Web	Web
Mobile	Mobile
Server	Server

Leanplum helps mobile teams orchestrate multi-channel campaigns — from messaging to the in-app experience — all from a single mobile marketing platform.

Good to know: This page is about the Leanplum Segment destination, which receives data from Segment. There's also a page about the Leanplum Segment source, which sends data *to* Segment!

Getting Started

1. From the Segment web app, click **Catalog**.

2. Search for “Leanplum” in the Catalog, select it, and choose which of your sources to connect the destination to.
3. Navigate to your “App Settings” within the Leanplum UI and open up your “Keys & Settings”
4. Copy the “App ID” into the Segment Settings UI under “Leanplum Application ID”
5. We recommend creating 2 Leanplum Destinations, 1 for production and 1 for development.
6. Copy the “Production” or “Development” value into the corresponding Leanplum destination and add it to the Segment Settings UI under “Leanplum Client Key” depending on your chosen environment.

If you have opted to use our server-side sources, we will begin passing data through our servers or from your users’ devices to Leanplum within minutes after you enable it on your source destinations page.

In order to use Leanplum’s Push Notifications, Messaging channels or A/B testing in your mobile app, you will need to use our iOS or Android SDKs. Read on to find out how to set this up.

iOS

CocoaPods is the dependency manager we use for Objective-C projects. If you already have CocoaPods installed and have a podfile, skip to step 3.

1. Install CocoaPods by running the following command:
   `sudo gem install cocoapods`
   For issues with installing CocoaPods, refer to the cocoapods website
2. Add a podfile. In your terminal, navigate to your app’s directory. Add a podfile to your app by running the following command:
   `pod init`
3. Open your podfile by running the following command:
   `open -a Xcode Podfile`
4. Insert the following line of code into your Podfile:
   `pod 'LeanplumSegment', '~> 1.0.1'`
5. Now, install the SDK by running the following command:
   `pod install`
6. Import the `LeanplumSegment` integration:
   `#import <LeanplumSegment/SEGLeanplumIntegrationFactory.h>`
   Add the following lines into your AppDelegate:

```
String *const SEGMENT_WRITE_KEY = @" [YOUR_SEGMENT_WRITE_KEY] ";
SEGAnalyticsConfiguration *config =
    [SEGAnalyticsConfiguration configurationWithWriteKey:SEGMENT_WRITE_KEY];
[config use:[SEGLeanplumIntegrationFactory instance]];
[SEGAnalytics setupWithConfiguration:config];
```

   Make sure to place your Segment Write Key within the code. This block of code also calls for Leanplum start.

For addition documentation you can also check Leanplum docs.

Android

1. Install the Segment-Leanplum Android SDK by adding the following maven URL to your project `build.gradle` file:

```
allprojects {
    repositories {
        jcenter()
        maven {
          url "http://www.leanplum.com/leanplum-sdks/"
        }
    }
}
```

2. Install the Segment-Leanplum integration by adding these lines to your module's `build.gradle` file:

```
dependencies {
  compile 'com.segment.analytics.android:analytics:4.0.4'
  compile 'com.leanplum.segment:LeanplumIntegration:1.1.0'
}
```

3. Add the following permissions to your applications AndroidManifest.xml:

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

If you want to use the advanced features of Leanplum, also add the additional permissions, as described in Leanplum's Android documentation.

4. Add the following lines to your Application or Controller:

```
private static final String SEGMENT_WRITE_KEY = " ... ";

Analytics analytics = new Analytics
  .Builder(getApplicationContext(), SEGMENT_WRITE_KEY)
  .use(LeanplumIntegration.FACTORY)
  .build();
```

Note: *There is no need to explicitly call `Leanplum.start`, as it is called within the `LeanplumIntegration`.*

5. In addition to that you can also use the advanced features of Leanplum. Once the Leanplum SDK is successfully registered, Segment executes a callback:

```
analytics.onIntegrationReady(LeanplumIntegration.LEANPLUM_SEGMENT_KEY,
    new Analytics.Callback() {
      @Override
      public void onReady(Object instance) {
        Leanplum.addVariablesChangedHandler( ... );
      }
    });
```

That's it! Now you can use the Segment SDK and also the advanced features of the Leanplum SDK.

Page

If you're not familiar with the Segment Specs, take a look to understand what the Page method does. An example call would look like:

```
analytics.page()
```

When you call `page` as a user moves to a new area, you can advance your user's 'state' through your application. We will call Leanplum's `advance` method when this happens.

Identify

If you're not familiar with the Segment Specs, take a look to understand what the Identify method does. An example call would look like:

```
analytics.identify({
  userId: 'user123',
  traits: {
    name: 'Michael Bolton',
    gender: 'male'
  }
})
```

As soon as a user logs-in or signs-up, you'll want to make an `identify` call which takes the `userId` of a user and

any `traits` you know before setting them in Leanplum.

Track

If you're not familiar with the Segment Specs, take a look to understand what the Track method does. An example call would look like:

```
analytics.track({
  userId: 'user123',
  event: 'Item Purchased',
  properties: {
    revenue: 39.95,
    shippingMethod: '2-day'
  }
})
```

Calling a `track` event will log a user event with Leanplum so that you can figure out how to increase engagement, virality, or whatever key action is critical to your business.

Additional Features

The features listed below are only supported if you have installed our iOS or Android SDKs.

Push Notifications

As every analytics provider deals with push notifications and in-app messaging differently, we currently don't support them out of the box. To use these features with your Leanplum account, you can add a couple of small code snippets to your app, then you're ready to go!

iOS

1. Login to the iOS provisioning portal.
2. the Identifiers > App IDs, select your app, click Edit, and enable Push Notifications.
3. lick Create Certificate for each of the Development and Production certificates and follow the onscreen instructions. You should not reuse existing certificates so that we can track delivery failures properly.
4. ownload your new certificate files from your browser. Open the files on your computer, which will launch Keychain.
5. Keychain, select the new certificates, expand them to view the private key, and then right click to export them as .p12 files. You must enter a password.
6. Leanplum, go to your app's Keys & Settings (App Settings > {Your app} > Keys & Settings). Under Push Notifications, upload your .p12 files to Leanplum and enter your passphrase from step 5 above.
7. onfigure your app to use push notifications in your app delegate's `applicationDidFinishLaunching` method (you may choose any combination of formats.

You are now ready to send push notifications from your Leanplum UI! If you need some code snippets, check out the Leanplum docs here.

Android

1. opy the permissions XML below into the AndroidManifest.xml and insert your package name into the name fields where it says [com.YOUR_PACKAGE].

```
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="[com.YOUR_PACKAGE]" xmlns:tools="http://schemas.android.com/tools">
    <uses-sdk
      android:minSdkVersion="16"
      tools:overrideLibrary="com.leanplum, com.google.android.gms"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
  <!-- Optional. Prevents the device from sleeping when a message is received. -->
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
```

1. Add the receiver XML below into the same manifest file and replace [com.YOUR_PACKAGE] with your package name.

```
  <receiver
      android:name="com.leanplum.GcmBroadcastReceiver"
      android:permission="com.google.android.c2dm.permission.SEND" >
      <intent-filter>
          <action android:name="com.google.android.c2dm.intent.RECEIVE" />
          <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
          <category android:name="[com.YOUR_PACKAGE].gcm" />
      </intent-filter>
  </receiver>
```

1. Register your service in the manifest file using the below code.

```
  <service android:name="com.leanplum.LeanplumPushService" />
```

We've put together two example projects for sending push notifications through GCM and Firebase for you to check out:

- LP-Segment-GCM-Example
- LP-Segment-Firebase-Example

A/B Testing

As with push notifications, A/B testing variables are dealt with in different ways by each of our destinations. Leanplum only requires two lines of code for each of your variables to get you set up!

iOS

1. Set the value with the macro `DEFINE_VAR_FLOAT` in ObjC, or `LPVar.define` in Swift. To access the value in your code, use the `floatValue` method. This should be set any time before calling `Leanplum.start`.

```
// Objective-C
  DEFINE_VAR_FLOAT(shootSpeed, 1.0);  // How fast your ship shoots.
    ...
    [Leanplum onVariablesChanged:^() {
      // Move ship according to its speed.
      [myShip moveWithSpeed:shootSpeed.floatValue];
    }];
```

```
  // Swift
  var shootSpeed = LPVar.define("shootSpeed", with:1.0); // How fast your ship shoots.
    ...
    Leanplum.onVariablesChanged {
    // Move ship according to its speed.
      myShip.moveWithSpeed(shootSpeed?.floatValue())
    }
```

For more information about A/B Testing Variables on iOS in Leanplum, see their docs.

1. If you want to define any other type of data, Boolean, String, Color, Assets, Dictionary, or Array, take a look at the Leanplum docs here

Android

1. Define your variable using var.define any time before calling `Leanplum.start`.

```
static Var<String> welcomeLabel = Var.define("welcomeLabel", "Welcome!");
```

2. Register a valueChangedHandler on that variable.

```
welcomeLabel.addValueChangedHandler(new VariableCallback<String>() {
  @Override public void handle(Var<String> stringVar) {
    stringVar.value();
  }
});
```

For more information about A/B Testing Variables on Android in Leanplum, see their docs.

Engage

You can send computed traits and audiences generated using Engage to this destination as a **user property**. To learn more about Engage, schedule a demo.

For user-property destinations, an identify call is sent to the destination for each user being added and removed. The property name is the snake_cased version of the audience name, with a true/false value to indicate membership. For example, when a user first completes an order in the last 30 days, Engage sends an Identify call with the property `order_completed_last_30days: true`. When the user no longer satisfies this condition (for example, it's been more than 30 days since their last order), Engage sets that value to `false`.

When you first create an audience, Engage sends an Identify call for every user in that audience. Later audience syncs only send updates for users whose membership has changed since the last sync.

Real-time to batch destination sync frequency

Real-time audience syncs to Leanplum may take six or more hours for the initial sync to complete. Upon completion, a sync frequency of two to three hours is expected.

Settings

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

SETTING	DESCRIPTION
Leanplum Application Id *(required)*	`string`. The application ID. To find yours, select your app in the navigation column, and click Edit Apps. Under Keys, click Show.
Leanplum Client Key *(required)*	`string`. Either the Production or Development keys, depending on which API call you want to make.
Use Leanplum in Development Mode	`boolean`, defaults to `FALSE`. To use development mode, you must also input your Development key as the client key.
GCM Sender ID	`string`. The GCM sender ID that should be used for Push Notifications.
Track as Supplemental Events	`boolean`, defaults to `FALSE`. Select this option to send events from this source to Leanplum as non-session related events. Events that originate from this source will be marked as passive and will not be tied to a user's session. Events from this source will also not create a new user if one does not currently exist. This option should be considered when tracking events outside of your mobile app.
Use Leanplum Sender ID	`boolean`, defaults to `FALSE`. Select this if you don't have your own GCM Sender ID, and would like to use Leanplum's built in sender ID instead. If you have set your own GCM Sender ID and enable this setting, both IDs will be used.

This page was last modified: 09 Aug 2024

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