Additional versions of this destination are available

This page is about the Braze Web Mode (Actions) Destination. See below for information about other versions of the Braze destination:

- Braze Cloud Mode (Actions)
- Braze (Classic)

Braze, formerly Appboy, is an engagement platform that empowers growth by helping marketing teams to build customer loyalty through mobile, omni-channel customer experiences.

Benefits of Braze Web Mode (Actions) vs Braze Classic

Braze Web Mode (Actions) provides the following benefits over Braze Classic:

- **E-commerce mappings**. Users who can't follow the e-commerce spec due to incompatible event names (for example, Trip Booked vs Order Completed) can log purchases in Braze Web Mode (Actions).

Getting Started

1. From the Segment web app, click **Catalog**.
2. Search for “Braze” in the Catalog, select **Braze Web Mode (Actions)**, and choose which of your sources to connect the destination to.
3. Configure the Connection Settings. **API Key** and **SDK Endpoint** are required settings.

> If you’re using a device-mode connection, Braze’s SDK assigns a `device_id` and a backend identifier, `braze_id`, to every user. This allows Braze to capture anonymous activity from the device by matching on those identifiers instead of `userId`. This applies to *device-mode connections*.

Destination Settings

SETTING	DESCRIPTION
Allow Crawler Activity	Allow Braze to log activity from crawlers. See more details
Allow User Supplied Javascript	To indicate that you trust the Braze dashboard users to write non-malicious Javascript click actions, set this property to true. If enableHtmlInAppMessages is true, this option will also be set to true. See more details
API Key	*Required.* Found in the Braze Dashboard under Manage Settings → Apps → Web
App Version	Version to which user events sent to Braze will be associated with. See more details
Automatically Send In-App Messages	When this is enabled, all In-App Messages that a user is eligible for are automatically delivered to the user. If you’d like to register your own display subscribers or send soft push notifications to your users, make sure to disable this option.
Content Security nonce	Allows Braze to add the nonce to any `<script>` and `<style>` elements created by the SDK. See more details
Only Track Known Users	If enabled, this setting delays initialization of the Braze SDK until the user has been identified. When enabled, events for anonymous users will no longer be sent to Braze.
Device Property Allow List	By default, the Braze SDK automatically detects and collects all device properties in DeviceProperties. To override this behavior, provide an array of DeviceProperties. See more details
Disable Push Token Maintenance	By default, users who have already granted web push permission will sync their push token with the Braze backend automatically on new session to ensure deliverability. To disable this behavior, set this option to true
Do Not Load Font Awesome	Braze automatically loads FontAwesome 4.7.0 from the FontAwesome CDN. To disable this behavior set this option to true.
Enable Logging	Set to true to enable logging by default
Enable SDK Authentication	Set to true to enable the SDK Authentication feature.
SDK Endpoint	*Required.* Your Braze SDK endpoint. See more details
In-App Message Z Index	By default, the Braze SDK will show In-App Messages with a z-index of 1040 for the screen overlay, 1050 for the actual in-app message, and 1060 for the message’s close button. Provide a value for this option to override these default z-indexes.
Localization	By default, any SDK-generated user-visible messages will be displayed in the user’s browser language. Provide a value for this option to override that behavior and force a specific language. The value for this option should be a ISO 639-1 Language Code.

SETTING	DESCRIPTION
Manage Service Worker Externally	If you have your own service worker that you register and control the lifecycle of, set this option to true and the Braze SDK will not register or unregister a service worker. See more details
Minimum Interval Between Trigger Actions in Seconds	Provide a value to override the default interval between trigger actions with a value of your own. See more details
No Cookies	By default, the Braze SDK will store small amounts of data (user ids, session ids), in cookies. Pass true for this option to disable cookie storage and rely entirely on HTML 5 localStorage to identify users and sessions. See more details
Open Cards In New Tab	By default, links from Card objects load in the current tab or window. Set this option to true to make links from cards open in a new tab or window.
Open In-App Messages In New Tab	By default, links from in-app message clicks load in the current tab or a new tab as specified in the dashboard on a message-by-message basis. Set this option to true to force all links from in-app message clicks open in a new tab or window.
Require Explicit In-App Message Dismissal	By default, when an in-app message is showing, pressing the escape button or a click on the greyed-out background of the page will dismiss the message. Set this option to true to prevent this behavior and require an explicit button click to dismiss messages.
Safari Website Push ID	If you support Safari push, you must specify this option with the website push ID that you provided to Apple when creating your Safari push certificate (starts with “web”, e.g. “web.com.example.domain”).
SDK Version	*Required.* The version of the Braze SDK to use
Service Worker Location	By default, when registering users for web push notifications Braze will look for the required service worker file in the root directory of your web server at /service-worker.js. If you want to host your service worker at a different path on that server, provide a value for this option that is the absolute path to the file, e.g. /mycustompath/my-worker.js. VERY IMPORTANT: setting a value here limits the scope of push notifications on your site. For instance, in the above example, because the service ,worker file is located within the /mycustompath/ directory, appboy.registerAppboyPushMessages MAY ONLY BE CALLED from web pages that start with http://yoursite.com/mycustompath/.
Session Timeout in Seconds	By default, sessions time out after 30 minutes of inactivity. Provide a value for this configuration option to override that default with a value of your own.

Available Presets

Braze Web Device Mode (Actions) has the following presets:

PRESET NAME	TRIGGER	DEFAULT ACTION
Track Calls	Event type = "track" and event != "Order Completed"	Track Event
Identify Calls	Event type = "identify" Event type = "group"	Update User Profile
Order Completed calls	Event type = "track" and event = "Order Completed"	Track Purchase

Available Actions

Build your own Mappings. Combine supported triggers with the following Braze Web Device Mode-supported actions:

Mapping limits per destination

Individual destination instances have support a maximum of 50 mappings.

Debounce Middleware

When enabled, it ensures that only events where at least one changed trait value are sent to Braze, and events with duplicate traits are not sent. Debounce functionality requires a frontend client to work. Therefore, it cannot be used with server-side libraries or with Engage.

Debounce Middleware is a **Web** action. The default Trigger is: `type = "identify" or type = "group"`

This action does not have any fields.

Track Event

Reports that the current user performed a custom named event.

Track Event is a **Web** action. The default Trigger is: `type = "track" and event != "Order Completed"`

Click to show / hide fields

FIELD	DESCRIPTION
Event Name *	Type: `STRING` The identifier for the event to track.
Event Properties	Type: `OBJECT` Hash of properties for this event.

Update User Profile

Updates a users profile attributes in Braze

Update User Profile is a **Web** action. The default Trigger is: `type = "identify" or type = "group"`

Click to show / hide fields

FIELD	DESCRIPTION
External User ID	Type: `STRING` The unique user identifier
Country	Type: `STRING` The country code of the user
Current Location	Type: `OBJECT` The user's current longitude/latitude.
Custom Attributes	Type: `OBJECT` Sets a custom user attribute. This can be any key/value pair and is used to collect extra information about the user.
Date of Birth	Type: `DATETIME` The user's date of birth
Email	Type: `STRING` The user's email
Email Subscribe	Type: `STRING` The user's email subscription preference: "opted_in" (explicitly registered to receive email messages), "unsubscribed" (explicitly opted out of email messages), and "subscribed" (neither opted in nor out).

FIELD	DESCRIPTION
First Name	Type: `STRING` The user's first name
Last Name	Type: `STRING` The user's last name
Gender	Type: `STRING` The user's gender: "M", "F", "O" (other), "N" (not applicable), "P" (prefer not to say) or nil (unknown).
Home City	Type: `STRING` The user's home city.
Image URL	Type: `STRING` URL of image to be associated with user profile.
Language	Type: `STRING` The user's preferred language.
Phone Number	Type: `STRING` The user's phone number
Push Subscribe	Type: `STRING` The user's push subscription preference: "opted_in" (explicitly registered to receive push messages), "unsubscribed" (explicitly opted out of push messages), and "subscribed" (neither opted in nor out).
Subscription Groups	Type: `OBJECT` A list of subscription group IDs and states to set. Subscription group states can be either "subscribed" or "unsubscribed". Subscription Group IDs are found in the Braze dashboard.

Track Purchase

Reports that the current user made an in-app purchase.

Track Purchase is a **Web** action. The default Trigger is: `type = "track" and event = "Order Completed"`

Click to show / hide fields

FIELD	DESCRIPTION
Purchase Properties	Type: `OBJECT` Hash of properties for this purchase. Keys are limited to 255 characters in length, cannot begin with a $, and can only contain alphanumeric characters and punctuation. Values can be numeric, boolean, Date objects, strings 255 characters or shorter, or nested objects whose values can be numeric, boolean, Date objects, arrays, strings, or null. Total size of purchase properties cannot exceed 50KB.
Products	Type: `OBJECT` List of products purchased by the user

Other features

Braze Web Mode (Actions) can use the following features of Braze.

In-app Messaging

Once configured, you can trigger in-app message display as a result of several different event types. By default, all In-App Messages that a user is eligible for are automatically delivered to the user upon a session start event. A new session automatically starts when a user loads your site. If you'd like to force a new session for a user, make an Identify call with the corresponding userId for that user.

If you don't want your site to display new In-App Messages as they're received, disable automatic display and register your own display subscribers. To do this:

Create your subscriber by calling:

```
analytics.ready(function() {
  window.appboy.subscribeToNewInAppMessages(function(inAppMessages) {
      // Display the first in-app message. You could defer display here by pushing this message to code      within i
n your own application.
      // If you don't want to use Appboy's built-in display capabilities, you could alternatively pass      the in-ap
p message to your own display code here.
      window.appboy.display.showInAppMessage(inAppMessages[0]);

    // Return an array with any remaining, unhandled messages to appboy's internal queue.
    // These will be part of the inAppMessages param the next time this subscriber is invoked.
      return inAppMessages.slice(1);
    });
});
```

The `inAppMessages` parameter will be an array of `appboy.ab.InAppMessage` subclass or `appboy.ab.ControlMessage` objects, each of which has various lifecycle event subscription methods.

Push Notifications

1. To support push notifications on Chrome, you'll need to enable FCM/GCM as well as configure your site. Check out steps one and two here for detailed instructions on both.
2. Browser Registration. In order for a browser to receive push notifications, you must register it for push by calling:

```
analytics.ready(function() {
  window.appboy.registerAppboyPushMessages();
});
```

Note: Place this snippet outside of your Segment Snippet within your `script` tag.
Note: This requests push permission from the user.

To show your own push-related UI to the user before requesting push permission (known as a soft push prompt), you can test to see if the user's browser supports push by calling:

```
analytics.ready(function() {
  if (window.appboy.isPushSupported()) {
    // Add your push logic
  }
 });
```

Braze recommends checking to see if this returns `true` since not all browsers can receive push notifications. See Soft Push Prompts for instructions on setting up a soft push prompt using Braze In-App Messages.

To unsubscribe a user, call:

```
analytics.ready(function() {
  window.appboy.unregisterAppboyPushMessages();
});
```

1. Set your GCM/FCM server API key and SenderID on the Braze dashboard. You can find more details for this in Braze's Initial SDK setup for web documentation.
2. To support push notifications on Safari, add your Website Push ID into your Segment Settings UI and Segment sends it when the Braze Web SDK initializes. To get your Website Push ID, follow the first two bullet points in these instructions.

Soft Push Prompts

1. Follow step one to create a "Prime for Push" in-app messaging Campaign on the Braze dashboard.
2. Add the following snippet to your site:

```
analytics.ready(function() {
  window.appboy.subscribeToNewInAppMessages(function(inAppMessages) {
    var message = inAppMessages[0];
    if (message != null) {
      var shouldDisplay = true;

      if (message instanceof appboy.ab.inAppMessage) {
        // Read the key/value pair for msg-id
        var msgId = message.extras["msg-id"];

        // If this is our push primer message
        if (msgId == "push-primer") {
          // We don't want to display the soft push prompt to users on browsers that don't support push, or if the us
er
          // has already granted/blocked permission
          if (!appboy.isPushSupported() || appboy.isPushPermissionGranted() || appboy.isPushBlocked())     {
            shouldDisplay = false;
          }
          // Prompt the user when the first button is clicked
          message.buttons[0].subscribeToClickedEvent(function() {
            appboy.registerAppboyPushMessages();
          });
        }
      }

      // Display the message
      if (shouldDisplay) {
        appboy.display.showInAppMessage(message);
      }
     }

    // Remove this message from the array of IAMs and return whatever's left
    return inAppMessages.slice(1);
   });
 });
```

For more details on this snippet, see Braze's Soft push prompt documentation.

> Place this snippet outside of your Segment Snippet within your `script` tag.

1. When you'd like to display the Soft Push to a user, call:

```
analytics.ready(function() {
 window.appboy.logCustomEvent("prime-for-push")
});
```

Enable SDK Authentication

When "Enable SDK Authentication" is enabled, Segment will set Braze's `enableSdkAuthentication` to `true`. When this feature is enabled, the Braze SDK will append the current user's last known JWT to network requests made to Braze Servers.

Important differences from the classic Braze destination

- Braze Web Mode (Actions) supports the Braze Web integration. Braze Cloud Mode (Actions) supports server and mobile sources, but to use mobile sources in device-mode, use the Braze Classic destination.

Migration from Braze Classic

Keep the following in mind if you plan to move to Braze (Actions) from the classic Braze destination.

Braze-web settings mapping

Search for setting..

All Cloud Device-web

BRAZE-WEB CLASSIC DESTINATION SETTING	HOW TO ENABLE IN BRAZE-WEB (ACTIONS)
CONNECTION SETTINGS	
IN-APP MESSAGING	
PAGE	
OTHER SETTINGS	

FAQ

How does the Debounce Middleware Action work?

The following Debounce Middleware logic is executed at the source-level:

When an Identify call is fired on a website, Segment first caches and compares the user traits object.

- If the user traits differ from what was previously cached, the data flows through destination filters, insert functions, and then through destination mappings.
- If user traits are the same as what's cached, Segment assumes that that data was already sent to Braze and a does not make a new request to Braze.

This page was last modified: 17 Oct 2024

Create free account