Amplitude (Classic) is in Maintenance mode

The Amplitude (Classic) Destination has entered maintenance mode. Future updates are limited to security updates and bug fixes. A new version of this destination is available. See Amplitude (Actions)

Amplitude is an event tracking and segmentation platform for your web and mobile apps. By analyzing the actions your users perform, you can gain a better understanding to drive retention, engagement, and conversion.

Segment's Amplitude destination code is open source and available on GitHub. You can view these repositories:

- Android
- iOS
- JavaScript
- Kotlin
- Swift

In addition to Segment's Amplitude documentation, Amplitude provides a Segment integration guide, as well.

To delete users based on GDPR regulations, you must include a secret key in the **Secret Key** setting of every Amplitude destination. You can find your Secret Key on the General Settings of your Amplitude project.

Identify high-value users with Historical Count analysis

Examine the exact moment in the customer journey that converts new users into high-value customers.

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

Getting Started

1. From the Segment web app, navigate to **Connections > Destinations** and click **Add Destination**.
2. Search for **Amplitude** select it.
3. Choose which sources to connect the destination to.
4. the destination settings, enter your Amplitude API key.
   You can find your Amplitude API key in the Amplitude project settings . It is a 32-character string of numbers and letters. Locate the project you want to receive your Segment data, copy that project's API key, and paste it into your Amplitude destination settings in Segment.

If you included Segment's JavaScript snippet on your page, then Amplitude's SDK loads on your page automatically and you can use Segment's to begin sending events right away.

Page and Screen

If you're not familiar with the Segment Specs, take a look to understand what the Page and Screen methods do. By default, Segment does not send these standard calls to Amplitude. However, you can enable them with the destination settings below, which you can find under the "Optional Settings" tab.

The example below shows a Page call from a server library.

```
analytics.page({
  userId: "some_user_id", // if using A.js client-side, you can leave out the `userId`
  category: "Merchant",
  name: "Settings",
})
```

The next example shows a call from a mobile library, which uses the Screen call instead of the Page call.

```
// Note: screen calls are only for mobile. you can't make them from A.js client-side.
analytics.screen({
  userId: "some_user_id",
  category: "Merchant",
  name: "Settings",
})
```

Page and Screen calls have two important properties: a *page name*, such as "Settings", and a *category*, such as "Merchant". How you pass these properties depends on which Segment library you use. Segment determines when to send events to Amplitude based on the settings you enable, and whether the call has a name or category included.

If you enable more than one of the following settings, Segment might send multiple events for the same call.

Event type settings for cloud-mode and Analytics.js

If you use Analytics.js (in either device- or cloud-mode), a mobile library in cloud-mode, or a Segment server library, the following settings are available. (Additional settings are available *only* for iOS and Android sources that send in device-mode.)

SETTING NAME	WHEN EVENTS ARE SENT TO AMPLITUDE	AMPLITUDE EVENT NAME	EXAMPLE FOR `{"NAME": "SETTINGS", "CATEGORY": "MERCHANT" }`
Track Named Pages	A `page`/`screen` *name* is provided	Loaded/Viewed (Category) (Name) Page/Screen	"Loaded Merchant Settings Page"
Track Categorized Pages	A `page`/`screen` *category* is provided	Loaded/Viewed (Category) Page/Screen	"Loaded Merchant Page"
Track All Pages	Always	Loaded/Viewed a Page/Screen	"Loaded a Page"

Before you choose a setting, read about the Amplitude event type volume considerations.

When you use the **Track Named Pages** or **Track Categorized Pages** settings, Segment sends a Page or Screen call that includes the name or category. This option stores the page and screen name as a top-level event type. However, Amplitude limits the number of distinct event types per project. Each unique Page and Screen name, Page and Screen category, and Track event counts towards the event type limit. Anything past the instrumentation limit is not visualized in Amplitude.

When you use the **Track All Pages** setting, Segment sends a `Loaded a Page` event type to Amplitude. When you use the generic event name, it is applied to all Page and Screen calls, so you don't hit the event type limit in your project in Amplitude. The page or screen name is still available as an attribute of the `Loaded a Page` event, and you can query it as an event property. The `Loaded a Page` event is counted as one event type, and Amplitude does not place any limits on the number of unique event property values in Amplitude.

Tip: These settings also apply to mobile Cloud-mode connections.

Event Type settings for iOS

The following settings are available on iOS for device-mode connections.

SETTING NAME	WHEN EVENTS WILL BE SENT TO AMPLITUDE	AMPLITUDE EVENT NAME	EXAMPLE FOR `{"NAME": "SETTINGS", "CATEGORY": "MERCHANT" }`
Track All Pages	Always	Viewed (Name)	"Viewed Settings"
Track All Screens	Always	Loaded a Screen	"Loaded a Screen"

When enabled, the "Track All Screens" setting includes the screen name and category as event properties, where the "Track All Pages" omits them. Most iOS implementations should use "Track All Screens".

Event Type settings for Android

The following settings are available on Android for device-mode connections.

SETTING NAME	WHEN EVENTS WILL BE SENT TO AMPLITUDE	AMPLITUDE EVENT NAME	EXAMPLE FOR `{"NAME": "SETTINGS", "CATEGORY": "MERCHANT" }`

SETTING NAME	WHEN EVENTS WILL BE SENT TO AMPLITUDE	AMPLITUDE EVENT NAME	EXAMPLE FOR `{"NAME": "SETTINGS", "CATEGORY": "MERCHANT" }`
Track Named Pages	A `screen` *name* is provided	Viewed (Category) (Name) Screen	"Viewed Merchant Settings Screen"
Track Categorized Pages	A `screen` *category* is provided	Viewed (Category) Screen	"Viewed Merchant Screen"
Track All Pages	Always	If a `screen` *name* is provided: `Viewed (Name) Screen`. Otherwise `Loaded a Screen`	"Viewed Settings Screen"
Track All Screens	Always	Loaded a Screen	"Loaded a Screen"

You can learn more about Page calls from the Page spec and Screen calls from the Screen spec.

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:

```
// On server-side
analytics.identify({
  "userId": "123",
  "anonymousId": "a80b66d5-b86d-41bd-866f-fe04ee7841af",
  "traits": {
    "email": "derek@example.com",
    "name": "Derek Sivers",
    "industry": "Music"
  }
})

// On client-side
analytics.identify({
  "email": "derek@example.com",
  "name": "Derek Sivers",
  "industry": "Music"
})
```

When you make an Identify call, Segment uses the `userId` you provide to set the User Id in Amplitude, and sets any `traits` you provide as Amplitude custom `user_properties`.

Merging users with Anonymous ID and User ID

To have Amplitude recognize an anonymous user and a known or logged-in user, make sure you include both the user's `userId` and the `anonymousId` they had before that in your Identify call. If you don't include the `anonymousId`, Amplitude can't tell that the anonymous user is the same person as the logged-in user.

If you're using a Segment server library or the Segment HTTP API, you must explicitly include both `anonymousId` and `userId`. If you're using Analytics.js in device-mode, or a bundled SDK, Segment automatically includes `anonymousId` for you.

Amplitude Device ID

You can set the Device ID in slightly different ways depending on the library and connection mode you're using (Device-mode vs Cloud-mode).

Default library behavior for Device ID

The table below represents default behavior.

LIBRARY	DEFAULT	FALLBACK

LIBRARY	DEFAULT	FALLBACK
A.js	Generated by Amplitude]	`anonymousId`
Server-side	`context.device.id`	`anonymousId`
iOS	Generated by Amplitude	n/a
Android	Generated by Amplitude	n/a

Prefer Anonymous ID for Device ID

If you're using the "Prefer Anonymous ID for Device ID" setting in client-side, server-side, or a mobile library with Cloud-mode enabled, the following rules apply.

LIBRARY	DEFAULT	FALLBACK
A.js	`anonymousId`	Generated by Amplitude
Server-side	`anonymousId`	`context.device.id`

Prefer Advertising ID for Device ID

This option is not currently available for mobile libraries using cloud-mode.

If you're using the "Prefer Advertising ID for Device ID" setting with one of our bundled mobile SDKs, the following rules apply.

LIBRARY	DEFAULT	FALLBACK
iOS	`anonymousId`	[Generated by Amplitude
Android	`anonymousId`	[Generated by Amplitude

Device ID priority

If you have multiple settings enabled, one setting or value can take priority of another. This table lists which settings, if enabled, take priority over other settings or values.

LIBRARY	PRIORITY (HIGHEST TO LOWEST)
A.js	Prefer Anonymous ID for Device ID Set Device ID From URL Parameter amp_device_id (Device-mode only) Device ID Generated by Amplitude
Server-side	Prefer Anonymous ID for Device ID `context.device.id`
iOS	Use AdvertisingId for Device ID (Device-mode only) Device ID Generated by Amplitude
Android	Use AdvertisingId for Device ID (Device-mode only) Device ID Generated by Amplitude

Using Device ID to merge users

For Amplitude to associate both device-mode and cloud-mode activity with the same user, you must pass the same `deviceID` to Amplitude. Otherwise, Amplitude creates two users - one for each of the `deviceID`'s set per the functionality outlined in the tables above.

You can get the `deviceID` from Amplitude in device-mode so you can return it on cloud-mode calls. The example

method below shows how you could log the `deviceId` in the `ready` function on the device, so you could send it to the server.

```
analytics.ready(function() {
  // Instead of console.log(...), you probably want to do upload_to_server(...)
  // or something to that effect.
  console.log(amplitude.getInstance().options.deviceId);
});
```

When a user logs in, be sure to send the same Amplitude `deviceID` in your Identify call. Otherwise, Amplitude creates two separate users: one for your anonymous user, and another for your logged-in user. This is handled automatically on mobile.

Track

If you're not familiar with the Segment Specs, take a look to understand what the Track method does. Amplitude supports several special properties, all of which are included in the following example:

```
// On server-side
analytics.track({
  "userId": "123",
  "event": "Subscription Started",
  "properties": {
    "plan": "Basic",
    "revenue": "32"
  },
  "context": {
    "ip": "8.8.8.8",
    "device": {
      "id": "2b6f0cc904d137be2e1730235f5664094b831186",
      "model": "iPhone 10",
      "brand": "Apple",
      "manufacturer": "Apple"
    },
    "os": {
      "name": "iOS",
      "version": "9.1"
    },
    "network": {
      "carrier": "T-Mobile"
    },
    "app": {
      "version": "3.5.1"
    },
    "location": {
      "country": "United States",
      "region": "California",
      "city": "San Francisco",
      "latitude": "37.7672319",
      "longitude": "-122.4021353"
    },
    "locale": {
      "language": "en-us"
    }
  }
})
```

Segment sends many of these properties automatically if you use Analytics.js, Segment's iOS source, or Segment's Android source.

For a complete list of special `context` keys see Segment's Common fields spec.

Log Revenue V2

Segment's iOS and Android sources can send revenue using Amplitude's preferred `logRevenueV2` method. Segment sets Amplitude's special revenue properties, such as `revenueType` and `productIdentifier`, which are used in Amplitude's Revenue Analysis and Revenue LTV charts. Segment uses the Amplitude `eventProperties` field to send any properties *not* mapped to Amplitude's special properties.

AMPLITUDE PROPERTY	SEGMENT PROPERTY	DESCRIPTION
`productId`	`productId`	An identifier for the product.
`quantity`	`quantity`	The quantity of products purchased. Note: revenue = `quantity` * `price`.
`price`	`price` or `revenue` (or `total` for mobile, see note below)	The price of the products purchased, and this can be negative.
`revenueType`	`revenueType`	The type of revenue (e.g. tax, refund, income).
`receiptSignature`	`receiptSignature` (Android only)	The receipt signature.
`receipt`	`receipt`	This is required if you want to verify the revenue event.
`eventProperties`	Any remaining properties	A NSDictionary or Map of event properties to include in the revenue event.

* If `properties.price` is not present, Segment uses `revenue` instead, and sends that as `price`. In Segment's iOS and Android components, if `revenue` isn't present either, Segment does an additional fallback and sends the `total`.

Property names should be `camelCase` for Android implementations, and `snake_case` for iOS implementations.

Amplitude doesn't support currency conversion

Normalize all revenue data to your currency of choice before sending it to Amplitude.

Revenue

For Segment's Analytics.js (device-mode), iOS, and Android sources, if you do not enable the preferred `logRevenueV2` setting, Segment sends the data using the deprecated `logRevenue` methods (which still work). If you record events using this old setting, fields such as `revenueType` aren't recorded in your events. This can reduce your ability to segment on those revenue events in the Amplitude platform.

AMPLITUDE PROPERTY	SEGMENT PROPERTY	DESCRIPTION
`productId`	`productId`	An identifier for the product.
`quantity`	`quantity`	The quantity of products purchased. Note: revenue = `quantity` * `price`.
`price`	`price` (or `revenue` or `total`, see note below)	The price of the products purchased, and this can be negative.
`receipt`	`receipt` (mobile only)	This is required to verify the revenue event.
`receiptSignature`	`receiptSignature` (Android only)	The receipt signature.
`revenueType`	`revenueType` (cloud-mode only)	The type of revenue (such as tax, refund, income).
`revenue`	`revenue` (cloud-mode only)	The revenue collected.
`eventProperties`	Any remaining properties (cloud-mode only)	A NSDictionary or Map of event properties to include in the revenue event.

In Segment's Analytics.js, iOS and Android sources, if `properties.price` isn't present, Segment falls back to `revenue`

and sends that as `price`. The Segment iOS and Android sources also do an additional fallback to `total`, if `revenue` isn't present either.

> **Tip** If your site allows users to perform a single transaction with multiple products (such as a shopping cart checkout), Segment recommends that you use an Order Completed event to track revenue with Amplitude.

Order Completed

Segment recommends that you use the Order Completed event to track revenue with Amplitude. This event allows you to define a list of products that a user purchased in a single transaction, which is the best way to track purchases for sites that have a shopping cart system.

You can currently use this event only for data coming from a server or web source. An `Order Completed` event from mobile using our bundled Amplitude integration will work the same as our standard `track` event documented above.

Here's an example of how you'd create an "Order Completed" event:

```
analytics.track({
  "userId": "e953c39d2597f0b8a79dd3c407baeb13bb58523a",
  "event": "Order Completed",
  "properties": {
    "checkoutId": "6727142daf49b93a601d3a31bc3d53aeae1d15ab",
    "orderId": "50314b8e9bcf000000000000",
    "affiliation": "Google Store",
    "total": 30,
    "revenue": 25,
    "shipping": 3,
    "tax": 2,
    "discount": 2.5,
    "coupon": "hasbros",
    "currency": "USD",
    "products": [
      {
        "productId": "507f1f77bcf86cd799439011",
        "sku": "45790-32",
        "name": "Monopoly: 3rd Edition",
        "price": 19,
        "quantity": 1,
        "category": "Games"
      },
      {
        "productId": "505bd76785ebb509fc183733",
        "sku": "46493-32",
        "name": "Uno Card Game",
        "price": 3,
        "quantity": 2,
        "category": "Games"
      }
    ]
  }
})
```

When you send an "Order Completed" event, an "Order Completed" event appears in Amplitude for that purchase. An Amplitude event called "Product Purchased" is also created for each product in the purchase. All event properties, except `products`, are sent as `event_properties` of the Amplitude "Order Completed" event. Information about each product is present *only* on the individual "Product Purchased" events.

Track Revenue Per Product

Amplitude has two different ways to track revenue associated with a multi-product purchase. You can choose which method you want to use using the **Track Revenue Per Product** destination setting.

If you disable the setting ("off"), Segment sends a single revenue event with the total amount purchased. Revenue data is added to the Amplitude "Order Completed" event. The "Product Purchased" events do not contain any native Amplitude revenue data.

If you enable the setting ("on"), Segment sends a single revenue event for each product that was purchased. Revenue data is added to each "Product Purchased" event, and the "Order Completed" event does not contain any native Amplitude revenue data.

Make sure you're formatting your events using the Track method spec, and pass at minimum a `revenue` property, as well as a `price` and `quantity` property for each product in the products list.

Group

If you're not familiar with the Segment Specs, take a look to understand what the Group method does.

> ⚠
> Groups are an enterprise-only feature in Amplitude and are only available if you've purchased the Accounts add-on.

The following example shows a Group call made from a server library:

```
// On server-side
analytics.group("some_group_id", {
  userId: "some_user_id",
  traits: {
    email: "the_group_email",
    some_other_property: "some_other_value",
  }
})
```

And this example shows a call made from a device-mode library that sends directly from the client:

```
// On client-side
analytics.group("some_group_id", {
  email: "the_group_email",
  some_other_property: "some_other_value",
})
```

Even if you don't have an enterprise Amplitude account, or don't have the Accounts add-on, Segment always adds groups as `user_properties` on a user record. As long as you specify the destination settings below, Segment adds a "group type" user property with a value of the "group value".

To use Amplitude's groups with Segment, you must enable the following destination settings and make sure you're sending them the data values they need to function. These settings act as a mapping from Segment group traits to Amplitude group types and values.

- **"Amplitude Group Type Trait"**: This specifies what trait in your Group calls contains the Amplitude "group type". In other words, it's how you tell Segment which trait to use as the group type.
- **"Amplitude Group Value Trait"**: This specifies what trait in your Group calls contains the Amplitude "group value". It's how you tell Segment which trait to use as the group value.

For example, if you specified `group_type` as the "Amplitude Group Type Trait", and `name` as the "Amplitude Group Value Trait", then the example call below...

```
analytics.group("082108c8-f51e-485f-9d2d-b6ba57ee2c40", {
  group_type: "Organization",
  name: "ExampleCorp, LLC",
  employees: "20",
  email: "hello@example.com"
});
```

Associates the current user with the group with type `"Organization"` and value `"ExampleCorp, LLC"`. On the device-mode version of the destination, that's all that happens. On Android, and in cloud-mode, Segment sends the traits you pass (in this case, `group_type`, `name`, `employees`, and `email`) as `group_properties` of that group.

Segment requires that all Group calls provide a group ID. What you provide as group ID doesn't matter, but you cannot leave group ID empty.

Legacy Group Behavior

If you do not provide "Amplitude Group Type/Value Trait", or one of the traits was not provided in your Group call, then Segment associated the user with a group with the type "[Segment] Group" and with the value "(Group Id)". No properties are associated with that group.

For example, the previous group call would associate the user with a group of type "[Segment] Group" and value "082108c8-f51e-485f-9d2d-b6ba57ee2c40".

Alias

Segment's Alias method maps to Amplitude's `usermap` endpoint. Making a Segment Alias call allows you to associate a Segment user's `previousId` with the user's `userId`, or what Amplitude refers to, respectively, as a `user_id` and a `global_user_id`.

By default, Segment does **NOT** send Alias events to Amplitude. To forward Alias events from Segment, go to your Amplitude destination settings in the Segment web app, and set the **Enable Alias** setting to "on".

Once enabled, Segment forwards Alias events from Segment's servers only. This means that Alias events reach Amplitude only when you're sending events from the client and have set your Amplitude instance's connection mode to "Cloud Mode", or are sending Alias events from a Segment server-side library (such as Node).


To use Alias, you must have the Amplitude Portfolio add-on enabled.

For more information, see the Segment Spec page for the Alias method.

SEGMENT IDENTIFIER NAME	EQUIVALENT AMPLITUDE IDENTIFIER NAME
`previousId`	`user_id`
`userId`	`global_user_id`

Mapping Users

You can map a Segment user's `previousId` to the user's `userId` in Amplitude by invoking a Segment Alias method with an argument for each value.

The example Alias call below maps the `previousId` with the value of `123` to the `userId` with a value of `456` in Amplitude. Both user `123` and `456` still have separate user profiles, but the profiles get merged together when you look at the user's behavior in Amplitude's Cross Project view.

This kind of mapping is useful for users who have different ids across different Amplitude projects. The user's `user_ids` act as child ids, and can all be mapped to a single `global_user_id` in Amplitude. This allows you to analyze the user's aggregate behavior in Amplitude's Cross Portfolio view.

```
analytics.alias({
  previousId: '123',
  userId: '456'
})
```

If you make an Alias call from the user's device, you don't need to explicitly pass a `previousId`. Segment device-mode Amplitude library sets the value of `oldId` to the value of the current user's previous `userId`. The example calls below show how to make an Alias call to map the `userId` `oldUserId` to the new `userId`, `finalUserId`:

```
analytics.identify('oldUserId')
analytics.alias('finalUserId')
// remember to identify with the new `userId`
analytics.identify('finalUserId')
```

Unmapping Users

You can also unmap users, for example if you aliased them in error. To unmap a user, pass the user's `previousId` as an integration-specific option. The example Alias call below sends a request to Amplitude that unlinks user `123` from *all* `global_user_ids` it was previously associated with.

```
analytics.alias({
  userId: '456',
  integrations: {
    Amplitude: {
      unmap: '123'
    }
  }
})
```

Advanced Amplitude features

sessionId

Segment doesn't have a concept for a session.

Device-mode calls to Amplitude include session information because Segment bundles Amplitude's SDK. To set up the same `sessionId` for cloud-mode calls to Amplitude, you must explicitly set the `session_id` as an integration-specific option, as in the example below.

```
{
  "userId": "1234",
  "traits": {
    "email": "someone@somewhere.com",
    "name": "Some Person",
    "industry": "Technology"
  },
  "context": {
    "ip": "00.0.00.00"
  },
  "timestamp": "2016-10-17T00:30:08.276Z",
  "integrations": {
    "Amplitude": {
      "session_id": "<Timestamp>"
    }
  }
}
```

You must pass the start time of a session as `<Timestamp>`.

When you pass a timestamp value from the `session_id` it must be in Unix format, otherwise it generates an error when it is delivered to Amplitude. For example, a date of January 1, 2020 and 9:30am UTC would be written as `2020-12-07T19:33:44+00:00` in ISO 8601, but `1577871000` in Unix epoch time. There are many tools and libraries available to help you convert your timestamps.

Setting event-level groups using Track calls

You can use Amplitude to set event-level groups. This means the group designation only applies for the specific event you are recording, and doesn't persist on the user. To specify these groups, provide an integration-specific `groups` property with key-value pairs corresponding to the `groupType-groupValue` pairs you want to appear in Amplitude.

```
analytics.track("Clicked Benefits Dropdown", {
  dropdownColor: "blue"
},
{
  integrations: {
    Amplitude: {
      groups: {
        onboarding_cohort: "Summer 2016"
      }
    }
  }
});
```

Setting Amplitude Version User Property using Identify calls

If you are sending event data to Amplitude in cloud-mode (through the Segment servers) and want to use the Amplitude Release objects feature, you can set the app version user property as in the example below. Make sure to send the version details in the `context` object and not as a standard user trait.

```
analytics.identify('testUser', {
  email: 'john@example.com',
  name: 'John Doe'
}, {
  context: {
    app: { 'version': "<value_here>", }
  }
});
```

Legacy group assignment using Identify calls

⚠️ **Note:** Segment will continue to support this behavior, however the preferred way to associate a user with a group in Amplitude is to use a Group call.

You can associate a user with a group by providing an integration-specific `groups` property, with the keys being Amplitude "group type" and the values being Amplitude "group value":

```
analytics.identify('user-id', {
  email: 'bill@example.com',
  country: 'USA'
}, {
  integrations: {
    Amplitude: {
      groups: {
        sports: ['basketball', 'tennis']
      }
    }
  }
});
```

This Identify event creates a new user (or updates an existing user) in Amplitude and sets their `sport` groups as `basketball` and `tennis`.

Location Tracking

This feature is only supported when you use the Segment iOS and Android sources, with Amplitude in device-mode.

This feature defaults to `enabled`. If a user granted your app location permissions, enable this setting so that the SDK will also grab the location of the user. Amplitude does not prompt the user for location permission, so your app must explicitly prompt to ask permission.

On iOS, the user's location is only recorded once per session. If you need to force update the location in Amplitude, you can use the native method `updateLocation` (iOS only) referenced in Amplitude's iOS SDK documentation . When you call `enableLocationListening` on the iOS SDK, it forces the SDK to update (and

overwrite) the initial location that was cached during app startup.

On Android, when enabled, this setting adds a latitude and longitude property to each Track call, which reflecte where geographically the event was triggered.

Even if you disable location listening, Amplitude's ingestion layer attempts to determine the user's location from their IP address. To prevent tracking of any location information, contact your Amplitude CSM to disable all location tracking.

Set AdvertisingId for DeviceId

This feature is only supported when you use the Segment iOS and Android sources, with Amplitude in device-mode.

Segment supports Amplitude's `useAdvertisingIdForDeviceId` method. For iOS, this allows you to use the `advertisingIdentifier` instead of `identifierForVendor` as the Device ID in Amplitude. This is useful for tying together data from advertising campaigns to analytics data.

> ⚠️
> Apple prohibits the use of `advertisingIdentifier` if you did not say that your app has advertising in your App Store application.

On Android, this setting relies on Google's Advertising ID. This method can return `null` if a Device ID has not been generated yet.

Increment Traits

This increments a user property by some numerical value. If the user property does not have a value set yet, Segment initializes it with a value of `0` before being incremented.

When you configure this setting (under **traitsToIncrement**), Segment calls Amplitude's `add` method on the Amplitude identity instance for each trait passed in an Identify call. The trait must have a numerical value so it can be incremented.

Set trait once

Supported on all components.

This sets the value of a user property only once. Subsequent operations on that user property will be ignored. Configure the `trait` you would like to `setOnce` in the integration settings pane. Segment then checks the `traits` object for the configured `trait` when `identify` is called.

Log out of sessions

This feature is only supported when you use the Segment iOS and Android sources, with Amplitude in device-mode.

Out-of-session events have a `session_id` of `-1`, and are not considered part of the current session. This means they do not extend the current session. This might be useful if you are logging events triggered by push notifications, for example. To set an out of session event, send the a Track call with an integration option property `outOfSession` set to `true`.

The example below shows how you might set this on iOS:

```
[[SEGAnalytics sharedAnalytics]
  track: @"Push Notification Viewed"
  properties: nil
  options: @{
    @"integrations": @{
      @"Amplitude": @{
        @"outOfSession": @YES
      }
    }
  }
];
```

The following example shows how you might set this on Android:

```
Properties properties = new Properties();
Map<String, Object> amplitudeOptions = new HashMap<>();
amplitudeOptions.put("outOfSession", true);

Options options = new Options().setIntegrationOptions("Amplitude", amplitudeOptions);
Analytics.with(context).track("Push Notification Viewed", properties, options);
```

Flush

The Segment mobile device-mode bundles for Amplitude map Segment's `flush` method to Amplitude's `uploadEvents` method.

Reset

The Segment mobile device-mode bundles for Amplitude support logging out users in Amplitude using Segment's `reset` method. You don't need to alias users, as Amplitude merges user data on the backend so that any events up to that point from the same client are tracked under the same user.

Segment logs the user out by setting the `userId` to `nil` and calling Amplitude's method to regenerate a new `deviceId`.

Troubleshooting

Instrumentation Explorer

Amplitude offers a robust Instrumentation Explorer/Debugger. This is a helpful Chrome extension that shows each page interaction that sends an event to Amplitude.

I Don't See My Data In Amplitude

If you don't your data arrive in Amplitude, see the Analytics.js guide to validating data being transmitted to your third-party destination.

Also, Amplitude doesn't support fields with a value of an array with nested arrays.

For more information on the Amplitude/Segment integration, view Amplitude's Import Segment Data documentation.

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 Amplitude 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
API Key *(required)*	`string`. You can find your API Key on your Amplitude Settings page.
Append Fields To Event Properties	`text-map`, defaults to {}. Web Device-mode only. Configure event fields to be appended to `event_props` for all track calls. For example, entering `context.page.title` on the left and `pageTitle` on the right will set the value of `context.page.title` at `event_properties.pageTitle`.
Batch Events	`boolean`, defaults to `FALSE`. If true, events are batched together and uploaded only when the number of unsent events is greater than or equal to `eventUploadThreshold` or after `eventUploadPeriodMillis` milliseconds have passed since the first unsent event was logged.
Set Device ID From URL Parameter amp_device_id	`boolean`, defaults to `FALSE`. If true, the SDK will parse device ID values from url parameter `amp_device_id` if available.
Enable Location Listening	`boolean`, defaults to `TRUE`. Mobile Only. If a user has granted your app location permissions, enable this setting so that the SDK will also grab the location of the user. Amplitude will never prompt the user for location permission, so this must be done by your app.
Endpoint	`select`. Cloud-mode Only (will not work in device-mode). Choose the endpoint corresponding to your region.
Event Upload Period Millis (for batching events)	`number`, defaults to 30000. Amount of time in milliseconds that the SDK waits before uploading events if `batchEvents` is `true`.
Event Upload Threshold (for batching events)	`number`, defaults to 30. Minimum number of events to batch together per request if `batchEvents` is `true`.
Force Https	`boolean`, defaults to `FALSE`. If true, the events will always be uploaded to HTTPS endpoint. Otherwise the SDK will use the embedding site's protocol.
Group Type Trait	`string`. What trait Segment should use as your Amplitude "group type" in group calls. If, for example, you set this to be `industry`, then `traits["industry"]` will be sent as `groupType` to Amplitude.
Group Value Trait	`string`. What trait Segment should use as your Amplitude "group value" in group calls. If, for example, you set this to be `plan`, then `traits["plan"]` will be sent as `groupValue` to Amplitude.

SETTING	DESCRIPTION
Map Query Params to Custom Property	`map`, defaults to {}. When sending data via server side or Cloud Mode, you can send the custom query params that are automatically collected by `analytics.js` (or whatever you manually send under `context.page.search`), by entering a custom property name you would like to map that under on the left hand side. On the right hand side, please choose whether you want the query params to be set on the user profile or event metadata level. Whatever you put on the left hand side we will map the entire query parameters string from the `context.page.search`.
Prefer Anonymous ID for Device ID	`boolean`, defaults to `FALSE`. By default, Segment will use `context.device.id` as the Amplitude `device_id`, using `anonymousId` if `context.device.id` isn't present. Enable this setting to flip this behavior; `anonymousId` will be used as the `device_id`, falling back to `context.device.id` if it isn't present. In browsers, enabling this setting means the user's anonymous ID, which you can set using `analytics.user().anonymousId('ID_GOES_HERE')`, will be set as the Amplitude device ID. Otherwise, Amplitude's default logic for determining device IDs will be used.
Save Referrer, URL Params, GCLID Once Per Session	`boolean`, defaults to `TRUE`. If true then includeGclid, includeReferrer, and includeUtm will only track their respective properties once per session. New values that come in during the middle of the user's session will be ignored. Set to false to always capture new values.
Secret Key	`string`. Your Amplitude Secret Key (Only needed for user deletion)
Enable Alias	`boolean`, defaults to `FALSE`. Server-Side Only. Enabling this setting allows your Amplitude destination instance to send `alias` events to Amplitude's `usermap` endpoint. By default, Segment's Amplitude integration does not support `alias`, so when this setting is disabled, your Segment Amplitude destination will reject `alias` events as unsupported.
Send To Batch Endpoint	`boolean`, defaults to `FALSE`. Server-Side Only. If true, events are sent to Amplitude's `batch` endpoint rather than to their `httpapi` endpoint. Because Amplitude's `batch` endpoint throttles traffic less restrictively than the Amplitude `httpapi` endpoint, enabling this setting may help to reduce 429s - or throttling errors - from Amplitude. Amplitude's `batch` endpoint throttles data only when the rate of events sharing the same `user_id` or `device_id` exceeds an average of 1,000/second over a 30-second period. More information about Amplitude's throttling is available here in their docs: https://developers.amplitude.com/#429s-in-depth.
Track All Pages to Amplitude	`boolean`, defaults to `FALSE`. This will track **Loaded a Page** events to Amplitude for all `page` method calls. We keep this disabled by default, since Amplitude isn't generally used for pageview tracking.
Track All Screens	`boolean`, defaults to `FALSE`. Mobile only. Sends a "Loaded Screen" event and the screen name as a property to Amplitude. Moving forward, this is the preferred method of tracking screen events in Amplitude.
Track Categorized Pages to Amplitude	`boolean`, defaults to `TRUE`. This will track events to Amplitude for `page` method calls that have a `category` associated with them. For example `page('Docs', 'Index')` would translate to **Viewed Docs Page**.
Track GCLID	`boolean`, defaults to `FALSE`. If true, captures the gclid url parameter as well as the user's initial_gclid via a set once operation.
Track Named Pages to Amplitude	`boolean`, defaults to `TRUE`. This will track events to Amplitude for `page` method calls that have a `name` associated with them. For example `page('Signup')` would translate to **Viewed Signup Page**. Remember that `name` includes `category`, so `page('Conversion', 'Signup')` would translate to a **Viewed Conversion Signup Page** event in Amplitude.

SETTING	DESCRIPTION
Track products once	`boolean`, defaults to `FALSE`. *Beta feature* Amplitude recently added support to submit an array of products on "Order Completed" events. If this setting is set to true, we will send all the products in one single event to Amplitude.
Track Referrer to Amplitude	`boolean`, defaults to `TRUE`. Client Side Only - Enabling this will send referrer information as a user property to Amplitude when you call Segment's `page` method.
Track Revenue Per Product	`boolean`, defaults to `FALSE`. Client and server only. This setting allows you to specify whether you would like to track an Amplitude Revenue event per individual product in a user transaction or a single Revenue event for the combined revenue of all products. This setting is only relevant if you are using our eCommerce spec and passing us an Order Completed event with a list of products.
Track Session Events to Amplitude	`boolean`, defaults to `FALSE`. (Optional) This enables the sending of start and end session events for mobile products. Amplitude's libraries track sessions automatically and this option is not necessary for session tracking.
Track UTM Properties to Amplitude.	`boolean`, defaults to `TRUE`. If Amplitude is connected in device-mode this will send the UTM properties found in the querystring. If Amplitude is connected in cloud-mode this will send the UTM properties found in the `context.campaign` object. (Note: The Analytics.js library automatically collects the `context.campaign` object)
Traits to Append	`array`, defaults to . Server-Side and Mobile Only. Configure values to be appended to the user property array via identify.traits.
Traits To Increment	`array`, defaults to . Configure `trait` to increment on identify. If the trait is present, it will increment the trait given the numerical value passed in when you call `identify` with the trait.
Traits to Prepend	`array`, defaults to . Server-Side and Mobile Only. Configure values to be prepended to the user property array via identify.traits.
Traits to Set Once	`array`, defaults to . Server-Side and Mobile Only. Configure values to be set only once via identify.traits.
Unset Params Referrer On New Session	`boolean`, defaults to `FALSE`. If false, the existing referrer and `utm_parameter` values will be carried through each new session. If set to true, the referrer and `utm_parameter` user properties, which include `referrer`, `utm_source`, `utm_medium`, `utm_campaign`, `utm_term`, and `utm_content`, will be set to null upon instantiating a new session. **Note**: This only works if Track Referrer or Track UTM Properties to Amplitude are set to true.
Use AdvertisingId for DeviceId	`boolean`, defaults to `FALSE`. Mobile Only (will *not* work in cloud-mode). Allows users to use advertisingIdentifier instead of identifierForVendor as the Device ID.
Use Amplitude Referral	`boolean`, defaults to `FALSE`. Let Amplitude handle referral tracking behavior. If the "Save Referrer, URL Params, GLCID Once Per Session" setting isn't giving the desired behavior, this setting will fix it. Note: This setting may cause Amplitude to not fully respect the "Prefer Anonymous ID for Device ID" setting (Amplitude may set the device ID upon initialization before it gets set to the proper Anonymous ID) if using Analytics.js 1.0. Consider [updating to Analytics.js 2.0] (https://segment.com/docs/connections/sources/catalog/libraries/website/javascript/upgrade-to-ajs2/)
Send Custom Language and Country Properties	`boolean`, defaults to `FALSE`. Enable this option if you want to send additional 'language' and 'country' parameters inside of event_properties. This is separate from the language and country collected from your user's context. (For example, you want to send the language that a video is played in). You can send these in your properties, for example: `analytics.track('Video Played', {language: 'Japanese'});`

SETTING	DESCRIPTION
Use Log Revenue V2 API	`boolean`, defaults to `TRUE`. Use Amplitude's logRevenueV2 API, which allows for the tracking of event properties with the revenue event. Track an event with "price" and "quantity" properties, and it will log total revenue = price * quantity. You may also set a revenueType property to designate the type of revenue (ex: purchase, refund, etc). Negative prices can be used to indicate revenue lost.
Version Name	`string`. Optional. You can assign a version name for your page, and we'll send it to Amplitude for more detailed events.

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