do continuous experimentation embedded into their development process. ABsmartly's real-time analytics help engineering and product teams ensure that new features will improve the customer experience without breaking or degrading performance and/or business metrics.

This destination is maintained by ABsmartly. For any issues with the destination, contact ABsmartly's Support.

Benefits of ABsmartly (Actions) vs ABsmartly Classic

- **Easier Setup**: Actions-based destinations are easier to configure with clear default settings, letting you quickly get started.
- **Control and clearer mapping**: Actions-based destinations enable you to define the mapping between the data Segment receives from your source and the data Segment sends to ABsmartly.

Getting started

1. From the Segment web app, click **Catalog**.
2. Search for "ABsmartly" in the Catalog, select **ABsmartly (Actions)**, and choose which of your sources to

connect the destination to.

3. Add the following Connection Settings:
   - **Collector Endpoint**: Your ABsmartly Collector REST Endpoint. Usually `https://<your-subdomain>.absmartly.io/v1`
   - **API Key**: An existing API Key. Created under Settings > API Keys in the ABsmartly Web Console.
   - **Environment**: The environment where the events are originated matching an existing environment in ABsmartly. Created under Settings > Environments in the ABsmartly Web Console.
4. Enable the *Track Calls* mapping to send events to ABsmartly.

Destination Settings

SETTING	DESCRIPTION
API Key	*Required.* ABsmartly SDK API Key. Create SDK Api Keys in the Settings > API Keys section of the ABsmartly Web Console
Collector Endpoint	*Required.* ABsmartly Collector endpoint, for example: https://you-subdomain.absmartly.io/v1 - Contact ABsmartly Support if you don't know your Collector Endpoint.
Environment	*Required.* Environment name. Environment name needs to match what's in the Web Console. Create Environments in the Settings > Environments section of the ABsmartly Web Console

Available Presets

ABsmartly (Actions) has the following presets:

PRESET NAME	TRIGGER	DEFAULT ACTION
Page Calls	Event type = "page"	Track Goal
Exposures (Verbatim)	Event type = "track" and event = "Experiment Viewed"	Track Exposure
Screen Calls	Event type = "screen"	Track Goal
Track Calls	Event type = "track" and event != "Experiment Viewed"	Track Goal

Available Actions

Build your own Mappings. Combine supported triggers with the following ABsmartly-supported actions:

Mapping limits per destination

Individual destination instances have support a maximum of 50 mappings.

- Track Exposure
- Track Goal

Track Exposure

Send an experiment exposure event to ABsmartly

Track Exposure is a **Cloud** action. The default Trigger is: `type = "track" and event = "Experiment Viewed"`

Click to show / hide fields

FIELD	DESCRIPTION
ABsmartly Exposure Payload *	Type: `OBJECT` The ABsmartly exposure payload without any goals. Generated by the ABsmartly SDK and should not be modified.
Agent	Type: `STRING` Optional agent identifier that originated the event. Used to identify which SDK generated the event.
Application	Type: `STRING` Optional application name that originated this event. Must exist if not empty. Create Applications in the Settings > Applications section of the ABsmartly Web Console

Track Goal

Send a goal event to ABsmartly

Track Goal is a **Cloud** action. The default Trigger is: `type = "track" and event != "Experiment Viewed"`

Click to show / hide fields

FIELD	DESCRIPTION
Units *	Type: `OBJECT` The units of the goal to track. Mapping of unit name to source property in the event payload. Create Units in the Settings > Units section of the ABsmartly Web Console
Goal Name *	Type: `STRING` The name of the goal to track
Goal Properties *	Type: `OBJECT` Custom properties of the goal
Agent	Type: `STRING` Optional agent identifier that originated the event. Used to identify which SDK generated the event.
Application	Type: `STRING` Optional application name that originated this event. Must exist if not empty. Create Applications in the Settings > Applications section of the ABsmartly Web Console

If you need support setting things up, you can contact the ABsmartly support team on Slack or via email.

Sending exposures to Segment

It can be useful to send experiment exposures to Segment for visibility from other destinations. The Segment Spec includes the Experiment Viewed semantic event for this purpose.

By default, the *Track Calls* mapping will filter and not send any events with the name `Experiment Viewed` to ABsmartly.

You can install a custom event logger in ABsmartly and send exposures directly to Segment.

```
analytics.ready(function() {
    // initialize ABsmartly SDK
    const sdk = new absmartly.SDK({
        endpoint: 'https://your-absmartly-endpoint.absmartly.io/v1',
        apiKey: '<YOUR-API-KEY>',
        environment: 'development',
        application: 'YOUR-APP',
        eventLogger: (context, eventName, data) => {
            if (eventName == "exposure") {
                // filter only relevant and interesting exposures
                // if the assigned flag is false, this exposure was a treatment call that did not result in an assign
ment
                // this can happen if, for example, the experiment is no longer running, but treatment() calls are st
ill in the application code
                if (exposure.assigned) {
                    analytics.track("Experiment Viewed", {
                        experiment_id: exposure.id,
                        experiment_name: exposure.name,
                        variation_id: exposure.variant,
                        variation_name: "ABCDEFG"[exposure.variant],
                    });
                }
            }
        },
    });

    const context = sdk.createContext(request);
    context.attribute("user_agent", navigator.userAgent);

    context.ready().then((response) => {
        console.log("ABSmartly Context ready!");
        console.log(context.treatment("test-exp"));
    }).catch((error) => {
        console.log(error);
    });
});
```

Publishing experiment exposures through Segment

To publish experiment exposures through Segment, you must first configure and enable the *Exposures (Verbatim)* mapping in your ABsmartly (Actions) destination.

By enabling the *Exposures (Verbatim)* mapping in Segment, you replace the direct flow of exposure events from the ABsmartly SDK to the ABsmartly collector and instead send them to Segment for processing by the destination function.

This can be achieved by instantiating the ABsmartly SDK with a custom context publisher.

The custom publisher will publish an `Experiment Viewed` Segment event with ABsmartly's exposure data in the `properties.exposure` field as well as the normal semantic data that Segment recommends for this event.

Here is an example in Javascript.

```
analytics.ready(function() {
    // initialize ABSmartly SDK
    const sdk = new absmartly.SDK({
        endpoint: 'https://your-absmartly-endpoint.absmartly.io/v1',
        apiKey: '<YOUR-API-KEY>',
        environment: 'development',
        application: 'YOUR-APP',
    });

    // ABSmartly publisher implementation that publishes ABSmartly exposures to Segment,
    // instead of directly to the ABSmartly Collector
    // these will then be pushed by the ABSmartly segment integration to the ABSmartly collector
    class SegmentContextPublisher extends absmartly.ContextPublisher {
        constructor(segment) {
            super();

            this._segment = segment;
        }

        publish(request, sdk, context) {
            // NOTE: only exposures are expected to come via this route
            // other types of events should be tracked through the Segment API
            if (request.exposures) {
                for (const exposure of request.exposures) {
                    this._segment.track(`Experiment Viewed`, {
                        experiment_id: exposure.id,
                        experiment_name: exposure.name,
                        variation_id: exposure.variant,
                        variation_name: "ABCDEFG"[exposure.variant],
                        exposure: Object.assign({},
                            {
                                exposures: [exposure],
                            },
                            // add anything else in the a/b smartly payload that are not exposures or goals
                            ...Object.entries(request)
                                .filter(e => (e[0] !== 'exposures') && (e[0] !== 'goals'))
                                .map(e => ({[e[0]]: e[1]}))
                        )
                    });
                }
            }

            return Promise.resolve();
        }
    }

    // set this as the default publisher - all contexts created from now on will use it by default
    sdk.setContextPublisher(new SegmentContextPublisher(analytics));

    const request = {
        units: {
            userId: analytics.user().id(),
            anonymousId: analytics.user().anonymousId(),
        },
    };

    window.context = sdk.createContext(request);
    context.attribute("user_agent", navigator.userAgent);

    context.ready().then((response) => {
        console.log("ABSmartly Context ready!");
        console.log(context.treatment("test-exp"));
    }).catch((error) => {
        console.log(error);
    });
});
```

Migration from the classic ABsmartly destination

To migrate from the classic ABsmartly destination to ABsmartly (Actions), disconnect the classic ABsmartly destination before enabling the ABsmartly (Actions) destination to avoid duplicate experimentation events.

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 ABsmartly (Actions) 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`. ABsmartly SDK API Key. Create SDK Api Keys in the Settings > API Keys section of the ABsmartly Web Console
Collector Endpoint *(required)*	`string`. ABsmartly Collector endpoint, for example: https://you-subdomain.absmartly.io/v1 - Contact ABsmartly Support if you don't know your Collector Endpoint.
Environment *(required)*	`string`. Environment name. Environment name needs to match what's in the Web Console. Create Environments in the Settings > Environments section of the ABsmartly Web Console

This page was last modified: 27 Mar 2024

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