Destination Info

- Accepts Identify, and Track calls
- Refer to it as **MoEngage** in the Integrations object

Components

Browser iOS Android Connection Modes

Device-mode	**Cloud-mode**
✓ Web	✓ Web
✓ Mobile	✓ Mobile
⊘ Server	✓ Server

MoEngage is an Intelligent Customer Engagement Platform. MoEngage allows brands to personalize every customer interaction and drive better engagement, retention, loyalty and lifetime value.

The MoEngage and Segment integration allows you to send the users you have tracked on Segment, along with their route data, to MoEngage for further targeting and campaigning. This Destination enables you to:

- **Import data from Segment to MoEngage**: Moengage offers a side-by-side (device mode) SDK integration for your Android, iOS, and web applications and a server-to-server integration for your backend services.
- **Sync Twilio Engage (cohorts)**: Send Segment Cohorts to MoEngage for use in MoEngage Segments and campaigns.

The MoEngage Destination source code is open-sourced and freely available on GitHub for anyone to view:

CONNECTION MODE	MAINTAINED BY	GITHUB LINK
iOS	MoEngage	MoEngage-Segment-Swift
Android	MoEngage	moengage-segment-integration
Web	Segment	analytics.js-integrations

Getting Started

Once you add the Segment-MoEngage library to your app, you can enable MoEngage from the Segment App. These new settings can take up to an hour to propagate to your existing users. For new users, it'll be instantaneous!

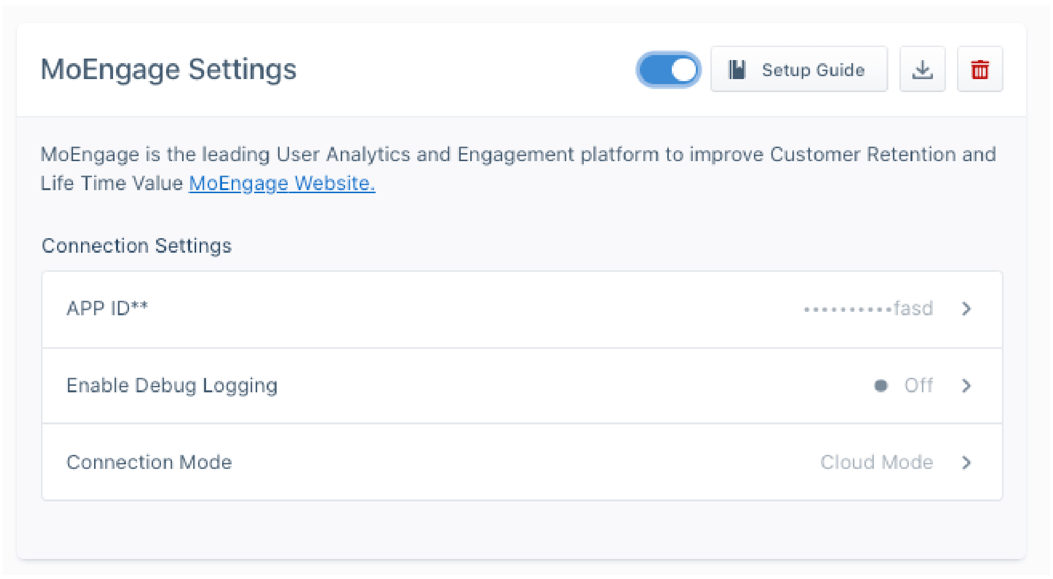
The Segment-MoEngage Integration is a bundled integration, meaning it requires that you add a client-side integration to your app.

Setup MoEngage in your Segment Workspace

To setup MoEngage do the following :

1. First get your key(AppID) from the MoEngage dashboard. Navigate to `Dashboard --> Settings --> App --> General`.
2. Go to your **Segment workspace**, go to **Destinations** and select **MoEngage**.
3. Enable the MoEngage Destination.
4. Go to the MoEngage Settings and enter the MoEngage AppID, obtained in **Step1**.
5. Save the changes.
6. Make sure you set the **Connection Mode** to `Device Mode`. MoEngage requires this setting to use of features like push notifications and any in-app features of the MoEngage SDK.

These new settings will take up to an hour to propagate to your existing users. For new users it'll be instantaneous! Segment-MoEngage Integration is a bundled integration, requires client side integration.


MoEngage Settings

MoEngage is the leading User Analytics and Engagement platform to improve Customer Retention and Life Time Value MoEngage Website.

Connection Settings

APP ID**	••••••••••fasd
Enable Debug Logging	● Off
Connection Mode	Cloud Mode

Identify

Use Identify to track user-specific attributes. This is the same as tracking user attributes on MoEngage. MoEngage supports traits supported by Segment as well as custom traits. If you set `traits.id`, MoEngage sets that as the Unique ID for that user.

MoEngage supports anonymous identifiers in Device-mode only. If you use the MoEngage destination in Cloud-mode, use a known user identifier.

The Identify method follows the format below:

```
analytics.identify('12090000-00001992', {
  name: 'John Doe',
  email: 'john.doe@example.com'
});
```

Track

Use track to track events and user behavior in your app.

```
analytics.track('Article Completed', {
  title: 'How to Create a Tracking Plan',
  course: 'Intro to Analytics',
});
```

This will send the event to MoEngage with the associated properties. Tracking events is essential and will help you create segments for engaging users.

Reset

If your app or website supports the ability for a user to logout and login with a new identity, then you'll need to call reset method in `analytics.js`.

```
analytics.reset();
```

iOS

To get started with MoEngage on iOS, first integrate your app with the MoEngage-Segment-Swift library. You can integrate MoEngage and Segment with Swift Package Manager.

Note: This document covers the integration with analytics-swift, if you have integrated with analytics-ios refer this documentation for details on how to integrate.

To install with SPM use the MoEngage-Segment-Swift library and set the branch as master or version as 1.0.0 and above.

Configure the Segment SDK:

Now head to the App Delegate file, and setup the Segment SDK by

1. Importing `Segment`, `Segment_MoEngage` and `MoEngageSDK`.
2. Initialize `MoEngageSDKConfig` object and call `initializeDefaultInstance` method of `MoEngageInitializer`.
3. Initialize `MoEngageDestination` as shown below:

Under your Analytics-Swift library setup, add the MoEngage plugin using the `analytics.add(plugin: ...)` method. The MoEngage dashboard now tracks all of your events.

```
let analytics = Analytics(configuration: Configuration(writeKey: "<YOUR WRITE KEY>")
                    .flushAt(3)
                    .trackApplicationLifecycleEvents(true))
analytics.add(plugin: MoEngageDestination())
```

Configure the MoEngage SDK:

```
 import Segment_MoEngage
 import MoEngageSDK
 ...
 func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOpti
onsKey:  Any]?) -> Bool {
 ...

     let sdkConfig = MoEngageSDKConfig(withAppID: "YOUR APP ID")
     MoEngageInitializer.shared.initializeDefaultInstance(sdkConfig: sdkConfig)
 ...
 }
```

Tracking User Attribute

User attributes are specific traits of a user, like email, username, mobile, gender etc. **identify** lets you tie a user to their actions and record traits about them. It includes a unique User ID and any optional traits you know about them.

```
 Analytics.main.identify("a user's id", traits: @["email":"a user's email address"])
```

Read more about identify calls.

Tracking Events

Segment uses event tracking to track user behavior in an app. `track` calls let you record the actions your users perform. Every action triggers an “event”, which can also have associated attributes.

```
 Analytics.main.track("Item Purchased", properties: @["item":"Sword of Heracles"])
```

Read more about track calls.

Reset Users

The `reset` method clears the SDK’s internal stores for the current user. This is useful for apps where users can log in and out with different identities over time.

```
Analytics.main.reset()
```

Read more about the reset method.

Install / Update Differentiation

Since your app might already be on the App Store, you must specify whether your app update would be an `UPDATE` or an `INSTALL`. To differentiate between those, use one of the method below:

```
//For new Install call following
MoEngageSDKAnalytics.sharedInstance.appStatus(.install)

//For an app update call following
MoEngageSDKAnalytics.sharedInstance.appStatus(.update)
```

Read more on install/update differentiation.

Set the data center

By default, the data center setting in the SDK is set to `data_center_01`. Follow the steps in the MoEngage - Data Center to update the data center value. If not set correctly, several features of the MoEngage SDK will not function.

MoEngage iOS SDK Features

Along with tracking your user's activities, you can use the MoEngage iOS SDK for more effective user engagement:

Push Notifications:

Push Notifications are a great way to keep your users engaged and informed about your app. You have following options while implementing push notifications in your app:

Segment Push Implementation:

1. In your application's application:didRegisterForRemoteNotificationsWithDeviceToken: method, add the following:

   ```
   Analytics.main.registeredForRemoteNotifications(deviceToken: deviceToken)
   ```

2. In your application's application:didReceiveRemoteNotification: method, add the following:

   ```
     Analytics.main.receivedRemoteNotification(userInfo: userInfo)
   ```

3. If you integrated the application:didReceiveRemoteNotification:fetchCompletionHandler: in your app, add the following to that method:

   ```
     Analytics.main.receivedRemoteNotification(userInfo: userInfo)
   ```

4. If you implemented handleActionWithIdentifier:forRemoteNotification:, add the following to that method:

   ```
     Analytics.main.handleAction(identifier: identifier, userInfo: userInfo)
   ```

MoEngage Push Implementation: For information about the MoEngage push implementation, see **Push Notifications** in the MoEngage documentation.

In-app messaging

In-App Messages are custom views which you can send to a set of users to show custom messages, give new offers, or direct to a specific page. For more information about In-app messaging, see MoEngage - In-App NATIV.

Cards

Create targeted or automated App Inbox/NewsFeed messages that can be grouped into various categories, and target your users with different updates or offers that can stay in the Inbox/Feed over a designated period of time. For more information about cards, see MoEngage - Cards.

Compliance

To make the App compliant with policies (such as GDPR) while using MoEngage's SDK, follow the instructions in this doc.

Segment Docs

For more info on using **Segment for iOS** refer to **Developer Docs** provided by Segment.

Android

To use MoEngage in an Android app, you must perform the following steps to set up your environment.

maven central 2.4.2

To enable the full functionality of MoEngage (like Push Notifications, InApp Messaging), complete the following steps in your Android app.

Note: This document covers the integration with analytics-kotlin, if you have integrated with analytics-android refer this documentation for details on how to integrate.

Adding the MoEngage Dependency

Along with the Segment dependency, add the below dependency in your `build.gradle` file.

```
implementation("com.moengage:moengage-segment-kotlin-destination:$sdkVersion") {
        transitive = true
    }
```

with `$sdkVersion` replaced by the latest version of the MoEngage SDK.

The MoEngage SDK depends on the below Jetpack libraries provided by Google for its functioning, make you add them if not done already.

```
    implementation("androidx.core:core:1.6.0")
    implementation("androidx.appcompat:appcompat:1.3.1")
    implementation("androidx.lifecycle:lifecycle-process:2.4.0")
```

Refer to the SDK Configuration documentation to know more about the build config and other libraries used by the SDK.

Register MoEngage with Segment SDK

After adding the dependency, you must register the integration with Segment SDK. To do this, import the MoEngage integration:

```
import com.segment.analytics.kotlin.destinations.moengage.MoEngageDestination
```

Add the following line:

```
Analytics("<YOUR WRITE KEY>", context)
  .add(MoEngageDestination(application))
```

Initialize the MoEngage SDK

Copy the APP ID from the Settings Page `Dashboard --> Settings --> App --> General` and initialize the MoEngage SDK in the onCreate method of the `Application` class

> **Note:** MoEngage recommend that you initialize the SDK on the main thread inside `onCreate()` and not create a worker thread and initialize the SDK on that thread.

```
// this is the instance of the application class and "YOUR_APP_ID" is the APP ID from the dashboard.
      val moEngage = MoEngage.Builder(this, "YOUR_APP_ID")
          .enablePartnerIntegration(IntegrationPartner.SEGMENT)
          .build()
      MoEngage.initialiseDefaultInstance(moEngage)
```

Exclude MoEngage Storage File from Auto-Backup

Auto backup service of Android periodically backs up the Shared Preference file, Database files, and so on.

For more information, refer to Auto Backup.

As a result of the backup, MoEngage SDK identifiers are backed up and restored after re-install. The restoration of the identifier results in your data being corrupted and the user not being reachable using push notifications.

To ensure data isn't corrupted after a backup is restored, opt-out of MoEngage SDK storage files.

Refer to the documentation for further details.

Install or update differentiation

This is required for migrations to the MoEngage Platform so the SDK can determine whether the user is a new user on your app, or an existing user who updated to the latest version.

If the user was already using your application and has just updated to a new version which has the MoEngage SDK, below is an example call:

```
 MoEAnalyticsHelper.setAppStatus(context, AppStatus.UPDATE)
```

If this is a fresh install:

```
MoEAnalyticsHelper.setAppStatus(context, AppStatus.INSTALL)
```

MoEngage Android SDK Features

Configure Push Notifications

Copy the Server Key from the FCM console and add it to the MoEngage Dashboard. To upload it, go to the Settings Page `Dashboard --> Settings --> Channel --> Push --> Mobile Push --> Android` and add the Server Key and package name. **Please make sure you add the keys both in Test and Live environment.**

Add meta information for push notification

To display push notifications, some metadata regarding the notification is required. For example, the small icon and large icon drawables are mandatory.

Refer to the MoEngage - NotificationConfig API reference for all the possible options.

Use the `configureNotificationMetaData()` to pass on the configuration to the SDK.

```
    val moEngage = MoEngage.Builder(this, "YOUR_APP_ID")
        .enablePartnerIntegration(IntegrationPartner.SEGMENT)
        .configureNotificationMetaData(NotificationConfig(
            smallIcon = R.drawable.small_icon,
            largeIcon = R.drawable.large_icon
        ))
        .build()
    MoEngage.initialiseDefaultInstance(moEngage)
```

Configuring Firebase Cloud Messaging

For showing Push notifications there are 2 important steps:

1. Registration for Push, for example generating push token.
2. Receiving the Push payload from Firebase Cloud Messaging(FCM) service and showing the notification on the device.

Push registration and receiving handled by the application

Opt-out of MoEngage registration

To opt-out of MoEngage token registration mechanism disable token registration while configuring FCM in the `MoEngage.Builder` as shown below

```
    val moEngage = MoEngage.Builder(this, "YOUR_APP_ID")
        .enablePartnerIntegration(IntegrationPartner.SEGMENT)
        .configureNotificationMetaData(NotificationConfig(
            smallIcon = R.drawable.small_icon,
            largeIcon = R.drawable.large_icon
        ))
        .configureFcm(FcmConfig(false))
        .build()
    MoEngage.initialiseDefaultInstance(moEngage)
```

Pass the push token to the MoEngage SDK

The Application must pass the Push Token received from FCM to the MoEngage SDK for the MoEngage platform to send out push notifications to the device. Use the below API to pass the push token to the MoEngage SDK.

```
MoEFireBaseHelper.getInstance().passPushToken(applicationContext,token)
```

Please make sure token is passed to MoEngage SDK whenever push token is refreshed and on application update. Passing token on application update is important for migration to the MoEngage Platform.

Passing the Push payload to the MoEngage SDK

To pass the push payload to the MoEngage SDK call the MoEngage API from the `onMessageReceived()` from the Firebase receiver. Before passing the payload to the MoEngage SDK you should check if the payload is from the MoEngage platform using the helper API provided by the SDK.

```
if (MoEPushHelper.getInstance().isFromMoEngagePlatform(remoteMessage.data)) {
    MoEFireBaseHelper.getInstance().passPushPayload(applicationContext, remoteMessage.data)
 } else {
    // your app's business logic to show notification
 }
```

Push Registration and Receiving handled by SDK

Add the below code in your manifest file.

```
<service android:name="com.moengage.firebase.MoEFireBaseMessagingService">
  <intent-filter>
  <action android:name="com.google.firebase.MESSAGING_EVENT" />
  </intent-filter>
</service>
```

When the MoEngage SDK handles push registration, it optionally provides a callback to the Application whenever a new token is registered or the token is refreshed.

An application can get this callback by implementing `FirebaseEventListener` and registering for a callback in the Application class `onCreate()` using `MoEFireBaseHelper.getInstance().addEventListener()`

Refer to the MoEngage - API reference for more details on the listener.

Callbacks

Segment recommends that you add the callbacks in the onCreate() of the Application class since these callbacks can be triggered even when the application is in the background.

Token Callback

When MoEngage SDK handles push registration, it optionally provides a callback to the application whenever a new token is registered or the token is refreshed. To get the token callback implement the TokenAvailableListener and register for the callback using MoEFireBaseHelper.getInstance().addTokenListener().

Non-MoEngage Payload

If you're using the receiver provided by the SDK in your application's manifest file, SDK provides a callback in case a push payload is received for any other server apart from MoEngage Platform. To get a callback implement the NonMoEngagePushListener and register for the callback using MoEFireBaseHelper.getInstance().addNonMoEngagePushListener().

Declare and configure Rich Landing Activity:

A rich landing page can be used to open a web URL inside the app through a push campaign.

The configuration below is only required if you want to add a parent activity to the Rich landing page. If not, you can move to the next section. To use a rich landing page you need to add the below code in the AndroidManifest.xml

Add the following snippet and replace `[PARENT_ACTIVITY_NAME]` with the name of the parent activity; `[ACTIVITY_NAME]` with the activity name which should be the parent of the Rich Landing Page

```
<activity
  android:name="com.moe.pushlibrary.activities.MoEActivity"
  android:label="[ACTIVITY_NAME]"
  android:parentActivityName="[PARENT_ACTIVITY_NAME]" >
</activity>
```

You are now all set up to receive push notifications from MoEngage. For more information on features provided in MoEngage Android SDK refer to the following links:

- Push Notifications
- Location Triggered
- In-App messaging
- Notification Center
- API Reference
- Compliance
- Release Notes

Identify

Use Identify to track user-specific attributes. This is the same as tracking user attributes on MoEngage. MoEngage supports traits supported by Segment as well as custom traits. If you set traits.id, MoEngage sets that as the Unique ID for that user.

MoEngage supports anonymous identifiers in Device-mode only. If you use the MoEngage destination in Cloud-mode, use a known user identifier.

Track

Use track to track events and user behavior in your app. This will send the event to MoEngage with the associated properties. Tracking events is essential and will help you create segments for engaging users.

Reset

If your app supports the ability for a user to logout and login with a new identity, then you'll need to call reset for the Analytics client.

Sample Implementation

Refer to this GitHub repository for sample implementation

Web

The MoEngage WebSDK offers the ability to send push notifications to Google Chrome, Opera and Firefox browsers. Complete the following steps after you've configured Segment's `analytics.js.`

Integration

1. Setup your MoEngage Web SDK settings at MoEngage Dashboard

Configure the web settings on the MoEngage dashboard to start using MoEngage <> Segment integration.

If you have selected `HTTPS` mode of integration in the settings, complete the following steps:

2 Set up for HTTPS websites

2.a Download the required files (HTTPS)

For HTTPS Web Push to work, you need to host two files in the `root` directory of your web server. These two files will be available for you to download at the web settings page.

- `manifest.json`
- `serviceworker.js`

Note: Please make sure the name of the serviceworker file is `serviceworker.js`. Please contact MoEngage support at support@moengage.com if you wish to have some other name for the serviceworker file.

2.b Add link to manifest in HTML (HTTPS)

Add the following line in the <head> tag of your page.

```
<head>
  ...
 <link rel="manifest" href="/manifest.json">
  ...
</head>
```

2.c Use your existing manifest or serviceworker file (HTTPS)

If you already have these files,

1. **Manifest** - Add the sender ID you saved on MoEngage dashboard as the `gcm_sender_id`. If you've used `MoEngage Shared Project` while setting up, your sender id is `540868316921`. Edit your `manifest.json` as follows:

   ```
     {
    ...
    "gcm_sender_id": "GCM_SENDER_ID",
    ...
     }
   ```

3. **Service Worker** - Add the following line to the top of your `serviceworker.js` file

```
importScripts("//cdn.moengage.com/webpush/releases/serviceworker_cdn.min.latest.js?date="+
new Date().getUTCFullYear()+""+new Date().getUTCMonth()+""+new Date().getUTCDate());
```

Identify

Use Identify to track user specific attributes. This is equal to tracking user attributes on MoEngage. MoEngage supports traits supported by Segment as well as custom traits.

MoEngage supports anonymous identifiers in Device-mode only. If you use the MoEngage destination in Cloud-mode, use a known user identifier.

Track

Use track to track events and user behavior in your app. This will send the event to MoEngage with the associated properties. Tracking events is essential and will help you create segments for engaging users.

Reset

If your website supports the ability for a user to logout and login with a new identity, then you'll need to call reset method in `analytics.js`.

Test Mode and Debugging

While updating the MoEngage settings on the Segment Dashboard, you can enable the logging functionality of the MoEngage SDK to see the SDK logs on the browser console. Just set `Enable Debug Logging` to `On` and the SDK loads in debug mode.

Note: When you enable debug mode, the events and attributes of the users send to the `TEST` environment of your MoEngage App.

MoEngage Web SDK Features

For information about optional features, see the documentation below:

- Configure opt in type
- Self-handled opt-ins
- SDK callbacks

On-Site Messaging

On-site Messaging Campaigns allow you to show personalized pop-ups and non-intrusive banners on your website.

Web SDK integration for On-site Messaging will automatically start working on all the pages where the web SDK is integrated.

For more information, refer to Configure and Integrate On-site Messaging.

Web Personalization

Web Personalization is used to personalize the website experience for each user. A few popular use cases for Web Personalization include the home page banner personalization basis user behavior, localizing the website content basis user geography, testing the performance of new page layouts for improved performance, modifying the content shown on any webpage as per the user behavior.

For more information, refer to Configure and Integrate Web Personalization.

Use Twilio Engage with MoEngage

You can send Computed traits and Audiences to MoEngage as custom attributes or custom events.

- Traits and audiences sent using the `identify` call will appear in MoEngage as *Tracked Custom Attributes* with value as *True*.
- Traits and audiences sent using the `track` call will appear in MoEngage as *Tracked User Events*.

When connecting the calculated trait/audience to the MoEngage destination, you may select the method of your choice (or opt to utilise both).

Sync Time

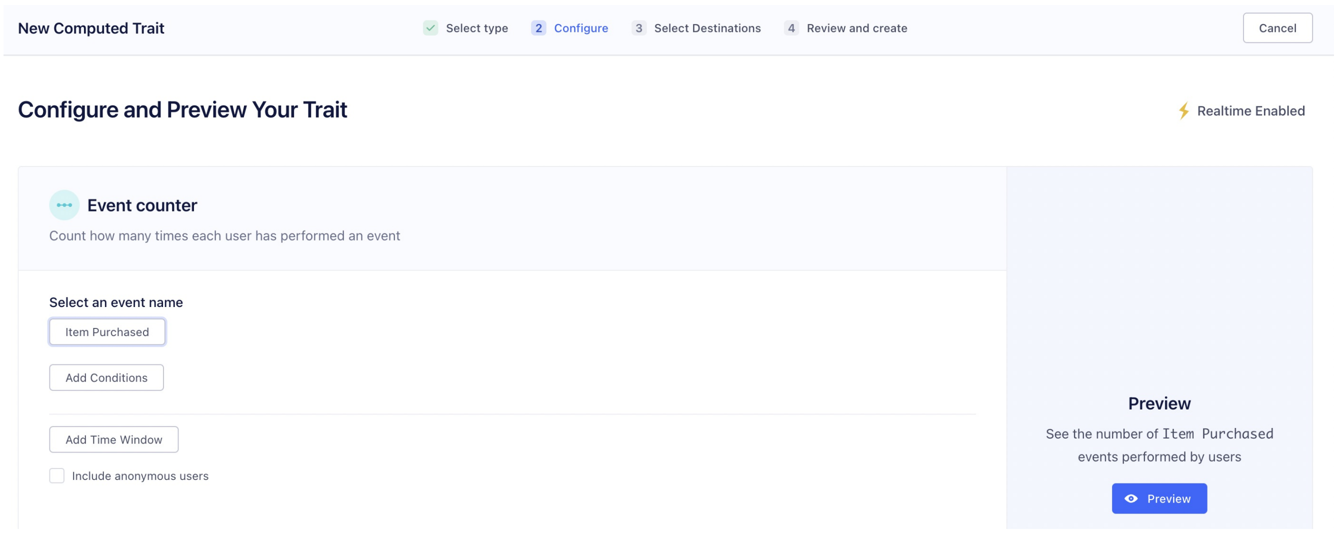
The default integration for MoEngage <> Twilio Engage connection is **Real Time.** But there are some filters that will disqualify the persona from syncing in real-time, including some time-based filters which restrict your audience's size at the time of message send.

Computed Traits using Identify calls

To generate custom attributes in MoEngage, you may provide Computed Traits defined in Engage as `Identify` calls. The Computed Trait's value is used to set the custom attribute.

Create a Segment Computed Trait

1. In Segment, navigate to the *Computed Traits* in *Engage.*
2. Click *New Computed Trait.*
3. Create your computed trait. A lightning bolt in the top corner of the page will indicate if the computation updates in real-time.


New Computed Trait

✓ Select type 2 Configure 3 Select Destinations 4 Review and create

Cancel

Configure and Preview Your Trait

Realtime Enabled

Event counter

Count how many times each user has performed an event

Select an event name

Item Purchased

Add Conditions

Add Time Window

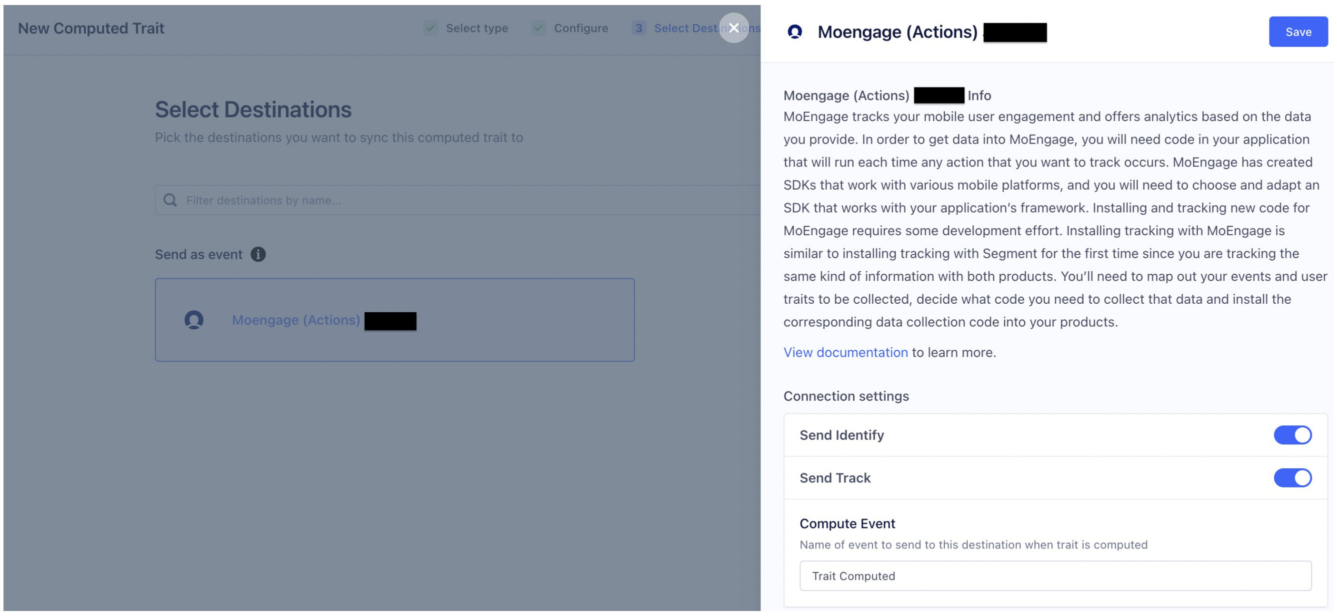
Include anonymous users

Preview

See the number of Item Purchased events performed by users

Preview

4. Next, select *MoEngage* as your destination.


New Computed Trait — ✓ Select type ✓ Configure 3 Select Destinations

Select Destinations

Pick the destinations you want to sync this computed trait to

Filter destinations by name...

Send as event

Moengage (Actions) [redacted]

Moengage (Actions) [redacted] Save

Moengage (Actions) [redacted] Info

MoEngage tracks your mobile user engagement and offers analytics based on the data you provide. In order to get data into MoEngage, you will need code in your application that will run each time any action that you want to track occurs. MoEngage has created SDKs that work with various mobile platforms, and you will need to choose and adapt an SDK that works with your application's framework. Installing and tracking new code for MoEngage requires some development effort. Installing tracking with MoEngage is similar to installing tracking with Segment for the first time since you are tracking the same kind of information with both products. You'll need to map out your events and user traits to be collected, decide what code you need to collect that data and install the corresponding data collection code into your products.

View documentation to learn more.

Connection settings

Send Identify: on

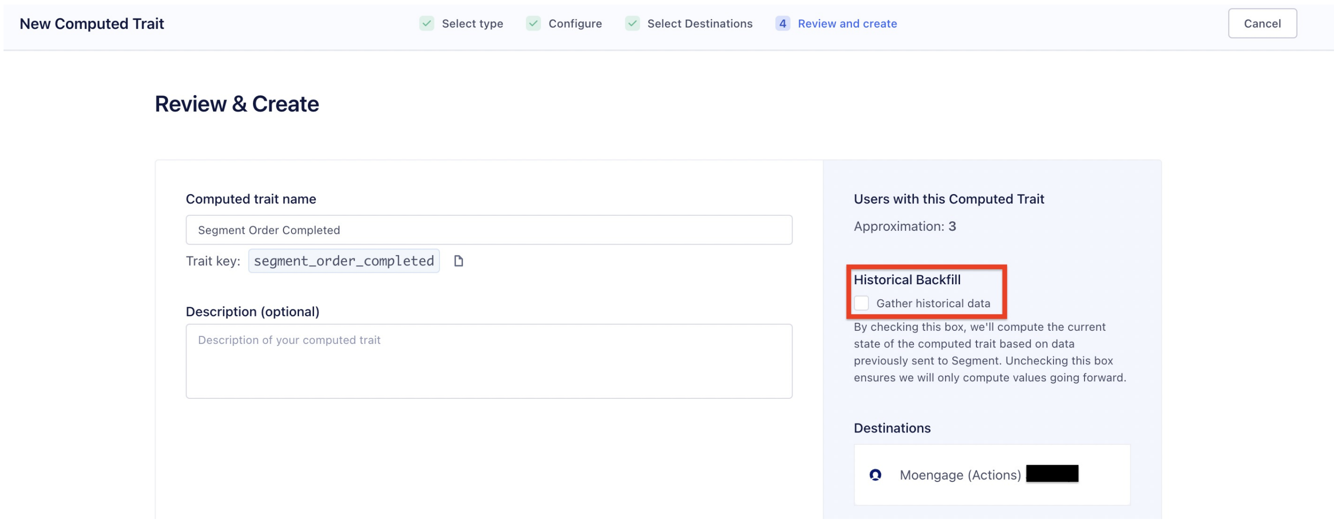
Send Track: on

Compute Event

Name of event to send to this destination when trait is computed

Trait Computed

5. Preview by clicking *Review & Create*.
Note- By default, Segment queries all historical data to set the current value of the computed trait and audience. To omit this data, uncheck *Historical Backfill*.


New Computed Trait — ✓ Select type ✓ Configure ✓ Select Destinations 4 Review and create — Cancel

Review & Create

Computed trait name

Segment Order Completed

Trait key: `segment_order_completed`

Description (optional)

Description of your computed trait

Users with this Computed Trait

Approximation: 3

Historical Backfill

☐ Gather historical data

By checking this box, we'll compute the current state of the computed trait based on data previously sent to Segment. Unchecking this box ensures we will only compute values going forward.

Destinations

Moengage (Actions) [redacted]

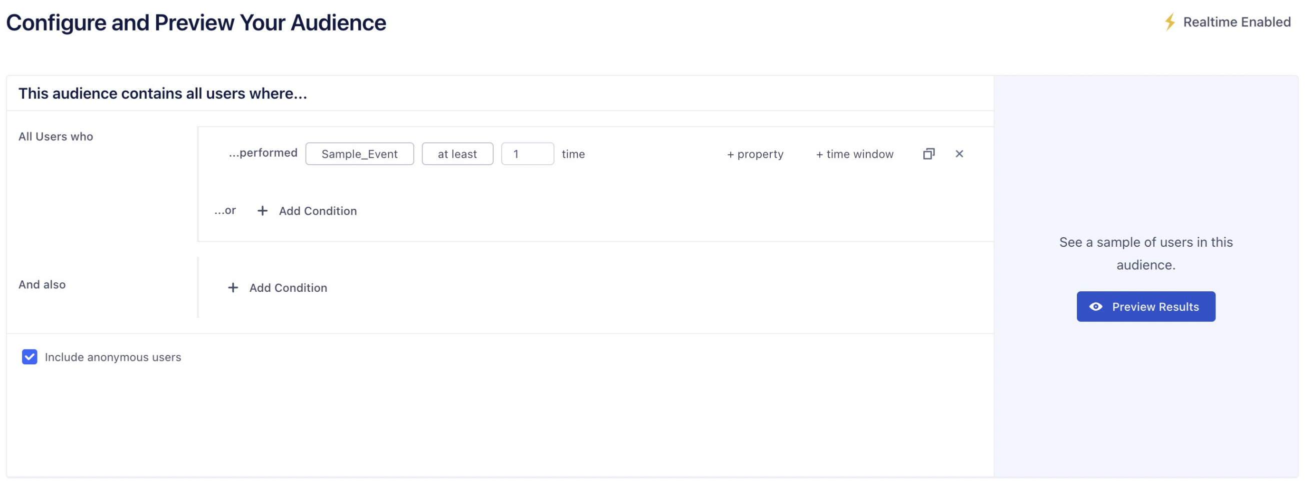
6. In the computed trait, adjust the connection settings based on how you would like your data sent to MoEngage.

Create a Segment Audience

1. In Segment, navigate to the *Audience* in *Engage*

2. Click *New*.

3. Create your audience. A lightning bolt in the top corner of the page will indicate if the computation updates in real-time.


Configure and Preview Your Audience

Realtime Enabled

This audience contains all users where...

All Users who ...performed Sample_Event at least 1 time + property + time window

...or + Add Condition

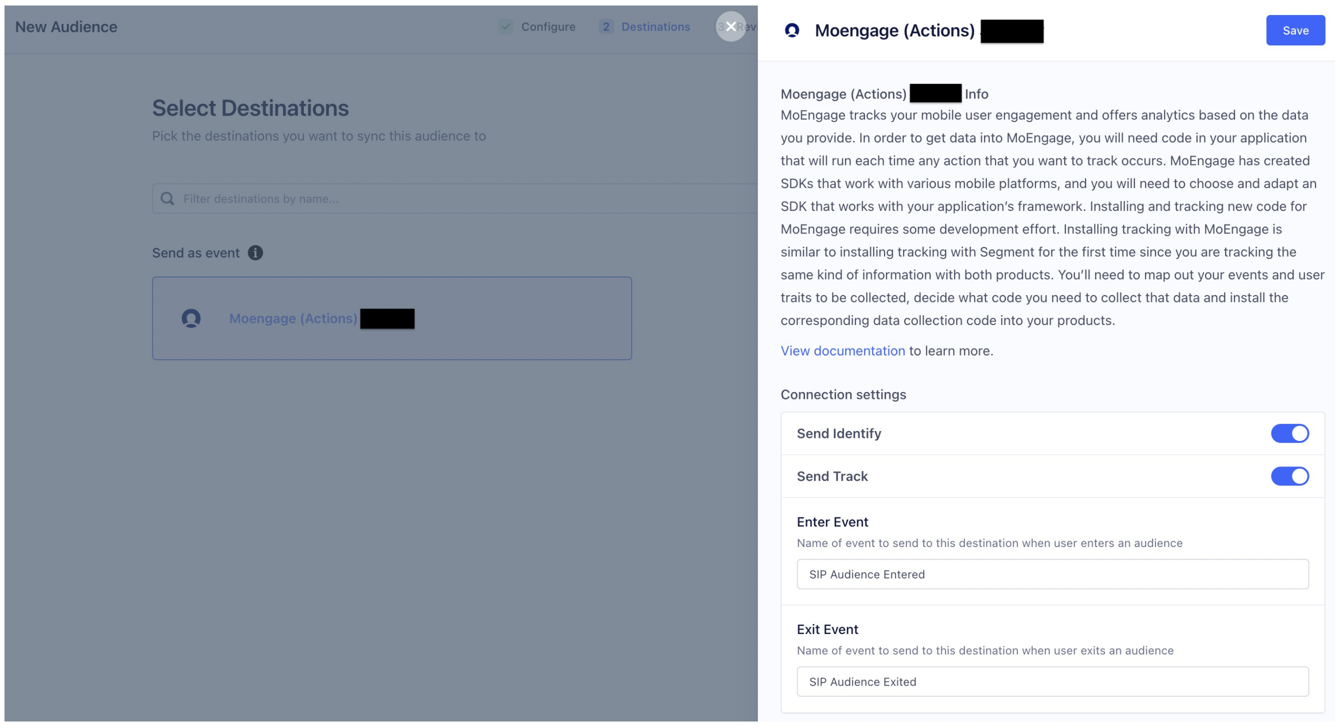
And also + Add Condition

☑ Include anonymous users

See a sample of users in this audience.

Preview Results

4. Next, select *MoEngage* as your destination.


New Audience
Configure
2 Destinations
Select Destinations
Pick the destinations you want to sync this audience to
Filter destinations by name...
Send as event
Moengage (Actions) [redacted]

Moengage (Actions) [redacted]

Save

Moengage (Actions) [redacted] Info

MoEngage tracks your mobile user engagement and offers analytics based on the data you provide. In order to get data into MoEngage, you will need code in your application that will run each time any action that you want to track occurs. MoEngage has created SDKs that work with various mobile platforms, and you will need to choose and adapt an SDK that works with your application's framework. Installing and tracking new code for MoEngage requires some development effort. Installing tracking with MoEngage is similar to installing tracking with Segment for the first time since you are tracking the same kind of information with both products. You'll need to map out your events and user traits to be collected, decide what code you need to collect that data and install the corresponding data collection code into your products.

View documentation to learn more.

Connection settings

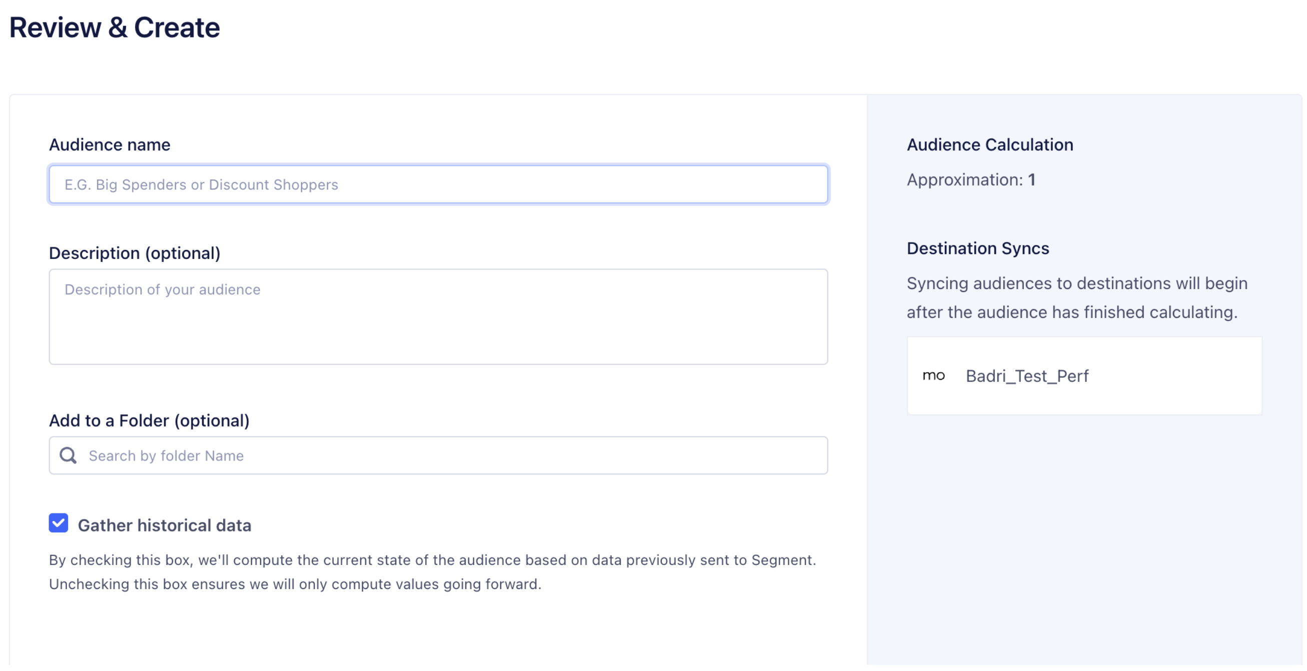
Send Identify ●

Send Track ●

Enter Event
Name of event to send to this destination when user enters an audience
SIP Audience Entered

Exit Event
Name of event to send to this destination when user exits an audience
SIP Audience Exited

5. Preview your audience by clicking *Review & Create*.
Note- By default, Segment queries all historical data to set the current value of the audience. To omit this data, uncheck *Historical Backfill*.


Review & Create

Audience name

E.G. Big Spenders or Discount Shoppers

Description (optional)

Description of your audience

Add to a Folder (optional)

Search by folder Name

☑ Gather historical data

By checking this box, we'll compute the current state of the audience based on data previously sent to Segment. Unchecking this box ensures we will only compute values going forward.

Audience Calculation

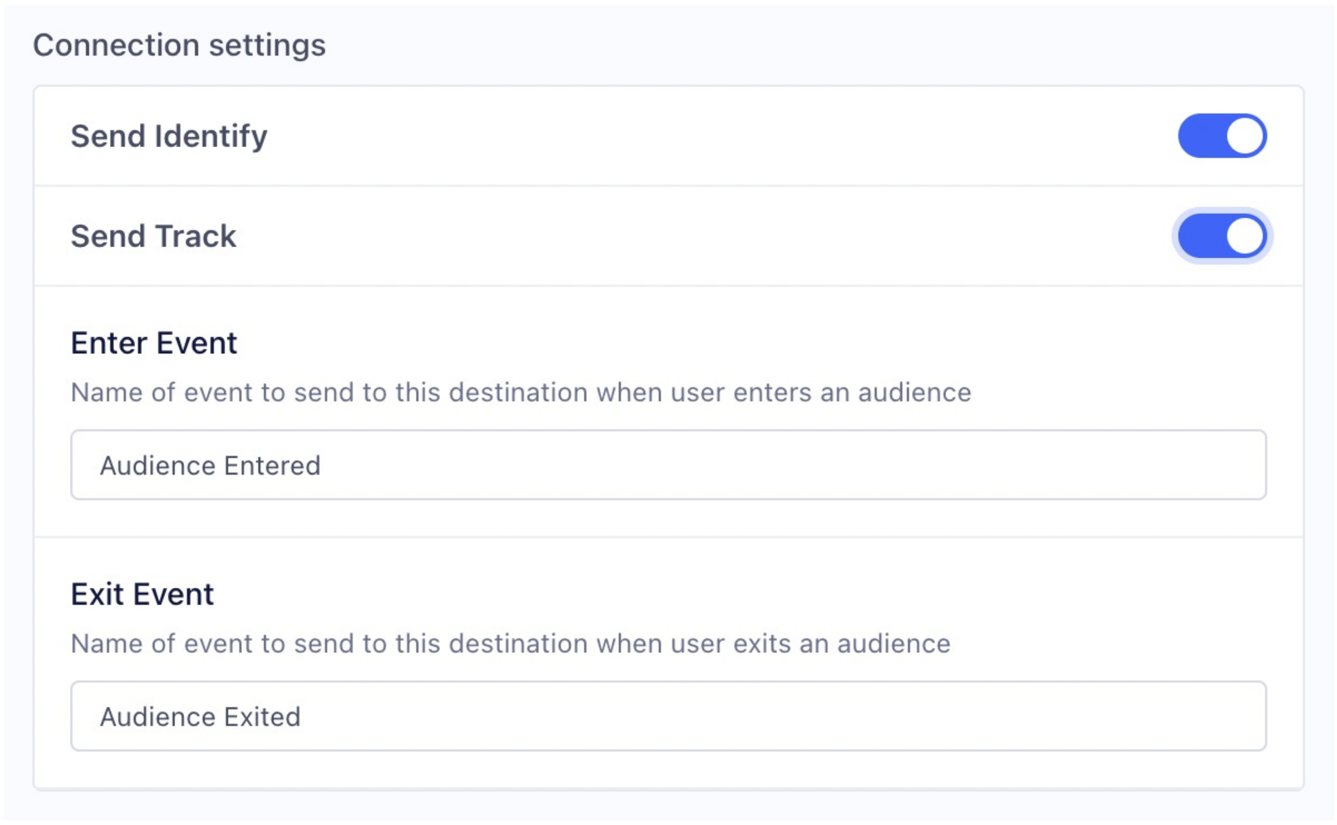
Approximation: **1**

Destination Syncs

Syncing audiences to destinations will begin after the audience has finished calculating.

mo Badri_Test_Perf

6. In the computed trait or audience settings, adjust the connection settings based on how you would like your data sent to MoEngage.


Connection settings

Send Identify: on

Send Track: on

Enter Event

Name of event to send to this destination when user enters an audience

Audience Entered

Exit Event

Name of event to send to this destination when user exits an audience

Audience Exited

Connect Twilio Engage to MoEngage

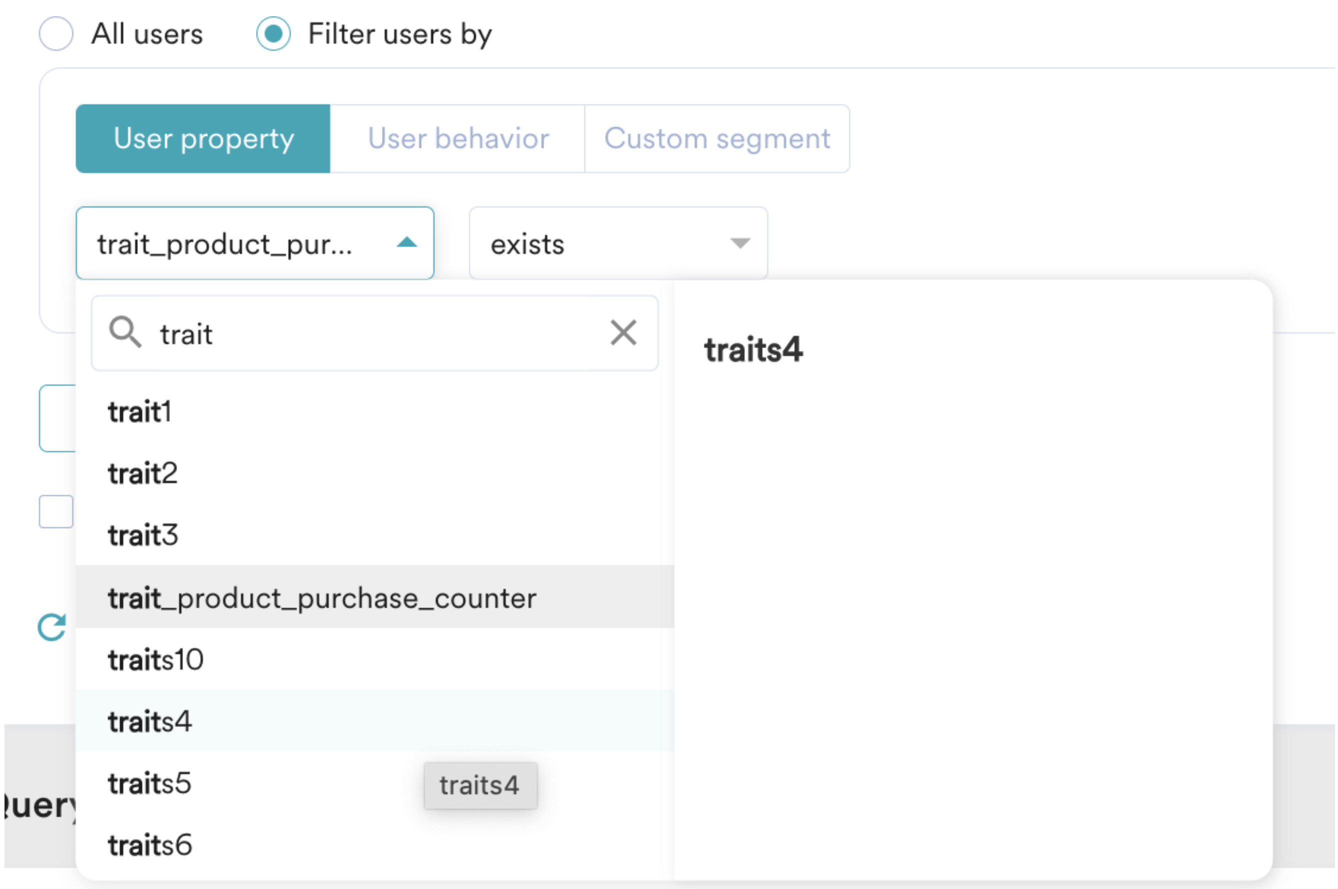
First, link MoEngage to Twilio Engage to send Computed Traits or Audiences. The first time you generate new Computed Trait or Audience, you can choose MoEngage as the destination for the Engage data.

1. Go to the Destinations tab in your space.
2. Search for MoEngage and add the MoEngage Destination to your Space.
3. On the set up screen, enter your `App Id`, `App Key`, and your `Endpoint Region`.

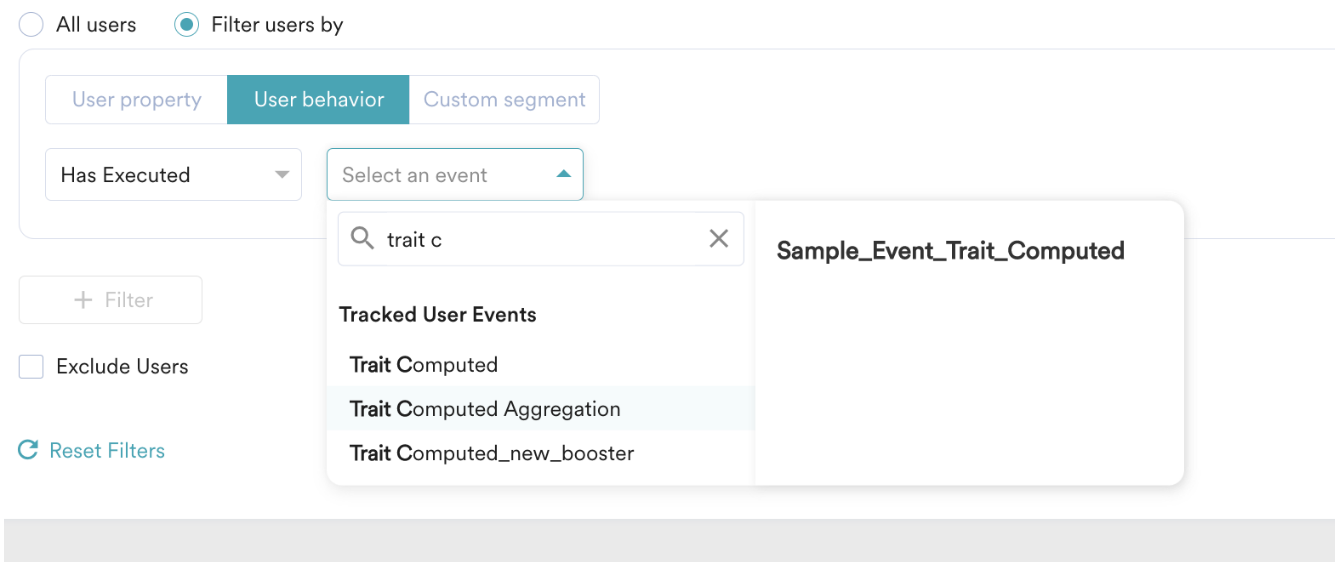
Segment users in MoEngage

To create a segment of these users, navigate to *Segments » Create Segment.* Next, based on which call you used:

- **Identify**: Select *User Property* and select the specific attribute.



- **Track**: Select *User Behaviour* and select the specific event.



Settings

Segment lets you change these destination settings from the Segment app without having to touch any code.

SETTING	DESCRIPTION
APP ID *(required)*	`string`. You can find the APP_ID key under App Settings on the MoEngage dashboard. Please make sure you have uploaded the pem file for sending push on iOS and GCM Key for Android.
Enable Debug Logging	`boolean`, defaults to `FALSE`. If you are in TEST mode and would like debug logs in your browser console, enable this setting. You should not have this option enabled for your production sources.

This page was last modified: 27 Oct 2023

Need support?

Questions? Problems? Need more info? Contact Segment Support for assistance!

Visit our Support page

Help improve these docs!

Edit this page

Request docs change

Was this page helpful?

Yes

No

Get started with Segment

Segment is the easiest way to integrate your websites & mobile apps data to over 300 analytics and growth tools.

Your work e-mail

Request Demo

or

Create free account