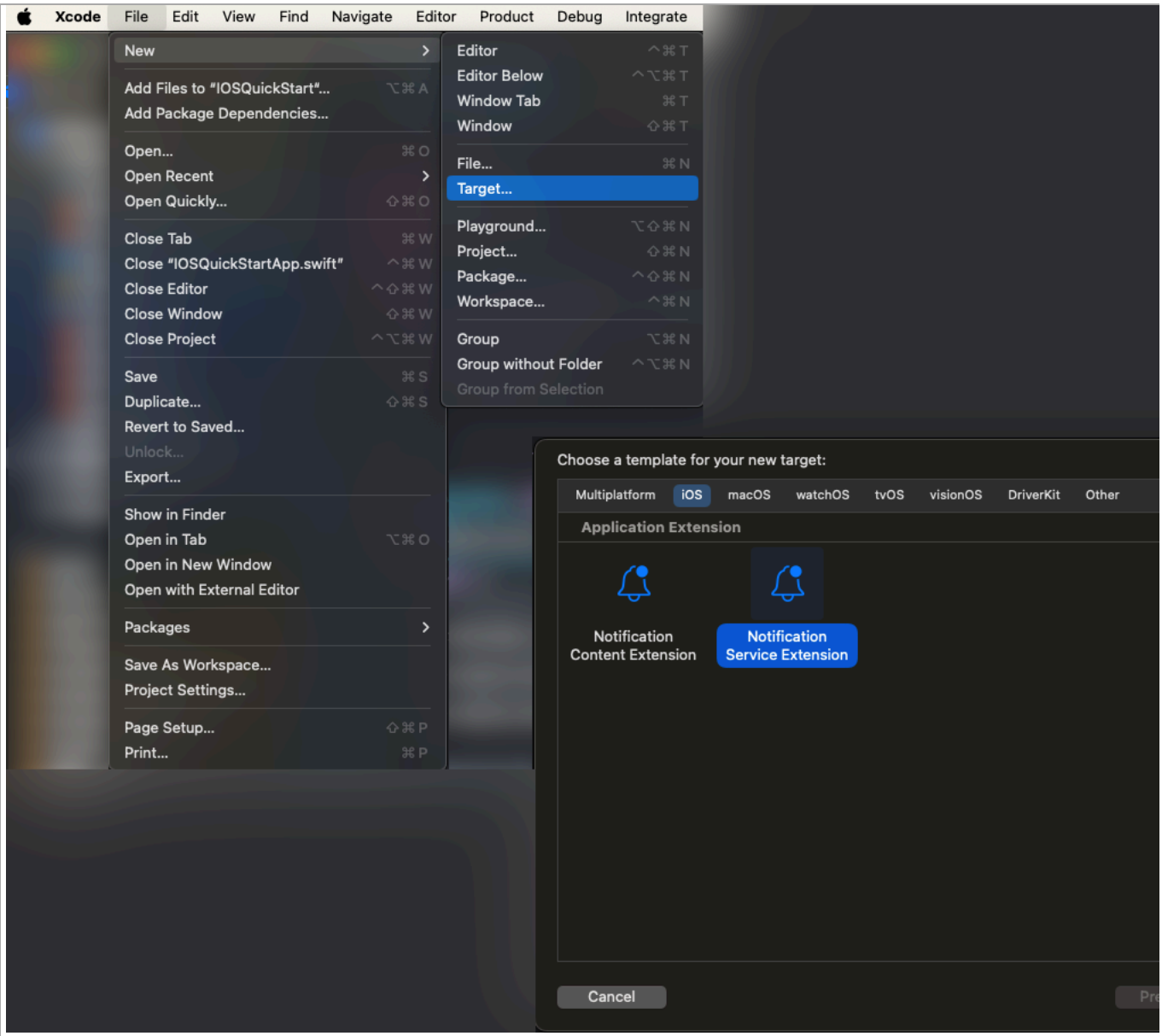
- **set the `delegate`;**
- **pass the APNS token to Firebase in `AppDelegate.application(_:didFinishLaunchingWithOptions:)`. This ensure and FCM token availability in background mode, for example during a `BGTask`. Since the APNS toker `AppDelegate.application(_:didFinishLaunchingWithOptions:)`, use the saved value (e.g., in `UserDefaults`).**



Step 8. Create a Notification Service Extension

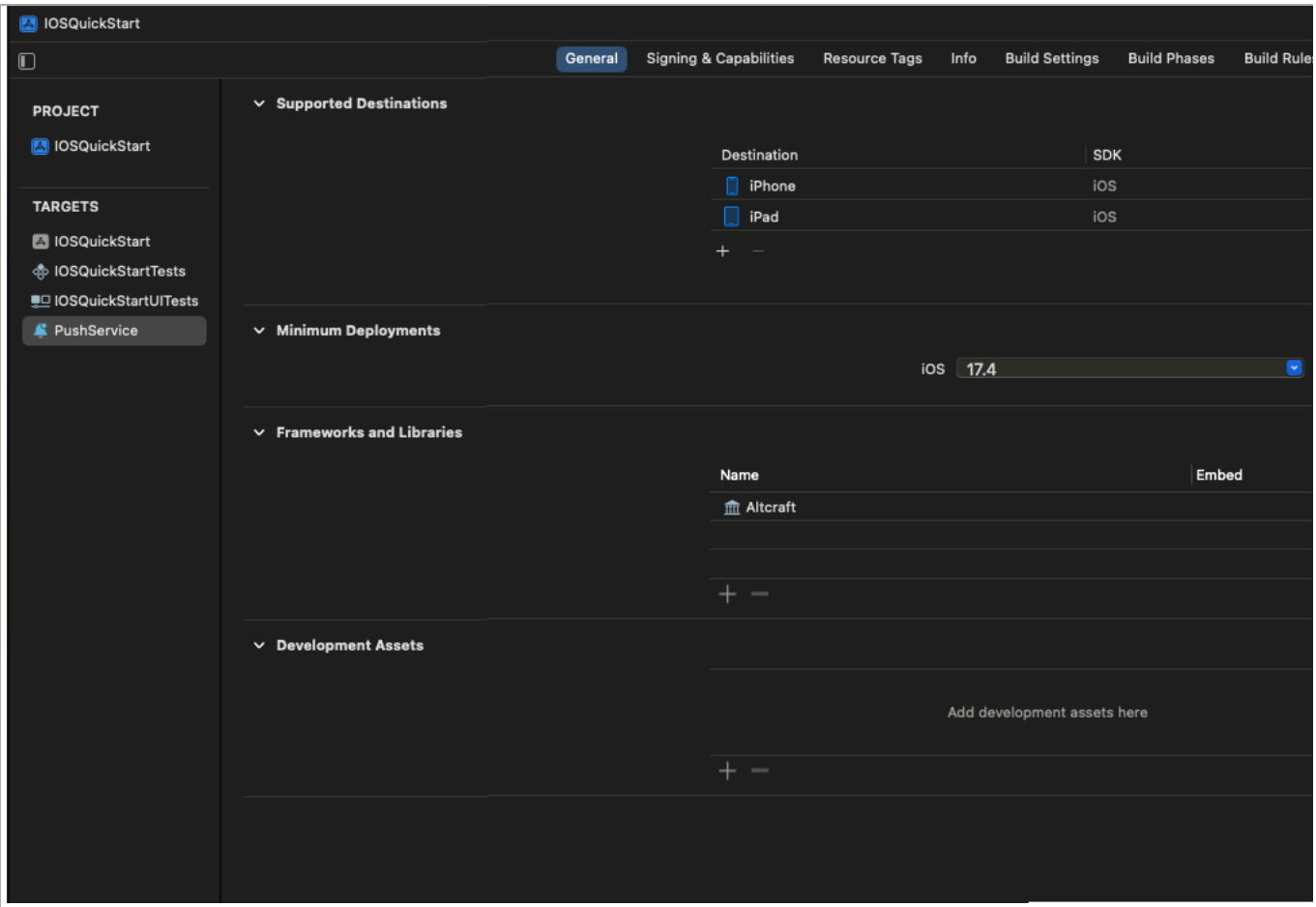
Create a Notification Service Extension:

- **File — New — Target — Notification Service Extension;**
- **Choose a Product Name for the extension target;**
- **Activate it.**



In the Notification Service Extension:

- **General:**
  - **Set Minimum Deployments — the Xcode build setting that defines the minimum OS version sup Extension;**
  - **Add the Altcraft library under Frameworks, Libraries and Embedded Content.**



- **Signing & Capabilities:**
  - **AppGroups — specify the `AppGroup` identifier.**

Then configure the SDK inside your `UNNotificationServiceExtension`. Replace the entire automatically gene below:

Example implementation of `UNNotificationServiceExtension`

```
class NotificationService: UNNotificationServiceExtension {

    /// - important! Set your App Group identifier.
    let appGroupID = "group.your.id"

    /// - important! Set the provider to JWT if you are using JWT authentication.
    let jwtProvider = JWTProvider()

    override func didReceive(
        _ request: UNNotificationRequest,
        withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void
    ) {

        AltcraftNSE.shared.setAppGroup(groupName: appGroupID)
        AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)

        if AltcraftNSE.shared.isAltcraftPush(request) {
            AltcraftNSE.shared.handleNotificationRequest(
                request: request,
                contentHandler: contentHandler
            )
        } else {
            contentHandler(request.content)
        }
    }

    override func serviceExtensionTimeWillExpire() {
        AltcraftNSE.shared.serviceExtensionTimeWillExpire()
    }
}
```

TIP: Note

If you are not using JWT authentication, remove the call to `AltcraftSDK.shared.setJWTProvider(provider: jwtP`

Step 9. SDK initialization

Create a configuration variable and initialize using `initialization()`, choosing `ios-firebase` as the preferr

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.setProviderPriorityList([Constants.ProviderName.firebase])
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

In this example, `ios-firebase` is set as the preferred provider. You can also set APNS as the preferred pr

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.setProviderPriorityList([Constants.ProviderName.apns])
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

You may omit `setProviderPriorityList`; the default priority will be used — `ios-apns -> ios-firebase -> ios-huaw`

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

Step 10. Subscribe to push notifications

Subscribe to push notifications using `pushSubscribe()` and pass the required parameters:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
 profileFields: ["_fname": "user_first_name", "_lname": "user_last_name"]
)
```

SDK Functionality

TIP

Before proceeding, you need to configure the SDK for your application. The detailed setup guide is av

Working with subscription statuses

Changing subscription status

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Push subscription managment
    └── pushSubscriptionFunctions: PublicPushSubscriptionFunctions
        // Subscribe (status = SUBSCRIBED)
        ├── pushSubscribe(
        │       sync: Bool = true,
        │       profileFields: [String: Any?]? = nil,
        │       customFields: [String: Any?]? = nil,
        │       cats: [CategoryData]? = nil,
        │       replace: Bool? = nil,
        │       skipTriggers: Bool? = nil
        │   ): Void
        // Unsubscribe (status = UNSUBSCRIBED)
        ├── pushUnSubscribe(
        │       sync: Bool = true,
        │       profileFields: [String: Any?]? = nil,
        │       customFields: [String: Any?]? = nil,
        │       cats: [CategoryData]? = nil,
        │       replace: Bool? = nil,
        │       skipTriggers: Bool? = nil
        │   ): Void
        // Suspend (status = SUSPENDED)
        └── pushSuspend(
                sync: Bool = true,
                profileFields: [String: Any?]? = nil,
                customFields: [String: Any?]? = nil,
                cats: [CategoryData]? = nil,
                replace: Bool? = nil,
                skipTriggers: Bool? = nil
            ): Void
```

Subscription status change functions:

- `func pushSubscribe()` **— subscribes to push notifications;**
- `func pushUnSubscribe()` **— unsubscribes from push notifications;**
- `func pushSuspend()` **— suspends the push subscription (no notifications are delivered, but no unsubsc profile);**
- `func unSuspendPushSubscription()` **— used to implement** `LogIn` **/** `LogOut` **transitions.**

All functions in this group have the same signature and parameters:

---

sync: `Bool`
Default: `true`
Required: No
Description: A flag that sets whether the request is executed synchronously.

Successful request execution:

On success, these functions produce an event with code 230, 231 or 232, containing an `event.value` **def**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "c52b28d2"
           ├ provider: "ios-apns"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: {"raw":"18.6.2","ver":"[\"18.0\", \"6.0\", \"2.0\"]"}
           │  ├ _device_type: "mob"
           │  ├ _ad_track: false
           │  ├ _device_name: "iPhone"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0300"
           │  ├ _os: "IOS"
           │  └ _device_model: "iPhone14,7"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, the `event.value["response_with_http_code"]` contains:

- **httpCode – transport-level status code;**
- **Response – a public struct containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData?` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (SubscriptionData)**
    - **subscription categories (CategoryData)**
    - **if the request fails,** `profile = nil`

▶ **Data structures**

If `sync == false`

```
ResponseWithHttpCode
  ├ httpCode: Int?
  └ response: Response?
     ├ error: Int?
     ├ errorText: String?
     └ profile: ProfileData? = nil
```

For asynchronous requests, `event.value["response_with_http_code"]` **contains:**

- **httpCode – transport-level status code;**
- **Response – a public struct containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData?` **— profile data, always** `nil` **for async requests.**

Failed request execution:

If a request in this group fails, an event with one of the following codes is created:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only** `httpCode` **if the Altcraft server was unavailable;**
- `error` **and** `errorText` **if the server returned an error.**

Retrieving event values

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event in
    let codes: Set<Int> = [
        230, 231, 232,
        430, 431, 432,
        531, 532, 533
    ]

    guard let code = event.eventCode, codes.contains(code) else { return }
    guard let responseWithHttp = event.value?["response_with_http_code"] as? ResponseWithHttp else { return }

    let httpCode = responseWithHttp.httpCode

    let response = responseWithHttp.response
    let error = response?.error
    let errorText = response?.errorText

    let profile = response?.profile
    let profileId = profile?.id
    let profileStatus = profile?.status
    let profileIsTest = profile?.isTest

    let subscription = profile?.subscription
    let subscriptionId = subscription?.subscriptionId
    let hashId = subscription?.hashId
    let provider = subscription?.provider
    let subscriptionStatus = subscription?.status

    let fields = subscription?.fields

    let cats = subscription?.cats
    let firstCat = cats?.first
    let catName = firstCat?.name
    let catTitle = firstCat?.title
    let catSteady = firstCat?.steady
    let catActive = firstCat?.active
}
```

The `fields` inside `subscription` may include your custom subscription fields. They are of type `[String: JS` provides helpers to simplify field value extraction:

```
public var stringValue: String? {
    if case let .string(value) = self { return value }
    return nil
}

/// Returns the numeric value if case is `.number`, otherwise `nil`.
public var numberValue: Double? {
    if case let .number(value) = self { return value }
    return nil
}

/// Returns the bool value if case is `.bool`, otherwise `nil`.
public var boolValue: Bool? {
    if case let .bool(value) = self { return value }
    return nil
}

/// Returns the object dictionary if case is `.object`, otherwise `nil`.
public var objectValue: [String: JSONValue]? {
    if case let .object(value) = self { return value }
    return nil
}

/// Returns the array if case is `.array`, otherwise `nil`.
public var arrayValue: [JSONValue]? {
    if case let .array(value) = self { return value }
    return nil
}
```

Example of extracting the field `"_device_name"`:

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event ->
    if event.eventCode == 230 {
        if let responseWithHttp = event.value?["response_with_http_code"] as? ResponseWithHttp {
            let response = responseWithHttp.response
            let profile = response?.profile
            let subscription = profile?.subscription

            // extract fields?["_device_name"] as a string value
            if let deviceName = subscription?.fields?["_device_name"]?.stringValue {
                print(deviceName)
            }
        }
    }
}
```

---

profileFields: `[String: Any?]?`

Default: `nil`
Required: No
Description: A dictionary containing profile fields.

The parameter may include system fields (e.g., `_fname` — first name, `_lname` — last name) and optional f platform UI). Allowed JSON-compatible value types:

- **String**
- **Bool**
- **Numbers**
- `nil`
- **Objects** `[String: Any]`
- **Arrays**

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `[String: Any?]?`

Default: `nil`
Required: No
Description: A dictionary containing subscription fields.

The parameter may include system fields (e.g., `_device_model` — device model, `_os` — OS), as well as opt platform UI). Allowed types:

- **String**
- **Bool**
- **Int**
- **Float**
- **Double**
- `nil`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

Nested objects, arrays, and collections are not allowed.

> TIP: Note
>
> **Most system subscription fields are automatically collected by the SDK and added to push requests. T** `"_os_language"`, `"_device_type"`, `"_device_model"`, `"_device_name"`, `"_os_ver"`, `"_ad_track"`, `"_ad_id"`.

---

cats: `[CategoryData]?`

Default: `nil`
Required: No
Description: Subscription categories.
Category structure:

```
public struct CategoryData: Codable {
    public var name: String?
    public var title: String?
    public var steady: Bool?
    public var active: Bool?
}
```

When sending a push request with categories, use only `name` (category name) and `active` (category act used in request processing. The `title` and `steady` fields are populated when retrieving subscription inf

Request example:

```
let cats: [CategoryData] = [CategoryData(name: "football", title: nil, steady: nil, active: true)]
```

Response example:

```
[ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft p in the resource is used, the request returns an error:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `Bool?`

Default: `nil`

Required: No

Description: When enabled, all subscriptions of other profiles with the same push token in the current after a successful request.

---

skipTriggers: `Bool?`

Default: `nil`

Required: No

Description: When enabled, the profile containing this subscription will be ignored by mailing and scen

---

Request examples

Example: subscribing to push notifications

Minimal working call:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe()
```

Passing all available parameters:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    sync: true,
    profileFields: ["_fname":"Andrey", "_lname":"Pogodin"],
    customFields:  ["developer":true],
    cats: [CategoryData(name: "developer_news", active: true)],
    replace: false,
    skipTriggers:  false
)
```

> TIP: Note
>
> **For `pushSubscribe`, `pushSuspend`, and `pushUnSubscribe`, the SDK automatically retries the request if the HTT range. No retry occurs if the code is outside this range.**

Function `unSuspendPushSubscription()`

`func unSuspendPushSubscription()` is designed for `LogIn` / `LogOut` transitions. It works as follows:

- **searches for subscriptions with the same push token as the current one that do not belong to the p JWT;**
- **changes the status of the found subscriptions from `subscribed` to `suspended`;**
- **changes the status of the subscriptions in the profile referenced by the current JWT from `suspended` and contains subscriptions);**
- **returns a `ResponseWithHttpCode?` where `response.profile` is the current profile referenced by the JWT (if `response.profile` is `nil`).**

Example:

```
AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

Recommended `LogIn` / `LogOut` flow

`LogIn` flow:

- **An anonymous user opens the app. The user has `JWT_1`, pointing to database #1Anonymous;**
- **The user subscribes to push; a profile is created in #1Anonymous;**
- **The user registers and receives `JWT_2`, pointing to #2Registered;**

- **Call `unSuspendPushSubscription()` — the anonymous subscription in #1Anonymous is suspended;**
- **The SDK searches for a profile in #2Registered to restore the subscription;**
- **Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()`**
- **After receiving `nil`, call `pushSubscribe()` to create a new profile in #2Registered.**

`LogOut` flow:

- **The user logs out in the app (`LogOut`);**
- **The user receives `JWT_1`, pointing to #1Anonymous;**
- **Call `unSuspendPushSubscription()`, which suspends the subscription in #2Registered and sets the statu**
- **The request returns `#1Anonymous != nil` — the subscription already exists, no new one is required.**

Example implementation:

```
func logIn() {
    JWTManager.shared.setRegJWT()
    // JWT is set for an authorized user
    AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription { result in
        if result?.httpCode == 200, result?.response?.profile?.subscription == nil {
            AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
                // specify the required parameters
            )
        }
    }
}

func logOut() {
    JWTManager.shared.setAnonJWT()
    // JWT is set for an anonymous user
    AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription { result in
        if result?.httpCode == 200, result?.response?.profile?.subscription == nil {
            AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
                // specify the required parameters
            )
        }
    }
}
```

Requesting subscription status

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Subscription managment
    └── pushSubscriptionFunctions: PublicPushSubscriptionFunctions
        // Last profile subscription status
        ├── getStatusOfLatestSubscription(
        │       completion: @escaping (ResponseWithHttp?) -> Void
        │   ): Void
        // Subscription status corresponding to the current token/provider
        ├── getStatusForCurrentSubscription(
        │       completion: @escaping (ResponseWithHttp?) -> Void
        │   ): Void
        // Status of the latest subscription for the provider (if nil – the current one is used)
        └── getStatusOfLatestSubscriptionForProvider(
                provider: String? = nil,
                completion: @escaping (ResponseWithHttp?) -> Void
            ): Void
```

Subscription status functions:

- **`func getStatusOfLatestSubscription()` — the last subscription status in the profile;**
- **`func getStatusOfLatestSubscriptionForProvider()` — the subscription status for the current token/provider;**
- **`func getStatusForCurrentSubscription()` — the latest subscription status by provider. If `nil` is specified,**

`public func getStatusOfLatestSubscription(completion: @escaping (ResponseWithHttp?) -> Void): Void`

Retrieves the status of the profile's latest subscription. The `completion` receives a `ResponseWithHttp?` cont — the most recently created subscription in the profile. If no such subscription exists, `nil` is passed.

Example:

```
// Last profile subscription status
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusOfLatestSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

`public func getStatusForCurrentSubscription(completion: @escaping (ResponseWithHttp?) -> Void): Void`

Retrieves the subscription status for the current token/provider. The `completion` receives a `ResponseWithHt` `response?.profile?.subscription` — the subscription found by the current push token and provider. If no suc passed.

Example:

```
// Subscription status corresponding to the current token/provider
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusForCurrentSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

`public func getStatusOfLatestSubscriptionForProvider(provider: String? = nil, completion: @escaping (ResponseWithHttp?`

Retrieves the latest subscription status by provider. The `completion` receives a `ResponseWithHttp?` containi the most recently created subscription for the specified push provider. If no provider is specified, the c no such subscription exists, `nil` is passed.

Example:

```
// Status of the latest subscription for the provider (if nil — the current one is used)
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusOfLatestSubscriptionForProvider(provider: String? = nil, completion: @escaping (Respo
```

Below is an example of extracting profile, subscription, and category data from the status response. T retrieval functions:

Data from status-retrieval functions

```
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusForCurrentSubscription{ responseWithHttp in
    // HTTP code
    let httpCode = responseWithHttp?.httpCode

    // Response
    let response = responseWithHttp?.response
    let error = response?.error
    let errorText = response?.errorText

    // Profile
    let profile = response?.profile
    let profileId = profile?.id
    let profileStatus = profile?.status
    let profileIsTest = profile?.isTest

    // Subscription
    let subscription = profile?.subscription
    let subscriptionId = subscription?.subscriptionId
    let hashId = subscription?.hashId
    let provider = subscription?.provider
    let subscriptionStatus = subscription?.status

    // Fields (dictionary [String: JSONValue])
    let fields = subscription?.fields

    // Cats (array of CategoryData)
    let cats = subscription?.cats

    // CategoryData (every element of cats array)
    let firstCat = cats?.first
    let catName = firstCat?.name
    let catTitle = firstCat?.title
    let catSteady = firstCat?.steady
    let catActive = firstCat?.active
}
```

On a successful request, an event with code 234 is created. On error — an event with code 434.

Provider push token management

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Managing push tokens and providers (FCM / HMS / APNs)
    └── pushTokenFunctions: PublicPushTokenFunctions
        // Manually save the token (String for FCM/HMS, Data for APNs)
        ├── setPushToken(
        │       provider: String,
        │       pushToken: Any?
        │   ): Void
        // Asynchronously retrieve the current push token (completion is optional)
        ├── getPushToken(
        │       completion: ((TokenData?) -> Void)? = nil
        │   ): Void
        // Set the Firebase Cloud Messaging provider (nil — remove)
        ├── setFCMTokenProvider(
        │       _ provider: FCMInterface?
        │   ): Void
        // Set the Huawei Mobile Services provider (nil — remove)
        ├── setHMSTokenProvider(
        │       _ provider: HMSInterface?
        │   ): Void
        // Set the Apple Push Notification service provider (nil — remove)
        ├── setAPNSTokenProvider(
        │       _ provider: APNSInterface?
        │   ): Void
        // Apply a new provider priority list and initiate token update
        ├── changePushProviderPriorityList(
        │       _ list: [String]
        │   ): Void
        // Delete the push token of the specified provider and call completion
        ├── deleteDeviceToken(
        │       provider: String,
        │       completion: @escaping () -> Void
        │   ): Void
        // Force token update
        └── forcedTokenUpdate(
                completion: (() -> Void)? = nil
            ): Void
```

SDK functions for working with a provider's token:

- `func setPushToken()` — **manually sets the device push token and provider in** `UserDefaults`**;**
- `func getPushToken()` — **retrieves the current push token;**
- `func setAPNSTokenProvider()` — **sets or removes the APNs push-token provider;**
- `func setFCMTokenProvider()` — **sets or removes the FCM push-token provider;**
- `func setHMSTokenProvider()` — **sets or removes the HMS push-token provider;**
- `func changePushProviderPriorityList()` — **dynamically switches the provider and updates the subscriptio**
- `func deleteDeviceToken()` — **deletes the push token for the specified provider;**
- `func forcedTokenUpdate()` — **deletes the current push token and then refreshes it.**

`public func setPushToken(provider: String, pushToken: Any?): Void`

Used to manually set the device push token and provider in `UserDefaults`**. This serves as a simplified wa without implementing provider protocols.**

Not recommended for passing the token. The preferred approach is to implement `FCMInterface`**,** `HMSInterf`

Example:

```
// Manually save the token (String for FCM/HMS, Data for APNs)
AltcraftSDK.shared.pushTokenFunctions.setPushToken(provider: String, pushToken: Any?)
```

`public func getPushToken(completion: ((TokenData?) -> Void)? = nil): Void`

Passes a `TokenData(provider: String, token: String)` **object to** `completion`**, containing the current device pus token is unavailable,** `nil` **is passed.**

Example:

```
// Asynchronously retrieve the current push token (completion is optional)
AltcraftSDK.shared.pushTokenFunctions.getPushToken(completion: ((TokenData?) -> Void)? = nil)
```

Example of requesting push-token data:

```
AltcraftSDK.shared.pushTokenFunctions.getPushToken { data in
    let provider = data?.provider
    let token = data?.token
}
```

`public func setFCMTokenProvider(_ provider: FCMInterface?): Void`

Sets or removes the Firebase Cloud Messaging provider. Pass `nil` **to disable the provider.**

Example:

```
// Set the Firebase Cloud Messaging provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(_ provider: FCMInterface?)
```

> **TIP: Important**
>
> **Call `setFCMTokenProvider()` in `AppDelegate.application(_:didFinishLaunchingWithOptions:)` before `AltcraftSDK.initial` registration at process start, regardless of lifecycle state or whether the app launches in foreground**

`public func setHMSTokenProvider(_ provider: HMSInterface?): Void`

Sets or removes the Huawei Mobile Services provider. Pass `nil` to disable the provider.

```
// Set the Huawei Mobile Services provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(_ provider: HMSInterface?)
```

> **TIP: Important**
>
> **Call `setHMSTokenProvider()` in `AppDelegate.application(_:didFinishLaunchingWithOptions:)` before `AltcraftSDK.initial` registration at process start, regardless of lifecycle state or whether the app launches in foreground**

`public func setAPNSTokenProvider(_ provider: APNSInterface?): Void`

Sets or removes the Apple Push Notification service provider. Pass `nil` to disable the provider.

```
// Set the Apple Push Notification service provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(_ provider: APNSInterface?)
```

> **TIP: Important**
>
> **Call `setAPNSTokenProvider()` in `AppDelegate.application(_:didFinishLaunchingWithOptions:)` before `AltcraftSDK.initia` registration at process start, regardless of lifecycle state or whether the app launches in foreground**

`public func changePushProviderPriorityList(_ list: [String]): Void`

Enables dynamic switching of the push provider with token refresh. Pass a new array with a different p `[Constants.ProviderName.firebase, Constants.ProviderName.apns, Constants.ProviderName.huawei]`.

Example:

```
// Apply a new provider priority list and initiate token update
AltcraftSDK.shared.pushTokenFunctions.changePushProviderPriorityList(_ list: [String])
```

`public func deleteDeviceToken(provider: String, completion: @escaping () -> Void): Void`

Deletes the push token for the specified provider. The token is invalidated and removed from the local push provider's server. After deletion, a new token can be requested.

Example:

```
// Delete the push token of the specified provider (invalidated locally and on the server)
AltcraftSDK.shared.pushTokenFunctions.deleteDeviceToken(provider: String, completion: @escaping () -> Void)
```

`public func forcedTokenUpdate(completion: (() -> Void)? = nil): Void`

Deletes the current push token and then refreshes it.

Example:

```
// Force token update (determine provider –> delete (except APNs) –> request new –> update)
AltcraftSDK.shared.pushTokenFunctions.forcedTokenUpdate(completion: (() -> Void)? = nil)
```

Example of registering providers

We do not recommend using `setPushToken` to set the push token. Instead, configure token retrieval for ea

Recommended provider registration in `AppDelegate`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        let appGroup = "your appGroup id"

        AltcraftSDK.shared.setAppGroup(groupName: appGroup)
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())

        // apns provider
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())

        // fcm provider
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        // hms provider
        AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(HMSProvider())

        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        let config = AltcraftConfiguration.Builder().setApiUrl("your api url").build()
        AltcraftSDK.shared.initialization(configuration: config)

        // other functions

        return true
    }
}
```

Managing Push Notifications

Push Notification Initialization

To initialize push notifications, call the following method in `application(_:didFinishLaunchingWithOptions:)`:

`AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)`

This method assigns `UNUserNotificationCenter.current().delegate` to the Altcraft SDK `NotificationManager`, requ notifications (alert/sound/badge), and registers the application with APNS via `registerForRemoteNotificatio`

> **TIP: Important**
>
> **The system permission dialog is shown only on the first request (or if the status has not yet been dete responded earlier, the dialog will not appear again.**

Passing Push Notifications to the SDK

The SDK includes the `AltcraftNSE` class, intended for use inside a Notification Service Extension. It prov incoming push notifications (`UNNotificationRequest`) and registers mobile events in the NSE.

```
// Used inside Notification Service Extension (NSE) — facade for handling Altcraft push notifications,
// detecting Altcraft-specific requests, and registering mobile events without direct dependency on UIApplication
AltcraftNSE
// Shared singleton instance for centralized NSE logic
├─ public static let shared: AltcraftNSE
// Provides access to mobile event registration API
├─ public let mobileEventFunctions: PublicMobileEventFunctions
// Check whether the notification request belongs to Altcraft
├─ public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool
// Handle an incoming Altcraft notification request and deliver the modified content
├─ public func handleNotificationRequest(
│     request: UNNotificationRequest,
│     contentHandler: @escaping (UNNotificationContent) -> Void
│  ): Void
// Called when NSE is about to time out — returns the best available notification content
├─ public func serviceExtensionTimeWillExpire(): Void
// Set the App Group identifier and initialize Core Data in the shared container
├─ public func setAppGroup(groupName: String): Void
// Register a JWT provider for secure token management in Altcraft SDK
└─ public func setJWTProvider(provider: JWTInterface): Void
```

`AltcraftNSE` class methods:

- **`func isAltcraftPush()` — checks the notification source;**
- **`func handleNotificationRequest()` — accepts a `UNNotificationRequest` from the Notification Service Extens SDK;**
- **`func serviceExtensionTimeWillExpire()` — called when the Notification Service Extension execution time**

`public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool`

An SDK method that checks whether the notification source is Altcraft based on a marker in `userInfo`.

Usage example:

```
AltcraftNSE.shared.isAltcraftPush(_ request: UNNotificationRequest) -> Bool
```

`public func handleNotificationRequest(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (`

An SDK method that accepts a `UNNotificationRequest` from the Notification Service Extension, processes notification.

Usage example:

```
AltcraftNSE.shared.handleNotificationRequest(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificatior
```

`public func serviceExtensionTimeWillExpire()`

A method called by the system when the Notification Service Extension execution time expires (~30 se

Usage example:

```
AltcraftNSE.shared.serviceExtensionTimeWillExpire()
```

`public func setAppGroup(groupName: String): Void`

A method for setting the `AppGroup` identifier (required).

Usage example:

```
AltcraftNSE.shared.setAppGroup(groupName: "group.your.app.id")
```

`public func setJWTProvider(provider: JWTInterface): Void`

A method for setting a JWT provider:

```
AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)
```

Receiving Foreground Notifications in the App (Optional)

By default, this functionality does not require configuration. If push notification handling is delegated t `registerForPushNotifications`), `NotificationManager` automatically:

- **is set as `UNUserNotificationCenterDelegate`,**
- **processes incoming notifications,**
- **displays notifications in the foreground using standard parameters (banner/alert + badge/sound).**

The setup described below is required only if:

- **access to notification content in the foreground is needed (for example, analytics or a custom UI),**
- **or you want to manually control notification presentation (show/hide, change presentation options)**

To configure foreground push notification behavior, use `NotificationManager`:

```
// Access to the SDK push notification manager
let manager = AltcraftSDK.shared.notificationManager

// Control of foreground push notification handling.
// true  — handling and presentation choice are delegated to the app.
// false — the SDK uses the standard notification presentation logic.
manager.customPushProcessing = false

// Callback for foreground push notification handling.
// Called when a notification is received while the app is active.
// completion must be called only when customPushProcessing = true.
manager.onForegroundNotification = { notification, complete in
    // Custom notification handling logic

    // Important: call complete(...) only if manager.customPushProcessing = true
    if #available(iOS 14.0, *) {
        complete([.banner, .badge, .sound])
    } else {
        complete([.alert, .badge, .sound])
    }

    // or [] to suppress presentation
}
```

If `customPushProcessing = false`, the SDK applies standard notification presentation parameters, and the ca additional app logic. If `customPushProcessing = true`, the app must call `complete(...)`; otherwise, the notifica foreground.

Handling Notification Click Events in the App (Optional)

By default, this functionality does not require additional configuration. If push notification handling is d `NotificationManager` automatically:

- **handles notification taps or actions,**

- **sends a push event of type `open` to Altcraft,**
- **executes the standard notification action handling logic.**

The setup described below is required only if:

- **the app needs access to notification data on click,**
- **custom navigation or action handling logic is required,**
- **or notification click handling must be fully intercepted by the app.**

To delegate click handling to the app, use `NotificationManager`:

```
let manager = AltcraftSDK.shared.notificationManager

// Control of notification click handling.
// true  — click handling is fully delegated to the app.
// false — the SDK uses the standard click handling logic.
manager.customClickProcessing = true

// Notification click handling callback.
// Called when the user taps a notification or performs an action.
// completion must be called when customClickProcessing = true.
manager.onNotificationClick = { response, complete in
    // Notification data
    let userInfo = response.notification.request.content.userInfo

    // Custom click handling logic
    print("push click")

    // Complete handling
    complete()
}
```

If `customClickProcessing = false`, the SDK handles notification clicks on its own, sends the corresponding e
In this case, the `onNotificationClick` callback (if set) is invoked only for additional app logic.

If `customClickProcessing = true`, the app must call `complete()`. Without calling `complete()`, iOS will treat the e
may lead to incorrect app behavior.

Mobile events

```
// PublicMobileEventFunctions object

AltcraftSDK.shared
└── let mobileEventFunctions: PublicMobileEventFunctions

    // Send a mobile event to the server
    └── func mobileEvent(
            sid: String,
            eventName: String,
            sendMessageId: String? = nil,
            payload: [String: Any?]? = nil,
            matching: [String: Any?]? = nil,
            matchingType: String? = nil,
            profileFields: [String: Any?]? = nil,
            subscription: (any Subscription)? = nil
            utm: UTM? = nil
        )
```

Case: Transferring advertising campaign information

Information about the app advertising campaign that led to the installation can be sent to the platform
`customFields` parameters of the `pushSubscribe` function, as well as the `utm` or `payload` parameters of the `m`
receiving a UTM tag in the app as a string, you need to call the `mobileEvent()` function:

Transfer via payload

```
AltcraftSDK.shared.mobileEventFunctions.mobileEvent(
    sid: "your sid",
    eventName: "app_install",
    payload: ["utm": "your utm tag"]
)
```

696a08e91275353a39ee3991

пт, 16 янв. 2026 г. 12:46:17

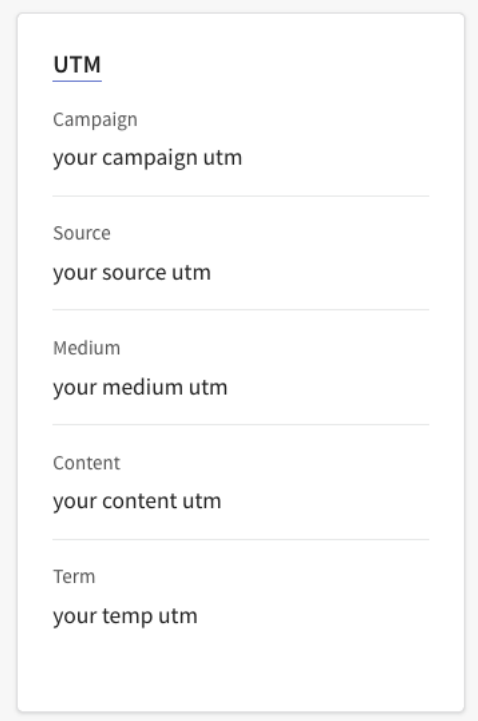
Открыт Pixel

Достигнута цель

- **Цели:** android_app_install
- **Пиксель:** #1 pixel
- **UTM Content:** your content utm
- **UTM Source:** your source utm
- **UTM Campaign:** your campaign utm
- **UTM Term:** your temp utm

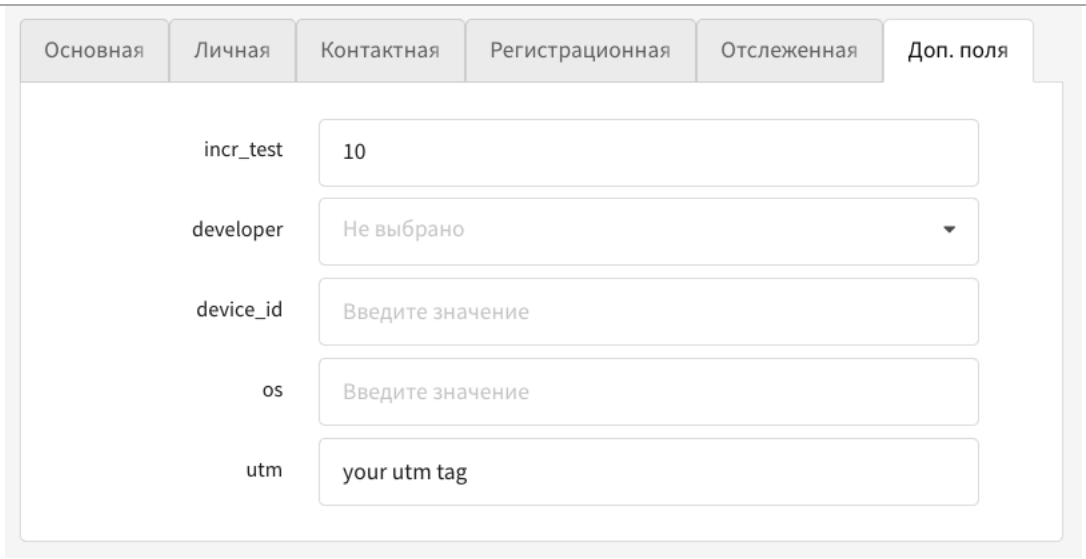
Transfer to UTM fields

```
AltcraftSDK.shared.mobileEventFunctions.mobileEvent(
    sid: "your sid",
    eventName: "app_install",
    utm: UTM(
        campaign: "your campaign utm",
        content: "your content utm",
        keyword: "your keyword utm",
        medium: "your medium utm",
        source: "your source utm",
        temp: "your temp utm"
    )
)
```



Transfer to a custom profile field

```
// transfer a UTM tag to a profile field
// profile fields must be created in the platform in advance
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: ["utm": "your utm tag"]
)
```



Data sent using any of these methods can be used to create a segment in the platform.

Please note that when setting up on iOS, the application must use AppsFlyer, otherwise the UTM tag w application.

Use `mobileEvent()` from `PublicMobileEventFunctions` to register a mobile event. Parameters:

---

sid: `String`

Required: Yes

Description: String pixel identifier to which mobile events are linked.

---

eventName: `String`

Required: Yes

Description: Mobile event name.

---

sendMessageId: `String?`

Required: No

Description: SMID of the sent message (when the event is tied to a specific mailing).

---

payload: `[String: Any?]?`

Required: No

Description: Event data — a map with string keys; only scalar value types are allowed:

- `String`
- `Bool`
- `Int`
- `Int64` / `UInt64` **(or** `NSNumber` **equivalents)**
- `Float`
- `Double`
- `nil` / `NSNull`

TIP: Note

Non-serializable objects (e.g., `Date` **without conversion, custom classes) will cause encoding errors.**

---

matching: `[String: Any?]?`

Required: No Description: A map for passing values with matching types/identifiers (e.g., `["email" : "use`

---

matchingType: `String?`

Required: No

Description: Matching type.

---

profileFields: `[String: Any?]?`

Required: No
Description: Profile fields — a map with string keys and values (JSON-compatible types):

- **Scalar values:**
  - `String`
  - `Bool`
  - `Int`
  - `Int64` / `UInt64` **(or** `NSNumber` **equivalents)**
  - `Float`
  - `Double`
  - `nil` / `NSNull`
- **Objects:** `[String: Any?]`
- **Lists:** `[Any?]`
- **Array of maps:** `[[String: Any?]]`

> **TIP: Note**
> `profileFields` **is used only with JWT authorization.**

---

utm: `UTM?`
Required: No
Description: UTM tags. Added via the `UTM` **struct, where each UTM type is a property:**

```
public struct UTM{
    /// UTM Mailing
    public var campaign: String?
    /// UTM Content
    public var content: String?
    /// UTM Keyword/Term
    public var keyword: String?
    /// UTM Medium
    public var medium: String?
    /// UTM Source
    public var source: String?
    /// UTM Temp
}
```

---

subscription: `(any Subscription)?`
Required: No Description: A subscription to be added to the profile, one of: `EmailSubscription`, `SmsSubscript` `CcDataSubscription`.

> **TIP: Note**
> **Used only with JWT authorization.**

`Subscription` **interface implementations**
Purpose — a common protocol for all subscription types. Serialization — via `Codable`**, discriminator field**
Common fields (for all implementations):

Field	Type	Required	Descripti
`resource_id`	`Int`	**Yes**	**Identifier of the resource/source of the subscription**
`status`	`String?`	**No**	**Subscription status (e.g., active/suspended)**
`priority`	`Int?`	**No**	**Delivery priority for this subscription**
`custom_fields`	`[String: JSONValue]?`	**No**	**Custom key–value fields for advanced segmentation**
`cats`	`[String]?`	**No**	**Subscription categories**
`channel`	`String`	**Yes**	**Channel type; fixed by the specific implementation**

> **TIP: Note**
> `JSONValue` **is a serializable enum/type of JSON values in your SDK. All values must be JSON-compatible**

Subscription types:
EmailSubscription (`channel = "email"` **)**

Required fields

Field	Type	Required	
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`email`	`String`	**Yes**	**Recipient email address**

Optional fields

Field	Type	Required	De
`status`	`String?`	**No**	**Subscription status**
`priority`	`Int?`	**No**	**Subscription priority**
`customFields`	`[String: JSONValue]?`	**No**	**Standard and custom subscription fields**
`cats`	`[String]?`	**No**	**Subscription categories**

SmsSubscription (`channel = "sms"`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`phone`	`String`	**Yes**	**Phone number in international format**

Optional fields

Field	Type	Required	De
`status`	`String?`	**No**	**Subscription status**
`priority`	`Int?`	**No**	**Subscription priority**
`customFields`	`[String: JSONValue]?`	**No**	**Standard and custom subscription fields**
`cats`	`[String]?`	**No**	**Subscription categories**

PushSubscription (`channel = "push"`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`provider`	`String`	**Yes**	**Provider (for example, `"ios-apns"`)**
`subscriptionId`	`String`	**Yes**	**Unique subscription identifier on the provider side**

Optional fields

Field	Type	Required	De
`status`	`String?`	**No**	**Subscription status**
`priority`	`Int?`	**No**	**Subscription priority**
`customFields`	`[String: JSONValue]?`	**No**	**Standard and custom subscription fields**
`cats`	`[String]?`	**No**	**Subscription categories**

CcDataSubscription (`channel ∈ {"telegram_bot","whatsapp","viber","notify"}`)

Required fields

Field	Type	Required	Description
`resourceId`	`Int`	**Yes**	**Altcraft resource ID**
`channel`	`String`	**Yes**	**One of: `"telegram_bot"` , `"whatsapp"` , `"viber"` , `"notify"`**
`ccData`	`[String: JSONValue]`	**Yes**	**Channel-specific data (for example, chat ID, phone number, tokens)**

Optional fields

Field	Type	Required	De
`status`	`String?`	**No**	**Subscription status**
`priority`	`Int?`	**No**	**Subscription priority**
`customFields`	`[String: JSONValue]?`	**No**	**Standard and custom subscription fields**
`cats`	`[String]?`	**No**	**Subscription categories**

Profile Fields Update

> **INFO: Important**
>
> **Updating profile fields using the `updateProfileFields()` function works only when using JWT authorizatio**

```
AltcraftSDK.shared
└── let profileFunctions: PublicProfileFunctions
    // Updates Altcraft profile fields
    └── func updateProfileFields(
        profileFields: [String: Any?]? = nil,
        skipTriggers: Bool? = nil
    ): Void
```

To update profile fields, use the `updateProfileFields()` function. It has the following parameters:

---

profileFields: `[String: Any?]?`

Required: No
Description: Profile fields — a map containing the values for the fields that need to be updated (JSON-c

- **Scalar values:**
  - `String`
  - `Bool`
  - `Int`
  - `Int64` / `UInt64` **(or equivalent** `NSNumber`**)**
  - `Float`
  - `Double`
  - `nil` / `NSNull`
- **Objects:** `[String: Any?]`
- **Lists:** `[Any?]`
- **Array of maps:** `[[String: Any?]]`

INFO: Note

For the update to succeed, the fields must have been previously added to the subscriber's profile.

---

skipTriggers: `Bool?`
Default: `nil`
Required: No
Description: When enabled, the profile containing this subscription will be ignored in mailing triggers a

Receiving SDK events in the app

```
// Events object functions

AltcraftSDK
// Singleton, SDK entry point
└── static let shared: AltcraftSDK
    // SDK event stream (one active subscriber)
    └── eventSDKFunctions: SDKEvents
        // Subscribe to SDK events (replaces an existing subscriber)
        ├── subscribe(
        │       callback: @escaping (Event) -> Void
        │   ): Void
        // Unsubscribe from events (callback remains assigned, delivery stops)
        └── unsubscribe(): Void
```

Only one active subscriber to SDK events can exist in the app.

Subscribing to events

`public func subscribe(callback: @escaping (Event) -> Void): Void`

Subscribes to and receives SDK events in the app. When an SDK event occurs, it invokes `callback` **and p subclass).**

Example:

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event in
  // handle event
}
```

All events provided by the SDK are instances of `Event` **or its subclasses:**

- **Event — general event (information, successful requests);**
- **Error — error event;**
- **RetryError — error event for a request that the SDK will automatically retry.**

Each event contains:

- **function — the name of the function that produced the event;**
- **eventCode — internal SDK event code (see "SDK events");**
- **eventMessage — event message;**
- **eventValue — arbitrary payload** `[String: Any?]?` **added to some events;**
- **date — event timestamp.**

SDK events classes

```
open class Event: Hashable {
    public let id = UUID()
    public let function: String
    public let message: String?
    public let eventCode: Int?
    public let value: [String: Any?]?
    public let date: Date
}

open class ErrorEvent: Event {
    public override init(
        function: String,
        message: String? = nil,
        eventCode: Int? = nil,
        value: [String: Any?]? = nil,
        date: Date = Date()
    ) {
        super.init(
            function: function,
            message: message,
            eventCode: eventCode,
            value: value,
            date: date
        )
    }
}

public class RetryEvent: ErrorEvent {
    public override init(
        function: String,
        message: String? = nil,
        eventCode: Int? = 0,
        value: [String: Any?]? = nil,
        date: Date = Date()
    ) {
        super.init(
            function: function,
            message: message,
            eventCode: eventCode,
            value: value,
            date: date
        )
    }
}
```

Example contents of a successful push subscription event

```
├ function: processResponse
├ eventCode: 230
├ message: "successful request: push/subscribe"
├ value
│  ├ http code: 200
│  └ response
│     ├ error: 0
│     ├ errorText: ""
│     └ profile
│        ├ id: // profile ID
│        ├ status: subscribed
│        ├ isTest: false
│        └ subscription
│           ├ subscriptionId: // push token of the subscription
│           ├ hashId: // hash ID
│           ├ provider: "ios-apns"
│           ├ status: subscribed
│           ├ fields
│           │  ├ _ad_id: // google service ad id
│           │  ├ _ad_track: false
│           │  ├ _app_id: "AltcraftMobile"
│           │  ├ _app_iid: "1.0.0"
│           │  ├ _app_ver: {"raw":"1.0.0","ver":[1,0]}
│           │  ├ _device_model: "iPhone14,7"
│           │  ├ _device_name: "iPhone"
│           │  ├ _device_type: "mob"
│           │  ├ _os: "Android"
│           │  ├ _os_language: "ru"
│           │  ├ _os_tz: "+0300"
│           │  └ _os_ver: {"raw":"18.6.2","ver":"[\"18.0\", \"6.0\", \"2.0\"]"}
│           └ cats
│              └ [ { name: "cats_1", title: "cats_1", steady: false, active: false } ]
└ date: Tue Aug 12 15:49:20 GMT+03:00 2025
```

Unsubscribing from events

`public func unsubscribe(): Void`

Unsubscribes from SDK events. It stops event delivery while keeping the `callback` assigned.

Example:

```
// unsubscribe from SDK events
AltcraftSDK.shared.eventSDKFunctions.unsubscribe()
```

List of all SDK events

Event list

Code	Description
200	SDK configuration is installed
201	push provider set
203	received Altcraft push notification
204	push is posted
205	SDK data has been cleared
206	SDK background task is registered
207	SDK background task completed
208	SDK background task is registered
209	SDK background task completed
230	subscribe request succeeded
231	token update request succeeded
232	unsuspend request succeeded
233	status request succeeded
234	push event delivered successfully
235	mobile event delivered successfully
401	the configuration is not set
402	userTag is null. It is impossible to identify the user
403	error in CoreData. Re-initialize the SDK
404	failed to load model from framework
405	App Group name was not set
406	forcing a push token update is not possible: the operation is not supported for APNs
407	the CoreData entity name is not valid
408	entity not found by ID
409	SDK initialization timeout has expire
422	unsuspend request data is nil
423	profile request data is nil
430	subscribe request failed
431	token update request failed
432	unsuspend request failed
433	status request failed
434	push event delivery failed
435	mobile event delivery failed
450	invalid userInfo format
451	uid in the push data is null; it is impossible to send a push event to the server
452	could not download image data from URL
453	‘media’ key is missing or URL string is invalid
454	‘buttons’ key is missing or is invalid
455	invalid button identifier (see event value)
456	out of range for identifier (see event value)
457	unknown button identifier (see event value)
471	invalid apiUrl value — empty or null
472	invalid resource token value — resource token is empty
473	invalid provider. Available — ios-apns, ios-firebase, ios-huawei
474	non-JSON object has been provided
475	invalid values: not all values are primitives
476	unsupported subscription type. Available: EmailSubscription, SmsSubscription, PushSubscription, CcDataSubscription
501	config data is nil
502	current token is nil
503	userTag is null. It is impossible to identify the user
504	notification permission denied
505	SDK background task expired
520	subscribe request data is nil
521	update request data is nil
524	push event request data is nil
525	mobile event request data is nil

Code	Description
529	common data is nil
530	subscribe request failed (retryable)
531	token update request failed (retryable)
534	push event delivery failed (retryable)
535	mobile event delivery failed (retryable)
540	JWT token is nil
541	JWT parsing error
542	invalid matching
543	auth data is nil
561	failed to create request
562	failed processing the response
571	invalid request URL
572	invalid response format
573	invalid push subscribe request data
574	invalid push event request data

In-App events:

Code	Description
236	successful request: inapp/unsubscribe
237	successful request: inapp/subscribe
238	successful request: inapp/status
239	successful request: inapp/event
240	successful request: inapp/placements
436	failed request: inapp/unsubscribe
437	failed request: inapp/placements
438	failed request: inapp/subscribe
439	failed request: inapp/status
440	failed request: inapp/event
536	in-app status request data is nil
537	in-app event request data is nil

Authentication events:

Code	Description
241	successful request: profile/authenticate
441	failed request: profile/authenticate
402	user is not authenticated

In-App Notifications

```
AltcraftSDK
// Singleton, SDK entry point
└── static let shared: AltcraftSDK
    // Public In-App notification functions
    └── inAppFunctions: PublicInAppFunctions
        // Subscribe to app lifecycle for automatic In-App display
        ├── registerLifecycleTracking(): Void
        // Set current screen marker (used for In-App notification filtering)
        ├── setScreen(placement: String): Void
        // Request available In-App placements
        ├── getInAppPlacements(): Void
        // Trigger a custom In-App notification by name
        ├── inAppTrigger(name: String): Void
        // Configure In-App notification animation (static)
        ├── setAnimation(_ block: @escaping @MainActor (InAppAnimator) -> Void): Void
        // Subscribe to In-App notifications (status = SUBSCRIBED)
        ├── inAppSubscribe(
        │     sync: Bool = true,
        │     profileFields: [String: Any?]? = nil,
        │     customFields: [String: Any?]? = nil,
        │     cats: [CategoryData]? = nil,
        │     replace: Bool? = nil,
        │     skipTriggers: Bool? = nil
        │   ): Void
        // Unsubscribe from In-App notifications (status = UNSUBSCRIBED)
        ├── inAppUnSubscribe(
        │     sync: Bool = true,
        │     profileFields: [String: Any?]? = nil,
        │     customFields: [String: Any?]? = nil,
        │     cats: [CategoryData]? = nil,
        │     replace: Bool? = nil,
        │     skipTriggers: Bool? = nil
        │   ): Void
        // In-App notification subscription status
        └── getInAppSubscriptionStatus(
              completion: @escaping (ResponseModel.ResponseWithHttpCode<ResponseModel.ResponseWithProfile>?) -> Void
            ): Void
```

Functions for working with In-App notifications:

- `func registerLifecycleTracking()` — **registers app lifecycle observers for correct In-App trigger handlin**
- `func setScreen()` — **sets the current screen marker, used for filtering In-App notifications;**
- `func getInAppPlacements()` — **requests available In-App placements from the server;**
- `func inAppTrigger()` — **triggers a custom In-App notification by name;**
- `static func setAnimation()` — **configures In-App notification display animation;**
- `func inAppSubscribe()` — **subscribes to In-App notifications;**
- `func inAppUnSubscribe()` — **unsubscribes from In-App notifications;**
- `func getInAppSubscriptionStatus()` — **retrieves the In-App notification subscription status.**

`public func registerLifecycleTracking(): Void`

Registers app lifecycle observers for automatic In-App notification display. After registration, the SDK automatically shows In-App notifications when the app comes to the foreground.

> TIP: Important
>
> **Call `registerLifecycleTracking()` as early as possible at app start — in `AppDelegate.application(_:didFinishLaun` lifecycle observers are registered before the first foreground transition. If the call is delayed, In-App n app launch may be missed.**

Example:

```
AltcraftSDK.shared.inAppFunctions.registerLifecycleTracking()
```

`public func setScreen(screen: String): Void`

Sets the current screen marker. The marker value is used by the Altcraft platform to filter In-App notifi notifications only on specific app screens.

Call this function when transitioning to a new screen so the SDK always knows which screen the user

Example:

```
AltcraftSDK.shared.inAppFunctions.setScreen(screen: "home_screen")
```

`public func inAppTrigger(name: String): Void`

Manually triggers the display of an In-App notification by its name. Used for custom triggers when a no response to a user action or an in-app event.

Example:

```
AltcraftSDK.shared.inAppFunctions.inAppTrigger(name: "my_custom_trigger")
```

`public func getInAppPlacements(): Void`

Requests available In-App placements from the Altcraft server. The request result can be obtained thro

Example:

```
AltcraftSDK.shared.inAppFunctions.getInAppPlacements()
```

`public static func setAnimation(_ block: @escaping @MainActor (InAppAnimator) -> Void): Void`

Configures a custom animation for In-App notification display. Accepts a closure with an `InAppAnimator` **p** `UIView` **and** `companyId` **after static resources are loaded.**

`InAppAnimator` **protocol:**

```
public protocol InAppAnimator {
    /// Campaign identifier associated with the animator
    var companyId: Int { get }

    /// Applies changes to the view without animation
    func prepare(_ block: (UIView) -> Void)

    /// Performs a UIKit animation for the In-App view
    func animate(
        duration: TimeInterval,
        delay: TimeInterval,
        options: UIView.AnimationOptions,
        animations: @escaping (UIView) -> Void,
        completion: ((Bool) -> Void)?
    )
}
```

Example:

```
AltcraftSDK.inAppFunctions.setAnimation { animator in
    animator.animate(
        duration: 0.3,
        delay: 0,
        options: .curveEaseInOut,
        animations: { view in
            view.alpha = 1.0
        },
        completion: nil
    )
}
```

`public func inAppSubscribe(sync: Bool, ...): Void`

Subscribes to In-App notifications. After a successful call, the user's profile will be subscribed to recei Altcraft platform.

Function parameters are the same as `pushSubscribe()`**:**

- `sync` **— synchronous request flag;**
- `profileFields` **— profile fields;**
- `customFields` **— subscription fields;**
- `cats` **— subscription categories;**
- `replace` **— when enabled, all subscriptions of other profiles with the same push token are set to** `uns`
- `skipTriggers` **— when enabled, the profile will be ignored by mailing and scenario triggers.**

Parameters are configured the same way as in `pushSubscribe()`**.**

Example:

```
AltcraftSDK.shared.inAppFunctions.inAppSubscribe()
```

`public func inAppUnSubscribe(sync: Bool, ...): Void`

Unsubscribes from In-App notifications. After a successful call, the user's profile will be unsubscribed f

Function parameters are the same as `pushUnSubscribe()`**.**

Example:

```
AltcraftSDK.shared.inAppFunctions.inAppUnSubscribe()
```

`public func getInAppSubscriptionStatus(completion: @escaping (ResponseModel.ResponseWithHttpCode<ResponseModel.Respons`

Retrieves the current In-App notification subscription status. The `completion` **receives a** `ResponseWithHttpC` `response?.profile?.subscription` **— the profile's current In-App notification subscription. If no subscription**

Example:

```
AltcraftSDK.shared.inAppFunctions.getInAppSubscriptionStatus { responseWithHttp in
    let httpCode = responseWithHttp?.httpCode
    let response = responseWithHttp?.response
    let profile = response?.profile
    let subscription = profile?.subscription
}
```

Authentication

```
AltcraftSDK
// Singleton, SDK entry point
└── static let shared: AltcraftSDK
    // Public authentication functions
    └── authFunctions: PublicAuthFunctions
        // Authenticate user
        ├── authenticate(): Void
        // Log out user
        └── logOut(): Void
```

Functions for user authentication:

- `func authenticate()` — **authenticates the user on the Altcraft platform;**
- `func logOut()` — **logs out the user, ending the session.**

`public func authenticate(): Void`

Authenticates the user on the Altcraft platform. Sends a request to the server with the current authent and binds the device to the user's profile.

After successful authentication, the user gains access to personalized In-App notifications.

> TIP: Important
>
> **Call `authenticate()` only when the user actually logs into the app and is known to the client (for exampl through a login form, OAuth, or another authentication mechanism in your app).**

Example:

```
AltcraftSDK.shared.authFunctions.authenticate()
```

`public func logOut(): Void`

Logs out the user — ends the current authentication session on the Altcraft platform. After this call, th session mode. Personalized In-App notifications are unavailable in this mode, as is subscription manag

> TIP: Important
>
> **Call `logOut()` only when the user actually logs out of the app and switches to anonymous mode.**

Example:

```
AltcraftSDK.shared.authFunctions.logOut()
```

Additional functions

Functional profile field updates

`public func actionField(key: String) -> ActionFieldBuilder`

A helper for the functional profile field update process. Supports:

```
    .set(value)
    .unset(value)
    .incr(value)
    .add(value)
    .delete(value)
    .upsert(value)
```

Example:

```
// "_fname" — field to update
// .set("Andrey") — sets the field to "Andrey"
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: AltcraftSDK.shared.pushSubscriptionFunctions.actionField(key: "_fname").set(value: "Andrey")
)
```

Clearing SDK data

`public func clear(completion: (() -> Void)? = nil): Void`

Clears SDK data and cancels all pending background tasks. Removes `CoreData` records and clears `UserD`

Example:

```
// Full SDK reset (cache/database/settings), then call completion
AltcraftSDK.shared. clear(completion: (() -> Void)? = nil)
```

The function accepts an optional `completion` called after cleanup and task cancellation.

Manual Push Event Registration

> **TIP: Note**
>
> **Use these only if you implement your own notification handling and do not pass them to Altcraft SDK. notifications automatically; these methods are needed only for custom client-side processing.**

```
// PublicPushEventFunctions

AltcraftSDK
//   Singleton, SDK entry point
└── public static let shared: AltcraftSDK
    // Manual sending of push events (delivery / open)
    └── public let pushEventFunctions: PublicPushEventFunctions
        // Record Altcraft push delivery (fires a delivery event)
        ├── public func deliveryEvent(from request: UNNotificationRequest): Void
        // Record Altcraft push open (fires an open event)
        └── public func openEvent(from request: UNNotificationRequest): Void
```

Manual event registration functions:

- `func deliveryEvent()` **— registers a delivery event;**
- `func openEvent()` **— registers an open event.**

`public func deliveryEvent(from request: UNNotificationRequest): Void`

Registers a delivery event in the Altcraft platform. Pass `UNNotificationRequest` in the `request` parameter.

Example:

```
// Record Altcraft push delivery (fires a delivery event)
AltcraftSDK.shared.pushEventFunctions.deliveryEvent(from: UNNotificationRequest)
```

`public func openEvent(from request: UNNotificationRequest): Void`

Registers an open event in the Altcraft platform. Pass `UNNotificationRequest` in the `request` parameter.

```
// Record Altcraft push open (fires an open event)
AltcraftSDK.shared.pushEventFunctions.openEvent(from: UNNotificationRequest)
```

SDK Configuration

Prerequisites

- **Push provider SDKs are integrated into the app project (see the push provider integration guides).**
- **The [Altcraft package] is added via Swift Package Manager.**

App preparation

Target settings

Configure the following app target settings:

- ☑ **General**

When using Swift Package Manager, verify that the Altcraft library is added to the app target under Fra Content.

- ☑ **Signing & Capabilities**

Add these capabilities:

- **Push Notifications**
- **App Groups**
- **Background Modes (Background fetch, Background processing)**
- ☑ **Info**

Add the key Permitted background task scheduler identifiers with the value "lib.Altcraft.bgTask.systen SDK can register a `bgTask` responsible for retrying failed requests in background mode.

- ☑ **Optional**

If you use Firebase Cloud Messaging, add the key `FirebaseAppDelegateProxyEnabled` (Boolean) with the value method swizzling performed by Firebase Messaging by default.

AppDelegate settings

In `AppDelegate.application(_:didFinishLaunchingWithOptions:)`, do the following:

- **pass the `AppGroup` identifier to the SDK**
- **register the SDK `BGTask` (background tasks)**
- **(optional) initialize `UNUserNotificationCenter` features on the SDK side**

Setting the AppGroup identifier

Adding an `AppGroup` identifier is required to exchange data with the Notification Service Extension and f `setAppGroup()`:

```
// Configure the App Group and initialize Core Data for the shared container
AltcraftSDK.shared.setAppGroup(groupName: String)
```

Registering the BGTask

Registering SDK background tasks enables retrying failed requests in the background. Use `registerBack`

```
// Register the SDK background task and schedule periodic execution (about every 3 hours)
AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
```

Initializing UNUserNotificationCenter features (optional)

The SDK includes the `NotificationManager` class, which:

- **configures `UNUserNotificationCenter` and requests notification permissions (alert/sound/badge);**
- **calls `UIApplication.registerForRemoteNotifications()` and ensures correct APNs registration;**
- **displays notifications in foreground;**
- **handles notification taps and action buttons — navigates to a specified URL or opens the app;**
- **registers the push open event ("open").**

To use the SDK `NotificationManager`, call this function in `AppDelegate.application(_:didFinishLaunchingWithOptions`

```
func registerForPushNotifications(for application: UIApplication, completion: ((_ granted: Bool, _ error: Error?) -> Void)? = nil): Void
```

This function sets the delegate, requests the user's permission, and registers the app for push notifica

Example of a correct `AppDelegate.application` setup

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        let appGroup = "group.your.id"

        //ID App Group Setup
        AltcraftSDK.shared.setAppGroup(groupName: appGroup)

        //Background SDK task registration
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()

        //SDK UNUserNotificationCenter functions initialization
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        //Other functions

        return true
    }
    //Other functions AppDelegate
}
```

Using this class is optional; you can replace it with your own implementation. To do so:

- **set the notification center delegate:** `UNUserNotificationCenter.current().delegate = self`**;**
- **request notification permissions and call** `UIApplication.shared.registerForRemoteNotifications()` **strictly o**
- **implement** `userNotificationCenter(_:willPresent:withCompletionHandler:)` **to display notifications in foregro**
- **call** `AltcraftSDK.shared.pushEventFunctions.openEvent(from:)` **in** `userNotificationCenter(_:didReceive:withCompletio` **notification tap (registering the "open" event);**
- **implement custom logic for handling notification taps (URL/Deep Link/buttons), and ensure a fallba valid link is present).**

JWT protocol setup (optional)

JWTInterface is the protocol for requesting a JWT token. It provides a current JWT token from the app protocol if API requests use JWT authentication. JWT confirms that user identifiers are authenticated JWT authentication is required if the matching type differs from push data from the subscription (for e an email or phone number).

SDK protocol:

```
public protocol JWTInterface {
    func getToken() -> String?
}
```

App-side implementation

```
class JWTProvider: JWTInterface {
    func getToken() -> String? {
         // your code that returns a JWT
      }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

In `AppDelegate`**, specify:**

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> B
    // Other code

    AltcraftSDK.setJWTProvider(provider: JWTProvider())
  }
  // Other functions
}
```

> TIP: Note
>
> `getJWT()` **is a synchronous function. The SDK execution thread will pause until JWT is obtained. It's re immediately from cache (in-memory,** `UserDefaults` **or** `Keychain` **) to speed up requests. Ideally, prepare a app start) and store it in cache so the token is available without delays. Returning** `nil` **is acceptable**

Provider push token request and deletion protocols

These token request/deletion protocol implementations ensure the current token is used and allow dy client's request. Implement only the protocols for the push providers used in your project.

Register providers in `application(_:didFinishLaunchingWithOptions:)` (AppDelegate). This registration point gu deterministic registration at process start, including in background.

Apple Push Notification service

APNSInterface — the protocol for requesting an APNs push token. This protocol has no token deletion f protocols.

SDK protocol:

```
public protocol APNSInterface {

    func getToken(completion: @escaping (String?) -> Void)
}
```

Recommended app-side implementation

```
class APNSProvider: APNSInterface {
    func getToken(completion: @escaping (String?) -> Void) {
        // your code that returns the APNs token
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // other code

    AltcraftSDK.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
  }
  // other functions
}
```

Firebase Cloud Messaging

FCMInterface — the protocol for requesting and deleting a Firebase Cloud Messaging push token.

SDK protocol:

```
public protocol FCMInterface {
    func getToken(completion: @escaping (String?) -> Void)

    func deleteToken(completion: @escaping (Bool) -> Void)
}
```

Recommended app-side implementation

```
class FCMProvider: FCMInterface {

    /// Retrieves the current FCM token
   func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token (as Data), retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// Set the APNs token for FCM before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// Request the FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Other code

    AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
  }
  // Other functions
}
```

Huawei Mobile Services

HMSInterface — the protocol for requesting and deleting a Huawei Mobile Services push token. SDK protocol:

```
public protocol HMSInterface {

    func getToken(completion: @escaping (String?) -> Void)

    func deleteToken(completion: @escaping (Bool) -> Void)
}
```

Recommended app-side implementation

```
class HMSProvider: HMSInterface {

    func getToken(completion: @escaping (String?) -> Void) {

        // variable containing the APNs token (as a String), which is later
        // passed to HmsInstanceId.getInstance().getToken(apnsToken)
        guard let apnsToken = getAPNsTokenFromUserDefault() else {
            completion(nil)
            return
        }

        let token = HmsInstanceId.getInstance().getToken(apnsToken)
        completion(token)
    }


    func deleteToken(completion: @escaping (Bool) -> Void) {
        HmsInstanceId.getInstance().deleteToken()
        completion(true)
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Other code

    AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
  }
  // Other functions
}
```

Preparing the Notification Service Extension

Adding an NSE is required for Rich Push support and guaranteed delivery event registration.

Create a Notification Service Extension:

- **File — New — Target — Notification Service Extension;**
- **Choose a Product Name for the extension target;**
- **Activate it.**

In the Notification Service Extension:

- ✅ **General**
- **Set Minimum Deployments — this Xcode build setting defines the minimum OS version required to r Extension.**
- **Add the Altcraft library under Frameworks, Libraries and Embedded Content.**
- ✅ **Signing & Capabilities**
- **Specify the `AppGroup` identifier.**

Then configure the SDK inside `UNNotificationServiceExtension`:

- **Import the `Altcraft` library.**
- **In the method `didReceive(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNoti`**
  - **Pass the `appGroup` identifier to `AltcraftNSE.shared.setAppGroup(groupName: appGroupsName)`. The identifier main app target.**
  - **Set the JWT provider (if JWT authentication is used): `AltcraftNSE.shared.setJWTProvider(provider: jwt`**
  - **Check whether the incoming notification belongs to Altcraft using `AltcraftNSE.shared.isAltcraftPus`**
  - **If the notification is from Altcraft, pass the `UNNotificationRequest` and the `contentHandler` to `AltcraftNSE.shared.handleNotificationRequest(request: request, contentHandler: contentHandler)`.**

The example below can be used as a ready-to-use `UNNotificationServiceExtension` class. Replace all autoge implementation if no extra logic is required. If you already use `UNNotificationServiceExtension`, integrate th implementation.

Example implementation of `UNNotificationServiceExtension`

```
class NotificationService: UNNotificationServiceExtension {

    /// - important! Set app groups identifier.
    let appGroupID = "group.your.id"

    /// - important! Set the provider to JWT if you are using JWT authentication.
    let jwtProvider = JWTProvider()

    override func didReceive(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationContent) -> Voi

        AltcraftNSE.shared.setAppGroup(groupName: appGroupID)
        AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)

        if AltcraftNSE.shared.isAltcraftPush(request) {
            AltcraftNSE.shared.handleNotificationRequest(request: request, contentHandler: contentHandler)
        } else {
            contentHandler(request.content)
        }
    }

    override func serviceExtensionTimeWillExpire() {
        AltcraftNSE.shared.serviceExtensionTimeWillExpire()
    }
}
```

TIP: Note

The app target and the Notification Service Extension target run in separate processes and do not sh process notifications correctly, the SDK must also register the `JWTProvider` and AppGroup ID in the NS `request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void)`. This e authorization token when sending the push delivery event ("delivery_event").

SDK initialization

Initialization parameters

Use the `AltcraftConfiguration` class to pass configuration parameters:

```
public final class AltcraftConfiguration {
    private let apiUrl: String
    private let rToken: String?
    private var appInfo: AppInfo?
    private var providerPriorityList: [String]?
}
```

The configuration object is built after calling the `build()` function of `AltcraftConfiguration.Builder(): Altcr`

Parameter descriptions:

apiUrl `String`

Required: Yes

Description: Altcraft API endpoint URL

---

rToken `String?`

Default: `nil`

Required: No

Description: Altcraft role token (identifies a resource, DB, or account). Used when the only matching ty issued by a provider.

---

appInfo `AppInfo?`

Default: `nil`

Required: No

Description: Basic app metadata for Firebase Analytics. To set this parameter, use the public SDK stru

```
public struct AppInfo: Codable {

    /// Unique App ID (Firebase `app_id`)
    public var appID: String

    /// Unique app instance ID (Firebase `app_instance_id`)
    public var appIID: String

    /// App version (Firebase `app_version`)
    public var appVer: String
}
```

---

enableLogging `Bool?`

Default: `nil`
Required: No
Description: Enables/disables logging. Logs can be found by `[Altcraft SDK]` prefix.

---

providerPriorityList `[String]?`
Default: `nil`
Required: No
**Description: Provider priority list. Used to automatically refresh the subscription's push token if the hig
unavailable. Priority is defined by the index in the list: the element at index 0 has the highest priority.**
Usage example:

```
providerPriorityList = [
    Constants.ProviderName.apns,
    Constants.ProviderName.firebase,
    Constants.ProviderName.huawei
]
```

With this configuration:

- **The SDK first requests the APNs token; if APNs is unavailable — FCM; if FCM is unavailable — HMS**
- **Works provided the corresponding provider protocols are implemented in the app.**

For convenient assignment, the SDK exposes public constants in `enum Constants`:

```
public enum Constants {
    public enum ProviderName {
        /// Provider name for Firebase
        public static let firebase = "ios-firebase"

        /// Provider name for APNs
        public static let apns = "ios-apns"

        /// Provider name for HMS
        public static let huawei = "ios-huawei"
    }
}
```

If not specified, the default provider priority is:

```
apns -> firebase -> huawei
```

The parameter value may contain a single element — in this case, only that provider will be used regar
You can omit this parameter if:

- **only one provider is used in the project;**
- **the default priority fits your needs.**

Performing initialization

> **TIP: Note**
>
> **Call `AltcraftSDK.shared.initialization()` when needed, but only after setting the `AppGroup` identifier and re
> implement the SDK protocols. Make requests after the configuration is set.**

Use `public func initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)` to initial

```
// SDK initialization and configuration
AltcraftSDK.shared.initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)
```

**The `initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)` function expects a
required values set and built:**

```
 let config = AltcraftConfiguration.Builder()
        .setApiUrl("your apiUrl")
        .setRToken("your rToken")
        .setAppInfo(AppInfo(appID: "your appID", appIID: "your appIID", appVer: "your appVer"))
        .setProviderPriorityList([Constants.ProviderName.firebase])
        .build()
```

Example of the correct initialization order in `AppDelegate.application(_:didFinishLaunchingWithOptions:)` (after setting the `AppGroup` ID and registering pro

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        // Firebase configuration can be placed here if Firebase is used in the project

        let appGroup = "group.altcraft.apns.example"

        AltcraftSDK.shared.setAppGroup(groupName: appGroup)
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        let config = AltcraftConfiguration.Builder()
            .setApiUrl("your apiUrl")
            .setRToken("your rToken")
            .setAppInfo(AppInfo(appID: "your appID", appIID: "your appIID", appVer: "your appVer"))
            .setProviderPriorityList([Constants.ProviderName.firebase])
            .build()

        AltcraftSDK.shared.initialization(configuration: config) { success in
            if success {
               // actions after successful initialization
            }
        }

        // Other functions

        return true
    }
    // Other AppDelegate functions
}
```

Public SDK API

class AltcraftSDK

```
AltcraftSDK
// Singleton entry point to the SDK
├ public static let shared: AltcraftSDK
// SDK initialization with configuration (completion is called on main)
├ public func initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil): Void
// Sets App Group and initializes Core Data for the shared container
├ public func setAppGroup(groupName: String): Void
// Registers a JWT provider for token retrieval
├ public func setJWTProvider(provider: JWTInterface): Void
// Completely clears SDK data (cache / DB / settings), then calls completion
├ public func clear(completion: (() -> Void)? = nil): Void
|
// SDK event stream (single active subscriber)
├ public let eventSDKFunctions: SDKEvents
|  // Subscribe to SDK events (replaces the existing subscriber)
|  ├ func subscribe(callback: @escaping (Event) -> Void): Void
|  // Unsubscribe from events (callback remains set, delivery stops)
|  ├ func unsubscribe(): Void
|  // Event model (base type for all events and errors)
|  ├ Event
|  |   // Base event model
|  |   id: UUID
|  |   function: String
|  |   message: String?
|  |   eventCode: Int?
|  |   value: [String: Any?]?
|  |   date: Date
|  |   // Initializer (normalizes function, filters nil values)
|  |   init(function: String, message: String? = nil, eventCode: Int? = nil, value: [String: Any?]? = nil, date: Date = Date())
|  |   // Equality by unique id
|  |   static func ==(lhs: Event, rhs: Event) -> Bool
|  |   // Hashing by id
|  |   func hash(into hasher: inout Hasher): Void
|  |
|  |   // Non-retryable errors (4xx class)
|  |   ErrorEvent : Event
|  |   init(function: String, message: String? = nil, eventCode: Int? = nil, value: [String: Any?]? = nil, date: Date = Date())
|  |
|  |   // Retryable errors (typically 5xx)
|  |   RetryEvent : ErrorEvent
|  |   init(function: String, message: String? = nil, eventCode: Int? = 0, value: [String: Any?]? = nil, date: Date = Date())
|
// Push token and provider management (FCM / HMS / APNs)
├  public let pushTokenFunctions: PublicPushTokenFunctions
|  // Set Firebase Cloud Messaging provider (nil to remove)
|  ├ public func setFCMTokenProvider(_ provider: FCMInterface?): Void
|  // Set Huawei Mobile Services provider (nil to remove)
|  ├ public func setHMSTokenProvider(_ provider: HMSInterface?): Void
|  // Set Apple Push Notification service provider (nil to remove)
|  ├ public func setAPNSTokenProvider(_ provider: APNSInterface?): Void
|  // Asynchronously get the current push token (completion optional)
|  ├ public func getPushToken(completion: ((TokenData?) -> Void)? = nil): Void
|  // Manually save a token (String for FCM/HMS, Data for APNs); provider: "ios-firebase" | "ios-huawei" | "ios-apns"
|  ├ public func setPushToken(provider: String, pushToken: Any?): Void
|  // Apply a new provider priority list and trigger token update
|  ├ public func changePushProviderPriorityList(_ list: [String]): Void
|  // Delete a token for the selected provider and call completion
|  ├ public func deleteDeviceToken(provider: String, completion: @escaping () -> Void): Void
|  // Force token update (resolve provider → delete token (except APNs) → request a new one)
|  └ public func forcedTokenUpdate(completion: (() -> Void)? = nil): Void
|
// Push subscription management (status changes, status requests)
├  public let pushSubscriptionFunctions: PublicPushSubscriptionFunctions
|  // Subscribe (status = SUBSCRIBED)
|  ├ public func pushSubscribe(
|  |     sync: Bool = true,
|  |     profileFields: [String: Any?]? = nil,
|  |     customFields: [String: Any?]? = nil,
|  |     cats: [CategoryData]? = nil,
|  |     replace: Bool? = nil,
|  |     skipTriggers: Bool? = nil
|  |   ): Void
|  // Unsubscribe (status = UNSUBSCRIBED)
|  ├ public func pushUnSubscribe(
|  |     sync: Bool = true,
|  |     profileFields: [String: Any?]? = nil,
|  |     customFields: [String: Any?]? = nil,
|  |     cats: [CategoryData]? = nil,
|  |     replace: Bool? = nil,
|  |     skipTriggers: Bool? = nil
```

```
    ): Void
  // Suspend (status = SUSPENDED)
  ├─ public func pushSuspend(
  │     sync: Bool = true,
  │     profileFields: [String: Any?]? = nil,
  │     customFields: [String: Any?]? = nil,
  │     cats: [CategoryData]? = nil,
  │     replace: Bool? = nil,
  │     skipTriggers: Bool? = nil
  │   ): Void
  // UnSuspend subscription (single request, no retries / persistence)
  ├─ public func unSuspendPushSubscription(
  │     completion: @escaping (ResponseWithHttp?) -> Void
  │   ): Void
  // Status of the latest profile subscription
  ├─ public func getStatusOfLatestSubscription(
  │     completion: @escaping (ResponseWithHttp?) -> Void
  │   ): Void
  // Subscription status for the current token / provider
  ├─ public func getStatusForCurrentSubscription(
  │     completion: @escaping (ResponseWithHttp?) -> Void
  │   ): Void
  // Latest subscription status for a provider (current one if nil)
  └─ public func getStatusOfLatestSubscriptionForProvider(
        provider: String? = nil,
        completion: @escaping (ResponseWithHttp?) -> Void
     ): Void
 // add a functional profile field (set / incr / ...)
 └─ public func actionField(key: String) -> ActionFieldBuilder
// Registering Mobile Events
├─ public let mobileEventFunctions: PublicMobileEventFunctions
  // Registers an Altcraft mobile event
  └─ public func mobileEvent(
        sid: String,
        altcraftClientID: String = "",
        eventName: String,
        sendMessageId: String? = nil,
        payload: [String: Any?]? = nil,
        matching: [String: Any?]? = nil,
        matchingType: String? = nil,
        profileFields: [String: Any?]? = nil,
        subscription: (any Subscription)? = nil,
        utm: UTM? = nil
     ): Void
|
// Updating Profile Fields
├─ public let profileFunctions: PublicProfileFunctions
  // Updates Altcraft profile fields
  └─ public func updateProfileFields(
        profileFields: [String: Any?]? = nil,
        skipTriggers: Bool? = nil
     ): Void
|
// Manual push event sending (delivery / open)
├─  public let  pushEventFunctions: PublicPushEventFunctions
  // Register delivery of an Altcraft push (fires delivery event)
  ├─ public func deliveryEvent(from request: UNNotificationRequest): Void
  // Register opening of an Altcraft push (fires open event)
  └─ public func openEvent(from request: UNNotificationRequest): Void
|
// Periodic background task registration (BGAppRefreshTask)
├─ public let backgroundTasks: BackgroundTasks
  // Register a task and schedule periodic execution (~every 3 hours)
  └─ public func registerBackgroundTask(): Void
|
// In-app notification management (notification center delegate)
├─  public let notificationManager: NotificationManager
   // NotificationManager singleton (entry point to the notification manager)
   ├─ public static let shared: NotificationManager
   // ObjC-friendly accessor (equivalent to shared)
   ├─ public class func sharedInstance() -> NotificationManager
   //
   // NotificationCenter events (SDK posts these)
   ├─ public extension Notification.Name
   │  // Fired when a notification is about to be presented in foreground.
   │  // userInfo: ["notification": UNNotification]
   │  ├─ static let altcraftPushWillPresent: Notification.Name
   │  // Fired when user taps on a delivered notification or performs an action.
   │  // userInfo: ["response": UNNotificationResponse]
   │  └─ static let altcraftPushDidReceive: Notification.Name
```

```
    //
    // App-level delegation modes
    ├ public var customPushProcessing: Bool
    ├ public var customClickProcessing: Bool
    //
    // Callbacks for custom handling
    │  // Called when a foreground push arrives.
    │  // App must call completion when customPushProcessing = true.
    ├ public var onForegroundNotification: ((UNNotification, @escaping (UNNotificationPresentationOptions) -> Void) -> Void)?
    │  // Called when the user taps a notification (or performs an action).
    │  // App must call completion when customClickProcessing = true.
    ├ public var onNotificationClick: ((UNNotificationResponse, @escaping () -> Void) -> Void)?
    //
    // Push registration (delegate + requestAuthorization + registerForRemoteNotifications)
    ├ func registerForPushNotifications(for application: UIApplication, completion: ((_ granted: Bool, _ error: Error?) -> Void)? = nil): \
    //
    // UNUserNotificationCenterDelegate
    │  // Posts altcraftPushWillPresent; if customPushProcessing is enabled,
    │  // delegates the decision to the app,
    │  // otherwise shows banner/alert (+ badge/sound)
    ├ public func userNotificationCenter(
    │     _ center: UNUserNotificationCenter,
    │     willPresent notification: UNNotification,
    │     withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void
    │   ): Void
    │  // Posts altcraftPushDidReceive; if customClickProcessing is enabled,
    │  // delegates handling to the app,
    │  // otherwise parses userInfo, sends an open event and performs the action,
    │  // then calls completionHandler
    └ public func userNotificationCenter(
        _ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: @escaping () -> Void
      ): Void
|
// Public In-App notification functions
├  public let inAppFunctions: PublicInAppFunctions
    // Subscribe to app lifecycle for automatic In-App display
    ├ public func registerLifecycleTracking(): Void
    // Set current screen marker (used for In-App notification filtering)
    ├ public func setScreen(screen: String): Void
    // Request available In-App placements
    ├ public func getInAppPlacements(): Void
    // Trigger a custom In-App notification by name
    ├ public func inAppTrigger(name: String): Void
    // Configure In-App notification animation (static)
    ├ public static func setAnimation(_ block: @escaping @MainActor (InAppAnimator) -> Void): Void
    // Subscribe to In-App notifications (status = SUBSCRIBED)
    ├ public func inAppSubscribe(
    │     sync: Bool = true,
    │     profileFields: [String: Any?]? = nil,
    │     customFields: [String: Any?]? = nil,
    │     cats: [CategoryData]? = nil,
    │     replace: Bool? = nil,
    │     skipTriggers: Bool? = nil
    │   ): Void
    // Unsubscribe from In-App notifications (status = UNSUBSCRIBED)
    ├ public func inAppUnSubscribe(
    │     sync: Bool = true,
    │     profileFields: [String: Any?]? = nil,
    │     customFields: [String: Any?]? = nil,
    │     cats: [CategoryData]? = nil,
    │     replace: Bool? = nil,
    │     skipTriggers: Bool? = nil
    │   ): Void
    // In-App notification subscription status
    └ public func getInAppSubscriptionStatus(
        completion: @escaping (ResponseModel.ResponseWithHttpCode<ResponseModel.ResponseWithProfile>?) -> Void
      ): Void
|
// Public authentication functions
├  public let authFunctions: PublicAuthFunctions
    // Authenticate user
    ├ public func authenticate(): Void
    // Log out user
    └ public func logOut(): Void
```

class AltcraftNSE

```
// Used inside the Notification Service Extension (NSE) — handles processing Altcraft push notifications,
// determining Altcraft notifications, and registering mobile events without relying on UIApplication
AltcraftNSE
// Shared (singleton) instance for centralized NSE logic
├─ public static let shared: AltcraftNSE
// Provides access to the mobile event registration API
├─ public let mobileEventFunctions: PublicMobileEventFunctions
// Registers an Altcraft mobile event
├─ public func mobileEvent(
│      sid: String,
│      altcraftClientID: String = "",
│      eventName: String,
│      sendMessageId: String? = nil,
│      payload: [String: Any?]? = nil,
│      matching: [String: Any?]? = nil,
│      matchingType: String? = nil,
│      profileFields: [String: Any?]? = nil,
│      subscription: (any Subscription)? = nil,
│      utm: UTM? = nil
│   ): Void
// Updates Altcraft profile fields
├─ public func updateProfileFields(
│      profileFields: [String: Any?]? = nil,
│      skipTriggers: Bool? = nil
│   ): Void
// Checks if the notification is related to Altcraft
├─ public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool
// Processes the incoming Altcraft notification and passes the modified content to the contentHandler
├─ public func handleNotificationRequest(
│      request: UNNotificationRequest,
│      contentHandler: @escaping (UNNotificationContent) -> Void
│   ): Void
// Called by the system when the NSE execution time expires
├─ public func serviceExtensionTimeWillExpire(): Void
// Sets the App Group and initializes the shared data container
├─ public func setAppGroup(groupName: String): Void
// Registers a JWT provider for secure token handling
└─ public func setJWTProvider(provider: JWTInterface): Void
```

class AltcraftConfiguration

```
AltcraftConfiguration
// Altcraft SDK configuration class.
// Stores the base API URL and optional parameters (rToken, AppInfo, providerPriorityList, enableLogging).
// Created only via Builder. ObjC-compatible: AltcraftConfiguration / AltcraftConfiguration_Builder.
│
├─ @objc(AltcraftConfiguration_Builder) public final class Builder
│   // Builder for creating a configuration with validation.
│   // Allows setting required and optional fields and assembling the configuration object.
│   // Builder constructor
│   ├─ public override init()
│   // Set API URL (required)
│   ├─ @discardableResult public func setApiUrl(_ url: String) -> Builder
│   // Set resource token (optional)
│   ├─ @discardableResult public func setRToken(_ rToken: String?) -> Builder
│   // Set application metadata (Swift AppInfo model) (optional)
│   ├─ @discardableResult public func setAppInfo(_ info: AppInfo?) -> Builder
│   // Set application metadata (ObjC DTO AppInfoObjC) (optional, not available in Swift)
│   ├─ @available(swift, obsoleted: 1) @objc(setAppInfo:) @discardableResult public func setAppInfo(_ dto: AppInfoObjC?) -> Builder
│   // Set push provider priority list (optional)
│   ├─ @discardableResult public func setProviderPriorityList(_ list: [String]?) -> Builder
│   // SDK internal logging control (optional)
│   // true — enable, false — disable, nil — keep default behavior
│   ├─ @discardableResult public func setEnableLogging(_ enabled: Bool?) -> Builder
│   // Build a valid configuration (nil if validation fails)
│   // Validation: apiUrl is not empty; providerPriorityList is valid; appInfo resolves (Swift AppInfo has priority, otherwise AppInfoObjC ·
│   └─ public func build() -> AltcraftConfiguration?
// Access to configuration parameters
├─ public func getApiUrl() -> String
├─ public func getRToken() -> String?
├─ public func getAppInfo() -> AppInfo?
├─ public func getEnableLogging() -> Bool?
└─ public func getProviderPriorityList() -> [String]?
```

Structures

```
/// Stores a push token and provider name.
public struct TokenData: Codable {
    public let provider: String
    public let token: String
}

/// Application metadata for Firebase Analytics (if used).
public struct AppInfo: Codable {
    public var appID: String
    public var appIID: String
    public var appVer: String
}

/// API response wrapper with HTTP status code.
public struct ResponseWithHttp {
    public let httpCode: Int?
    public let response: Response?
}

/// Synchronous subscription request response.
public struct Response: Codable {
    public let error: Int?
    public let errorText: String?
    public let profile: ProfileData?
}

/// User profile data.
public struct ProfileData: Codable {
    public let id: String?
    public let status: String?
    public let isTest: Bool?
    public let subscription: SubscriptionData?
}

/// Current profile subscription.
public struct SubscriptionData: Codable {
    public let subscriptionId: String?
    public let hashId: String?
    public let provider: String?
    public let status: String?
    public let fields: [String: JSONValue]?
    public let cats: [CategoryData]?
}

/// Subscription category (name / title / flags).
public struct CategoryData: Codable {
    public var name: String?
    public var title: String?
    public var steady: Bool?
    public var active: Bool?
}

/// Email channel subscription (mobile events).
public struct EmailSubscription: Subscription, Codable {
    public let resourceId: Int
    public let email: String
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "email"
}

/// SMS channel subscription (mobile events).
public struct SmsSubscription: Subscription, Codable {
    public let resourceId: Int
    public let phone: String
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "sms"
}

/// Push channel subscription (mobile events).
public struct PushSubscription: Subscription, Codable {
    public let resourceId: Int
    public let provider: String
    public let subscriptionId: String
    public let status: String?
```

```
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "push"
}

/// Subscription with cc_data (Telegram, WhatsApp, Viber, Notify).
public struct CcDataSubscription: Subscription, Codable {
    public let resourceId: Int
    public let channel: String
    public let ccData: [String: JSONValue]
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
}

/// UTM parameters for mobile events.
public struct UTM: Codable {
    public let campaign: String?
    public let content: String?
    public let keyword: String?
    public let medium: String?
    public let source: String?
    public let temp: String?
}
```

Protocols

InAppAnimator:

```
/// Protocol for configuring In-App notification animation.
/// Provides access to UIView and companyId after static resources are loaded.
InAppAnimator
// Campaign identifier associated with the animator
├─ var companyId: Int { get }
// Applies changes to the view without animation
├─ func prepare(_ block: (UIView) -> Void): Void
// Performs a UIKit animation for the In-App view
└─ func animate(
     duration: TimeInterval,
     delay: TimeInterval,
     options: UIView.AnimationOptions,
     animations: @escaping (UIView) -> Void,
     completion: ((Bool) -> Void)?
   ): Void
```

JWTInterface:

```
JWTInterface
// Access to the current JWT
└─ func getToken() -> String?
   // Returns the current JWT or nil
```

APNSInterface:

```
APNSInterface
// APNs token retrieval contract
└─ func getToken(completion: @escaping (String?) -> Void)
   // Returns the APNs token via completion or nil
```

FCMInterface:

```
FCMInterface
// Contract for FCM token operations (retrieve / revoke)
├─ func getToken(completion: @escaping (String?) -> Void)
│  // Returns the FCM token via completion or nil
└─ func deleteToken(completion: @escaping (Bool) -> Void)
   // Deletes the FCM token; completion(true|false)
```

HMSInterface:

```
HMSInterface
// Contract for HMS token operations
├─ func getToken(completion: @escaping (String?) -> Void)
│  // Returns the HMS token via completion or nil
└─ func deleteToken(completion: @escaping (Bool) -> Void)
   // Deletes the HMS token; completion(true|false)
```

Subscription:

```
Subscription (protocol)
// Base protocol for all subscription types passed with a mobile event
```

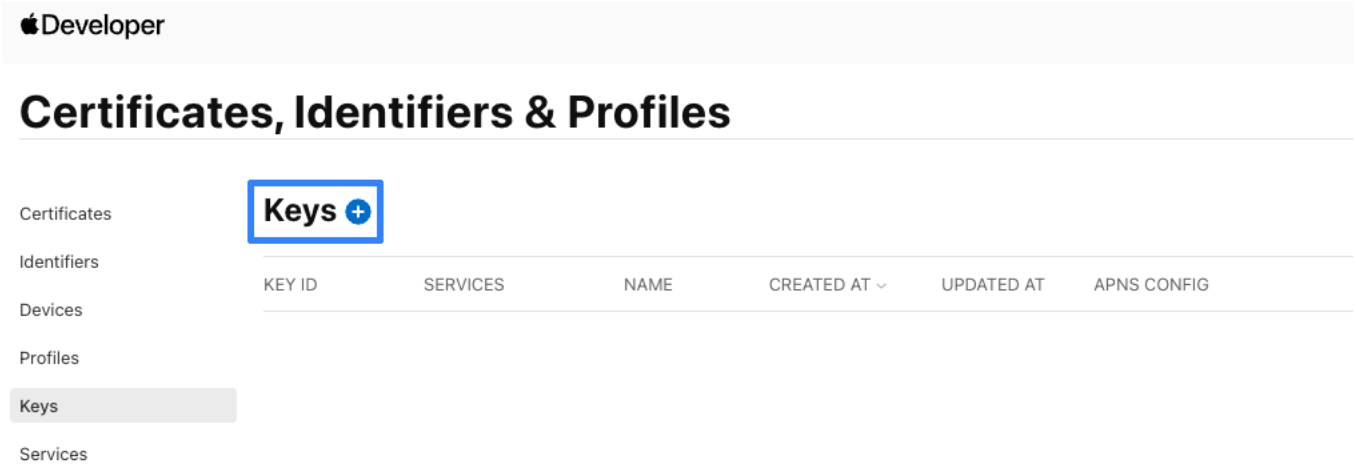
Apple Push Notification Service

You can configure APNS using either an authorization key or a certificate. The setup method depends c

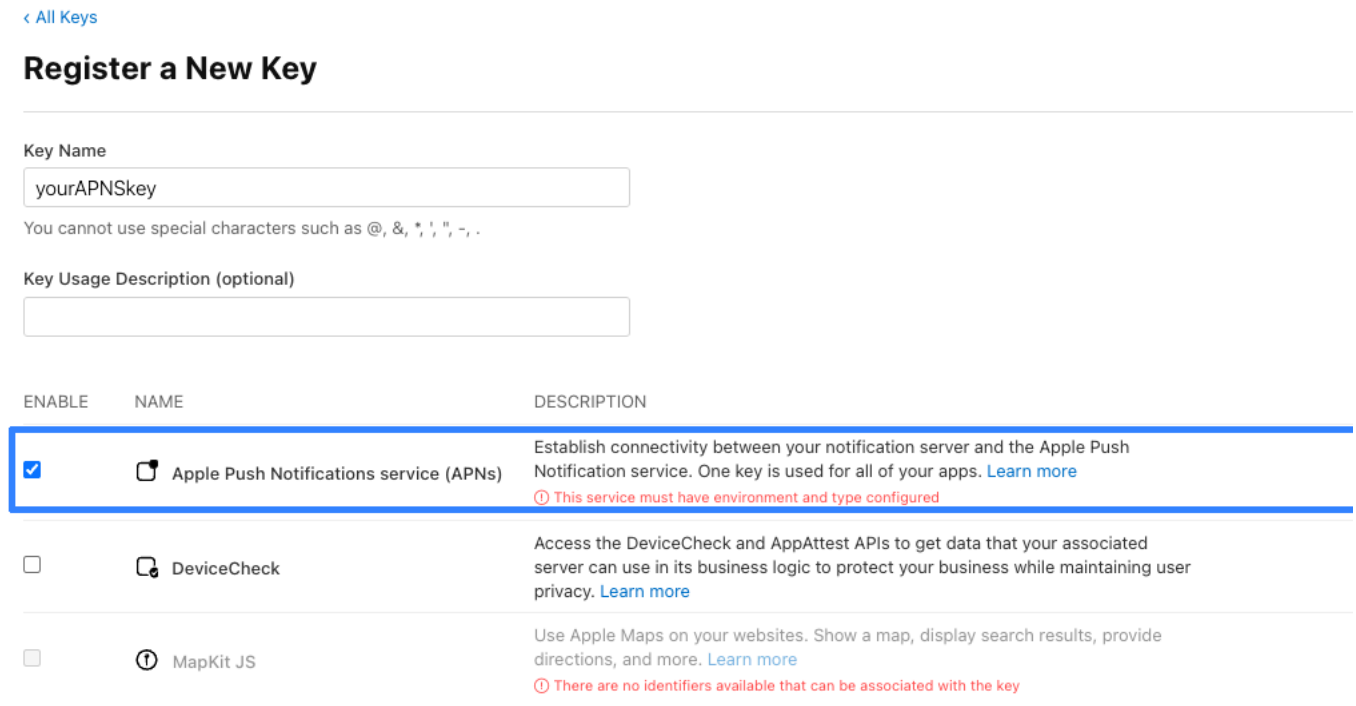
Option 1: using an Auth Key

Step 1. APNS configuration

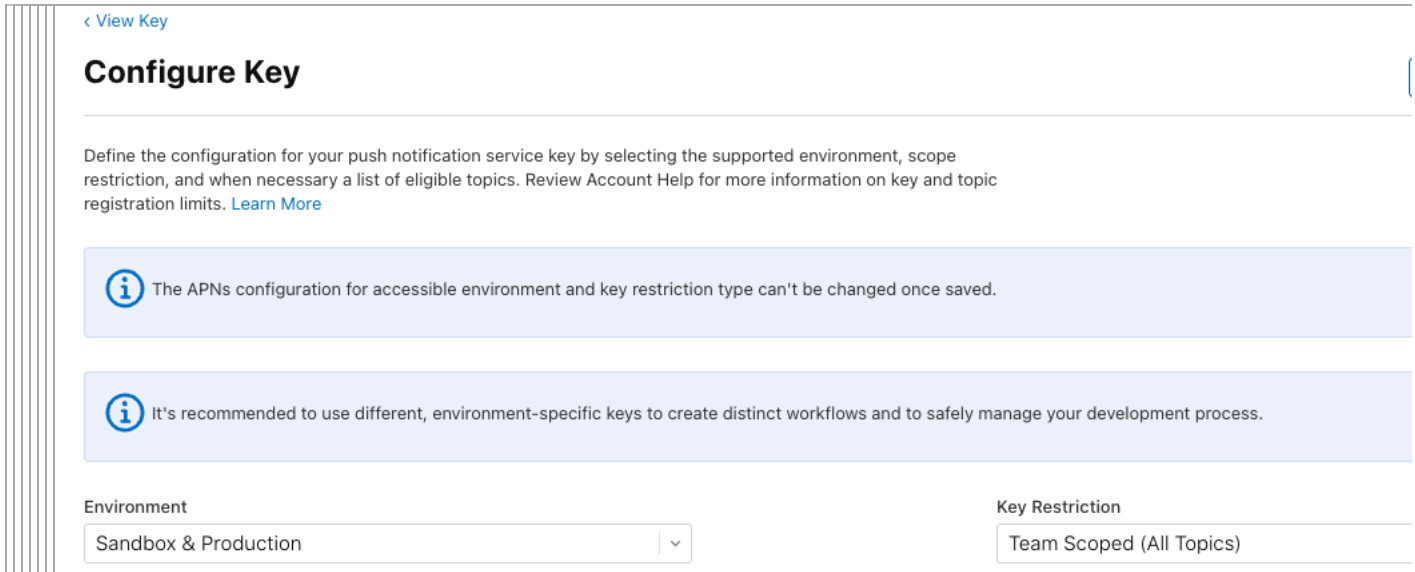
Go to the Apple Developer website, section Account. Navigate to Certificates, Identifiers & Profiles — I



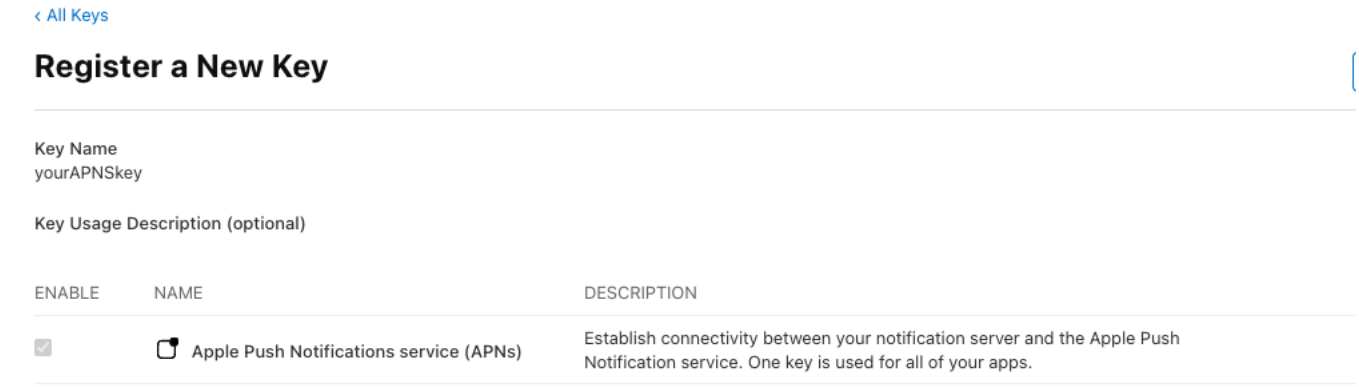
Select Apple Push Notification service:



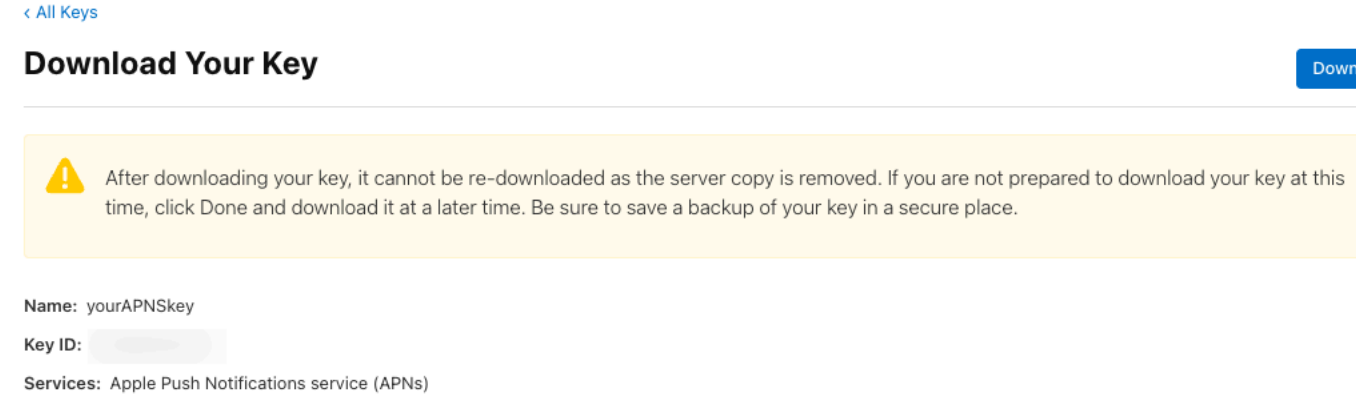
Configure APNs. Click Configure and select the Environment based on your setup:



Register the key:

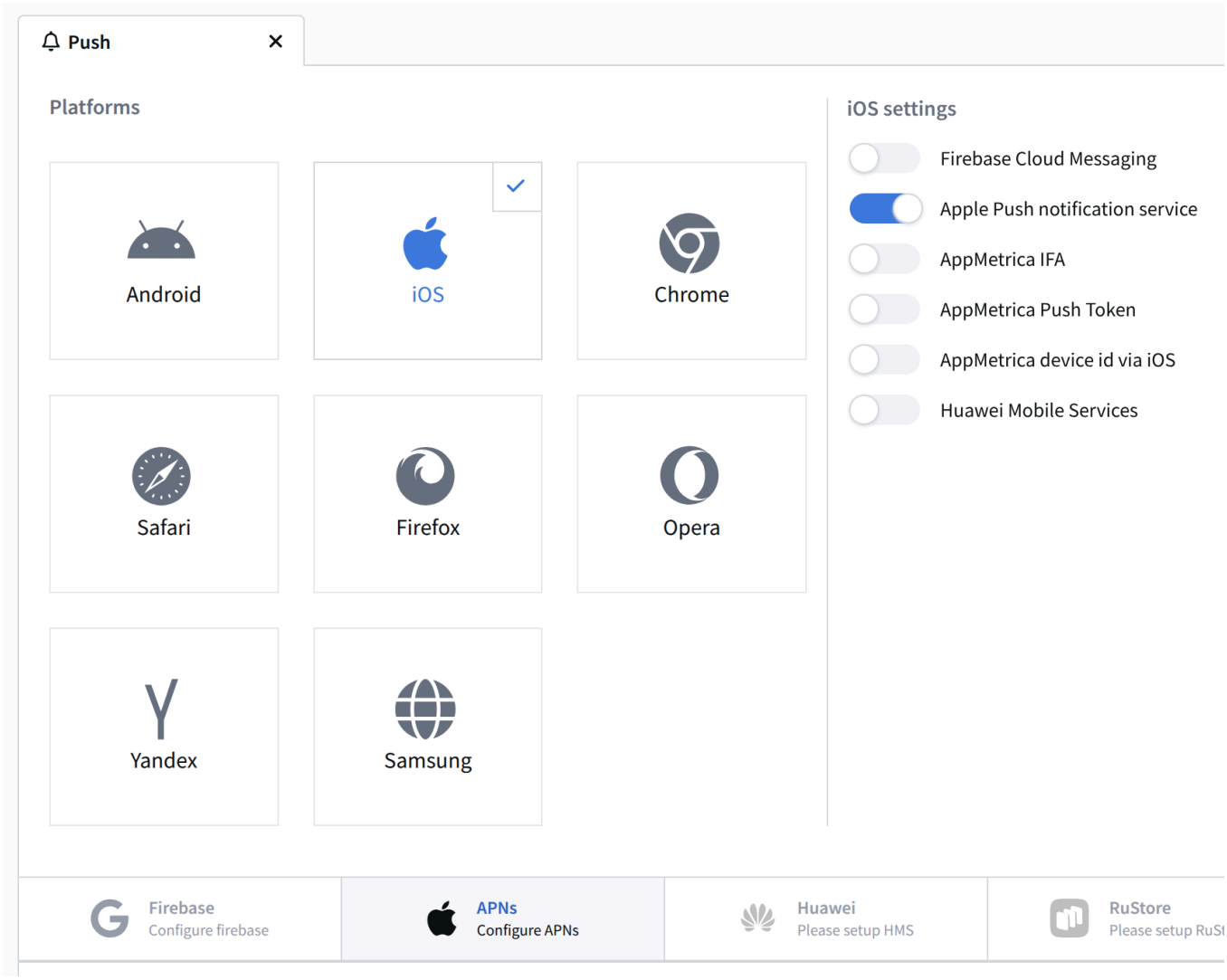


Apple Developer allows the created key to be downloaded only once. Make sure to store it securely. Af again:



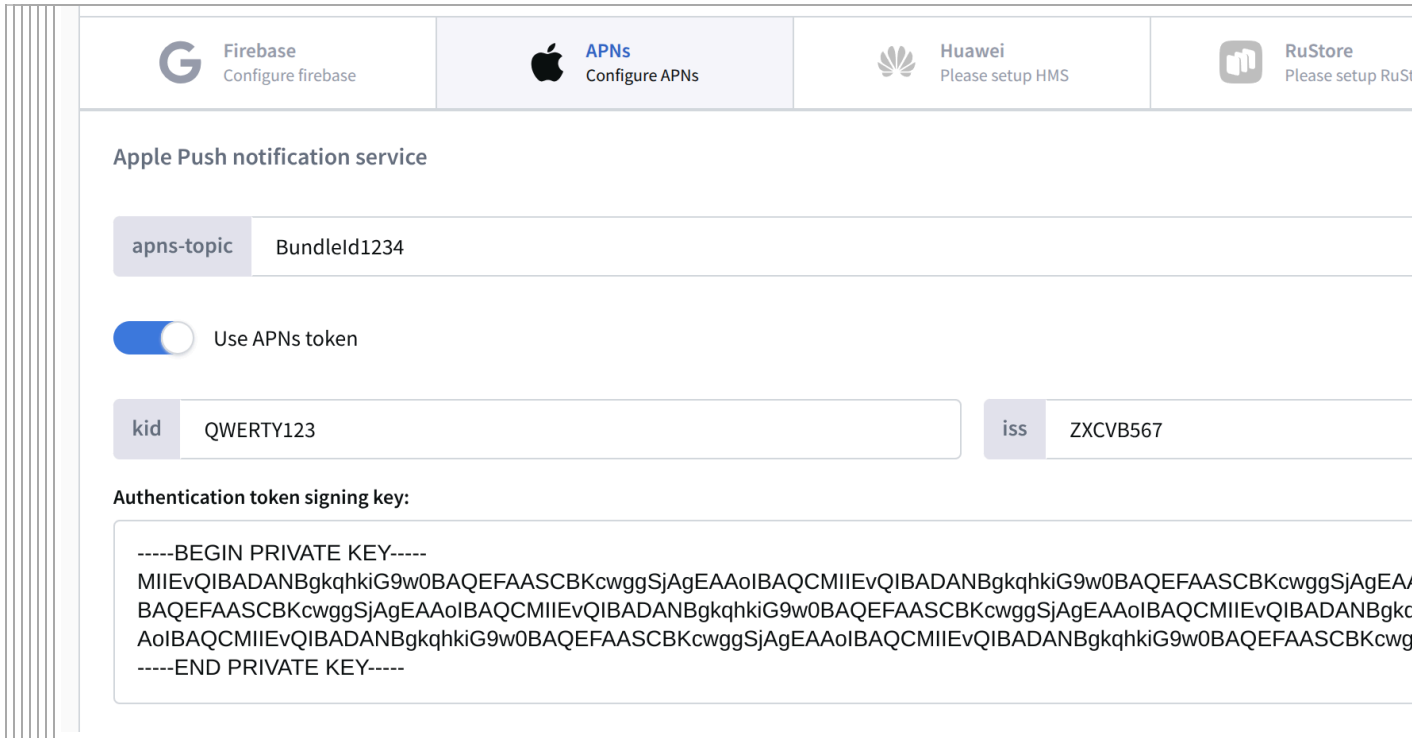
Step 2. Altcraft resource configuration

Configure the resource in the platform. Add a Push channel and enable Apple Push Notification service



Enable Use APNs token and fill in the following fields:

- **apns-topic — the app Bundle ID**
- **kid — the key ID created in the previous step. It is available in Certificates, Identifiers & Profiles — website**
- **iss — the Team ID from the Apple Developer website**
- **Signing key — the created Auth Key. To add it, open the downloaded p8 file in a text editor and cop**

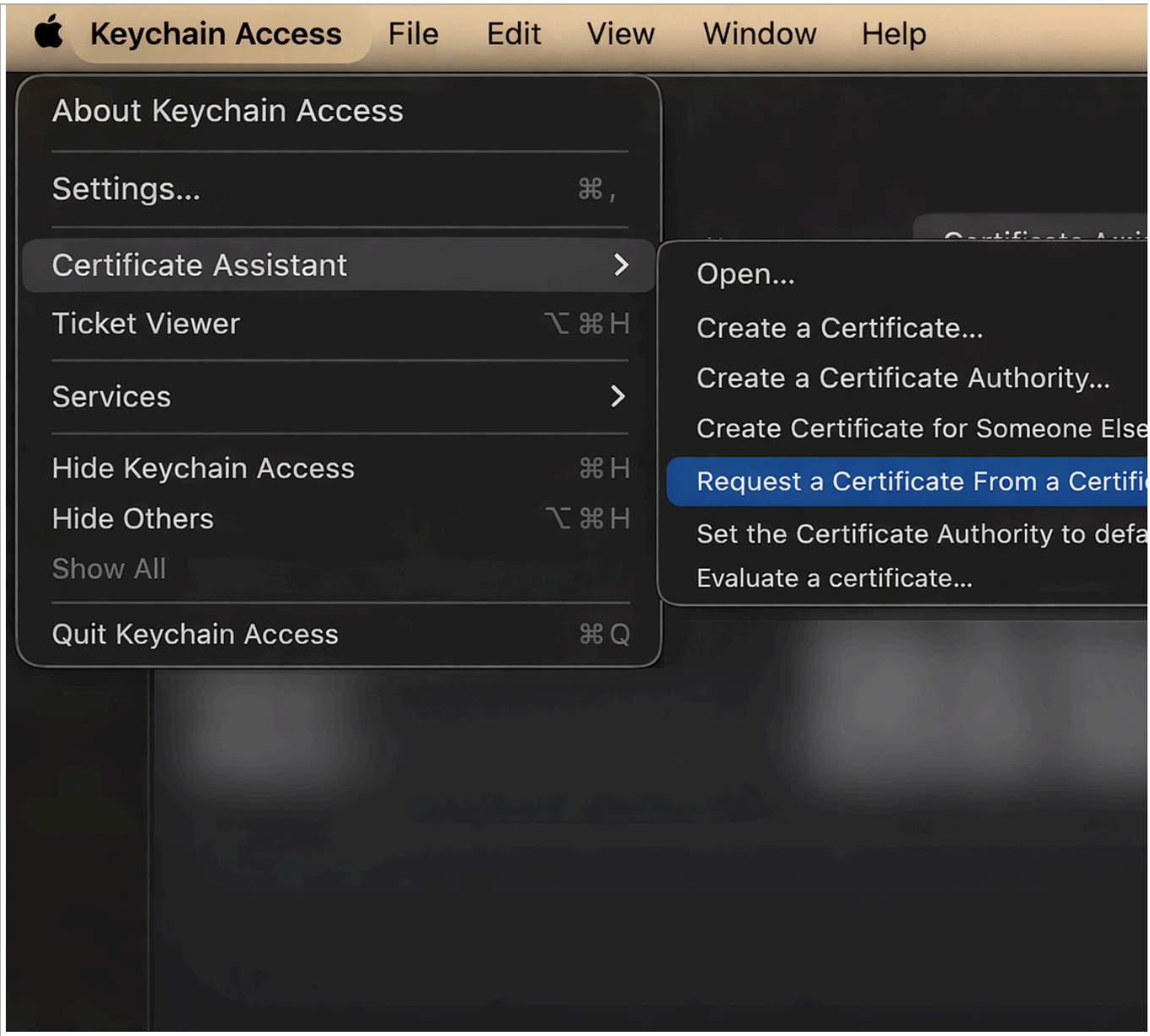


Save the resource. Provider configuration is complete. Next, follow the SDK initialization guide.

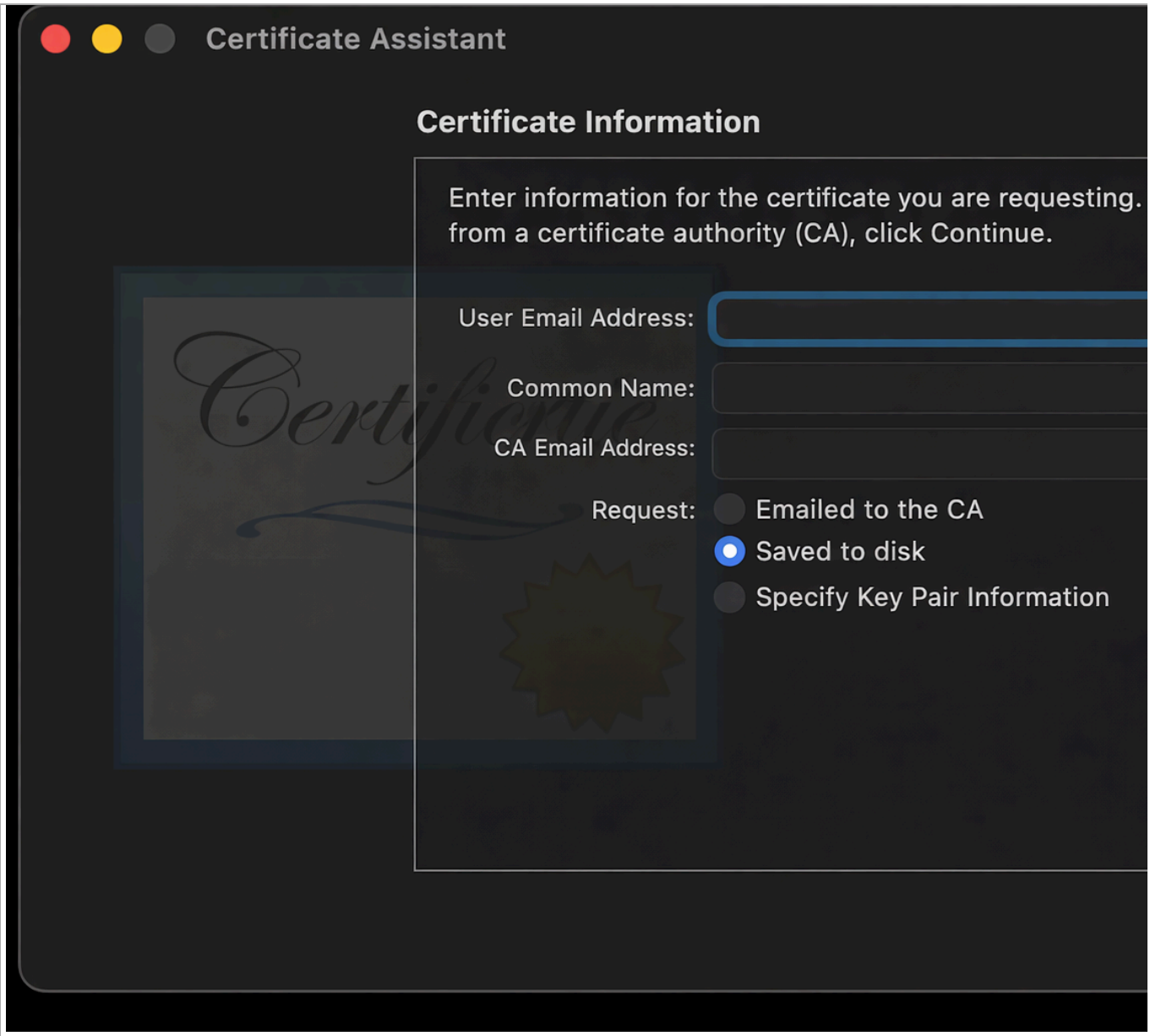
Option 2: using a certificate

Step 1. APNS configuration

Request a certificate from the Keychain Access certificate authority:

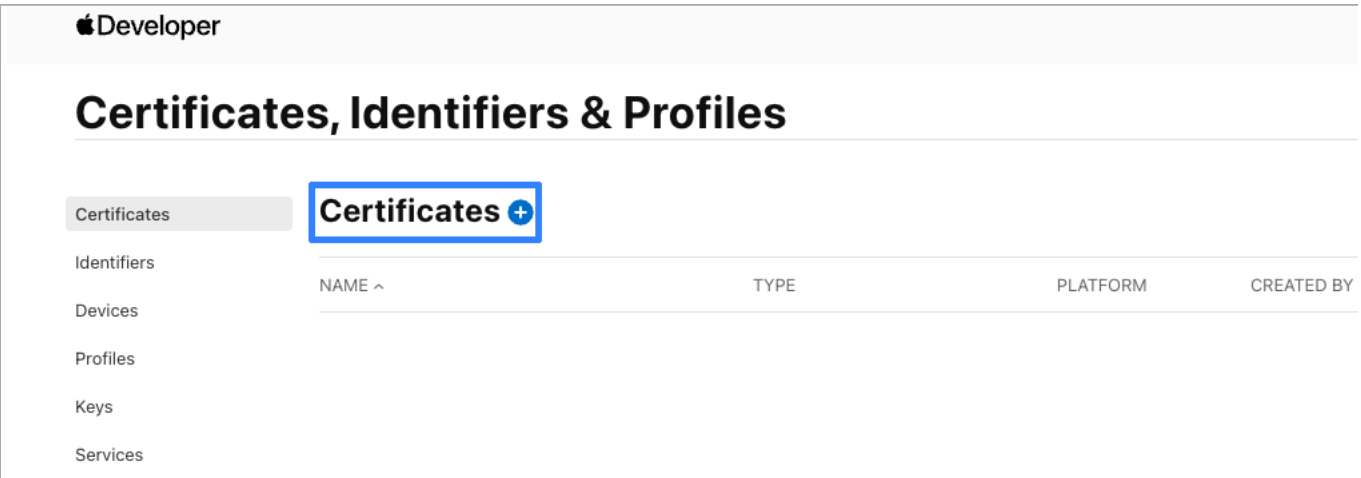


Fill in the required fields and save the file to disk:

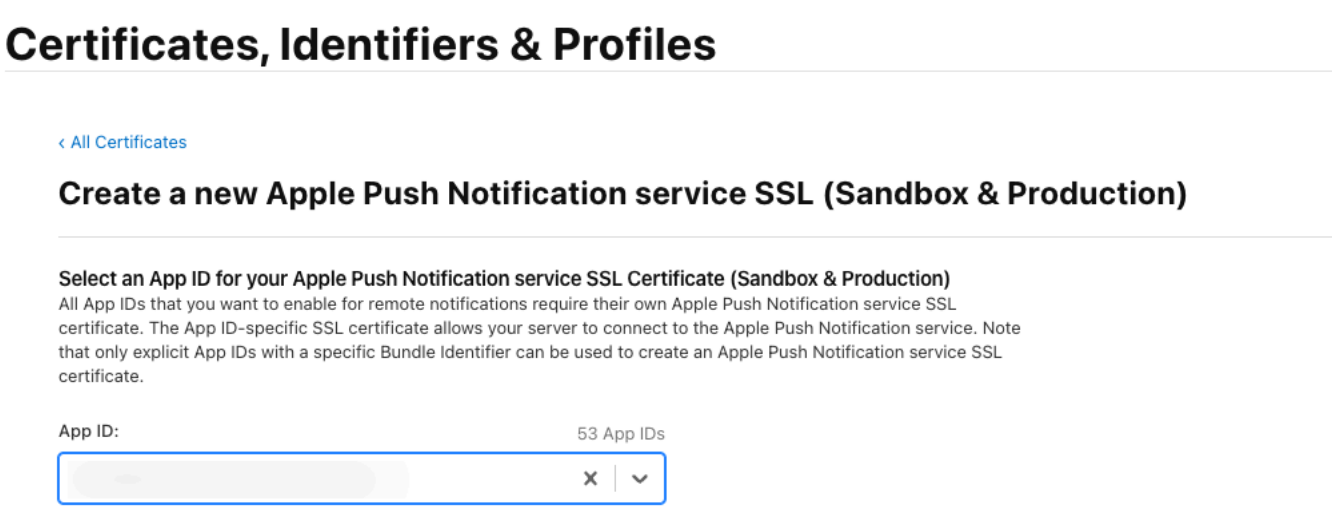


The saved .certSigningRequest file will be needed at the next step.

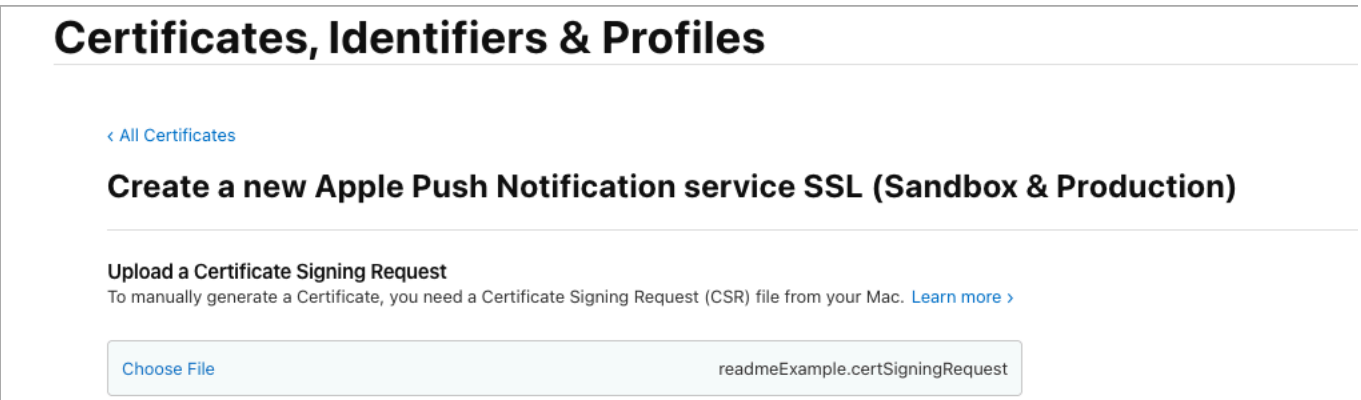
Go to the Apple Developer website, section [Account](). Navigate to Certificates, Identifiers & Profiles — certificate:



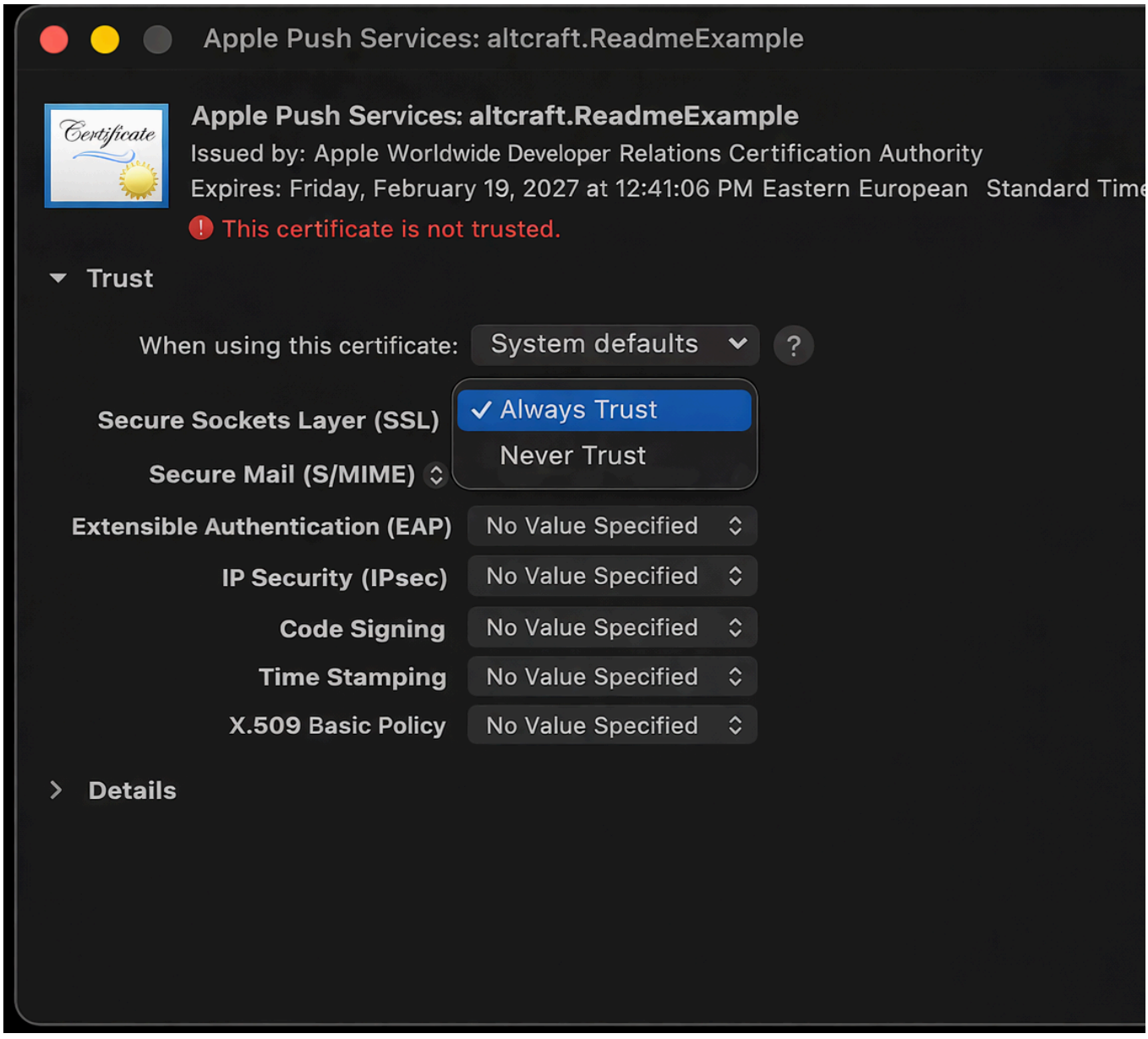
In Services, select Apple Push Notification service SSL (Sandbox) or (Sandbox and Production) depend choose your app from the list:



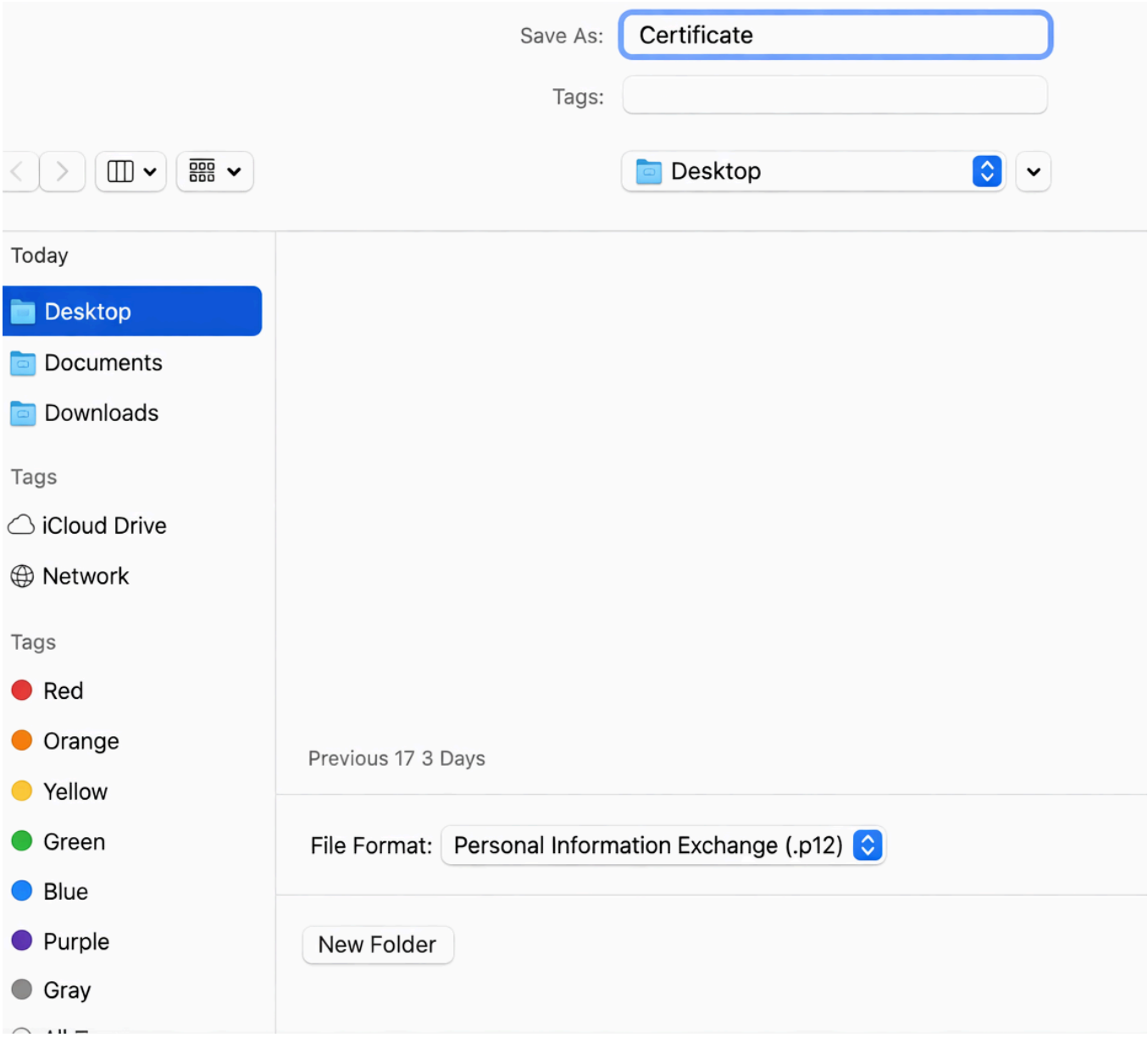
Upload the previously created Certificate Signing Request (.certSigningRequest file):



**Download the created certificate. Open the downloaded file in Keychain Access. Double-click the certi
Always Trust:**



Right-click the certificate and export it as a .p12 file. Set a password if required:



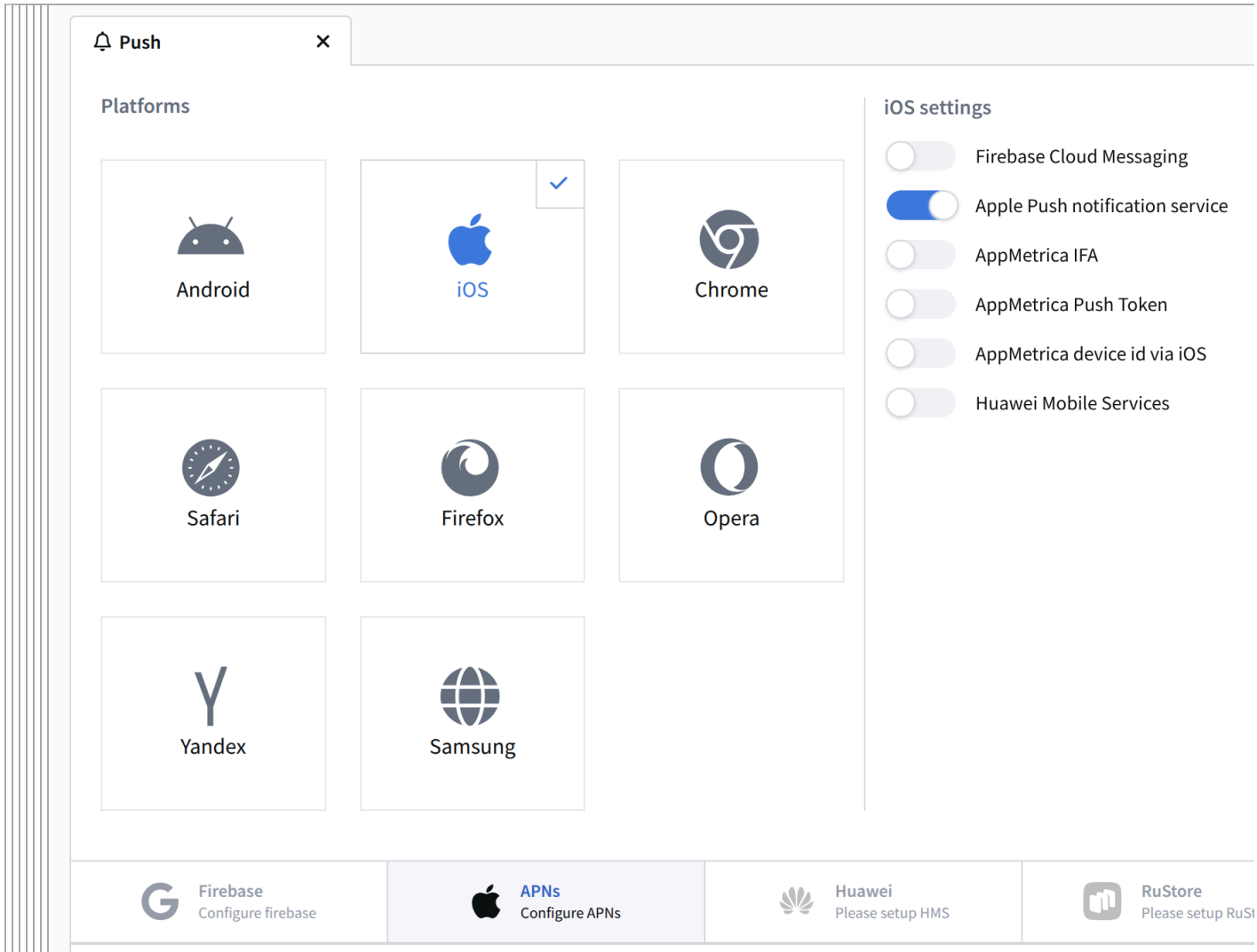
Using Terminal, navigate to the folder containing the exported certificate. Create a pem key using the `filename.p12 -out filename.key.pem -nocerts -nodes`**, where** `filename.p12` **is the exported file name and** `filename.` **PEM key file.**

Without closing Terminal or changing directories, create a pem certificate using `openssl pkcs12 -legacy -` `clcerts -nokeys`**, following the same naming logic (** `filename.p12` **as the source file,** `filename.crt.pem` **as the o**

Open both files in Terminal using nano. Their contents will be required in the platform fields at the nex

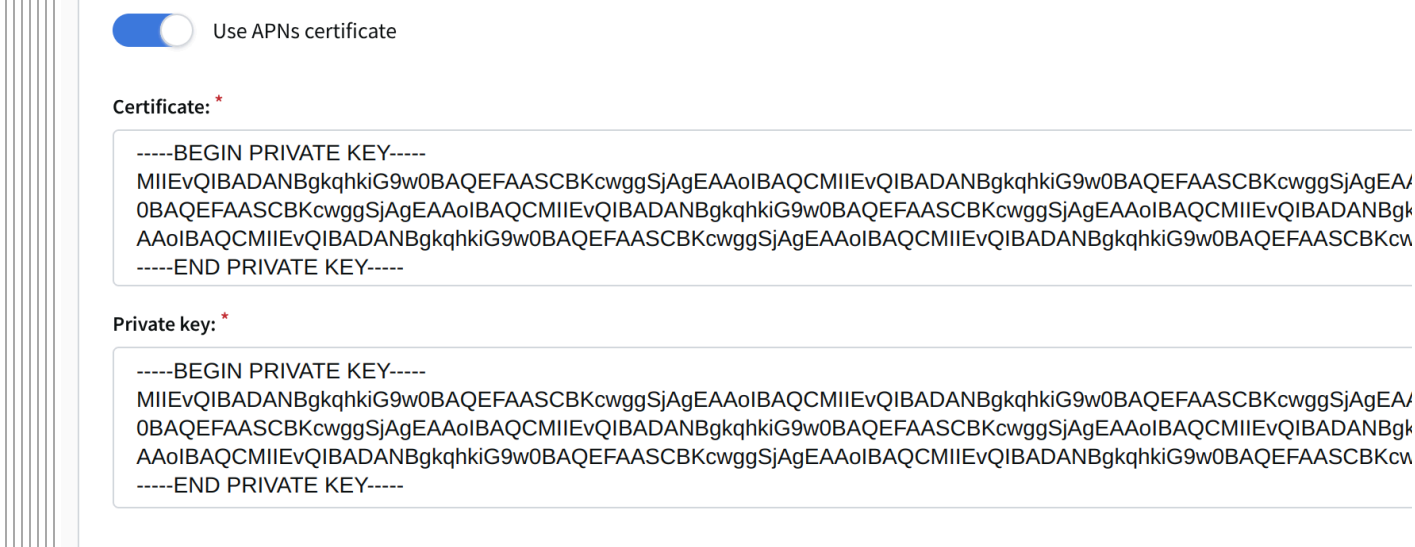
Step 2. Altcraft resource configuration

Configure the resource in the platform. Add a Push channel and enable Apple Push Notification service



Enable Use APNs certificate and fill in the following fields:

- **apns-topic — the app Bundle ID**
- **Certificate — the certificate file contents**
- **Private key — the certificate private key value**

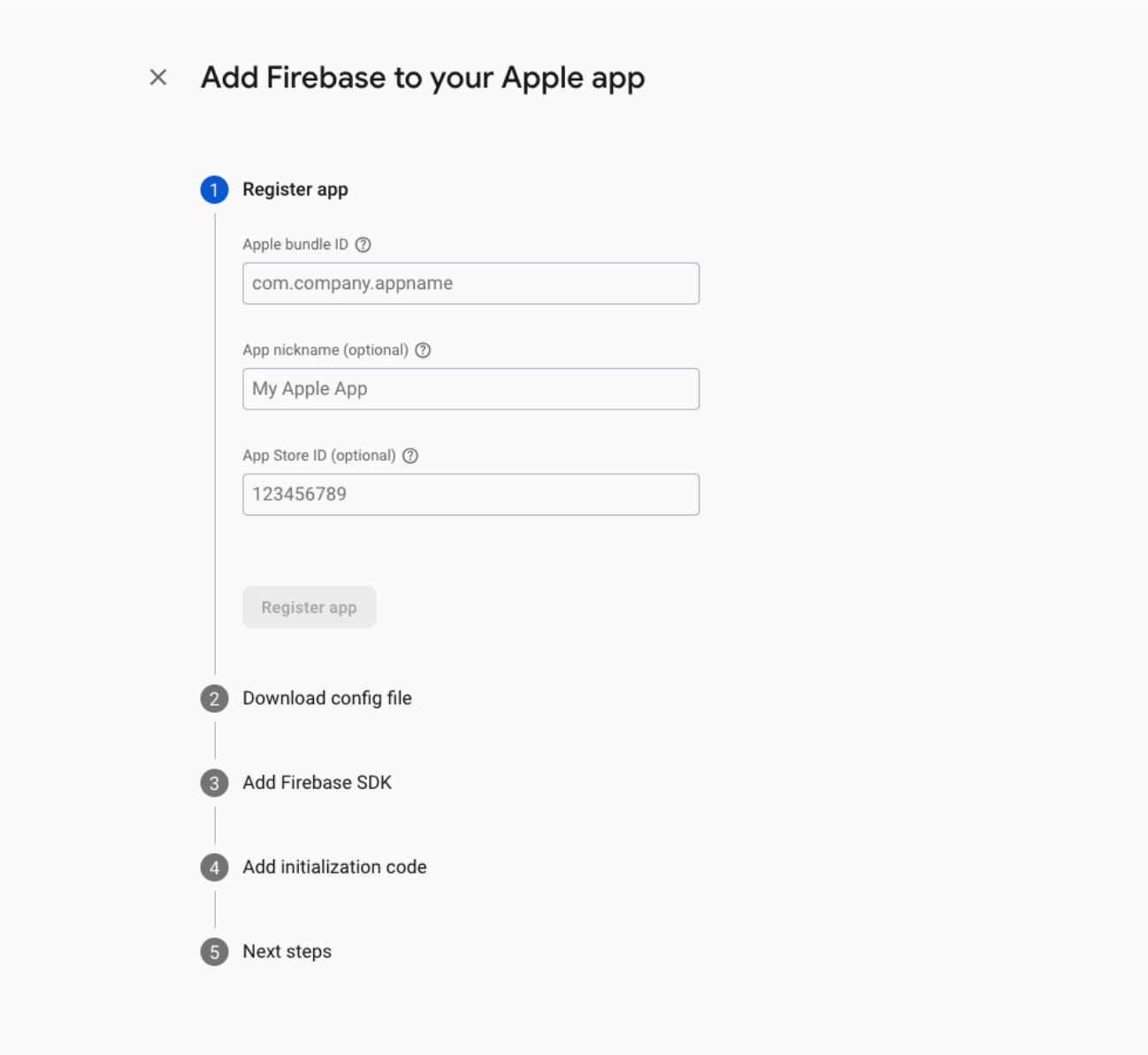


Save the resource. Provider configuration is complete. Next, follow the SDK initialization guide.

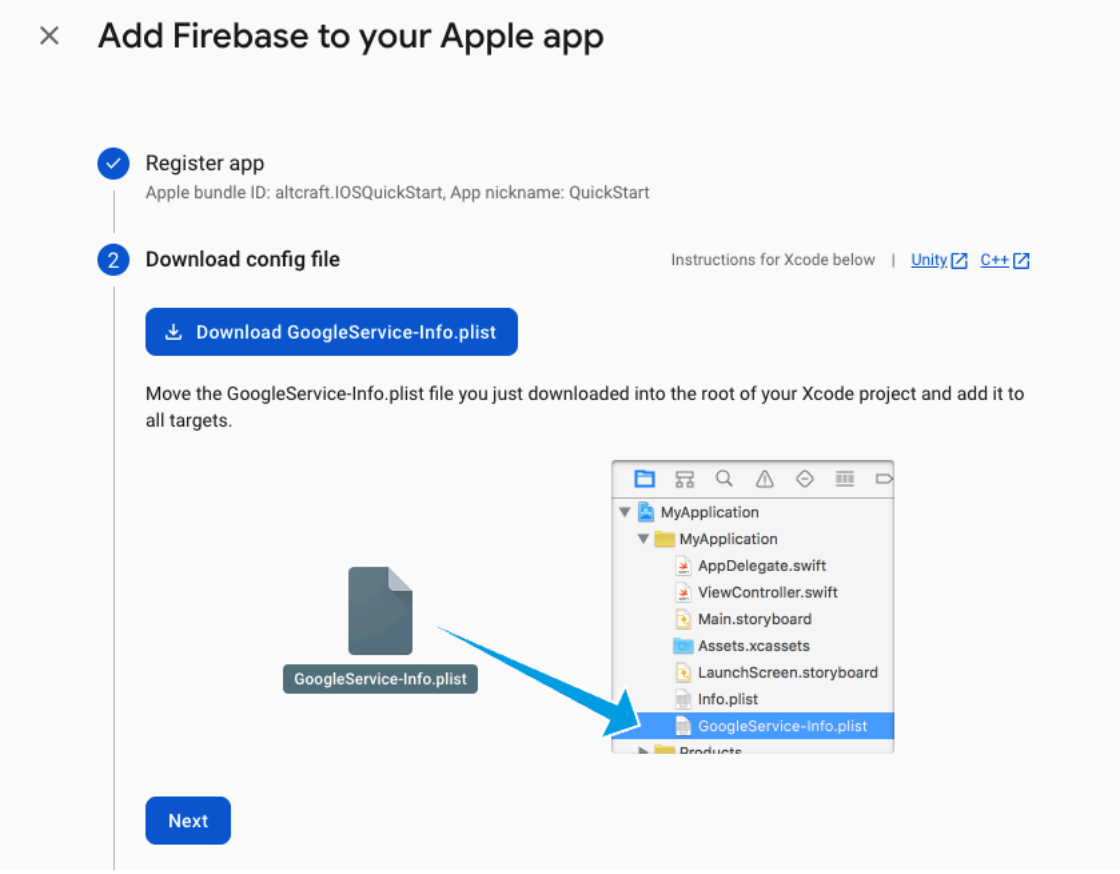
Firebase Cloud Messaging

Step 1. Firebase Cloud Messaging project setup

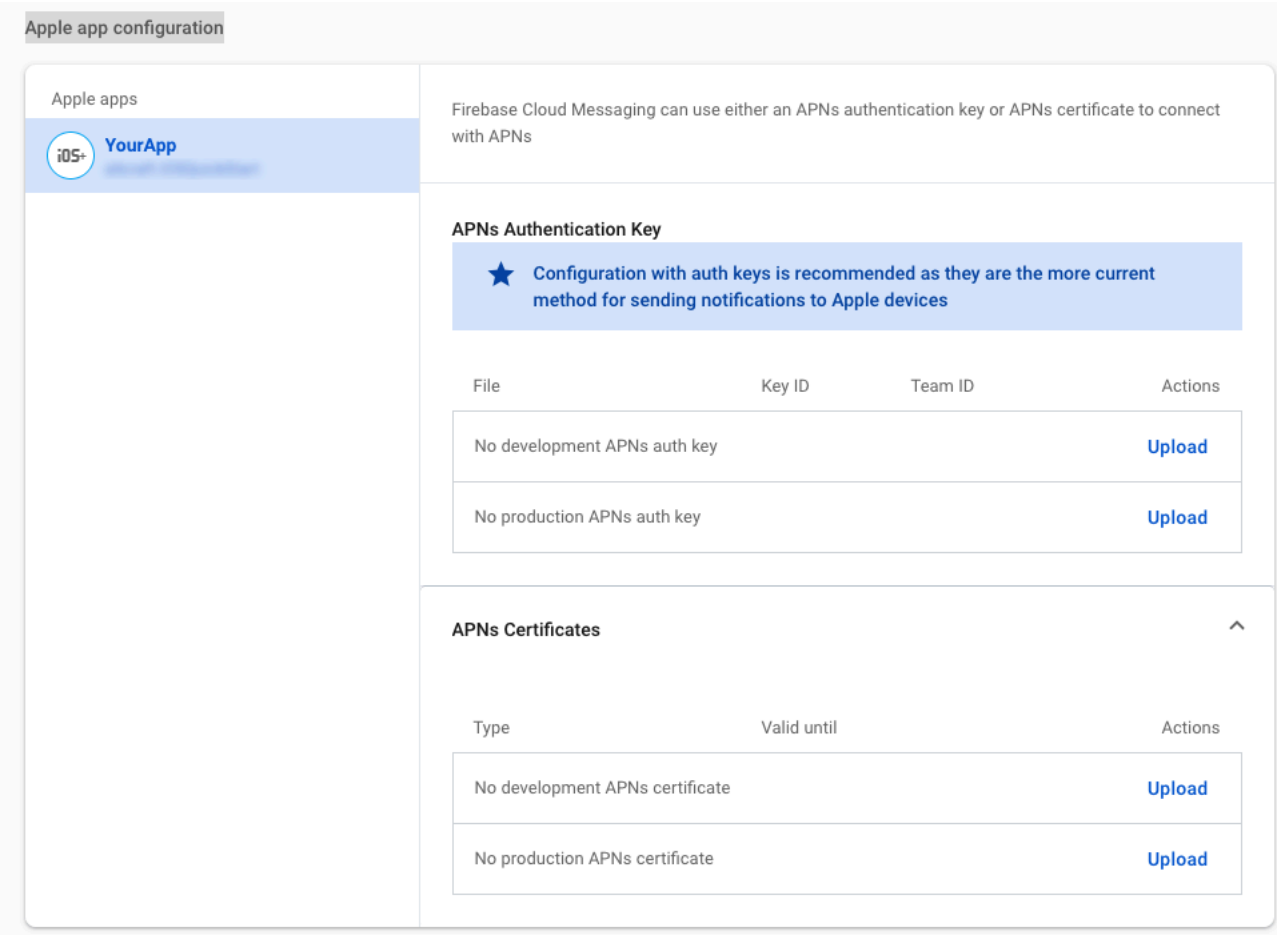
- **Create a [Firebase] project.**
- **Register a new app in the project:**



- **Download the `GoogleService-Info.plist` file.**

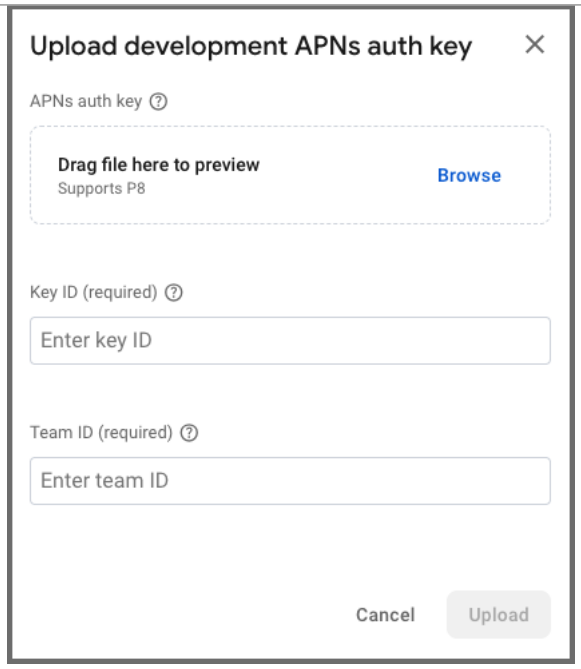


- **Go to Project Settings —> Cloud Messaging —> Apple app configuration.**
- **Add the required APNs Authentication Key data (the key must first be created in [Apple Developer](#)):**



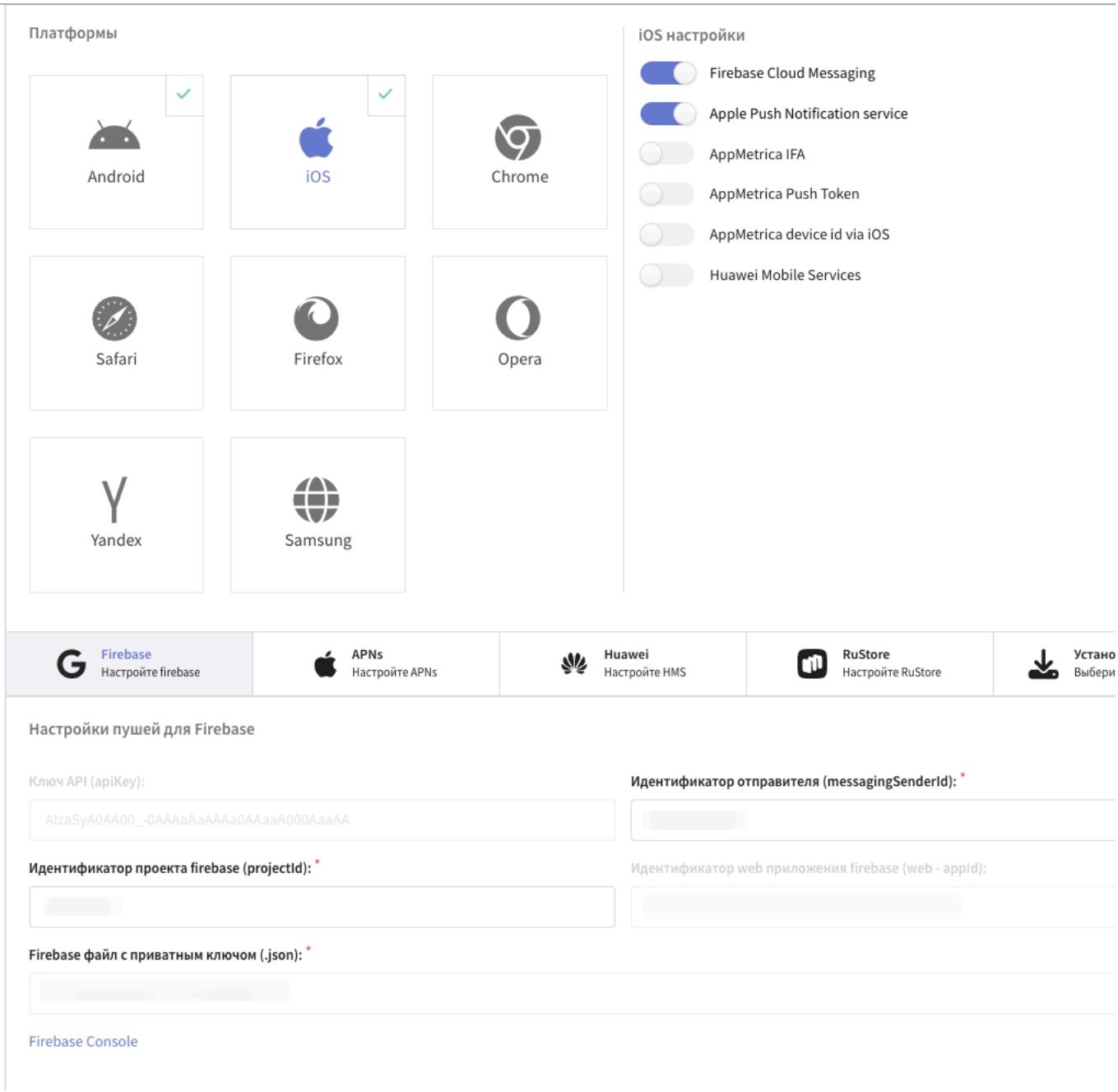
Enter the following data:

- **upload the `.p8` key file**
- **specify the `key ID`**
- **specify the `team ID`**



Step 2. Altcraft resource configuration

- **specify the** `messagingSenderId`
- **specify the** `projectId`
- **upload the Firebase private key file (.json). The file is created in Project settings —> Service accou**



Step 3. Integrating Firebase Cloud Messaging into the app project

- **Move the downloaded `GoogleService-Info.plist` file into the application folder (app).**
- **Add the Firebase package using Swift Package Manager — repository [Firebase](). The Firebase pack with your Xcode version.**
- **General:**
  - **Ensure that the FirebaseMessaging library is added under Frameworks, Libraries, and Embedd**
- **Signing & Capabilities:**
  - **PushNotifications**

Option 1: FirebaseAppDelegateProxyEnabled — YES (Default for Firebase Messaging)

1. **Add the `MessagingDelegate` protocol to `AppDelegate`:**

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate {}
```

This allows receiving a callback with the current FCM token through the `messaging(_:didReceiveRegistrationTo`

2. **Initialize Firebase in `didFinishLaunchingWithOptions`:**

```
FirebaseApp.configure()
Messaging.messaging().delegate = self
application.registerForRemoteNotifications()
```

3. **Implement the delegate method to receive the token:**

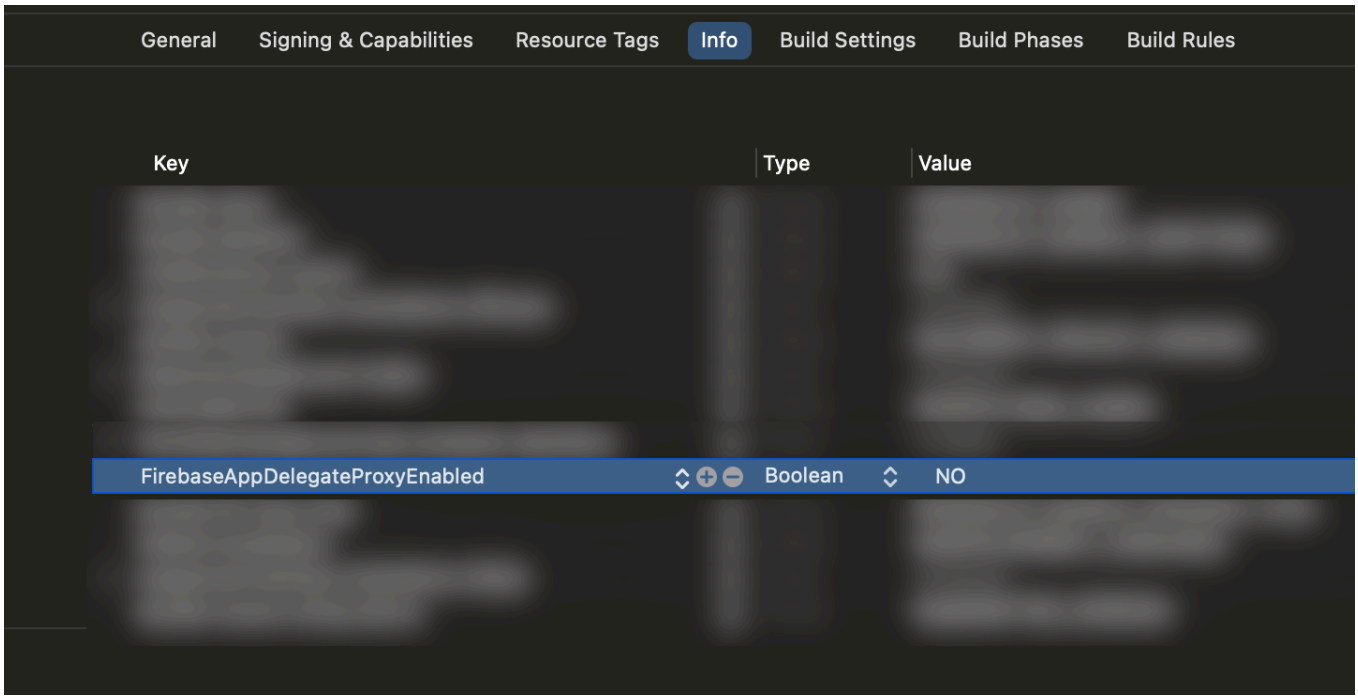
```
    func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
        guard let token = fcmToken else { return }
        print(" FCM Token: \(token)")
    }
```

Full example of Firebase initialization and FCM token retrieval

```
@main
struct IOSQuickStartApp: App {
   @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate

   var body: some Scene {
       WindowGroup {
           ContentView()
       }
   }
}

class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
   func application(
       _ application: UIApplication,
       didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
   ) -> Bool {
       /// Initialize Firebase
       FirebaseApp.configure()
       /// Set delegate
       Messaging.messaging().delegate = self
       application.registerForRemoteNotifications()

       return true
   }

   func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
       if let token = fcmToken {
          //FCM token
       }
   }
}
```

Option 2: FirebaseAppDelegateProxyEnabled - NO

1. **In the app target settings:**
    - **Info:**
        - **Add the key `FirebaseAppDelegateProxyEnabled (Boolean)` with the value `NO` in the app's Info.**



2. **The `MessagingDelegate` protocol is not required in this case. When `FirebaseAppDelegateProxyEnabled` is disab automatically swizzle `AppDelegate`, so the SDK handles APNs and FCM token setup and management ma therefore not needed.**
3. **Initialize Firebase in `didFinishLaunchingWithOptions`:**

```
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        FirebaseApp.configure()
        return true
    }
```

4. Retrieve and save the APNs token in `didRegisterForRemoteNotificationsWithDeviceToken`:

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        FirebaseApp.configure()
        return true
    }

    func application(
        _ application: UIApplication,
        didRegisterForRemoteNotificationsWithDeviceToken apnsToken: Data
    ) {
        // Save the token for later use when requesting the FCM token
        setAPNsTokenInUserDefault(apnsToken)
    }
}
```

Step 4. Manually setting the APNs token in Firebase Messaging before requesting the FCM token

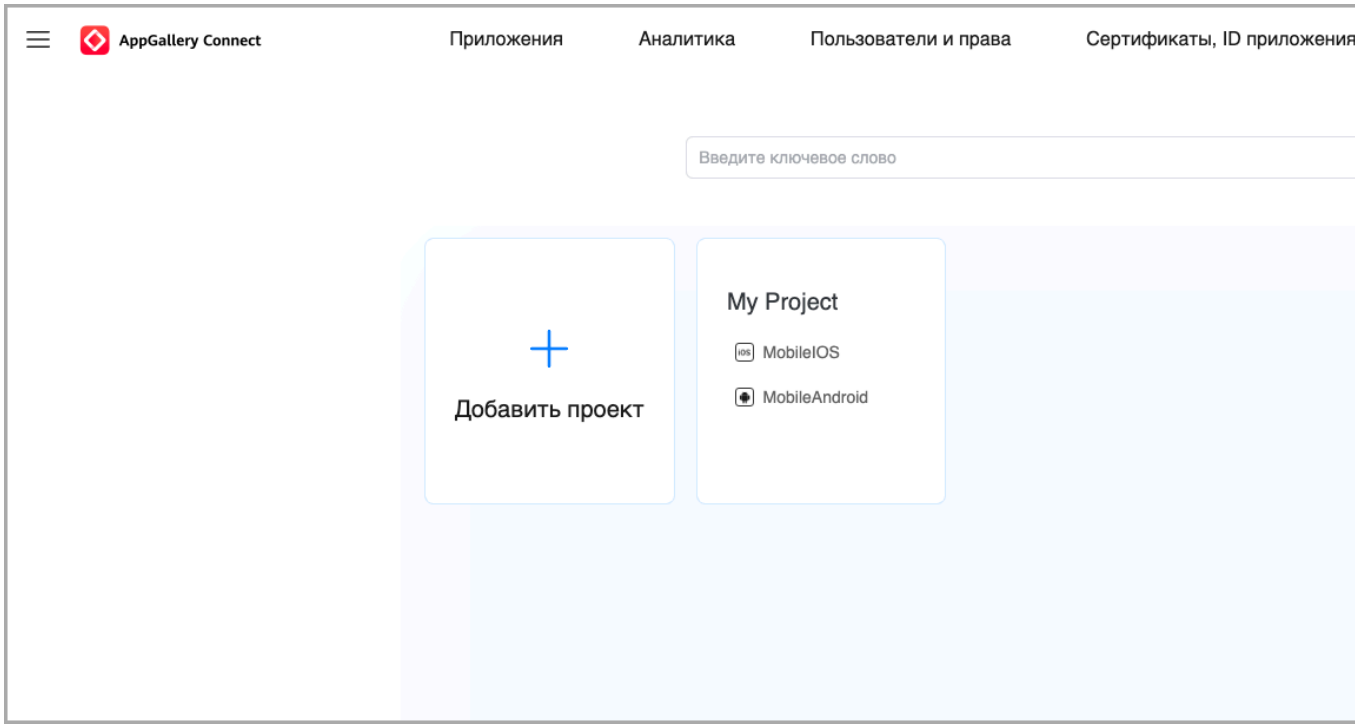
Example inside SDK provider

```
class FCMProvider: FCMInterface {

    /// Retrieves the current FCM token
    func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// Manually set the APNs token before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// Request FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

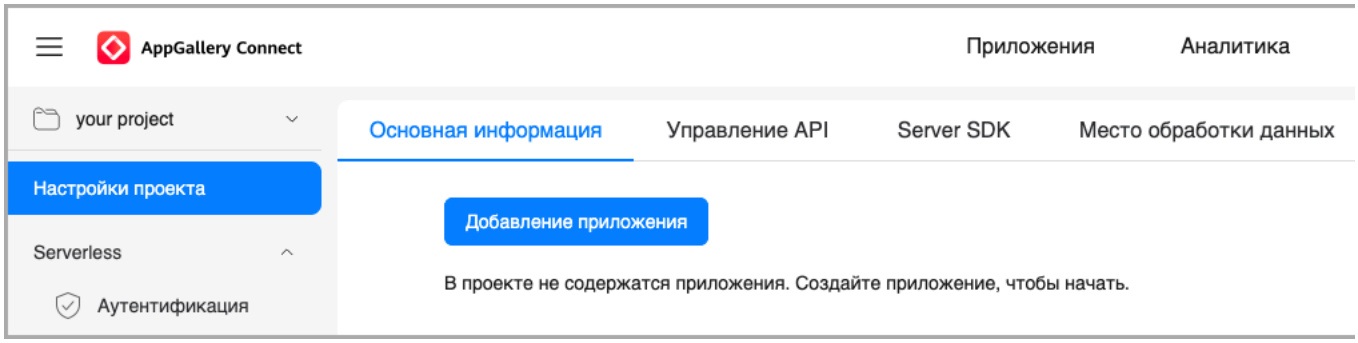
Huawei Mobile Services

Step 1. Huawei Mobile Services project setup

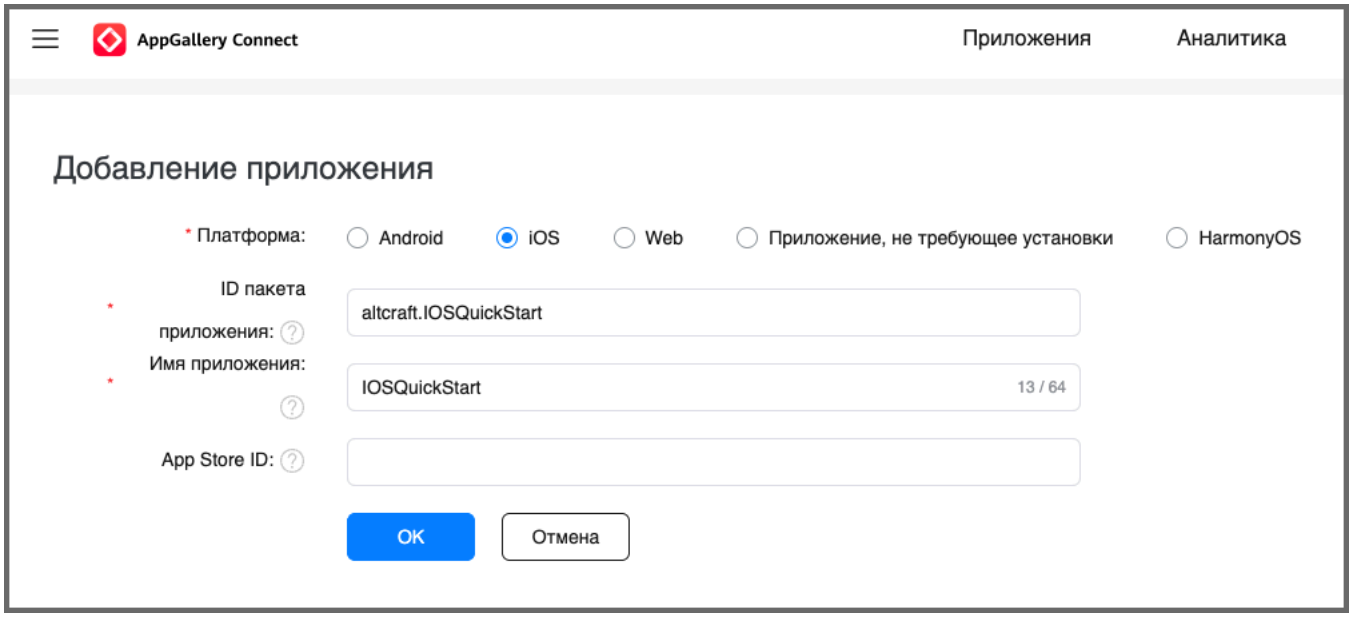
Create an [AppGallery Connect] project. Go to Develop and Services —> Add project:



Add a new app to the project:



Provide your app information:



Download the `agconnect-services.plist` file:

Настройка SDK

Добавьте базовый пакет SDK к вашему приложению.Подробнее

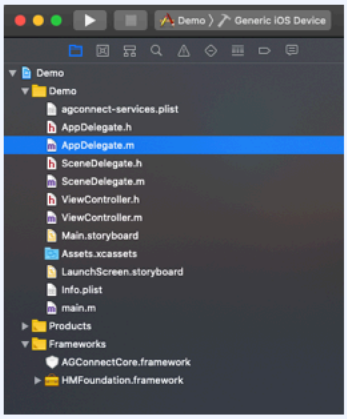
Шаг 1 Добавление файла конфигурации

(Пример ниже применим к Xcode.)

agconnect-services.plist

a) Загрузите файл agconnect-services.plist.

b) Добавьте загруженный файл agconnect-services.plist в проект Xcode.



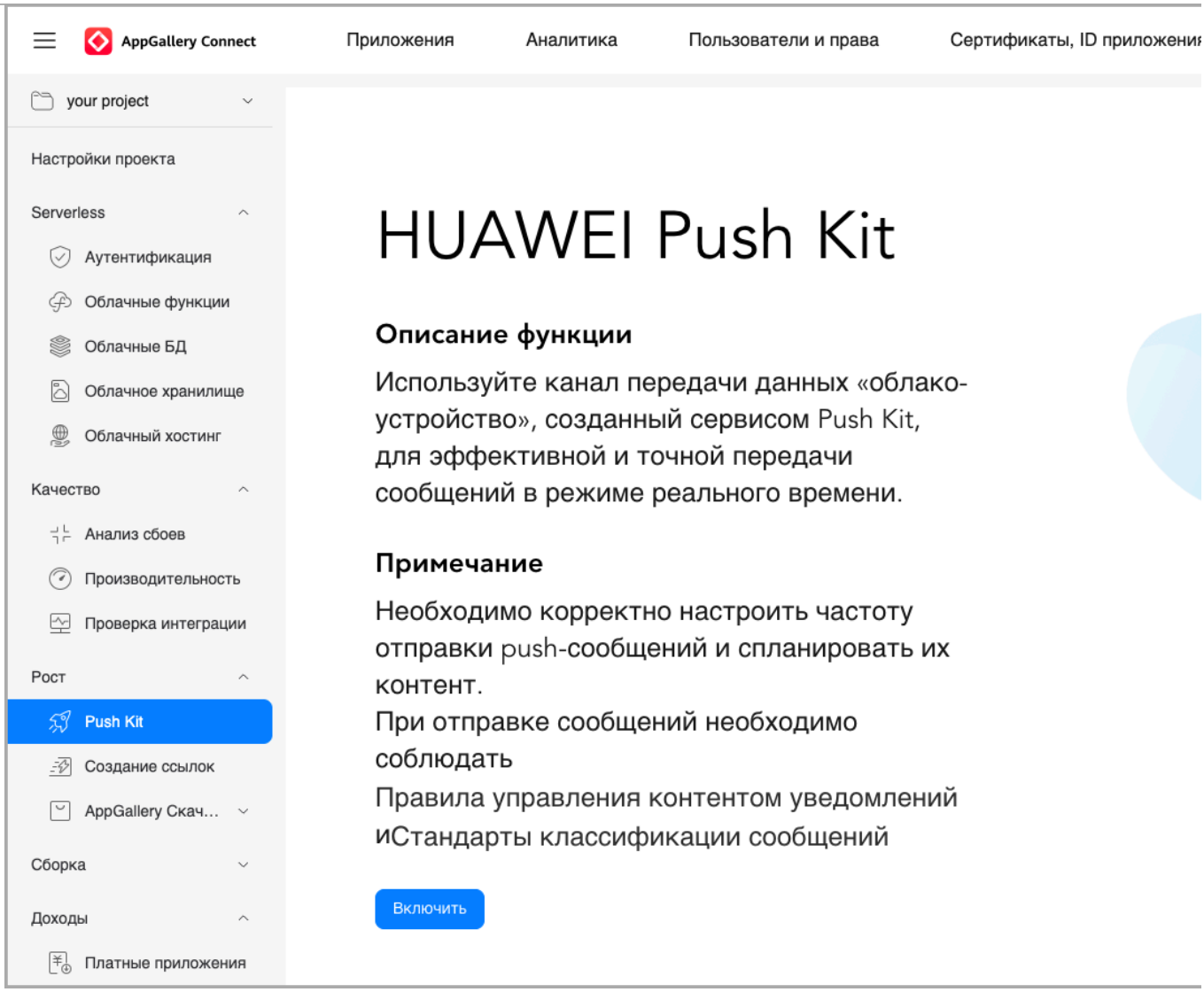
Далее Отмена

Шаг 2 Добавление SDK

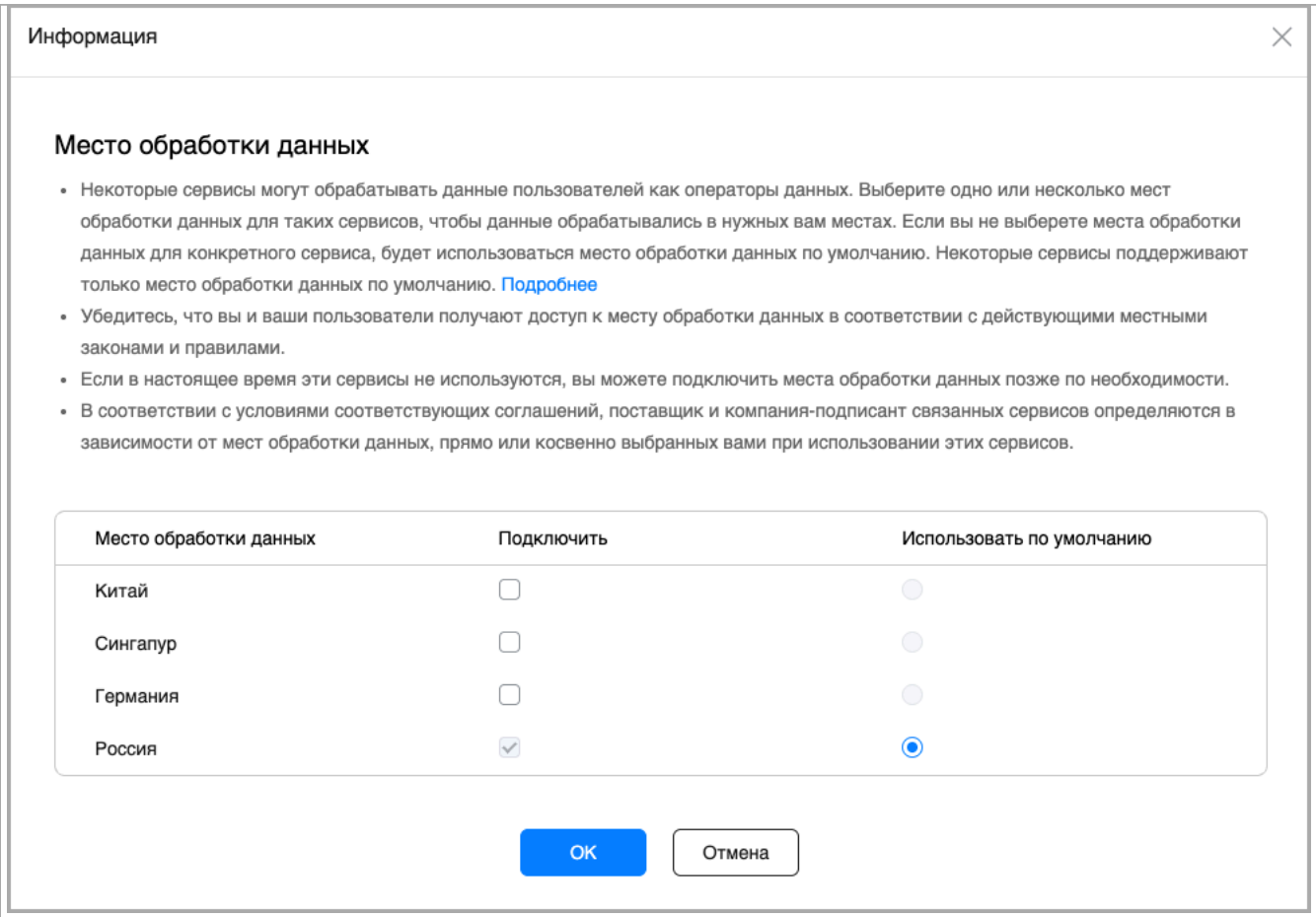
(Пример ниже применим к CocoaPods. Для подробной информации о других способах разработки см.Руководство.)

Шаг 3 Интеграция сервисов

In the left sidebar, under Grow, select Push Kit:

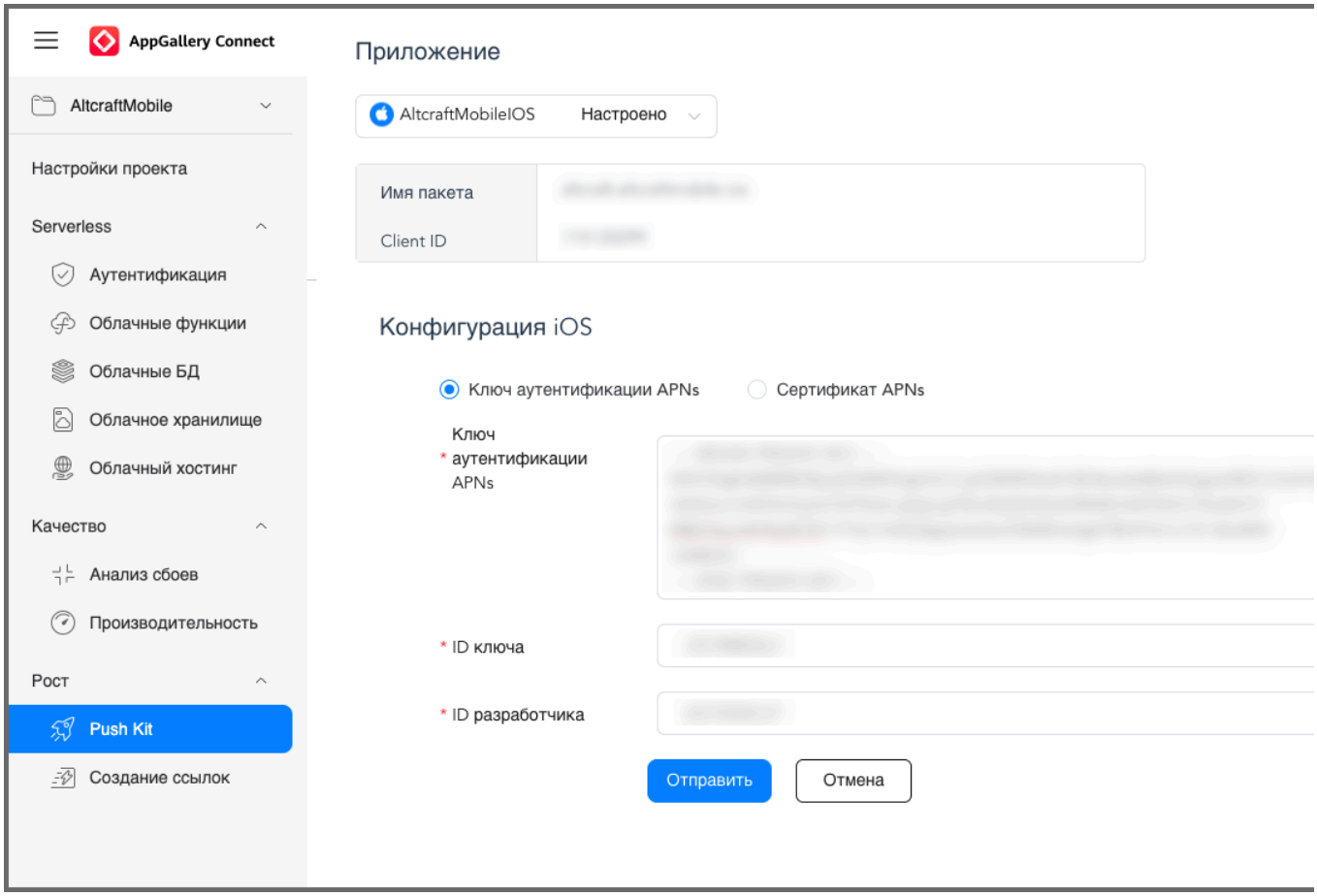


Click Enable —> Select data storage location:



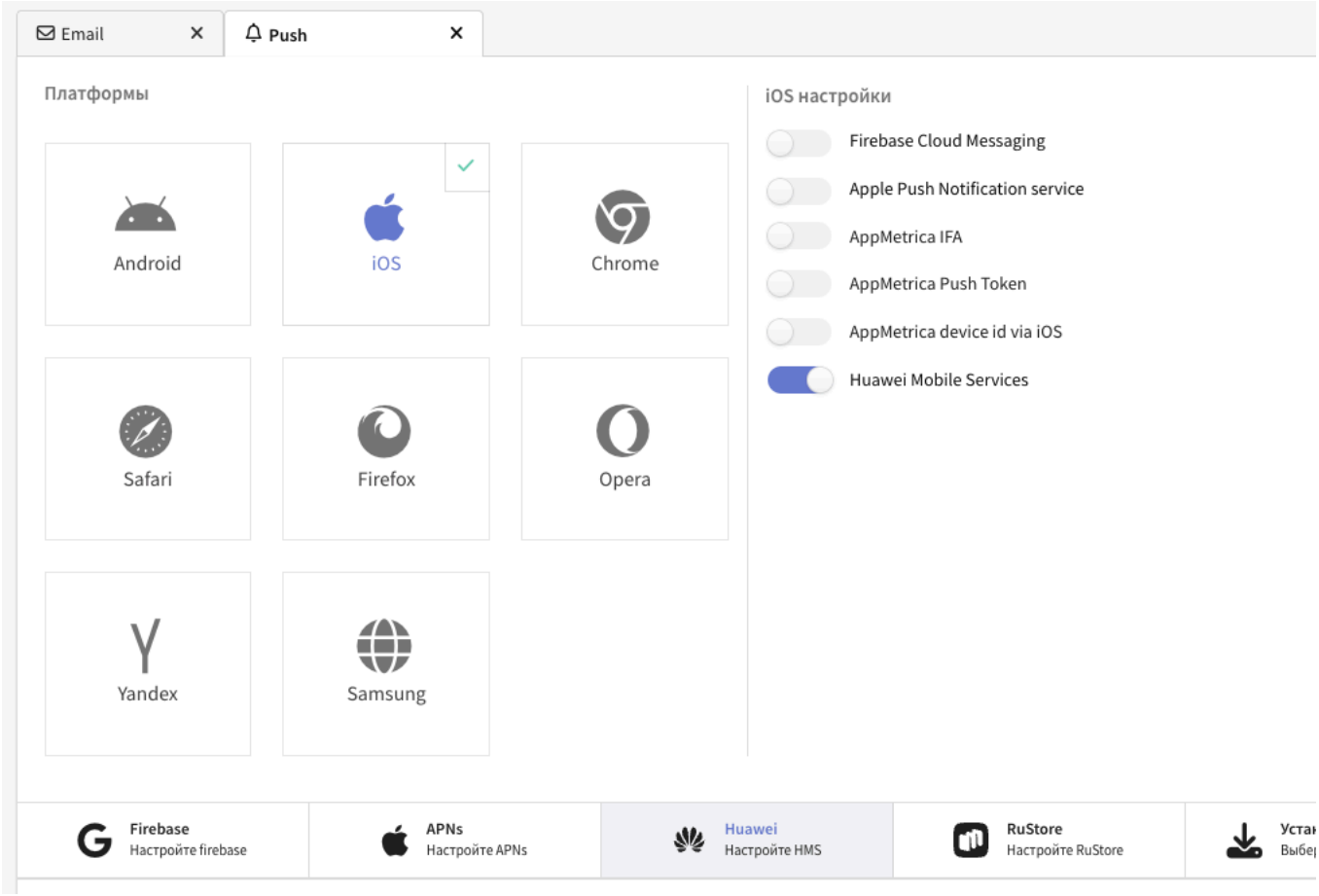
Then go to Push Kit —> Settings —> App —> iOS configuration —> Enable. In the opened iOS configurat

```
- **APNs Authentication Key** – the contents of the private .p8 key file opened in a text editor;
- **Key ID** – the `keyID` of the private .p8 key file (usually matches the file name – the value after `AuthKey_`);
- **Developer ID** – the `teamID` from Apple Developer.
```

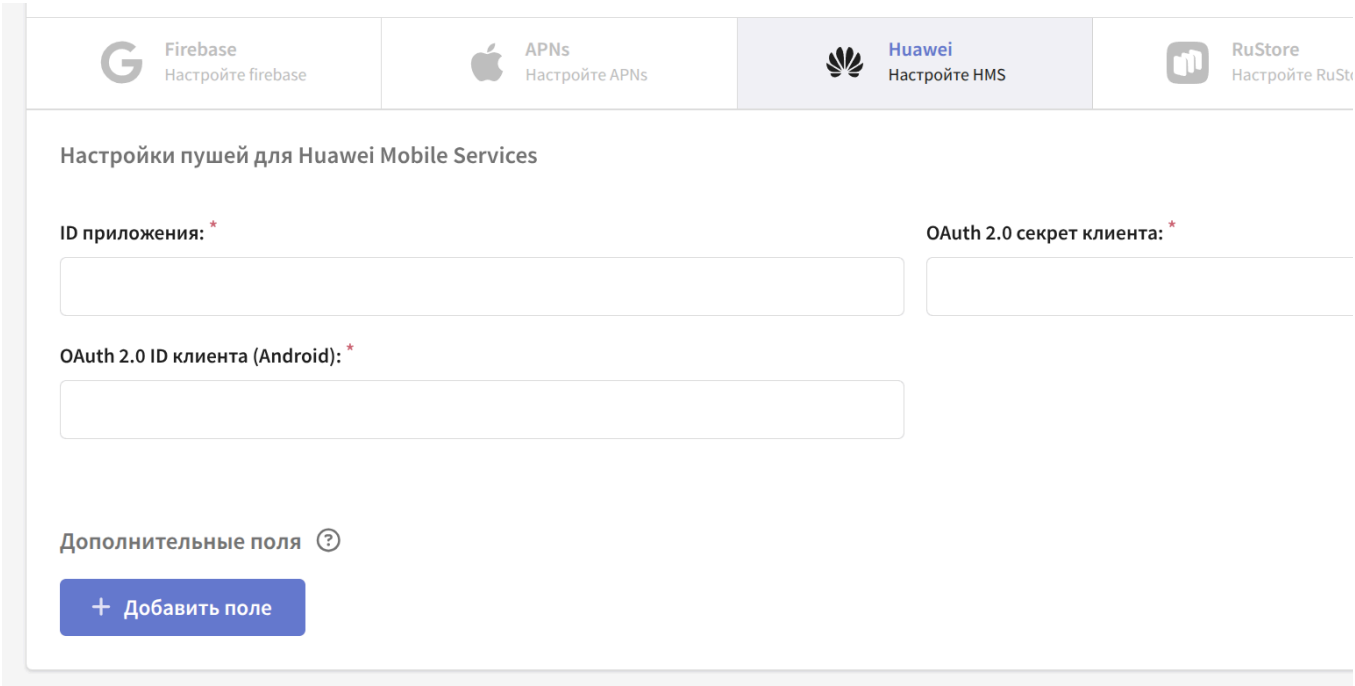


Step 2. Altcraft resource configuration

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existi
settings, select the iOS platform and activate the Huawei Mobile Services switch:



Scroll down to Huawei settings and enter the App ID, OAuth 2.0 client ID, and OAuth 2.0 client secret:



Note that the App ID is the Client ID from the App information section.
After making changes, don't forget to save the resource.

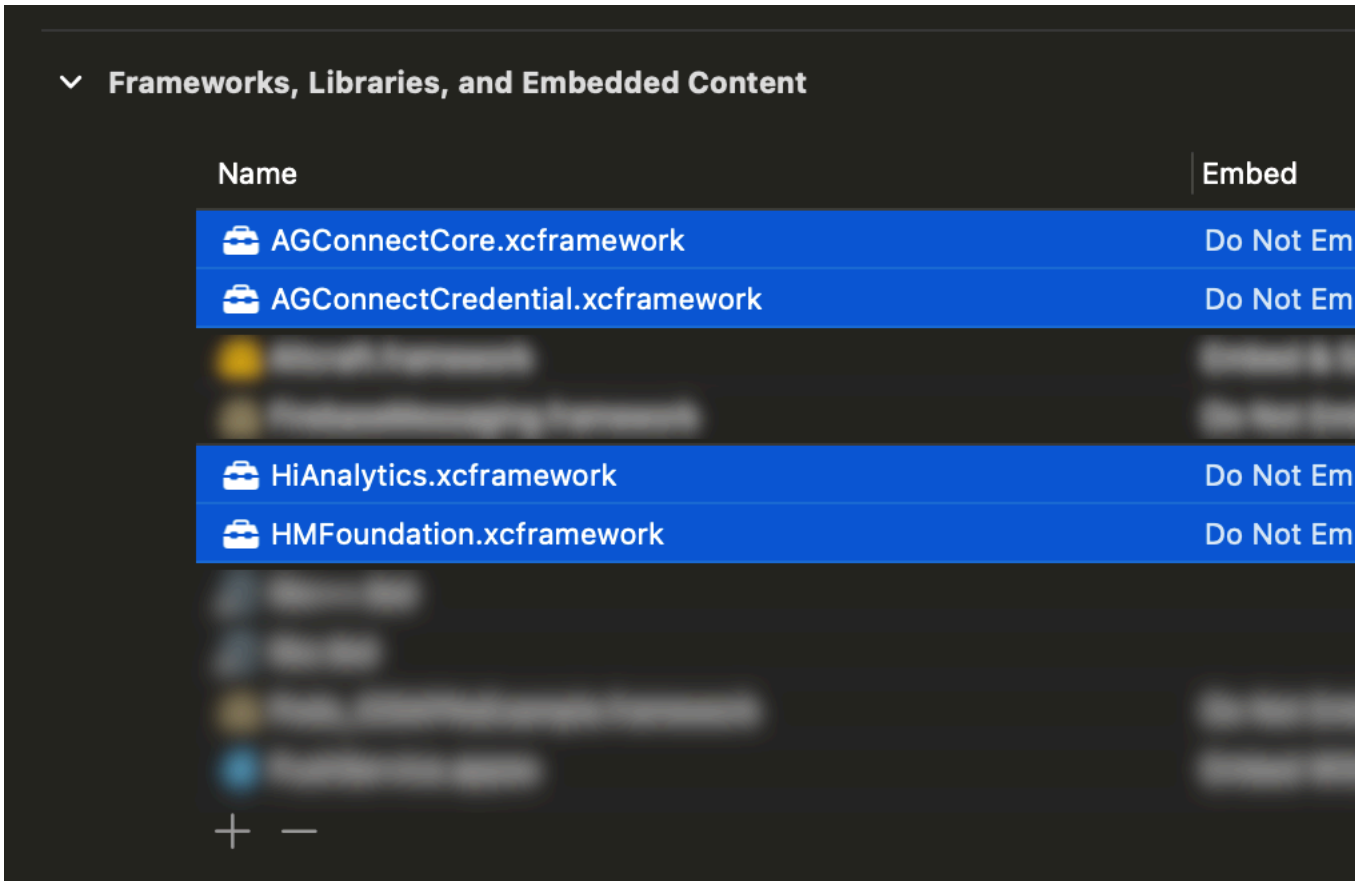
Step 3. Integrating Huawei Push Kit into the app project

Move the downloaded `agconnect-services.plist` **file to the root directory of the app module (app). Then ad**
Configure the app target:

- **General — Frameworks, Libraries, and Embedded Content**

Make sure the required frameworks are added and set to Do Not Embed:

- `AGConnectCore.xcframework`
- `AGConnectCredential.xcframework`
- `HiAnalytics.xcframework`
- `HMFoundation.xcframework`



- **Signing & Capabilities**

Add Push Notifications.

Step 4. Obtaining the HMS token

Get the HMS token using the `HmsInstanceId.getInstance().getToken(apnsToken)` function, passing the APNs tok

Example of obtaining and deleting an HMS token in an SDK interface implementation

```
// Be sure to import HmsPushSdk in the files where you use HMS functions

class HMSProvider: HMSInterface {

    /// Retrieves the current HMS token using the APNs token
    func getToken(completion: @escaping (String?) -> Void) {
        guard let apnsToken = getAPNsTokenFromUserDefault() else {
            completion(nil)
            return
        }

        let token = HmsInstanceId.getInstance().getToken(apnsToken)
        completion(token)
    }


    func deleteToken(completion: @escaping (Bool) -> Void) {
        HmsInstanceId.getInstance().deleteToken()
        completion(true)
    }
}
```

Quick Start

This article provides an example of quick integration of Altcraft mSDK for React Native.

Step 0. Prerequisites

- **A React Native project is created and runs on iOS/Android.**
- **Native push providers are integrated into the project:**
  - **Android: FCM, HMS, RuStore**
  - **iOS: APNS, FCM, HMS**
- **Altcraft mSDK is configured on the native side:**
  - **providers are registered (JWT / APNS / FCM / HMS / RuStore);**
  - **interception of incoming push notifications and their forwarding to the SDK is configured.**

Step 1. Package installation

```
npm i @altcraft/react-native-sdk
# or
yarn add @altcraft/react-native-sdk
```

Step 2. SDK configuration and initialization

Import the SDK and initialize it at app startup (for example, in `App.tsx`):

```
AltcraftSDK.initialize({
  apiUrl: 'https://pxl-your-instance.altcraft.com',
  rToken: null,
  appInfo: {
    appID: 'your-app-id',
    appIID: 'your-installation-id',
    appVer: '1.0.0',
  },
  providerPriorityList: null,
  enableLogging: true,

  // -------- ANDROID-ONLY (optional) --------
  android_icon: null,
  android_pushReceiverModules: null,
  android_pushChannelName: null,
  android_pushChannelDescription: null,
});
```

Step 3. Subscribing to push notifications

Push subscription is performed by calling `pushSubscribe()`:

```
AltcraftSDK.pushSubscribe(
  true,
  { _fname: 'user_first_name', _lname: 'user_last_name' }, // profileFields
  null,                                                   // customFields
  null,                                                   // cats
  null,                                                   // replace
  null                                                    // skipTriggers
);
```

- **`sync: true` — performs the request synchronously (the result can be obtained via an SDK event if su**
- **`sync: false` — performs the request asynchronously.**

(Optional) Subscribing to SDK events

To receive call results and internal SDK events, use `subscribeToEvents()`:

```
AltcraftSDK.subscribeToEvents((event: SdkEvent) => {
  // event.function — name of the SDK function that triggered the event
  // event.type — 'event' | 'error' | 'retryError'
  // event.code / event.message — event code and message
  // event.value — additional data (if any)
  console.log('[AltcraftSDK event]', event);
});
```

To unsubscribe:

```
AltcraftSDK.unsubscribeFromEvent();
```

SDK Configuration

Prerequisites

- **Push provider SDKs are integrated into the app project (see the push provider integration guides).**
- **The npm package is installed from the repository npm using one of the following commands:**

```
yarn add @altcraft/react-native-sdk
# or
npm i @altcraft/react-native-sdk
```

App preparation

Passing a JWT token to the SDK (optional)

JWT implementation is required if API requests use JWT authentication. JWT confirms that user identi
JWT authentication is mandatory if the matching type differs from push data from the subscription (for is an email or phone number).
JWT setup follows the same approach as in the native SDKs (Android / iOS).

Preparing to work with push providers

For iOS, preparation for working with push providers is done natively. Detailed instructions are availabl guide.

SDK initialization

Initialization parameters

Use the `AltcraftConfig` object to pass configuration parameters:

```
  apiUrl: string;

  rToken?: string | null;
  appInfo?: {
    appID: string;
    appIID: string;
    appVer: string;
  } | null;

  providerPriorityList?: string[] | null;
  enableLogging?: boolean | null;

  // -------- ANDROID-ONLY --------
  android_icon?: number | null;
  android_pushReceiverModules?: string[] | null;
  android_pushChannelName?: string | null;
  android_pushChannelDescription?: string | null;
};
```

Parameter descriptions:

apiUrl `string` Required: Yes
Description: Altcraft API endpoint URL

---

rToken `string | null` Default: `null`
Required: No
Description: Altcraft role token (identifies a resource / database / account). Used when the only matchi

---

appInfo `{ appID: string; appIID: string; appVer: string } | null` Default: `null` Required: No
Description: Basic application metadata.

---

enableLogging `boolean | null` Default: `null`
Required: No
Description: Enables or disables logging.

---

providerPriorityList `string[] | null` Default: `null`
Required: No
Description: Provider priority list (by index: 0 is the highest priority).

Usage examples:

Android

```
providerPriorityList: ['android-firebase', 'android-huawei', 'android-rustore']
```

iOS

```
providerPriorityList: ['ios-apns', 'ios-firebase', 'ios-huawei']
```

---

android_icon `number | null` ***(Android only)*** **Default:** `null`
Required: No
Description: `drawable` **resource ID used as the notification icon.**

---

android_pushReceiverModules `string[] | null` ***(Android only)*** **Default:** `null`
Required: No
Description: List of package names containing native `AltcraftPushReceiver` **implementations.**

---

android_pushChannelName `string | null` ***(Android only)*** **Default:** `null`
Required: No
Description: Base name of the push notification channel (visible in Android settings).

---

android_pushChannelDescription `string | null` ***(Android only)*** **Default:** `null`
Required: No
Description: Push channel description (visible in Android settings).

Performing initialization

Use the `initialize(config)` function to initialize the SDK:

```
await AltcraftSDK.initialize({
  apiUrl: 'https://pxl-example.altcraft.com',

  // optional
  rToken: null,
  enableLogging: true,
  providerPriorityList: ['android-firebase', 'android-huawei', 'android-rustore'],

  // android-only (optional)
  android_icon: null,
  android_pushReceiverModules: null,
  android_pushChannelName: null,
  android_pushChannelDescription: null,
});
```

SDK Functionality

TIP

Before proceeding, you need to configure the SDK for your application. The detailed setup guide is av

Working with subscription statuses

Changing subscription status

```
AltcraftSDK (React Native)
└─ pushSubscriptionFunctions
   // Subscribe (status = SUBSCRIBED)
   ├─ function pushSubscribe(
   │      sync: boolean | null = true,
   │      profileFields: Record<string, unknown> | null = null,
   │      customFields: Record<string, unknown> | null = null,
   │      cats: CategoryData[] | null = null,
   │      replace: boolean | null = null,
   │      skipTriggers: boolean | null = null
   │    ): void
   // Suspend (status = SUSPENDED)
   ├─ function pushSuspend(
   │      sync: boolean | null = true,
   │      profileFields: Record<string, unknown> | null = null,
   │      customFields: Record<string, unknown> | null = null,
   │      cats: CategoryData[] | null = null,
   │      replace: boolean | null = null,
   │      skipTriggers: boolean | null = null
   │    ): void
   // Unsubscribe (status = UNSUBSCRIBED)
   └─ function pushUnSubscribe(
         sync: boolean | null = true,
         profileFields: Record<string, unknown> | null = null,
         customFields: Record<string, unknown> | null = null,
         cats: CategoryData[] | null = null,
         replace: boolean | null = null,
         skipTriggers: boolean | null = null
       ): void
```

- `function pushSubscribe()` — **subscribes to push notifications;**
- `function pushUnSubscribe()` — **unsubscribes from push notifications;**
- `function pushSuspend()` — **suspends the push subscription (no notifications are delivered, but no unsu profile);**
- `function unSuspendPushSubscription()` — **used to implement** `LogIn` / `LogOut` **transitions.**

All functions in this group share the same signature and parameters:

---

sync: `Boolean`
Default: `true`
Required: No
Description: A flag that sets whether the request is executed synchronously.
Successful request execution:
On success, an event with code 230, 231, or 232 is created. Its `event.value` **depends on the sync flag:**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "7f31a9c4"
           ├ provider: "android-firebase"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: "{\"raw\":\"14\",\"ver\":\"[\\\"14.0\\\", \\\"0.0\\\", \\\"0.0\\\"]\"}"
           │  ├ _device_type: "mob"
           │  ├ _ad_track: "true"
           │  ├ _device_name: "Pixel"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0100"
           │  ├ _os: "Android"
           │  ├ _device_model: "Pixel 7"
           │  └ _app_ver: "1.0.0"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, the event value `event.value["response_with_http_code"]` contains:

- **httpCode — transport-level status code;**
- **response — an export type containing:**
  - `error: number | null` **— internal server error code *(0 if no errors)*,**
  - `errorText: string | null` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData | null` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (Subscription)**
    - **if the request fails, only `profile = null` is returned**

▶ **export type**

If sync == false

```
ResponseWithHttpCode
  ├ httpCode: number | null
  └ response: Response | null
     ├ error?: number | null
     ├ errorText?: string | null
     └ profile?: ProfileData | null // always null
```

For asynchronous requests, the event value `event.value["response_with_http_code"]` contains:

- **httpCode — transport-level status code;**
- **response — an export type containing:**
  - `error: number | null` **— internal server error code *(0 if no errors)*,**
  - `errorText: string | null` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData | null` **— always `null` for async requests.**

Failed request execution:

If a request in this group fails, an event with one of the following codes is created:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only `httpCode` if the Altcraft server was unavailable;**
- **`error` and `errorText` if the server returned an error.**

Retrieving event values

```
const RESPONSE_CODES = new Set<number>([
  230, 231, 232,
  430, 431, 432,
  531, 532, 533,
]);

subscribeToEvents((event: SdkEvent) => {
  if (event.code == null) return;
  if (!RESPONSE_CODES.has(event.code)) return;

  const raw = event.value?.['response_with_http_code'] as unknown;
  const responseWithHttp = raw as ResponseWithHttpCode | null;
  if (!responseWithHttp) return;

  const httpCode = responseWithHttp.httpCode;

  const response = responseWithHttp.response;
  const error = response?.error ?? null;
  const errorText = response?.errorText ?? null;

  const profile = response?.profile ?? null;
  const profileId = profile?.id ?? null;
  const profileStatus = profile?.status ?? null;
  const profileIsTest = profile?.isTest ?? null;

  const subscription = profile?.subscription ?? null;
  const subscriptionId = subscription?.subscriptionId ?? null;
  const hashId = subscription?.hashId ?? null;
  const provider = subscription?.provider ?? null;
  const subscriptionStatus = subscription?.status ?? null;

  const fields = subscription?.fields ?? null;

  const cats = subscription?.cats ?? null;
  const firstCat = cats?.[0] ?? null;
  const catName = firstCat?.name ?? null;
  const catTitle = firstCat?.title ?? null;
  const catSteady = firstCat?.steady ?? null;
  const catActive = firstCat?.active ?? null;
});
```

---

profileFields: `Record<string, unknown> | null`

Default: `null`
Required: No
Description: An object containing profile fields.

The parameter can include system fields (for example, `_fname` — first name, `_lname` — last name) and op the platform UI beforehand). Allowed JSON-compatible structures:

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Number**
  - `null`
- **Objects:** `Record<string, unknown>`
- **Lists:** `unknown[]`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `Record<string, unknown> | null`

Default: `null`
Required: No
Description: An object containing subscription fields.

The parameter can include system fields (for example, `_device_model` — device model, `_os` — OS) and op the platform UI beforehand). Allowed value types (JSON-compatible, scalars only):

- **String**
- **Boolean**
- **Number**
- `null`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

> **TIP: Note**
>
> **Most system subscription fields are automatically collected by the SDK and added to push requests. T**
> `"_os_language"`, `"_device_type"`, `"_device_model"`, `"_device_name"`, `"_os_ver"`, `"_ad_track"`, `"_ad_id"`.

---

cats: `CategoryData[] | null`
Default: `null`
Required: No
Description: Subscription categories.
Category structure:

```
  name?: string | null;
  title?: string | null;
  steady?: boolean | null;
  active?: boolean | null;
};
```

When sending a push request with categories, use only `name` (category name) and `active` (category act used in request processing. The `title` and `steady` fields are populated when retrieving subscription inf
Request example:

```
const cats: CategoryData[] = [
  { name: 'football', active: true },
  { name: 'hockey', active: true },
];
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft p in the resource is used, the request returns an error:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `boolean | null`
Default: `null`
Required: No
Description: When enabled, all subscriptions of other profiles with the same push token in the current after a successful request.

---

skipTriggers: `boolean | null`
Default: `null`
Required: No
Description: When enabled, the profile containing this subscription will be ignored by mailing and scen

---

Request examples

Example: subscribing to push notifications

Minimal working call:

```
AltcraftSDK.pushSubscribe();
```

Passing all available parameters:

```
AltcraftSDK.pushSubscribe(
  true,
  { _fname: 'Andrey', _lname: 'Pogodin' },
  { developer: true },
  [{ name: 'developer_news', active: true }],
  false,
  false
);
```

> **TIP: Note**
>
> **For `pushSubscribe`, `pushSuspend`, and `pushUnSubscribe`, the SDK automatically retries the request if the HTT range. No retry occurs if the code is outside this range.**

Function `unSuspendPushSubscription()`
The function `async function unSuspendPushSubscription(): Promise<ResponseWithHttpCode | null>` **is designed for** `Log follows:**

- searches for subscriptions with the same push token as the current one that do not belong to the p JWT;
- changes the status of the found subscriptions from `subscribed` to `suspended`;
- changes the status of the subscriptions in the profile referenced by the current JWT from `suspended` and contains subscriptions);
- returns `ResponseWithHttpCode | null`, where `response.profile` is the current profile referenced by the JW `response.profile` is `null`).

Recommended `LogIn` / `LogOut` flow

`LogIn` flow

- An anonymous user opens the app. The user has `JWT_1`, pointing to database #1Anonymous;
- The user subscribes to push; a profile is created in #1Anonymous;
- The user registers and receives `JWT_2`, pointing to database #2Registered;
- Call `unSuspendPushSubscription()` — the anonymous subscription in #1Anonymous is suspended;
- The SDK searches for a profile in #2Registered to restore the subscription;
- Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()`
- After receiving `null`, call `pushSubscribe()` to create a new profile in #2Registered.

`LogOut` flow

- The user logs out in the app (`LogOut`);
- The user receives `JWT_1`, pointing to database #1Anonymous;
- Call `unSuspendPushSubscription()`, which suspends the subscription in #2Registered and sets the statu
- The request returns profile #1Anonymous != null — the subscription already exists, no new one is r

Example implementation:

```
async function unSuspend(logIn: boolean): Promise<void> {
  setAuth(logIn);

  const result: ResponseWithHttpCode | null =
    await AltcraftSDK.unSuspendPushSubscription();

  if (!result) {
    AltcraftSDK.pushSubscribe();
    return;
  }

  const httpCode = result.httpCode ?? null;
  const subscription = result.response?.profile?.subscription ?? null;

  if (httpCode === 200 && subscription == null) {
    AltcraftSDK.pushSubscribe();
  }
}

  void unSuspend(true);
}

  void unSuspend(false);
}
```

Requesting subscription status

```
AltcraftSDK
└─ function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>
└─ function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>
└─ function getStatusOfLatestSubscriptionForProvider(
       provider: string | null
    ): Promise<ResponseWithHttpCode | null>
```

Subscription status functions:

- `function getStatusOfLatestSubscription()` — status of the profile's latest subscription;
- `function getStatusForCurrentSubscription()` — subscription status for the current token/provider;
- `function getStatusOfLatestSubscriptionForProvider()` — status of the latest subscription for a provider. If `r` provider is used.

`function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>`

Retrieves the status of the profile's latest subscription. Returns `ResponseWithHttpCode | null`, containing `r` most recently created subscription in the profile. If no such subscription exists, `null` is returned.

Example:

```
const result = await AltcraftSDK.getStatusOfLatestSubscription();
```

`function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>`

Retrieves the subscription status for the current token/provider. Returns `ResponseWithHttpCode | null`, cont — the subscription found by the current push token and provider. If no such subscription exists, `null` is

Example:

```
const result = await AltcraftSDK.getStatusForCurrentSubscription();
```

`function getStatusOfLatestSubscriptionForProvider(provider: string | null): Promise<ResponseWithHttpCode | null>`

Retrieves the latest subscription status by provider. Returns `ResponseWithHttpCode | null`, containing `respo` recently created subscription for the specified provider. If no provider is specified (`provider = null`), the no such subscription exists, `null` is returned.

Example:

```
const result = await AltcraftSDK.getStatusOfLatestSubscriptionForProvider(null);
```

Below is an example of extracting profile, subscription, and category data from the status response. T retrieval functions:

Data from status-retrieval functions

```
async function readStatus(): Promise<void> {
  const result = await AltcraftSDK.getStatusForCurrentSubscription();
  if (!result) return;

  const httpCode = result.httpCode ?? null;

  const response = result.response ?? null;
  const error = response?.error ?? null;
  const errorText = response?.errorText ?? null;

  const profile = response?.profile ?? null;
  const subscription = profile?.subscription ?? null;

  const cats = subscription?.cats ?? null;
}
```

Provider push token management

TIP

On the native side of the app, you can use the functions listed in the articles for the corresponding pl

```
AltcraftSDK
|
└   // Get current device token data
  ├──   function getPushToken(): Promise<TokenData | null>
  |
  |  // Save/update token manually for a specific provider
  └──   function setPushToken(provider: string, token: string | null): Promise<void>
```

SDK functions for working with a provider token:

- `function setPushToken()` **— sets the device push token and provider;**
- `function getPushToken()` **— retrieves the current push token.**

`setPushToken(provider, token)`

Sets the device push token and provider.

- **In React Native, `context` is not passed — token storage and persistence are handled on the native s**
- `token: null` **clears the token for the specified provider.**

Usage example:

```
await AltcraftSDK.setPushToken('FCM', token);
```

Example of passing a token when a new token is received:

```
async function onNewToken(provider: string, token: string) {
  await AltcraftSDK.setPushToken(provider, token);
}
```

`getPushToken()`

Returns the current device push token data and provider as `TokenData`.

Return format:

- `TokenData` — **an object with:**
  - `provider: string`
  - `token: string`

If the token is unavailable, `null` **is returned.**

Usage example:

```
const tokenData = await AltcraftSDK.getPushToken();
```

Example of requesting push token data:

```
async function readPushToken() {
  const data = await AltcraftSDK.getPushToken();

  const provider = data?.provider ?? null;
  const token = data?.token ?? null;

  return { provider, token };
}
```

Registering and passing push notifications to the SDK

In native code

To configure push notification handling (receiving and passing to the SDK) in React Native, use the nat

Android:

- **Connecting push providers**
- **Preparing the SDK for push providers**
- **Passing push notifications to the SDK**

Note that in an RN project on Android, the app entry point is `MainApplication : Application(), ReactApplicatio` **providers, not a separate** `Application` **class.**

iOS:

- **Connecting push providers**
- **Configuring the app's AppDelegate**
- **Preparing NSE**
- **Working with push notifications**

> INFO: Important
>
> **The Altcraft npm library** `@altcraft/react-native-sdk` **already includes native SDKs. All native classes are** **app code immediately after installing the package. Do not connect the SDK additionally via Gradle, C** **Manager.**

In React Native (Android only)

On Android, incoming push notifications can be received in the React part of the app using one of the p

- **FCM (Firebase Cloud Messaging):** `@react-native-firebase/app` **and** `@react-native-firebase/messaging`
- **HMS (Huawei Mobile Services):** `@hmscore/react-native-hms-push`

FCM

Install the required dependencies:

```
npm i @react-native-firebase/app @react-native-firebase/messaging
# or
yarn add @react-native-firebase/app @react-native-firebase/messaging
```

Push notification handling via FCM works as follows:

Background push

```
messaging().setBackgroundMessageHandler(async (rm) => {
  console.log('altcraft_lib push background');

  const payload = RemoteMessagetoStringMap(rm?.data);
  if (payload) {
    AltcraftSDK.takePush(payload);
  }
});
```

Foreground push

```
function registerFCMForegroundPush(): void {
  if (fcmFg || Platform.OS !== 'android') return;
  fcmFg = true;

  messaging().onMessage(async (rm) => {
    console.log('altcraft_lib push foreground');

    const payload = RemoteMessagetoStringMap(rm?.data);
    if (payload) {
      AltcraftSDK.takePush(payload);
    }
  });
}
```

To pass a push notification to the SDK, use the method:

```
AltcraftSDK.takePush(message: Record<string, string>): void
```

where `message` is the push notification data, pre-converted from `RemoteMessage` to `Record<string, string>` for

This approach can be combined with using the native push service `FirebaseMessagingService`: notifications simultaneously passed to React Native/Altcraft SDK for business logic.

HMS

Install the required dependencies:

```
npm i @hmscore/react-native-hms-push
# or
yarn add @hmscore/react-native-hms-push
```

Push notification handling via HMS works as follows:

Background push

```
HmsPushMessaging.setBackgroundMessageHandler(async (dataMessage) => {
  console.log('altcraft_lib push background');

  const payload = new RNRemoteMessage(dataMessage).getDataOfMap();
  if (payload) {
    AltcraftSDK.takePush(payload);
  }
});
```

Foreground push

```
let hmsFg = false;

function registerHMSForegroundPush(): void {
  if (hmsFg || Platform.OS !== 'android') return;
  hmsFg = true;

  HmsPushEvent.onRemoteMessageReceived((dataMessage) => {
    console.log('altcraft_lib push foreground');

    const payload = new RNRemoteMessage(dataMessage).getDataOfMap();
    if (payload) {
      AltcraftSDK.takePush(payload);
    }
  });
}
```

To pass a push notification to the SDK, use the method:

```
AltcraftSDK.takePush(message: Record<string, string>): void
```

where `message` is the push notification data, pre-converted from `RemoteMessage` to `Record<string, string>` for

This approach can be combined with using the native push service `HmsMessageService`: notifications can b simultaneously passed to React Native/Altcraft SDK for business logic.

Mobile events

```
// AltcraftSDK object function (React Native)

AltcraftSDK
└── function mobileEvent(
        sid: string,
        eventName: string,
        sendMessageId?: string | null,
        payload?: Record<string, unknown> | null,
        matching?: Record<string, unknown> | null,
        matchingType?: string | null,
        profileFields?: Record<string, unknown> | null,
        subscription?: Subscription | null,
        utm?: UTM | null
    ): void
```

Use the `mobileEvent()` function from `AltcraftSDK` to register a mobile event. The function accepts the foll

---

sid: `string`

Required: Yes Description: String identifier of the pixel to which mobile events are bound.

---

eventName: `string`

Required: Yes Description: Name of the mobile event.

---

sendMessageId: `string | null`

Required: No Description: SMID identifier of the sent message (if the event is related to a specific mail

---

payload: `Record<string, unknown> | null`

Required: No Description: Event data — a map with string keys. In the RN bridge, values are serialized t

- `string` **— passed as is**
- `number` / `boolean` / `bigint` — `String(value)`
- `object` / `array` — `JSON.stringify(value)`
- `null` / `undefined` **— the key is not passed (the field is removed from payload)**

> TIP: Note
>
> **Non-serializable objects (for example, `Date` without conversion, custom classes, functions) are passe may result in incorrect data. Pass JSON-compatible values.**

---

matching: `Record<string, unknown> | null`

Required: No Description: Map for passing values with matching types and identifiers (for example, `{"e` bridge, values are serialized to `string` using the same rules as for `payload`.

---

matchingType: `string | null`

Required: No Description: Matching type.

---

profileFields: `Record<string, unknown> | null`

Required: No Description: Profile fields — a map with string keys. In the RN bridge, values are serialize for `payload`.

> TIP: Note
>
> **The `profileFields` parameter is used only when JWT authorization is enabled.**

---

subscription: `Subscription | null`

Required: No Description: Parameter for adding a subscription for the selected channel.

The parameter value is one of the implementations of the `Subscription` union type:

- `EmailSubscription` **— email subscription**
- `SmsSubscription` **— SMS subscription**
- `PushSubscription` **— push subscription**
- `CcDataSubscription` **— subscription for Telegram, WhatsApp, Viber, Notify**

> TIP: Note
>
> **The `subscription` parameter is used only when JWT authorization is enabled.**

Common subscription fields (for all implementations)

Field	Type	Required	De
`type`	`string`	**Yes**	**Channel type (discriminator).**

Field	Type	Required	De
`resource_id`	`number`	**Yes**	**Resource/source subscription ID.**
`status`	`string | null`	**No**	**Subscription status (for example, active/suspended).**
`priority`	`number | null`	**No**	**Delivery priority.**
`custom_fields`	`{ [key: string]: string } | null`	**No**	**Custom subscription fields.**
`cats`	`string[] | null`	**No**	**Subscription categories.**

Subscription variants

EmailSubscription (`type = "email"`)

Field	Type	Required	
`resource_id`	`number`	**Yes**	**Altcraft resource ID**
`email`	`string`	**Yes**	**Email address**

SmsSubscription (`type = "sms"`)

Field	Type	Required	Description
`resource_id`	`number`	**Yes**	**Altcraft resource ID**
`phone`	`string`	**Yes**	**Phone number in international format**

PushSubscription (`type = "push"`)

Field	Type	Required	Descriptio
`resource_id`	`number`	**Yes**	**Altcraft resource ID**
`provider`	`string`	**Yes**	**Provider (for example, `"android-firebase"`)**
`subscription_id`	`string`	**Yes**	**Provider subscription unique identifier**

CcDataSubscription (`type = "cc_data"`)

Field	Type	Required	Description
`resource_id`	`number`	**Yes**	**Altcraft resource ID**
`channel`	`string`	**Yes**	**One of: `"telegram_bot"`, `"whatsapp"`, `"viber"`, `"notify"`**
`cc_data`	`{ [key: string]: string }`	**Yes**	**Channel-specific data (for example, chat ID, phone number, tokens)**

utm: `UTM | null`

Required: No Description: UTM tags. Passed as a `UTM` structure, where each UTM type is a separate fie

```
  campaign?: string | null;
  content?: string | null;
  keyword?: string | null;
  medium?: string | null;
  source?: string | null;
  temp?: string | null;
};
```

Usage examples

```
AltcraftSDK.mobileEvent('your sid', 'app_open');
```

Passing data via payload

```
AltcraftSDK.mobileEvent(
  'your sid',
  'purchase',
  null,
  {
    orderId: 'A-1001',
    amount: 3990,
    currency: 'RUB',
    items: [{ sku: 'sku-1', qty: 2 }],
  }
);
```

Passing UTM tags

```
AltcraftSDK.mobileEvent(
  'your sid',
  'app_install',
  null,
  null,
  null,
  null,
  null,
  null,
  {
    campaign: 'your campaign utm',
    content: 'your content utm',
    keyword: 'your keyword utm',
    medium: 'your medium utm',
    source: 'your source utm',
    temp: 'your temp utm',
  }
);
```

Passing `matching` / `matchingType`

```
AltcraftSDK.mobileEvent(
  'your sid',
  'profile_update',
  null,
  { source: 'settings' },
  { email: 'user@example.com' },
  'email'
);
```

***Passing `subscription` (JWT)**

```
const sub: PushSubscription = {
  type: 'push',
  resource_id: 10,
  provider: 'android-firebase',
  subscription_id: 'provider-sub-id',
  status: 'active',
  priority: 1,
  custom_fields: { utm: 'your utm tag' },
  cats: ['promo', 'news'],
};

AltcraftSDK.mobileEvent(
  'your sid',
  'jwt_bind_subscription',
  null,
  null,
  null,
  null,
  null,
  sub,
  null
);
```

> **TIP: Note**
>
> **When passing `subscription` via the RN bridge, some fields are serialized to strings (for example, `custom_` JSON strings). Pass JSON-compatible values.**

Receiving SDK events in the app

In React Native, SDK events are passed from the native layer via `NativeEventEmitter`. Only one active sub the app.

Events are emitted under the name `AltcraftSdkEvent`. Each event received in JS has the following format

```
  function: string;                // SDK function name
  code: number | null;             // event code
  message: string;                 // event message
  type: 'event' | 'error' | 'retryError';
  value?: Record<string, unknown> | null; // extra payload
};
```

Event types:

- `event` — **informational events and successful operations;**
- `error` — **execution errors;**
- `retryError` — **errors of operations that the SDK retries automatically.**

Subscribing to events

To subscribe, use `subscribeToEvents`.

When you subscribe again, the previous subscriber is automatically removed.

```
subscribeToEvents((event) => {
  switch (event.type) {
    case 'event':
      // handle success event
      break;

    case 'error':
      // handle error
      break;

    case 'retryError':
      // handle retryable error
      break;
  }
});
```

Unsubscribing from events

To stop receiving events, use `unsubscribeFromEvent`.

```
unsubscribeFromEvent();
```

After calling it, the subscriber is removed and SDK events are no longer delivered to JavaScript.

List of all SDK events

Event list

Code	Description
200	SDK configuration is installed
201	push provider set
202	received a notification unrelated to the Altcraft Platform
203	received Altcraft push notification
204	push is posted
205	SDK data has been cleared
206	waiting for SDK initialization to complete
207	waiting for push token update to complete
220	receiver override event
230	subscribe request succeeded
231	push suspend request succeeded
232	push unsubscribed request succeeded
233	push update request succeeded
234	push unsuspend request succeeded
235	profile status request succeeded
236	successful request: inapp/unsubscribe
237	successful request: inapp/subscribe
238	successful request: inapp/status
239	successful request: inapp/event
240	successful request: inapp/placements
241	successful request: profile/authenticate
401	the configuration is not set
402	userTag is null. It is impossible to identify the user
404	SDK initialization timeout has expired
405	mobile event parts is null
422	unsuspend request data is null
423	profile request data is null
430	subscribe request failed
431	suspend request failed
432	unsubscribed request failed
433	update request failed
434	unsuspend request failed
435	status request failed
436	failed request: inapp/unsubscribe
437	failed request: inapp/placements
438	failed request: inapp/subscribe
439	failed request: inapp/status
440	failed request: inapp/event
441	failed request: profile/authenticate

Code	Description
450	push data is null
451	uid in the push data is null or empty, it is impossible to send a push event to the server
452	error uploading the notification image
453	couldn’t create notification
454	foreground info is null
455	the notification channel has not been created
471	invalid provider. Available - android-firebase, android-huawei, android-rustore
472	invalid customFields: not all values are primitives
480	subscribe retry limit reached (request removed from DB)
484	push event retry limit reached (request removed from DB)
485	mobile event retry limit reached (request removed from DB)
501	config data is null
502	current push token is null
503	userTag is null. It is impossible to identify the user
504	no permission to send notifications
505	no internet connection, retry when connection is restored
506	failed to update the push token. The subscription request was rejected
520	push subscribe request data is null
521	token update request data is null
524	push event request data is null
525	mobile event request data is null
529	common data is null
530	subscribe request failed (retryable)
531	token update request failed (retryable)
534	push event delivery failed (retryable)
535	mobile event delivery failed (retryable)
540	JWT token is null
541	matching mode is null
542	JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS
543	JWT does not contain a payload
544	auth data is null
545	matching claim does not contain a matching ID
560	response data is null

In-App Notifications

```
AltcraftSDK
└── inAppFunctions
    // Set current screen marker
    ├── function setScreen(screenName: string): void
    //
    // Open an In-App screen by name
    ├── function openScreen(screenName: string): void
    //
    // Close an In-App screen by name
    ├── function closeScreen(screenName: string): void
    //
    // Close all open In-App screens
    ├── function closeAllScreens(): void
    //
    // Get array of active screen names
    ├── function getActiveScreens(): string[]
    //
    // Send screen opened event
    ├── function setScreenOpened(screenName: string): void
    //
    // Send screen closed event
    ├── function setScreenClosed(screenName: string): void
    //
    // Send screen action event
    ├── function setScreenAction(screenName: string, actionName: string): void
    //
    // Send screen stay time event
    └── function setScreenStayTime(screenName: string, stayTime: number): void
```

Functions for working with In-App notifications:

- `function setScreen()` — **sets the current screen marker, used for In-App notification filtering;**
- `function openScreen()` — **opens an In-App screen by name;**
- `function closeScreen()` — **closes an In-App screen by name;**
- `function closeAllScreens()` — **closes all open In-App screens;**

- `function getActiveScreens()` — **returns an array of active screen names;**
- `function setScreenOpened()` — **sends a screen opened event;**
- `function setScreenClosed()` — **sends a screen closed event;**
- `function setScreenAction()` — **sends a screen action event;**
- `function setScreenStayTime()` — **sends a screen stay time event.**

`function setScreen(screenName: string): void`

Sets the current screen marker. The marker value is used by the Altcraft platform to filter In-App notifi be shown only on specific app screens.

Call this function when navigating to a new screen so the SDK always knows which screen the user is

Example:

```
AltcraftSDK.inAppFunctions.setScreen('home_screen');
```

`function openScreen(screenName: string): void`

Opens an In-App screen by its name.

Example:

```
AltcraftSDK.inAppFunctions.openScreen('screen_1');
```

`function closeScreen(screenName: string): void`

Closes an In-App screen by its name.

Example:

```
AltcraftSDK.inAppFunctions.closeScreen('screen_1');
```

`function closeAllScreens(): void`

Closes all open In-App screens.

Example:

```
AltcraftSDK.inAppFunctions.closeAllScreens();
```

`function getActiveScreens(): string[]`

Returns an array of active In-App screen names.

Example:

```
const activeScreens = AltcraftSDK.inAppFunctions.getActiveScreens();
```

`function setScreenOpened(screenName: string): void`

Sends a screen opened event. Event code: `inapp_screen_opened`**.**

Example:

```
AltcraftSDK.inAppFunctions.setScreenOpened('screen_1');
```

`function setScreenClosed(screenName: string): void`

Sends a screen closed event. Event code: `inapp_screen_closed`**.**

Example:

```
AltcraftSDK.inAppFunctions.setScreenClosed('screen_1');
```

`function setScreenAction(screenName: string, actionName: string): void`

Sends a screen action event. Event code: `inapp_screen_action`**.**

Example:

```
AltcraftSDK.inAppFunctions.setScreenAction('screen_1', 'button_click');
```

`function setScreenStayTime(screenName: string, stayTime: number): void`

Sends a screen stay time event (in milliseconds). Event code: `inapp_screen_stay_time`**.**

Example:

```
AltcraftSDK.inAppFunctions.setScreenStayTime('screen_1', 15000);
```

Authentication

```
AltcraftSDK
└── authFunctions
    // Authenticate user with JWT token
    ├── function authenticate(token: string): Promise<void>
    //
    // Log out and clear user data
    └── function logOut(): Promise<void>
```

Functions for user authentication:

- `function authenticate()` — **authenticates a user on the Altcraft platform with a JWT token;**
- `function logOut()` — **logs out the user, ending the session and clearing data.**

`function authenticate(token: string): Promise<void>`

Authenticates a user on the Altcraft platform. Sends a request to the server with the JWT token and bi
After successful authentication, the user gains access to personalized In-App notifications.

> TIP: Important
>
> **Call `authenticate()` only when the user actually logs into the app and is known to the client (for exampl via a login form, OAuth, or another authentication mechanism in your app).**

Example:

```
await AltcraftSDK.authFunctions.authenticate('your-jwt-token');
```

`function logOut(): Promise<void>`

Logs out the user — ends the current authentication session on the Altcraft platform. After calling, the session mode. Personalized In-App notifications are unavailable in this mode.

> TIP: Important
>
> **Call `logOut()` only when the user actually logs out of the app and switches to anonymous mode.**

Example:

```
await AltcraftSDK.authFunctions.logOut();
```

Additional SDK functions

Clearing SDK data

`AltcraftSDK.clear();`

The function clears SDK data and cancels all background tasks that are waiting to be executed.

Manual push event registration

> TIP: Note
>
> **Use these functions only if you implement your own notification handling logic and do not pass notific**

Android

In React Native, manual push event registration functions are available on the JS side:

- `deliveryEvent(message?: Record<string, string> | null, messageUID?: string | null)` — **registers notification de**
- `openEvent(message?: Record<string, string> | null, messageUID?: string | null)` — **registers notification open.**

Parameters:

- `message: Record<string, string> | null` — **notification data payload as a dictionary of strings.**
- `messageUID: string | null` — **message identifier.**

Passing rules:

- **you can pass only `message` (the SDK will take the message ID from the payload if it exists there);**
- **you can pass only `messageUID` (if the payload is unavailable but the ID is known);**
- **if both parameters are `null`, the event will not be registered correctly (no data to identify the mes**

iOS

Manual push event registration on the React Native side is not implemented. If manual event registrati native part of the app.
See the iOS guide (manual push event registration).

Public SDK API

default export AltcraftSDK

```
AltcraftSDK
// SDK initialization and configuration setup
├─ function initialize(config: AltcraftConfig): Promise<void>
// Full SDK data cleanup
├─ function clear(): Promise<void>
// Request notification permissions (platform-dependent)
├─ function requestNotificationPermission(): void

// Push subscription (fire-and-forget)
├─ function pushSubscribe(
│    sync?: boolean | null,
│    profileFields?: Record<string, unknown> | null,
│    customFields?: Record<string, unknown> | null,
│    cats?: CategoryData[] | null,
│    replace?: boolean | null,
│    skipTriggers?: boolean | null
│  ): void
├─ function pushSuspend(...): void
└─ function pushUnSubscribe(...): void

// In-App notifications
├─ function setScreen(screenName: string): void
├─ function closeScreen(screenName: string): void
├─ function closeAllScreens(): void
├─ function getActiveScreens(): string[]
├─ function setScreenOpened(screenName: string): void
├─ function setScreenClosed(screenName: string): void
├─ function setScreenAction(screenName: string, actionName: string): void
└─ function setScreenStayTime(screenName: string, stayTime: number): void

// Authentication
├─ function authenticate(token: string): Promise<void>
└─ function logOut(): Promise<void>

// Subscription status (async)
├─ function unSuspendPushSubscription(): Promise<ResponseWithHttpCode | null>
├─ function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>
├─ function getStatusOfLatestSubscriptionForProvider(provider?: string | null): Promise<ResponseWithHttpCode | null>
└─ function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>

// SDK events (single JS-side subscriber)
├─ function subscribeToEvents(handler: (event: SdkEvent) => void): void
└─ function unsubscribeFromEvent(): void

// Mobile event sending
└─ function mobileEvent(
sid: string,
eventName: string,
sendMessageId?: string | null,
payload?: Record<string, unknown> | null,
matching?: Record<string, unknown> | null,
matchingType?: string | null,
profileFields?: Record<string, unknown> | null,
utm?: UTM | null
): void

// Push provider token
├─ function setPushToken(provider: string, token: string | null): Promise<void>
└─ function getPushToken(): Promise<TokenData | null>

// Android: pass incoming push payload to SDK
└─ function takePush(message: Record<string, string>): void

// Manual push event registration (delivery / open) on RN side
├─ function deliveryEvent(message?: Record<string, string> | null, messageUID?: string | null): void
└─ function openEvent(message?: Record<string, string> | null, messageUID?: string | null): void

// iOS: write value to UserDefaults / App Group
└─ function setUserDefaultsValue(suiteName: string | null, key: string, value: unknown): void
```

Event Classes

```
SdkEvent
// Universal SDK event delivered to JS via NativeEventEmitter.
function: string
code: number | null
message: string
type: 'event' | 'error' | 'retryError'
value?: Record<string, unknown> | null
```

Exported Types

```
// Subscription category
  name?: string | null;
  title?: string | null;
  steady?: boolean | null;
  active?: boolean | null;
};

// Device push provider token
provider: string | null;
token: string | null;
};

// Application metadata
appID: string;
appIID: string;
appVer: string;
};

// API response with HTTP status code
httpCode: number | null;
response: Response | null;
};

error?: number | null;
errorText?: string | null;
profile?: ProfileData | null;
};

id?: string | null;
status?: string | null;
isTest?: boolean | null;
subscription?: SubscriptionData | null;
};

subscriptionId?: string | null;
hashId?: string | null;
provider?: string | null;
status?: string | null;
fields?: { [key: string]: string } | null;
cats?: CategoryData[] | null;
};

// UTM parameter container
campaign?: string | null;
content?: string | null;
keyword?: string | null;
medium?: string | null;
source?: string | null;
temp?: string | null;
};
```

AltcraftConfig

```
  apiUrl: string;

  rToken?: string | null;
  appInfo?: AppInfo | null;
  providerPriorityList?: string[] | null;
  enableLogging?: boolean | null;

  // -------- ANDROID-ONLY --------
  android_icon?: number | null;
  android_pushReceiverModules?: string[] | null;
  android_pushChannelName?: string | null;
  android_pushChannelDescription?: string | null;
};
```

In-App Notifications

Available via `AltcraftSDK.inAppFunctions`:

Function	Description
`setScreen(screenName: string)`	**Sets a screen marker for the current screen (used for In-App notification filtering)**
`openScreen(screenName: string)`	**Opens an In-App screen by name**
`closeScreen(screenName: string)`	**Closes an In-App screen by name**
`closeAllScreens()`	**Closes all open In-App screens**
`getActiveScreens(): string[]`	**Returns an array of active screen names**
`setScreenOpened(screenName: string)`	**Sends a screen opened event**
`setScreenClosed(screenName: string)`	**Sends a screen closed event**
`setScreenAction(screenName: string, actionName: string)`	**Sends a screen action event**
`setScreenStayTime(screenName: string, stayTime: number)`	**Sends a screen stay time event**

Authentication

Available via `AltcraftSDK.authFunctions`:

Function	Description
`authenticate(token: string)`	**Authenticates a user with a JWT token**
`logOut()`	**Logs out and clears user data**

Provider setup

Provider setup follows the instructions for the native SDKs of each platform. More details are available guide. Below are links to the documentation for native provider configuration:

Android:

- **Firebase Cloud Messaging**
- **Huawei Mobile Services**
- **RuStore**

iOS:

- **Apple Push Notification Service**
- **Firebase Cloud Messaging**
- **Huawei Mobile Services**

Quick Start

This article provides an example of quick integration of Altcraft mSDK for Flutter.

Step 0. Prerequisites

- **A Flutter project is created and runs on Android/iOS.**
- **Native push providers are integrated into the project:**
  - **Android: FCM, HMS, RuStore**
  - **iOS: APNS, FCM, HMS**
- **Altcraft mSDK is configured on the native side:**
  - **push providers are registered;**
  - **JWT provider is registered if needed;**
  - **incoming push notification interception and passing to the SDK is configured.**

> INFO: Important
>
> **The JWT provider and push token provider are registered natively: in `Application.onCreate` on Android ar not passed via the Dart API.**

Step 1. Package Installation

Add the package to your Flutter project:

```
flutter pub add altcraft_sdk
```

After installation, fetch dependencies:

```
flutter pub get
```

Step 2. SDK Configuration and Initialization

Import the SDK and perform initialization at application startup, for example in `main.dart`:

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await AltcraftSDK.initialize(
    const AltcraftConfig(
      apiUrl: 'https://pxl-example.altcraft.com',
      appInfo: AppInfo(
        appID: '1:000000000000:android:0000000000000000',
        appIID: '00000000-0000-4000-8000-000000000000',
        appVer: '1.0.0',
      ),
      providerPriorityList: ['android-firebase'],
    ),
  );

  runApp(const AppRoot());
}

class AppRoot extends StatelessWidget {
  const AppRoot({super.key});

  @override
  Widget build(BuildContext context) {
    return const Directionality(
      textDirection: TextDirection.ltr,
      child: SizedBox.shrink(),
    );
  }
}
```

Step 3. Push Subscription

Push subscription is performed by calling `pushSubscribe()`:

```
void subscribeToPush() {
  AltcraftSDK.pushSubscribe(
    sync: true,
    profileFields: const {
      '_fname': 'Ivan',
      '_lname': 'Petrov',
    },
    customFields: const {
      'source': 'flutter_app',
    },
    cats: const [
      {
        'name': 'developer_news',
        'active': true,
      },
    ],
    replace: null,
    skipTriggers: null,
  );
}
```

- `sync: true` — **executes the request synchronously on the native side. The Dart method still returns comes through SDK events.**
- `sync: false` — **executes the request asynchronously. If `sync` is not provided, the native SDK uses `tr`**
- `profileFields` — **profile fields.**
- `customFields` — **custom subscription fields.**
- `cats` — **subscription categories.**

The `pushSubscribe()` method returns `Future<void>`. In the example, calling without `await` is acceptable (fir result is available via SDK events.

Step 4. Subscribing to SDK Events

To receive call results and internal SDK events, use `subscribeToEvents()`:

```
StreamSubscription<SdkEvent>? sdkEventsSubscription;

void subscribeToSdkEvents() {
  sdkEventsSubscription = AltcraftSDK.subscribeToEvents().listen(
    (SdkEvent event) {
      // event.function – name of the SDK function that triggered the event
      // event.type – event type: event / error / retryError
      // event.code – event code
      // event.message – event message
      // event.value – additional data, if any
      print('[AltcraftSDK event] ${event.function}: ${event.message}');
    },
  );
}

Future<void> unsubscribeFromSdkEvents() async {
  await sdkEventsSubscription?.cancel();
  sdkEventsSubscription = null;
  await AltcraftSDK.unsubscribeFromEvents();
}
```

Step 5. Requesting Notification Permission for Android

For Android, you can request notification permission via the SDK:

```
Future<void> requestPushPermission() async {
  final bool granted = await AltcraftSDK.requestNotificationPermission();

  if (granted) {
    AltcraftSDK.pushSubscribe(sync: true);
  }
}
```

> **TIP**
>
> **For iOS, notification permissions are configured in the native part of the application along with `UNUserN` Notification Service Extension. The `requestNotificationPermission()` method in the current Flutter bridge i `false`.**

SDK Configuration

Prerequisites

- **The `altcraft_sdk` package is added to the Flutter project.**
- **Push notification provider SDKs are integrated into the application project.**
- **For Android, one or more providers are configured: FCM, HMS, RuStore.**
- **For iOS, one or more providers are configured: APNS, FCM, HMS.**
- **If API requests must be authorized via JWT, the JWT provider is registered on the native side of the**

```
flutter pub add altcraft_sdk
flutter pub get
```

Application Preparation

Passing JWT Token to the SDK (Optional)

JWT implementation is required if JWT authentication of API requests is used. JWT confirms that user application.

JWT authentication implementation is mandatory if a matching type other than push data from the sub phone, or internal user identifier.

In Flutter SDK, the JWT provider is not registered via the Dart API. It must be configured natively:

- **Android: in `Application.onCreate`;**
- **iOS: in `AppDelegate`.**

Setup is performed similarly to the native platform SDKs: Android and iOS.

Preparing for Push Providers

Push provider setup is performed on the native side of the application.

Android:

- **Connecting FCM**
- **Connecting HMS**
- **Connecting RuStore**
- **Passing push notifications to the SDK**

iOS:

- **Connecting APNS**
- **Connecting FCM**
- **Connecting HMS**
- **Setting up AppDelegate**
- **Preparing NSE**

> INFO: Important
>
> **The Flutter package provides the Dart API and platform bridge. JWT registration, push provider regist processing, and Notification Service Extension setup are performed in the native parts of the project.**

SDK Initialization

Initialization Parameters

To pass configuration parameters, use the `AltcraftConfig` object:

```
const AltcraftConfig(
  apiUrl: 'https://pxl-example.altcraft.com',
  rToken: 'your-role-token',
  appInfo: AppInfo(
    appID: '1:000000000000:android:0000000000000000',
    appIID: '00000000-0000-4000-8000-000000000000',
    appVer: '1.0.0',
  ),
  providerPriorityList: ['android-firebase'],
  enableLogging: true,
)
```

Parameter description:

apiUrl `String`
Required: Yes
Description: Altcraft API endpoint URL.

---

rToken `String?`
Default: `null`
Required: No
Description: Role token for resource, database, or account identification.

---

appInfo `AppInfo?`
Default: `null`
Required: No
Description: Basic application metadata.

`AppInfo` fields:

Field	Type	Description
`appID`	`String`	**Application identifier.**
`appIID`	`String`	**Application installation identifier.**
`appVer`	`String`	**Application version.**

---

providerPriorityList `List<String>?`
Default: `null`
Required: No
Description: Push provider priority list. Index `0` is the highest priority provider.

Example for Android:

```
providerPriorityList: ['android-firebase', 'android-huawei', 'android-rustore']
```

Example for iOS:

```
providerPriorityList: ['ios-apns', 'ios-firebase', 'ios-huawei']
```

---

enableLogging `bool?`
Default: `null`
Required: No
Description: Enables or disables SDK logging.

Android-only fields

The following `AltcraftConfig` fields apply only on Android:

Field	Type	Description
`icon`	`int?`	**Notification icon resource ID.**
`pushReceiverModules`	`List<String>?`	**Packages of modules with overridden PushReceiver.**
`pushChannelName`	`String?`	**Push notification channel name.**
`pushChannelDescription`	`String?`	**Push notification channel description.**

Performing Initialization

To initialize the SDK, use the `initialize(config)` function:

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await AltcraftSDK.initialize(
    const AltcraftConfig(
      apiUrl: 'https://pxl-example.altcraft.com',
      appInfo: AppInfo(
        appID: '1:000000000000:ios:0000000000000000',
        appIID: '00000000-0000-4000-8000-000000000001',
        appVer: '1.0.0',
      ),
      providerPriorityList: ['ios-apns'],
    ),
  );

  runApp(const AppRoot());
}

class AppRoot extends StatelessWidget {
  const AppRoot({super.key});

  @override
  Widget build(BuildContext context) {
    return const Directionality(
      textDirection: TextDirection.ltr,
      child: SizedBox.shrink(),
    );
  }
}
```

Working with App Group and UserDefaults on iOS

For iOS, the SDK provides methods for setting the App Group and writing string values to `UserDefaults`. T only on iOS; on Android the call completes successfully without action.

```
Future<void> configureIosAppGroup() async {
  if (!Platform.isIOS) return;

  const String appGroup = 'group.altcraft.flutter.example';

  await AltcraftSDK.setAppGroup(appGroup);
  await AltcraftSDK.setUserDefaultsValue(
    appGroup,
    'altcraft_jwt',
    'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtYXRjaGluZyI6ImVtYWlsIn0.signature',
  );
}
```

The `setUserDefaultsValue(suiteName, key, value)` method is also available on Android, where `suiteName` is ignc SDK's `SharedPreferences`. The `value` parameter has type `String?`: a string writes the value, `null` removes point to an available `UserDefaults suite`; if the suite is not found, the method will throw an `AltcraftExcepti`

Requesting Notification Permission

For Android, use `requestNotificationPermission()`:

```
Future<bool> requestAndroidNotificationPermission() {
  return AltcraftSDK.requestNotificationPermission();
}
```

The method returns `true` if permission is granted. On Android below version 13, permission is not requ the user denied, there is no active `Activity`, or a request error occurred, `false` is returned. On iOS, this permission and returns `false`; iOS permissions are configured in the native part of the application.

SDK Functionality

TIP

You need to set up the SDK for your application first. The detailed instructions are here.

Working with Subscription Statuses

Changing Subscription Status

```
AltcraftSDK (Flutter)
└─ Push subscription functions
    // Subscribe: status = subscribed
    ├─ static Future<void> pushSubscribe({
    │     bool? sync,
    │     Map<String, dynamic>? profileFields,
    │     Map<String, dynamic>? customFields,
    │     List<dynamic>? cats,
    │     bool? replace,
    │     bool? skipTriggers,
    │   })
    // Suspend: status = suspended
    ├─ static Future<void> pushSuspend({
    │     bool? sync,
    │     Map<String, dynamic>? profileFields,
    │     Map<String, dynamic>? customFields,
    │     List<dynamic>? cats,
    │     bool? replace,
    │     bool? skipTriggers,
    │   })
    // Unsubscribe: status = unsubscribed
    └─ static Future<void> pushUnSubscribe({
          bool? sync,
          Map<String, dynamic>? profileFields,
          Map<String, dynamic>? customFields,
          List<dynamic>? cats,
          bool? replace,
          bool? skipTriggers,
        })
```

- `pushSubscribe()` — **subscribes to push notifications;**
- `pushSuspend()` — **suspends push subscription (notifications stop arriving, but no unsubscribe event is**
- `pushUnSubscribe()` — **unsubscribes from push notifications;**
- `unSuspendPushSubscription()` — **used for creating** `LogIn` **and** `LogOut` **transitions.**

The functions `pushSubscribe()`**,** `pushSuspend()`**, and** `pushUnSubscribe()` **have the same signature.**

---

sync: `bool?`
Default: `true` **on the native side if** `null` **is passed from Dart**
Required: No
Description: Flag that sets synchronous execution of the request.

Successful request execution:

Upon successful execution, an SDK event with code 230, 231, or 232 will be created, containing `event.v`
synchronization flag:

If `sync == true`

```
ResponseWithHttpCode
  ├ http_code: 200
  └ response
     ├ error: 0
     ├ error_text: ""
     └ profile
        ├ id: "00000000000000000000000000"
        ├ status: "subscribed"
        ├ is_test: false
        └ subscription
           ├ subscription_id: "provider-subscription-id"
           ├ hash_id: "7f31a9c4"
           ├ provider: "android-firebase"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os: "Android"
           │  ├ _os_ver: "14"
           │  ├ _device_type: "mob"
           │  ├ _device_model: "Pixel 7"
           │  └ _app_ver: "1.0.0"
           └ cats
              └ [ { name: "developer_news", active: true } ]
```

For a synchronous request, the event value `event.value` under the key `response_with_http_code` provides:

- `http_code` — **transport response code;**
- `response` — **response data containing:**
  - `error` — **internal server error code *(0 if no errors)*;**
  - `error_text` — **error text *(empty string if no errors)*;**
  - `profile` — **profile and subscription data if the request is successful. If the request fails, only** `pr`

If `sync == false`

```
ResponseWithHttpCode
  ├ http_code: int?
  └ response: Map<String, dynamic>?
     ├ error: int?
     ├ error_text: String?
     └ profile: null // usually null for async request
```

For an asynchronous request, the event value `event.value` under the key `response_with_http_code` provides:

- `http_code` — **transport response code;**
- `response` — **response data containing:**
  - `error` — **internal server error code *(0 if no errors)*;**
  - `error_text` — **error text *(empty string if no errors)*;**
  - `profile` — **for asynchronous requests, always** `null`**.**

Request execution with an error:

If a request from this group of functions fails, an event will be created with the following codes:

Operations without automatic retry on the SDK side:

- **430 — subscription to notifications;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 — subscription to notifications;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event content:

- **only** `http_code`**, if the Altcraft server was unavailable;**
- `error` **and** `error_text`**, if the server returned an error.**

Retrieving event values

```
final responseCodes = {230, 231, 232, 430, 431, 432, 530, 531, 532};

AltcraftSDK.subscribeToEvents().listen((SdkEvent event) {
  if (event.code == null) return;
  if (!responseCodes.contains(event.code)) return;

  final raw = event.value?['response_with_http_code'];
  if (raw == null) return;

  final map = Map<String, dynamic>.from(raw as Map);
  final httpCode = map['http_code'] ?? map['httpCode'];
  final response = map['response'] as Map<dynamic, dynamic>?;

  final error = response?['error'];
  final errorText = response?['error_text'] ?? response?['errorText'];
  final profile = response?['profile'] as Map<dynamic, dynamic>?;
  final subscription = profile?['subscription'] as Map<dynamic, dynamic>?;

  print('httpCode=$httpCode error=$error errorText=$errorText profileId=${profile?['id']}');
  print('subscriptionId=${subscription?['subscription_id']}');
});
```

---

profileFields: `Map<String, dynamic>?`
Default: `null`
Required: No
Description: Object containing profile fields.
The parameter can accept both system fields (e.g., `_fname` — first name or `_lname` — last name) and opt the platform interface in advance). Allowed structures (JSON-compatible):

- **Scalar values:** `String`, `bool`, `num` **(int / double),** `null`
- **Objects:** `Map<String, dynamic>`
- **Lists:** `List<dynamic>`

If an invalid optional field is passed, the request will fail with an error:

```
SDK error: 430
http code: 400
error: 400
error_text: Platform profile processing error: with field "field_name": Incorrect field
```

```
AltcraftSDK.pushSubscribe(
  sync: true,
  profileFields: const {
    '_fname': 'Ivan',
    '_lname': 'Petrov',
    '_email': 'ivan.petrov@example.com',
  },
);
```

---

customFields: `Map<String, dynamic>?`
Default: `null`
Required: No
Description: Object containing subscription fields.
The parameter can accept both system fields (e.g., `_device_model` — device model or `_os` — operating sy manually in the platform interface in advance). Allowed value types (JSON-compatible, scalars only):

- `String`
- `bool`
- `num` **(int / double)**
- `null`

If an invalid optional field is passed, the request will fail with an error:

```
SDK error: 430
http code: 400
error: 400
error_text: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

```
AltcraftSDK.pushSubscribe(
  customFields: const {
    'source': 'flutter_app',
    'build': '100',
  },
);
```

TIP: Note

Most system subscription fields are automatically collected by the SDK and added to push requests. T
`"_os_tz"`, `"_os_language"`, `"_device_type"`, `"_device_model"`, `"_device_name"`, `"_os_ver"`, `"_ad_track"`, `"_ad_id"`.

cats: `List<dynamic>?`
Default: `null`
Required: No
Description: Subscription categories.
The category structure is defined by the `CategoryData` **class:**

```
class CategoryData {
  final String? name;
  final String? title;
  final bool? steady;
  final bool? active;
}
```

When sending a push request with categories, only use the `name` **(category name) and** `active` **(category and** `steady` **fields are not used in request processing — they are populated when retrieving subscriptior**

```
AltcraftSDK.pushSubscribe(
  cats: const [
    {'name': 'developer_news', 'active': true},
    {'name': 'product_updates', 'active': false},
  ],
);
```

Categories used in the request must be created and added to the resource in the Altcraft platform in a the resource is used in the request, the request will fail with an error:

```
SDK error: 430
http code: 400
error: 400
error_text: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `bool?`
Default: `null`
Required: No
Description: When this flag is activated, all subscriptions of other profiles with the same push token in `unsubscribed` **after a successful request.**

---

skipTriggers: `bool?`
Default: `null`
Required: No
Description: When this flag is activated, the profile containing this subscription will be ignored in maili

---

Request implementation examples

Example of a push subscription request

Minimal working setup:

```
AltcraftSDK.pushSubscribe();
```

Passing all available parameters:

```
AltcraftSDK.pushSubscribe(
  sync: true,
  profileFields: const {'_fname': 'Ivan', '_lname': 'Petrov'},
  customFields: const {'source': 'flutter_app'},
  cats: const [{'name': 'developer_news', 'active': true}],
  replace: false,
  skipTriggers: false,
);
```

> TIP: Note
>
> **For** `pushSubscribe`**,** `pushSuspend`**,** `pushUnSubscribe`**, the SDK provides automatic request retry if the HTTP re 599. The request is not retried if the response code is outside this range.**

`unSuspendPushSubscription()` **function**

The function `static Future<ResponseWithHttpCode?> unSuspendPushSubscription()` **is designed for creating** `LogIn` **ar follows:**

- **searches for subscriptions with the same push token as the current one, not belonging to the profi token;**
- **changes the status of found subscriptions from** `subscribed` **to** `suspended`**;**
- **changes the status in subscriptions of the profile pointed to by the current JWT from** `suspended` **to by the JWT exists and contains subscriptions);**
- **returns** `ResponseWithHttpCode?`**, where** `response?.profile` **is the current profile pointed to by the JWT (re exist).**

Recommended implementation of LogIn and LogOut transitions

`LogIn` **transition**

- **An anonymous user enters the application. This user is assigned `JWT_1`, pointing to database #1An**
- **Push subscription is performed, profile created in database #1Anonymous;**
- **The user registers and is assigned `JWT_2`, pointing to database #2Registered;**
- **Function `unSuspendPushSubscription()` is called — the anonymous user's subscription in database #1An**
- **Profile search is performed in database #2Registered to restore the subscription;**
- **Since no subscription with this push token exists in database #2Registered, the function returns `n`**
- **After receiving `null`, you can execute `pushSubscribe()` to create a new profile in database #2Registe**

`LogOut` **transition**

- **The user logs out from their profile on the application side (`LogOut`);**
- **The user is assigned `JWT_1`, pointing to database #1Anonymous;**
- **Function `unSuspendPushSubscription()` is called, which suspends the subscription in database #2Regist status in database #1Anonymous to `subscribed`;**
- **The request returns profile #1Anonymous != null — the subscription already exists, no new one is n**

Implementation example:

```
Future<void> unSuspend(bool logIn) async {
  setAuth(logIn);

  final ResponseWithHttpCode? result =
      await AltcraftSDK.unSuspendPushSubscription();

  if (result == null) {
    AltcraftSDK.pushSubscribe();
    return;
  }

  final int httpCode = result.httpCode;
  final subscription = result.response?['profile']?['subscription'];

  if (httpCode == 200 && subscription == null) {
    AltcraftSDK.pushSubscribe();
  }
}

void logIn() => unSuspend(true);
void logOut() => unSuspend(false);
```

Requesting Subscription Status

```
AltcraftSDK
├─ static Future<ResponseWithHttpCode?> getStatusOfLatestSubscription()
├─ static Future<ResponseWithHttpCode?> getStatusForCurrentSubscription()
└─ static Future<ResponseWithHttpCode?> getStatusOfLatestSubscriptionForProvider(
     String? provider,
   )
```

Subscription status request functions:

- **`getStatusOfLatestSubscription()` — status of the latest profile subscription;**
- **`getStatusForCurrentSubscription()` — status of the subscription for the current token/provider;**
- **`getStatusOfLatestSubscriptionForProvider()` — status of the latest subscription by provider. If `null` is spe token is used.**

`getStatusOfLatestSubscription()`

Function to retrieve the status of the latest profile subscription. Returns `ResponseWithHttpCode?`, containin the last created subscription in the profile. If no such subscription exists, `null` will be returned.

```
final ResponseWithHttpCode? result =
    await AltcraftSDK.takePush(const {'_uid': 'push-message-uid-0001'});
```

`getStatusForCurrentSubscription()`

Function to retrieve the subscription status for the current token/provider. Returns `ResponseWithHttpCode?`, `response?.profile?.subscription` — the subscription found by the current push token and provider. If no suc returned.

```
final ResponseWithHttpCode? result =
    await AltcraftSDK.getStatusForCurrentSubscription();
```

`getStatusOfLatestSubscriptionForProvider(provider)`

Function to retrieve the status of the latest subscription by provider. Returns `ResponseWithHttpCode?`, cont — the last created subscription with the specified provider. If the provider is not specified (`provider = nu` token is used. If no such subscription exists, `null` will be returned.

```
final ResponseWithHttpCode? result =
    await AltcraftSDK.getStatusOfLatestSubscriptionForProvider('android-firebase');
```

Below is an example of extracting profile, subscription, and category data from the response of status applies to all status retrieval functions:

Data from status retrieval functions

```
Future<void> readStatus() async {
  final ResponseWithHttpCode? result =
      await AltcraftSDK.getStatusForCurrentSubscription();

  if (result == null) return;

  final int httpCode = result.httpCode;
  final response = result.response;
  final int? error = response?['error'] ?? null;
  final String? errorText = response?['error_text'] as String?;
  final profile = response?['profile'] as Map<String, dynamic>?;
  final subscription = profile?['subscription'] as Map<String, dynamic>?;
  final cats = subscription?['cats'] ?? null;

  print('httpCode=$httpCode error=$error errorText=$errorText cats=$cats');
}
```

Managing Provider Push Tokens

> **TIP**
>
> **On the native side of the application, you can use the functions described in the articles for the corre**
> **iOS.**

```
AltcraftSDK
├─ static Future<TokenData?> getPushToken()
└─ static Future<void> setPushToken(String provider, String? token)
```

Functions for working with the provider token in the SDK:

- `setPushToken()` — **setting the device push token and provider;**
- `getPushToken()` — **retrieving the current push token.**

`setPushToken(provider, token)`

This function is designed to set the device push token and provider.

- **In Flutter,** `context` **is not passed — token saving and storage is performed on the native SDK side.**
- `token: null` — **means clearing the token for the specified provider.**

```
Future<void> saveFcmToken(String token) async {
  await AltcraftSDK.setPushToken('android-firebase', token);
}
```

Example of passing a token when receiving a new token:

```
Future<void> onNewToken(String provider, String token) async {
  await AltcraftSDK.setPushToken(provider, token);
}
```

`getPushToken()`

This function returns the current device push token and provider data as `TokenData`**.**

Format of returned data:

- `TokenData` — **object with fields:**
  - `provider: String` — **provider identifier (e.g.,** `android-firebase`**,** `ios-apns`**);**
  - `token: String` — **push token value.**

If the token is unavailable, `null` **will be returned.**

```
Future<void> readPushToken() async {
  final TokenData? data = await AltcraftSDK.getPushToken();

  final String? provider = data?.provider;
  final String? token = data?.token;

  print('provider=$provider token=$token');
}
```

Push Notification Registration and Transmission to SDK

In native code

To set up push notifications in a Flutter project, use the native SDK instructions.

Android:

- **Connecting push providers**
- **Preparing the SDK for push providers**
- **Passing push notifications to the SDK**

iOS:

- **Connecting push providers**
- **Setting up the application AppDelegate**
- **Preparing NSE**
- **Working with push notifications**

> INFO: Important
>
> **Flutter SDK provides the `takePush()` method for passing payload to the SDK on Android. On iOS, APNS a typically handled in the native part of the application and the Notification Service Extension.**

In Flutter code on Android

If incoming push notifications are handled on the Dart side, pass the payload to the SDK using the `take`

```
AltcraftSDK.takePush(Map<String, String> message)
```

where `message` is the push notification data converted to `Map<String, String>` format.

FCM (Firebase Cloud Messaging)

Install the required dependencies:

```
flutter pub add firebase_messaging
```

Handling push notifications via FCM:

Background push

```
@pragma('vm:entry-point')
Future<void> firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  final Map<String, String> payload = message.data.map(
    (String key, dynamic value) => MapEntry(key, value?.toString() ?? ''),
  );

  if (payload.isNotEmpty) {
    AltcraftSDK.takePush(payload);
  }
}
```

Foreground push

```
void registerFirebaseForegroundPush() {
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    final Map<String, String> payload = message.data.map(
      (String key, dynamic value) => MapEntry(key, value?.toString() ?? ''),
    );

    if (payload.isNotEmpty) {
      AltcraftSDK.takePush(payload);
    }
  });
}
```

Registering the background handler is done at application startup:

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  FirebaseMessaging.onBackgroundMessage(firebaseMessagingBackgroundHandler);
  runApp(const AppRoot());
}
```

This approach can be combined with using the native push service `FirebaseMessagingService`: notifications simultaneously passed to Flutter/Altcraft SDK for business logic.

Mobile Events

```
AltcraftSDK
└─ static Future<void> mobileEvent({
     required String sid,
     required String eventName,
     String? sendMessageId,
     Map<String, dynamic>? payload,
     Map<String, dynamic>? matching,
     String? matchingType,
     Map<String, dynamic>? profileFields,
     Map<String, dynamic>? subscription,
     Map<String, dynamic>? utm,
   })
```

To register a mobile event, use the `mobileEvent()` function from `AltcraftSDK`.

---

sid: `String`

Required: Yes

Description: String identifier of the pixel to which mobile events are bound.

---

eventName: `String`

Required: Yes

Description: Name of the mobile event.

---

sendMessageId: `String?`

Required: No

Description: SMID identifier of the sent message, if the event is related to a specific mailing.

---

payload: `Map<String, dynamic>?`

Required: No

Description: Event data — a map with string keys. Pass JSON-compatible values: `String`, `num`, `bool`, `Ma` values are passed to the native SDK as codec-safe `Map`; on iOS, the bridge converts them to JSON-com

> TIP: Note
>
> **Non-serializable objects may be converted to a string or dropped by the native side, so for stable resu data.**

---

matching: `Map<String, dynamic>?`

Required: No

Description: A map for passing matching values with types and identifiers, e.g., `{'email': 'ivan.petrov@exa` same rules as `payload`.

---

matchingType: `String?`

Required: No

Description: Matching type.

---

profileFields: `Map<String, dynamic>?`

Required: No

Description: Profile fields. Serialization follows the same rules as `payload`.

> TIP: Note
>
> **The `profileFields` parameter is used in scenarios where the profile is identified via JWT authorization**

---

subscription: `Map<String, dynamic>?`

Required: No

Description: Parameter for adding a subscription for a selected channel.

The SDK provides the `Subscription` class with factory constructors for each subscription type:

```
class Subscription {
  final String type;
  final int resourceId;
  final String? status;
  final int? priority;
  final String? email;
  final String? phone;
  final String? provider;
  final String? subscriptionId;
  final Map<String, dynamic>? customFields;
  final List<dynamic>? cats;
  final Map<String, dynamic>? ccData;
  final String? ccChannel;
}
```

> **TIP: Note**
>
> **The `subscription` parameter is only used when working with JWT authorization.**

Common subscription fields (for all implementations)

Field	Type	Required	Description
`type` / `channel`	`String`	**Yes**	**Channel type (discriminator). `toMap()` adds both keys.**
`resourceId` / `resource_id`	`int`	**Yes**	**Resource/source subscription ID.**
`status`	`String?`	**No**	**Subscription status.**
`priority`	`int?`	**No**	**Delivery priority.**
`customFields` / `custom_fields`	`Map<String, dynamic>?`	**No**	**Custom subscription fields.**
`cats`	`List<dynamic>?`	**No**	**Subscription categories. For `mobileEvent`, a list of strings is recommer**

Subscription variants

Email subscription (`type = "email"`)

Field	Type	Required	
`resourceId`	`int`	**Yes**	**Altcraft resource ID**
`email`	`String`	**Yes**	**Email address**

```
final emailSub = Subscription.email(
  resourceId: 10,
  email: 'user@example.com',
  status: 'subscribed',
  priority: 1,
  customFields: {'source': 'flutter_app'},
  cats: ['promo', 'news'],
);
```

SMS subscription (`type = "sms"`)

Field	Type	Required	Description
`resourceId`	`int`	**Yes**	**Altcraft resource ID**
`phone`	`String`	**Yes**	**Phone number in international format**

```
final smsSub = Subscription.sms(
  resourceId: 10,
  phone: '+79001234567',
  status: 'subscribed',
);
```

Push subscription (`type = "push"`)

Field	Type	Required	Descripti
`resourceId`	`int`	**Yes**	**Altcraft resource ID**
`provider`	`String`	**Yes**	**Provider (e.g., `"android-firebase"`)**
`subscriptionId`	`String`	**Yes**	**Unique subscription ID at the provider**

```
final pushSub = Subscription.push(
  resourceId: 10,
  provider: 'android-firebase',
  subscriptionId: 'provider-sub-id',
  status: 'subscribed',
  priority: 1,
  customFields: {'source': 'flutter_app'},
  cats: ['promo', 'news'],
);
```

CcData subscription (`type = "cc_data"`)

Field	Type	Required	Descript
`resourceId`	`int`	**Yes**	**Altcraft resource ID**
`channel`	`String`	**Yes**	**Custom channel name, e.g., `telegram_bot`**
`ccData`	`Map<String, dynamic>`	**Yes**	**Channel-specific data (chat ID, number, tokens)**

```
final ccSub = Subscription.ccData(
  resourceId: 10,
  channel: 'telegram_bot',
  ccData: {'chat_id': '123456'},
);
```

---

utm: `Map<String, dynamic>?`

Required: No

Description: UTM tags. The SDK provides the `UTM` class:

```
class UTM {
  final String? source;
  final String? medium;
  final String? campaign;
  final String? content;
  final String? keyword;
  final String? temp;
}
AltcraftSDK.mobileEvent(
  sid: 'pixel-00000000000000000000000000',
  eventName: 'app_install',
  utm: const {
    'campaign': 'summer_campaign',
    'content': 'main_banner',
    'keyword': 'mobile_sdk',
    'medium': 'push',
    'source': 'flutter_app',
    'temp': 'a1',
  },
);
```

Usage examples

Simple event

```
AltcraftSDK.mobileEvent(
  sid: 'pixel-00000000000000000000000000',
  eventName: 'app_open',
);
```

Passing data via payload

```
AltcraftSDK.mobileEvent(
  sid: 'pixel-00000000000000000000000000',
  eventName: 'purchase',
  payload: const {
    'orderId': 'A-1001',
    'amount': 3990,
    'currency': 'RUB',
    'items': [
      {'sku': 'sku-1', 'qty': 2},
    ],
  },
);
```

Passing matching / matchingType

```
AltcraftSDK.mobileEvent(
  sid: 'pixel-00000000000000000000000000',
  eventName: 'profile_update',
  payload: const {'source': 'settings'},
  matching: const {'email': 'ivan.petrov@example.com'},
  matchingType: 'email',
);
```

Passing subscription (JWT)

```
final pushSub = Subscription.push(
  resourceId: 10,
  provider: 'android-firebase',
  subscriptionId: 'provider-sub-id',
  status: 'subscribed',
  priority: 1,
  customFields: {'source': 'flutter_app'},
  cats: ['promo', 'news'],
);

AltcraftSDK.mobileEvent(
  sid: 'pixel-00000000000000000000000000',
  eventName: 'jwt_bind_subscription',
  subscription: pushSub.toMap(),
);
```

TIP: Note

Pass `subscription` via `Subscription.toMap()`. If the required fields for the selected channel are missing, th the subscription model and will send the mobile event without it.

Receiving SDK Events in the Application

In Flutter, SDK events are delivered to Dart via `Stream<SdkEvent>` (uses `EventChannel` under the hood).

```
class SdkEvent {
  final String function;
  final int? code;
  final String message;
  final String type;
  final Map<String, dynamic>? value;
  final String? timestamp;
}
```

Event types:

- `event` — **informational events and successful operations;**
- `error` — **execution errors;**
- `retryError` — **errors for operations where the SDK performs automatic retry.**

The `timestamp` field contains the event timestamp in `yyyy-MM-dd HH:mm:ss.SSS` format (provided by the nativ

Subscribing to Events

The method `subscribeToEvents()` returns a shared broadcast `Stream<SdkEvent>` that is cached on the Dart si listener, save the `StreamSubscription` and call `cancel()`.

```
StreamSubscription<SdkEvent>? sdkEventsSubscription;

void subscribeToSdkEvents() {
  sdkEventsSubscription = AltcraftSDK.subscribeToEvents().listen(
    (SdkEvent event) {
      switch (event.type) {
        case 'event':
          print('SDK event: ${event.message}');
          break;
        case 'error':
          print('SDK error: ${event.message}');
          break;
        case 'retryError':
          print('SDK retry error: ${event.message}');
          break;
        default:
          print('SDK unknown event type: ${event.type}');
      }
    },
  );
}
```

Unsubscribing from Events

```
Future<void> unsubscribeFromSdkEvents() async {
  await sdkEventsSubscription?.cancel();
  sdkEventsSubscription = null;
  await AltcraftSDK.unsubscribeFromEvents();
}
```

After calling `unsubscribeFromEvents()`, event delivery from the native SDK stops. To receive events again, c

Handling SDK Errors

The SDK may throw `AltcraftException` on method call errors via MethodChannel:

```
class AltcraftException implements Exception {
  final String code;
  final String message;
  final Object? details;
}
```

The standard MethodChannel implementation converts native-side errors into `AltcraftException`. Check methods on iOS return code `UNAVAILABLE_ON_IOS`. The `SdkNotInitializedException` class is exported by the pa such errors typically come as a regular `AltcraftException`:

```
try {
  await AltcraftSDK.takePush(const {'_uid': 'push-message-uid-0001'});
} on AltcraftException catch (e) {
  if (e.code == 'UNAVAILABLE_ON_IOS') {
    print('Method is Android-only');
    return;
  }
  print('SDK error (${e.code}): ${e.message}');
}
```

List of All SDK Events

Event list

Code	Description
200	SDK configuration is installed
201	push provider set
202	received a notification unrelated to the Altcraft Platform
203	received Altcraft push notification
204	push is posted
205	SDK data has been cleared
206	waiting for SDK initialization to complete
207	waiting for push token update to complete
220	receiver override event
230	subscribe request succeeded
231	push suspend request succeeded
232	push unsubscribed request succeeded
233	push update request succeeded
234	push unsuspend request succeeded
235	profile status request succeeded
236	push event delivered successfully
237	mobile event delivered successfully
401	the configuration is not set
402	userTag is null. It is impossible to identify the user
404	SDK initialization timeout has expired
405	mobile event parts is null
422	unsuspend request data is null
423	profile request data is null
430	subscribe request failed
431	suspend request failed
432	unsubscribed request failed
433	update request failed
434	unsuspend request failed
435	status request failed
436	push event delivery failed
437	mobile event request failed
450	push data is null
451	uid in the push data is null or empty, it is impossible to send a push event to the server
452	error uploading the notification image
453	couldn’t create notification
454	foreground info is null
455	the notification channel has not been created
471	invalid provider. Available - android-firebase, android-huawei, android-rustore
472	invalid customFields: not all values are primitives
480	subscribe retry limit reached (request removed from DB)
484	push event retry limit reached (request removed from DB)
485	mobile event retry limit reached (request removed from DB)
501	config data is null
502	current push token is null
503	userTag is null. It is impossible to identify the user
504	no permission to send notifications
505	no internet connection, retry when connection is restored
506	failed to update the push token. The subscription request was rejected
520	push subscribe request data is null
521	token update request data is null
524	push event request data is null
525	mobile event request data is null
529	common data is null

Code	Description
530	subscribe request failed (retryable)
531	suspend request failed (retryable)
532	unsubscribed request failed (retryable)
533	update request failed (retryable)
534	push event delivery failed (retryable)
535	mobile event delivery failed (retryable)
540	JWT token is null
541	matching mode is null
542	JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS
543	JWT does not contain a payload
544	auth data is null
545	matching claim does not contain a matching ID
560	response data is null

ActionFieldBuilder

The SDK provides `ActionFieldBuilder` for structured profile field management. The builder produces a ma
which should be merged with `profileFields` before passing to `pushSubscribe`, `updateProfileFields`, or similar

```
class ActionFieldBuilder {
  final String key;

  Map<String, dynamic> set(dynamic value);
  Map<String, dynamic> unset(dynamic value);
  Map<String, dynamic> incr(dynamic value);
  Map<String, dynamic> add(dynamic value);
  Map<String, dynamic> delete(dynamic value);
  Map<String, dynamic> upsert(dynamic value);
}

ActionFieldBuilder actionField(String key);
```

Available actions:

Method	Description
`set(value)`	Set the field to the given value.
`unset(value)`	Remove the field value.
`incr(value)`	Increment a numeric field by the given value.
`add(value)`	Add a value to a set/array field.
`delete(value)`	Remove a value from a set/array field.
`upsert(value)`	Set if the field is missing; update if it exists.

Usage example:

```
final profileFields = <String, dynamic>{
  ...actionField('_fname').set('Ivan'),
  ...actionField('_lname').set('Petrov'),
  ...actionField('login_count').incr(1),
  ...actionField('visited_pages').add('/settings'),
  ...actionField('old_field').unset(null),
};

AltcraftSDK.updateProfileFields(
  profileFields: profileFields,
  skipTriggers: false,
);
```

Additional SDK Functions

Clearing SDK Data

```
await AltcraftSDK.clear();
```

This function clears SDK data and cancels all background tasks awaiting execution.

Manual Push Event Registration

> **TIP: Note**
>
> **These functions are only used if you implement your own notification handling logic and do not pass t**

Android

In Flutter, manual push event registration functions are available on Android:

- `deliveryEvent(message, messageUID)` — **register notification delivery;**
- `openEvent(message, messageUID)` — **register notification opening.**

Parameters:

- `message: Map<String, String>?` — **notification data payload as a string dictionary.**
- `messageUID: String?` — **message identifier.**

```
AltcraftSDK.deliveryEvent(
  const {
    '_uid': 'push-message-uid-0001',
    '_provider': 'android-firebase',
  },
  'push-message-uid-0001',
);

AltcraftSDK.openEvent(
  const {
    '_uid': 'push-message-uid-0001',
    '_provider': 'android-firebase',
  },
  'push-message-uid-0001',
);
```

Passing rules:

- **you can pass only** `message` **(SDK will extract the message ID from the payload if present);**
- **you can pass only** `messageUID` **(if the payload is unavailable but the ID is known);**
- **if both parameters are** `null` **— the event will not be registered correctly (no data to identify the me**

iOS

Manual push event registration via Flutter is not implemented on iOS. Calls to `deliveryEvent()` **and** `openEv` `AltcraftException` **with code** `UNAVAILABLE_ON_IOS`**. If manual event registration is required, perform it in the n**
See the iOS instructions.

Updating Profile Fields

```
AltcraftSDK.updateProfileFields({
  Map<String, dynamic>? profileFields,
  bool? skipTriggers,
});
```

This function updates profile fields on the server.

- `profileFields` — **map of profile fields to update.**
- `skipTriggers` — **when this flag is activated, the profile will be ignored in mailing and scenario trigge**

```
AltcraftSDK.updateProfileFields(
  profileFields: const {
    '_fname': 'Ivan',
    '_lname': 'Petrov',
  },
  skipTriggers: false,
);
```

Forced Token Update

```
await AltcraftSDK.forcedTokenUpdate();
```

This function initiates a forced push token update.

Deleting Device Token

```
await AltcraftSDK.deleteDeviceToken('android-firebase');
```

This function deletes the device token for the specified provider.

Changing Push Provider Priority

```
await AltcraftSDK.changePushProviderPriorityList(
  const ['android-firebase', 'android-huawei', 'android-rustore'],
);
```

This function changes the push provider priority list.

Checking Altcraft Push

```
final bool isAltcraft = await AltcraftSDK.isAltcraftPush(
  const {
    '_ac_push': 'Altcraft',
    '_uid': 'push-message-uid-0001',
  },
);
```

This function checks whether a push message belongs to Altcraft. Returns `true` **if the message belong**
On iOS, the method always returns `false`**.**

Unlocking Initial Operations in Session

```
AltcraftSDK.unlockInitialOperationsInThisSession();
```

This function unlocks initial operations in the current session. The method is used on Android. On iOS, successfully.

App Group and UserDefaults

```
AltcraftSDK.setAppGroup(String? groupName);
AltcraftSDK.setUserDefaultsValue(String? suiteName, String key, String? value);
```

- `setAppGroup()` — **used on iOS to set the App Group, which defines the** `UserDefaults` **space shared betv Notification Service Extension. On Android, the method is a no-op. On iOS,** `groupName` **must not be** `n`
- `setUserDefaultsValue()` — **saves a string value to native storage:** `UserDefaults` **on iOS and** `SharedPreferen` **key is removed. On Android,** `suiteName` **is ignored.**

```
void configureIosAppGroup() {
  if (!Platform.isIOS) return;

  const String appGroup = 'group.altcraft.flutter.example';

  AltcraftSDK.setAppGroup(appGroup);
  AltcraftSDK.setUserDefaultsValue(
    appGroup,
    'altcraft_jwt',
    'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...',
  );
}
```

SDK Public API

class AltcraftSDK

The entry point to the SDK. All methods are static. The JWT provider and push token providers are regi on Android and `AppDelegate` on iOS. The Dart API passes configuration, subscription commands, mobile and auxiliary operations.

```
AltcraftSDK
// ---- Initialization and State ----
├─ static Future<void> initialize(AltcraftConfig config)
├─ static Future<void> clear()
├─ static Future<bool> requestNotificationPermission()
└─ static Future<void> unlockInitialOperationsInThisSession()

// ---- Push Subscription ----
├─ static Future<void> pushSubscribe({bool? sync, ...})
├─ static Future<void> pushSuspend({bool? sync, ...})
├─ static Future<void> pushUnSubscribe({bool? sync, ...})
└─ static Future<ResponseWithHttpCode?> unSuspendPushSubscription()

// ---- Subscription Status ----
├─ static Future<ResponseWithHttpCode?> getStatusOfLatestSubscription()
├─ static Future<ResponseWithHttpCode?> getStatusForCurrentSubscription()
└─ static Future<ResponseWithHttpCode?> getStatusOfLatestSubscriptionForProvider(String? provider)

// ---- SDK Events ----
├─ static Stream<SdkEvent> subscribeToEvents()
└─ static Future<void> unsubscribeFromEvents()

// ---- Mobile Events and Profile ----
├─ static Future<void> mobileEvent({required String sid, required String eventName, ...})
└─ static Future<void> updateProfileFields({Map<String, dynamic>? profileFields, bool? skipTriggers})

// ---- Push Tokens ----
├─ static Future<TokenData?> getPushToken()
├─ static Future<void> setPushToken(String provider, String? token)
├─ static Future<void> forcedTokenUpdate()
├─ static Future<void> deleteDeviceToken(String provider)
└─ static Future<void> changePushProviderPriorityList(List<String> priorityList)

// ---- Push Notifications ----
├─ static Future<void> takePush(Map<String, String> message)
├─ static Future<bool> isAltcraftPush(Map<String, String> message)
├─ static Future<void> deliveryEvent(Map<String, String>? message, String? messageUID)
└─ static Future<void> openEvent(Map<String, String>? message, String? messageUID)

// ---- Native storage ----
├─ static Future<void> setAppGroup(String? groupName)
└─ static Future<void> setUserDefaultsValue(String? suiteName, String key, String? value)
```

> INFO: Error Handling
>
> **Most methods return `Future<void>` and do not return the server response directly. Platform call errors results for subscription operations, profile updates, mobile events, and retry errors must be read via**

Initialization

`initialize(config)`

```
static Future<void> initialize(AltcraftConfig config)
```

Initializes the Altcraft SDK with the given configuration. Called once at application startup, before mai

Parameter	Type	Description
`config`	`AltcraftConfig`	**SDK configuration object. The `apiUrl` field is required.**

Returns: `Future<void>` — completes when the native SDK has finished initialization.

Throws: `AltcraftException` on configuration or initialization error. Main codes: `ALTCRAFT_INIT_INVALID_CONFIG`, `INITIALIZATION_FAILED`, `SDK_ERROR`.

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await AltcraftSDK.initialize(
    const AltcraftConfig(
      apiUrl: 'https://pxl-demo.altcraft.com',
      appInfo: AppInfo(
        appID: '1:100000000000:android:100000000000000',
        appIID: '10000000-0000-4000-8000-000000000001',
        appVer: '1.0.0',
      ),
      providerPriorityList: ['android-firebase'],
      enableLogging: true,
    ),
  );

  runApp(const MyApp());
}
```

`clear()`

```
static Future<void> clear()
```

Clears SDK data on the native side: local SDK storage, saved tokens, and deferred operations.

Returns: `Future<void>`.

Throws: `AltcraftException` **on error.**

`requestNotificationPermission()`

```
static Future<bool> requestNotificationPermission()
```

Requests the `POST_NOTIFICATIONS` **permission on Android 13+. On Android below 13, returns** `true` **since ru**
On iOS, this Dart method does not request permission and returns `false`**; iOS permissions are configur**
`UNUserNotificationCenter`**.**

Returns: `Future<bool>` **—** `true` **if permission is granted or not required,** `false` **on denial, missing Activity,**

```
final bool granted = await AltcraftSDK.requestNotificationPermission();

if (granted) {
  await AltcraftSDK.pushSubscribe(sync: true);
}
```

`unlockInitialOperationsInThisSession()`

```
static Future<void> unlockInitialOperationsInThisSession()
```

Unlocks initial operations in the current Android application session. On iOS, the method is a no-op and

---

Push Subscription

`pushSubscribe()` / `pushSuspend()` / `pushUnSubscribe()`

```
static Future<void> pushSubscribe({
  bool? sync,
  Map<String, dynamic>? profileFields,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
  bool? replace,
  bool? skipTriggers,
})

static Future<void> pushSuspend({
  bool? sync,
  Map<String, dynamic>? profileFields,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
  bool? replace,
  bool? skipTriggers,
})

static Future<void> pushUnSubscribe({
  bool? sync,
  Map<String, dynamic>? profileFields,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
  bool? replace,
  bool? skipTriggers,
})
```

Method	Status	Success	
`pushSubscribe()`	`subscribed`	`230`	`430`
`pushSuspend()`	`suspended`	`231`	`431`
`pushUnSubscribe()`	`unsubscribed`	`232`	`432`

All three methods have the same signature.

Parameter	Type	Description
`sync`	`bool?`	`true` — synchronous request on the SDK side; `false` — asynchronous. If `null` is passed, the native SDK uses `true`.
`profileFields`	`Map<String, dynamic>?`	Profile fields. Values should be JSON-compatible.
`customFields`	`Map<String, dynamic>?`	Custom subscription fields. Scalar JSON values are recommended.
`cats`	`List<dynamic>?`	Subscription categories. Pass a list of `Map`, e.g., `CategoryData(...).toMap()`.
`replace`	`bool?`	When `true`, the SDK may replace existing subscriptions with the same push token according to the native SDK's logi
`skipTriggers`	`bool?`	When `true`, the profile is ignored in mailing and scenario triggers.

Returns: `Future<void>` — means the command was accepted by the bridge/native SDK. Server result con

```
await AltcraftSDK.pushSubscribe(
  sync: true,
  profileFields: const {
    '_fname': 'Ivan',
    '_lname': 'Petrov',
  },
  customFields: const {
    'source': 'flutter_app',
  },
  cats: const [
    {'name': 'developer_news', 'active': true},
  ],
);
```

How to read an error from an event: if `event.type == 'error'` **or** `event.type == 'retryError'`**, check** `event.code` **the server response, the key** `response_with_http_code` **is typically used; nested fields come in** `snake_case`**.**

```
AltcraftSDK.subscribeToEvents().listen((SdkEvent event) {
  final raw = event.value?['response_with_http_code'];
  if (raw is! Map) return;

  final responseWithHttp = Map<String, dynamic>.from(raw);
  final int? httpCode = responseWithHttp['http_code'] as int?;
  final response = responseWithHttp['response'] as Map<dynamic, dynamic>?;
  final int? error = response?['error'] as int?;
  final String? errorText = response?['error_text'] as String?;

  if (event.type == 'error' || event.type == 'retryError') {
    print('SDK error code=${event.code} http=$httpCode error=$error text=$errorText');
  }
});
```

`unSuspendPushSubscription()`

```
static Future<ResponseWithHttpCode?> unSuspendPushSubscription()
```

Resumes a suspended push subscription. Used for `LogIn` / `LogOut` **transitions between profiles.**

Returns: `Future<ResponseWithHttpCode?>`**. Returns** `null` **if the native SDK did not return a response.**

Throws: `AltcraftException` **on platform call error.**

```
final ResponseWithHttpCode? result =
    await AltcraftSDK.unSuspendPushSubscription();

if (result == null) {
  await AltcraftSDK.pushSubscribe(sync: true);
  return;
}

final response = result.response;
final profile = response?['profile'] as Map<String, dynamic>?;
final subscription = profile?['subscription'] as Map<String, dynamic>?;

if (result.httpCode == 200 && subscription == null) {
  await AltcraftSDK.pushSubscribe(sync: true);
}
```

Subscription Status

`getStatusOfLatestSubscription()`

```
static Future<ResponseWithHttpCode?> getStatusOfLatestSubscription()
```

Returns the status of the latest profile subscription.

`getStatusForCurrentSubscription()`

```
static Future<ResponseWithHttpCode?> getStatusForCurrentSubscription()
```

Returns the subscription status for the current push token and current provider.

`getStatusOfLatestSubscriptionForProvider(provider)`

```
static Future<ResponseWithHttpCode?> getStatusOfLatestSubscriptionForProvider(
  String? provider,
)
```

Returns the status of the latest subscription for the specified provider. If `provider == null`**, the current t**

Parameter	Type	Description
`provider`	`String?`	**Provider identifier, e.g., `android-firebase` or `ios-apns`.**

All three methods return `Future<ResponseWithHttpCode?>`. If `result == null`, there is no response. If `result.res` returned an error — handle `error` and `error_text`.

```
final ResponseWithHttpCode? result =
    await AltcraftSDK.getStatusForCurrentSubscription();

if (result == null) return;

final response = result.response;
final int? error = response?['error'] as int?;
final String? errorText = response?['error_text'] as String?;

if (error != null && error != 0) {
  print('Status error: $errorText');
  return;
}

final profile = response?['profile'] as Map<String, dynamic>?;
final subscription = profile?['subscription'] as Map<String, dynamic>?;
print('status=${subscription?['status']} provider=${subscription?['provider']}');
```

SDK Events

`subscribeToEvents()`

```
static Stream<SdkEvent> subscribeToEvents()
```

Returns a shared broadcast `Stream<SdkEvent>` from `EventChannel` (`com.altcraft/events`). The stream is cache subscribers listen to one event source.

Returns: `Stream<SdkEvent>`.

`unsubscribeFromEvents()`

```
static Future<void> unsubscribeFromEvents()
```

Stops event delivery from the native SDK and clears the current native event sink. To receive events a again.

Returns: `Future<void>`.
Throws: `AltcraftException` **on platform call error.**

```
StreamSubscription<SdkEvent>? sdkEventsSubscription;

void startListening() {
  sdkEventsSubscription = AltcraftSDK.subscribeToEvents().listen(
    (SdkEvent event) {
      print('[${event.type}] ${event.function}: ${event.message}');
    },
  );
}

Future<void> stopListening() async {
  await sdkEventsSubscription?.cancel();
  sdkEventsSubscription = null;
  await AltcraftSDK.unsubscribeFromEvents();
}
```

Mobile Events and Profile

`mobileEvent()`

```
static Future<void> mobileEvent({
  required String sid,
  required String eventName,
  String? sendMessageId,
  Map<String, dynamic>? payload,
  Map<String, dynamic>? matching,
  String? matchingType,
  Map<String, dynamic>? profileFields,
  Map<String, dynamic>? subscription,
  Map<String, dynamic>? utm,
})
```

Registers an Altcraft mobile event.

Parameter	Type	Description
`sid`	`String`	**Pixel identifier. Must not be empty.**
`eventName`	`String`	**Event name. Must not be empty.**
`sendMessageId`	`String?`	**SMID of the sent message.**
`payload`	`Map<String, dynamic>?`	**Event data. Use JSON-compatible values.**
`matching`	`Map<String, dynamic>?`	**Matching data, e.g., `{'email': 'ivan.petrov@example.com'}`.**

Parameter	Type	Description
`matchingType`	`String?`	**Matching type.**
`profileFields`	`Map<String, dynamic>?`	**Profile fields.**
`subscription`	`Map<String, dynamic>?`	**Subscription for a channel. Pass `Subscription.toMap()`.**
`utm`	`Map<String, dynamic>?`	**UTM tags. Pass `UTM(...).toMap()` or a regular `Map`.**

Returns: `Future<void>` **— command accepted by SDK. Send result is tracked via SDK events (** `237` **,** `437` **,** `5`

Throws: `AltcraftException` **with code** `ERR` **if** `sid` **or** `eventName` **are empty/missing.**

```
await AltcraftSDK.mobileEvent(
  sid: 'pixel-10000000000000000000000000',
  eventName: 'purchase',
  payload: const {
    'order_id': 'A-1001',
    'amount': 3990,
    'currency': 'RUB',
  },
);
```

`updateProfileFields()`

```
static Future<void> updateProfileFields({
  Map<String, dynamic>? profileFields,
  bool? skipTriggers,
})
```

Updates profile fields on the server.

Parameter	Type	Description
`profileFields`	`Map<String, dynamic>?`	**Map of profile fields to update.**
`skipTriggers`	`bool?`	**When `true`, the profile is ignored in mailing and scenario triggers.**

Returns: `Future<void>` **— command accepted by SDK. Server result comes through SDK events (** `233` **,** `433`

Push Tokens

`getPushToken()`

```
static Future<TokenData?> getPushToken()
```

Returns the current device push token and provider data.

Returns: `Future<TokenData?>` **—** `TokenData` **object or** `null` **if the token is unavailable.**

Throws: `AltcraftException` **on platform call error.**

`setPushToken(provider, token)`

```
static Future<void> setPushToken(String provider, String? token)
```

Sets or clears the push token for the specified provider. If `token == null`**, the token is cleared.**

Parameter	Type	Description
`provider`	`String`	**Provider identifier, e.g.,** `android-firebase`**,** `android-huawei`**,** `android-rustore`**,** `ios-apns`**,** `ios-firebase`**,** `ios-huawei`**.**
`token`	`String?`	**Token value.** `null` **— clears the token.**

`forcedTokenUpdate()`

```
static Future<void> forcedTokenUpdate()
```

Initiates a forced push token update via the native SDK.

`deleteDeviceToken(provider)`

```
static Future<void> deleteDeviceToken(String provider)
```

Deletes the device token for the specified provider.

Parameter	Type	Desc
`provider`	`String`	**Provider identifier.**

`changePushProviderPriorityList(priorityList)`

```
static Future<void> changePushProviderPriorityList(List<String> priorityList)
```

Changes the push provider priority list. Index `0` **is the highest priority provider.**

Parameter	Type	
`priorityList`	`List<String>`	**New priority list.**

```
await AltcraftSDK.changePushProviderPriorityList(
  ['android-firebase', 'android-huawei', 'android-rustore'],
);
```

All methods in this group throw `AltcraftException` **on platform call error.**

Push Notifications

`takePush(message)`

```
static Future<void> takePush(Map<String, String> message)
```

Passes incoming push payload to the SDK. The method is available on Android. On iOS, the call will thr

`UNAVAILABLE_ON_IOS`**.**

Parameter	Type	Descr
`message`	`Map<String, String>`	**Push notification data payload.**

`isAltcraftPush(message)`

```
static Future<bool> isAltcraftPush(Map<String, String> message)
```

Checks whether a push message belongs to Altcraft.

Parameter	Type	Descr
`message`	`Map<String, String>`	**Push notification data payload.**

Returns: `Future<bool>` — `true` **if the message belongs to Altcraft. On iOS, always returns** `false`**.**

`deliveryEvent(message, messageUID)`

```
static Future<void> deliveryEvent(Map<String, String>? message, String? messageUID)
```

Manual push notification delivery registration. The method is available on Android. On iOS, the call will code `UNAVAILABLE_ON_IOS`**.**

Parameter	Type	
`message`	`Map<String, String>?`	**Notification data payload.**
`messageUID`	`String?`	**Message identifier.**

You can pass only `message` **or only** `messageUID`**. If both values are** `null`**, the SDK cannot correctly identify**

`openEvent(message, messageUID)`

```
static Future<void> openEvent(Map<String, String>? message, String? messageUID)
```

Manual push notification opening registration. The method is available on Android. On iOS, the call will code `UNAVAILABLE_ON_IOS`**.**

Native storage

`setAppGroup(groupName)`

```
static Future<void> setAppGroup(String? groupName)
```

Sets the App Group for the iOS SDK. On Android, the method is a no-op. Despite the nullable Dart signa must be a non-empty string; `null` **will return an** `AltcraftException` **with code** `ERR`**.**

Parameter	Type	Description
`groupName`	`String?`	**App Group identifier, e.g.,** `group.altcraft.flutter.example`**.**

`setUserDefaultsValue(suiteName, key, value)`

```
static Future<void> setUserDefaultsValue(String? suiteName, String key, String? value)
```

Saves a string value to native storage. On iOS, `UserDefaults` **is used; if** `suiteName` **is specified, the App Gr** `suiteName` **is ignored and the value is saved to the plugin's** `SharedPreferences` **(** `altcraft_sdk`**). If** `value == null`

Parameter	Type	Description
`suiteName`	`String?`	**App Group / suite name for iOS. Ignored on Android.**
`key`	`String`	**Storage key. Must not be** `null`**.**
`value`	`String?`	**Value to save.** `null` **removes the key.**

class AltcraftConfig

```
class AltcraftConfig {
  final String apiUrl;
  final String? rToken;
  final AppInfo? appInfo;
  final List<String>? providerPriorityList;
  final bool? enableLogging;

  // Android-only
  final int? icon;
  final List<String>? pushReceiverModules;
  final String? pushChannelName;
  final String? pushChannelDescription;
}
```

Altcraft SDK configuration class. In `toMap()`**, Android-only fields are passed with the** `android_` **prefix:** `an` `android_pushChannelName`**,** `android_pushChannelDescription`**.**

Field	Type	Required	Description
`apiUrl`	`String`	**Yes**	**Altcraft API URL.**
`rToken`	`String?`	**No**	**Role token for scenarios without JWT.**
`appInfo`	`AppInfo?`	**No**	**Application metadata.**
`providerPriorityList`	`List<String>?`	**No**	**Push provider priority. Index** `0` **is the highest priority.**
`enableLogging`	`bool?`	**No**	**Enables/disables native SDK logs.**
`icon`	`int?`	**No**	**Notification icon resource ID. Used only on Android.**
`pushReceiverModules`	`List<String>?`	**No**	**Packages of modules with overridden PushReceiver. Used only on Android.**
`pushChannelName`	`String?`	**No**	**Push notification channel name. Used only on Android.**
`pushChannelDescription`	`String?`	**No**	**Push notification channel description. Used only on Android.**

`AppInfo`**:**

```
class AppInfo {
  final String appID;
  final String appIID;
  final String appVer;
}
```

Field	Type	Description
`appID`	`String`	**Application identifier.**
`appIID`	`String`	**Application installation identifier.**
`appVer`	`String`	**Application version.**

Data Classes

`SdkEvent`

```
class SdkEvent {
  final String function;
  final int? code;
  final String message;
  final String type;
  final Map<String, dynamic>? value;
  final String? timestamp;
}
```

Universal SDK event, delivered to Dart via `Stream<SdkEvent>`.

Field	Type	Description
`function`	`String`	**Name of the SDK function that triggered the event.**
`code`	`int?`	**Event code.**
`message`	`String`	**Event message.**
`type`	`String`	**`event`, `error`, `retryError`.**
`value`	`Map<String, dynamic>?`	**Additional data, e.g., `response_with_http_code`.**
`timestamp`	`String?`	**Timestamp in `yyyy-MM-dd HH:mm:ss.SSS` format.**

`TokenData`

```
class TokenData {
  final String provider;
  final String token;
}
```

Device provider push token data.

Field	Type	Descri
`provider`	`String`	**Provider identifier.**
`token`	`String`	**Push token value.**

`ResponseWithHttpCode`

```
class ResponseWithHttpCode {
  final int httpCode;
  final Map<String, dynamic>? response;
}
```

API response wrapper with HTTP code. `ResponseWithHttpCode.fromMap()` understands both top-level keys: `ht` `response` remains in the format returned by the native bridge. In the current bridge, nested response fie

Dart Field	Type	Description
`httpCode`	`int`	**HTTP response code. If no HTTP code is in the map, it will be `0`.**
`response`	`Map<String, dynamic>?`	**Response body or `null`.**

Main fields of `response`:

Key	Type	Description
`error`	`int?`	**Internal server error code (`0` — no errors).**
`error_text`	`String?`	**Error text.**
`profile`	`Map<String, dynamic>?`	**Profile and subscription data.**

Nested `profile` structure:

Key	Type	
`id`	`String?`	**Profile identifier.**
`status`	`String?`	**Profile status.**
`is_test`	`bool?`	**Test mode flag.**
`subscription`	`Map<String, dynamic>?`	**Subscription data.**

Nested `subscription` structure:

Key	Type	Description
`subscription_id`	`String?`	**Subscription identifier at the provider.**
`hash_id`	`String?`	**Subscription hash.**
`provider`	`String?`	**Subscription provider.**
`status`	`String?`	**Subscription status (`subscribed`, `suspended`, `unsubscribed`).**
`fields`	`Map<String, dynamic>?`	**System and custom fields.**
`cats`	`List<dynamic>?`	**Subscription categories.**

`CategoryData`

```
class CategoryData {
  final String? name;
  final String? title;
  final bool? steady;
  final bool? active;
}
```

Subscription category. For sending to `pushSubscribe`, `pushSuspend`, or `pushUnSubscribe`, pass `CategoryData(...)` required fields.

Field	Type	Description
`name`	`String?`	**Category name.**
`title`	`String?`	**Category title. Usually populated in SDK responses.**
`steady`	`bool?`	**Stability flag. Usually populated in SDK responses.**
`active`	`bool?`	**Category active status.**

`Subscription`

```
class Subscription {
  final String type;
  final int resourceId;
  final String? status;
  final int? priority;
  final String? email;
  final String? phone;
  final String? provider;
  final String? subscriptionId;
  final Map<String, dynamic>? customFields;
  final List<dynamic>? cats;
  final Map<String, dynamic>? ccData;
  final String? ccChannel;
}
```

Subscription model for `mobileEvent()`. `toMap()` adds both `type` and `channel` keys; the native bridge reads (and `type` as fallback.

Factory constructors:

```
Subscription.email({
  required int resourceId,
  required String email,
  String? status,
  int? priority,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
})

Subscription.sms({
  required int resourceId,
  required String phone,
  String? status,
  int? priority,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
})

Subscription.push({
  required int resourceId,
  required String provider,
  required String subscriptionId,
  String? status,
  int? priority,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
})

Subscription.ccData({
  required int resourceId,
  required String channel,
  required Map<String, dynamic> ccData,
  String? status,
  int? priority,
  Map<String, dynamic>? customFields,
  List<dynamic>? cats,
})
```

TIP

For `mobileEvent(subscription: ...)`, pass `subscription.toMap()`. If the required fields for the subscription are assemble the subscription model and will send the event without it.

`UTM`

```
class UTM {
  final String? source;
  final String? medium;
  final String? campaign;
  final String? content;
  final String? keyword;
  final String? temp;
}
```

UTM parameter container for mobile events. Passed to `mobileEvent()` as `UTM(...).toMap()`.

ActionFieldBuilder

```
class ActionFieldBuilder {
  final String key;

  Map<String, dynamic> set(dynamic value);
  Map<String, dynamic> unset(dynamic value);
  Map<String, dynamic> incr(dynamic value);
  Map<String, dynamic> add(dynamic value);
  Map<String, dynamic> delete(dynamic value);
  Map<String, dynamic> upsert(dynamic value);
}

ActionFieldBuilder actionField(String key);
```

Builder for structured profile field management. Produces a map compatible with the native SDK.

Method	Description
`set(value)`	**Set the field to the given value.**
`unset(value)`	**Remove the field value.**
`incr(value)`	**Increment a numeric field by the given value.**
`add(value)`	**Add a value to a set/array field.**
`delete(value)`	**Remove a value from a set/array field.**
`upsert(value)`	**Set if the field is missing; update if it exists.**

```
final profileFields = <String, dynamic>{
  ...actionField('_fname').set('Ivan'),
  ...actionField('login_count').incr(1),
  ...actionField('visited_pages').add('/settings'),
};

await AltcraftSDK.updateProfileFields(profileFields: profileFields);
```

Exceptions

`AltcraftException`

```
class AltcraftException implements Exception {
  final String code;
  final String message;
  final Object? details;
}
```

Exception thrown by the Dart wrapper on `MethodChannel` errors. Check `code` and `message`.

Field	Type	Description
`code`	`String`	**Error code (`ERR`, `SDK_ERROR`, `UNAVAILABLE_ON_IOS`, `ALTCRAFT_INIT_INVALID_CONFIG`, etc.).**
`message`	`String`	**Error message.**
`details`	`Object?`	**Additional data, if the native bridge passed it.**

```
try {
  await AltcraftSDK.takePush(const {'_uid': 'push-message-uid-0001'});
} on AltcraftException catch (e) {
  if (e.code == 'UNAVAILABLE_ON_IOS') {
    print('This method is Android-only');
    return;
  }
  print('SDK error (${e.code}): ${e.message}');
}
```

`SdkNotInitializedException`

```
class SdkNotInitializedException extends AltcraftException
```

The class is exported by the package and may be used by custom/test implementations of the platform `MethodChannel` implementation, native-side errors are converted to a regular `AltcraftException`, so in the a check `e.code`.

Constants

Push notification providers:

Provider	
FCM (Android)	`android-firebase`
HMS (Android)	`android-huawei`
RuStore (Android)	`android-rustore`
APNS (iOS)	`ios-apns`
FCM (iOS)	`ios-firebase`
HMS (iOS)	`ios-huawei`

Subscription statuses:

Status	
Subscribed	`subscribed`
Unsubscribed	`unsubscribed`
Suspended	`suspended`

Provider Setup

Provider setup is performed according to the native SDK instructions for each platform. More details ca
SDK usage instructions, and below are links to the documentation for native provider setup.

Android:

- **Firebase Cloud Messaging**
- **Huawei Mobile Services**
- **RuStore**

iOS:

- **Apple Push Notification Service**
- **Firebase Cloud Messaging**
- **Huawei Mobile Services**

INFO: Important

In Flutter SDK, push providers and JWT are registered natively. The Dart API is used for SDK initializa
events, receiving SDK events, and auxiliary operations.