## Where is the changelog for the library?

You can see a changelog in the GitHub repository, detailing the changes made in each release.

## Can I use the library with Maven?

Yes! You can use the Segment library with Maven, or any other custom build system because the `core` SDK is simply a JAR.

```
<dependency>
    <groupId>com.segment.analytics.android</groupId>
    <artifactId>analytics</artifactId>
    <version>LATEST</version>
</dependency>
```

## How big is the Segment SDK?

The core Segment SDK is extremely lightweight. It contains just under 1k methods, the JAR weighs in at 123KB

and the dex size is 113KB.

## How can I swap out debugging and production keys?

If you're using Gradle and build flavors, you can provide an `analytics.xml` for each configuration with different Writekeys in each.

For other cases, you can also construct custom instances of the client and pass in a different key for each instance. Set it as the singleton instance, and use the same API everywhere else.

```
class MyApp extends Application {
  @Override public void onCreate() {
    Analytics analytics;
    if(BuildConfig.DEBUG) {
      analytics = new Analytics.Builder(this, debugWriteKey)...build();
    } else {
      analytics = new Analytics.Builder(this, releaseWriteKey)...build();
    }
    Analytics.setSingletonInstance(analytics); // Must be called before any calls to Analytics.with(context)

    // Now Analytics.with will return the custom instance
    Analytics.with(this).track("App Launched");
  }
}
```

## How can I use a destination-specific feature?

...for example, Mixpanel's push notifications?

If you're using Device-mode for a mobile destination, meaning that you are bundling the destination's mobile SDK, you can always access features from that tool's native SDK.

To make sure you use the same instance of these destinations as Segment does, you can register a listener that notifies you when the destinations are ready. This listener is called synchronously if the destinations are notified, and asynchronously if the destinations aren't yet ready.

```
analytics.onIntegrationReady("Crittercism", new Callback() {
  @Override public void onReady(Object instance) {
    // Crittercism uses static methods only, so the instance returned is null.
    Crittercism.leaveBreadcrumb();
  }
});
analytics.onIntegrationReady("Mixpanel", new Callback() {
  @Override public void onReady(Object instance) {
    MixpanelAPI mixpanelAPI = (MixpanelAPI) instance;
    mixpanelAPI.clearSuperProperties();
  }
});
```

Destinations that return `Void` use a shared instance, and you can call into the SDK directly. This API guarantees that the destinations are initialized first. If you ever decide to change the settings for the destination from the Segment App, the changes are reflected here.

```
analytics.onIntegrationReady(BundledIntegration.FLURRY, new Callback() {
  @Override public void onReady(Object instance) {
    // Flurry uses static methods only, so the instance returned is null.
    Flurry.setLogEnabled(true);
  }
});
```

## Why is my callback not being invoked?

If you use the destination callbacks described above, and don't receive a callback, check your Proguard configuration. Any easy way to verify that Proguard is the issue is to disable it completely for a run and see if the callbacks is invoked.

## How should I configure Proguard?

For the Segment SDKs, you can add the following snippet to your Proguard configuration:

```
-keep class com.segment.analytics.** { *; }
-keep class androidx.lifecycle.DefaultLifecycleObserver
```

You should also check the vendor documentation for any Device-mode destinations you are bundling, to see if they include any recommended Proguard configurations.

## How should I use Outbound's push notifications?

Outbound isn't a Device-mode destination, so you must set it up manually.

First, set up the GCM client as described in the instructions.

The Segment servers look for a `context.device.token` key to send to Outbound. If you are using Analytics-Android version 2.1.6 or later, you can then set the registration ID from the step above on the `context`, as in the example below:

`analytics.getContext().putDeviceToken(registrationId);`

The entire code flow looks a bit like this:

```
String registrationId = loadRegistrationId(); // look up a cached value
if(registrationId == null) {
  registrationId = register(SENDER_ID); // using GoogleCloudMessaging
  save(registrationId); // save the registration ID
}
analytics.getContext().putDeviceToken(registrationId);
```

## Do you support Phonegap or Cordova?

Yes! You can use Segment's browserify'd analytics-node package just like any other client-side JavaScript library.

## Are there any limitations for using Segment with Huawei?

No, there hasn't been any instances that show there are limitations when Segment tracks Huawei devices.

## Does LifecycleObserver (above version 2.6.0) work with Segment?

No. It depends on androidx-startup for initialization, this snippet prevents the Segment SDK from tracking app lifecycle events. The solution is to either remove the snippet completely or use `tools:node="merge"` instead of `tools:node="remove"`.

```
<provider
    android:name="androidx.startup.InitializationProvider"
    android:authorities="${applicationId}.androidx-startup"
    tools:node="merge"></provider>
```

This page was last modified: 28 Feb 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



Privacy

Terms

Website Data Collection Preferences