Intercom Events

Last Modified on 02/24/2021 8:57 am EST

SAP Open Connectors supports events via polling or webhooks depending on the API provider. For more information about our Events framework, see Events Overview.

Supported Events and Resources

SAP Open Connectors supports both polling and webhook events for Intercom. After receiving an event, SAP Open Connectors standardizes the payload and sends an event to the configured callback URL of your authenticated connector instance.

Polling

You can set up polling for the `events` resource. You can also copy the `events` configuration to poll other resources. See Configure Polling Through API for more information.

> **Note:** Unless configured for a specific time zone, polling occurs in UTC.

Configure Polling Through the UI

To configure polling through the UI, follow the same steps to authenticate a connector instance, and then turn on events. Select the resources to poll, and then click **Create Instance**. For more information, see Authenticate an Connector Instance with Events (UI) or the connector-specific authentication topic.

Configure Polling Through API

Use the `/instances` endpoint to authenticate with Intercom and create a connector instance with polling enabled.

> **Note:** The endpoint returns a connector instance token and id upon successful completion. Retain the token and id for all subsequent requests involving this connector instance.

To authenticate a connector instance with polling:

1. Construct a JSON body as shown below (see Parameters):

```
{
  "element": {
    "key": "intercom"
  },
  "providerData": {
    "code": ""
  },
  "configuration": {
    "oauth.api.key": "xxxxxxxxxxxxxxxxxxxx",
    "oauth.api.secret": "xxxxxxxxxxxxxxxxxxxxx",
    "oauth.callback.url": "https://xxxxxxxxxxxxxxxx/oauth",
    "oauth.user.refresher_time": null,
    "oauth.user.refresher_token": null,
    "event.notification.enabled": true,
    "event.poller.refresh_interval": "15",
    "event.vendor.type": "polling",
    "filter.response.nulls": "true",
    "event.poller.configuration": {
      "conversations": {
        "url": "/hubs/marketing/conversations?orderBy=updated_at desc",
        "idField": "id",
        "datesConfiguration": {
          "updatedDateField": "updated_at",
          "updatedDateFormat": "timestamp",
          "updatedDateTimezone": "GMT",
          "createdDateField": "created_at",
          "createdDateFormat": "timestamp",
          "createdDateTimezone": "GMT"
        }
      }
    }
  },
  "tags": [
    ""
  ],
  "name": ""
}
```

2. Call the following, including the JSON body you constructed in the previous step:

```
POST /instances
```

> **Note:** Make sure that you include the User and Organization keys in the header. For more information, see Authorization Headers, Organization Secret, and User Secret.

3. Locate the `token` and `id` in the response and save them for all future requests using the connector instance.

Example cURL with Polling

```
curl -X POST \
https://api.openconnectors.us2.ext.hana.ondemand.com/elements/api-v2/instances \
-H 'authorization: User , Organization ' \
-H 'content-type: application/json' \
-d '{
  "element": {
    "key": "intercom"
  },
  "providerData": {
    "code": "oiu3gr1g2roi12ruioiu190f9"
  },
  "configuration": {
    "oauth.api.key": "xxxxxxxxxxxxxxxxxxxx",
    "oauth.api.secret": "xxxxxxxxxxxxxxxxxxxxx",
    "oauth.callback.url": "https://mycoolapp.com/oauth",
    "event.notification.enabled": true,
    "event.poller.refresh_interval": "15",
    "event.vendor.type": "polling",
    "filter.response.nulls": "true",
    "event.poller.configuration": {
      "conversations": {
        "url": "/hubs/marketing/conversations?orderBy=updated_at desc",
        "idField": "id",
        "datesConfiguration": {
          "updatedDateField": "updated_at",
          "updatedDateFormat": "timestamp",
          "updatedDateTimezone": "GMT",
          "createdDateField": "created_at",
          "createdDateFormat": "timestamp",
          "createdDateTimezone": "GMT"
        }
      }
    }
  },
  "tags": [
    "Marketing"
  ],
  "name": "API Instance with Polling"
}
```

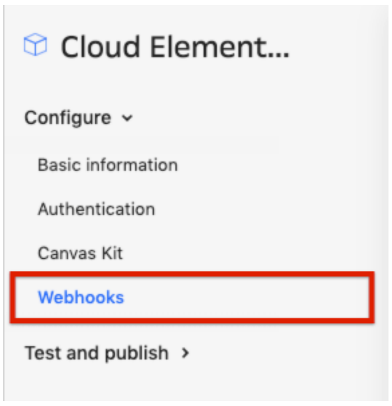
Webhooks

You can configure webhooks through the UI or through API in the JSON body of the `/instances` API call. First, you must set up webhooks in Intercom.

Set Up Webhooks

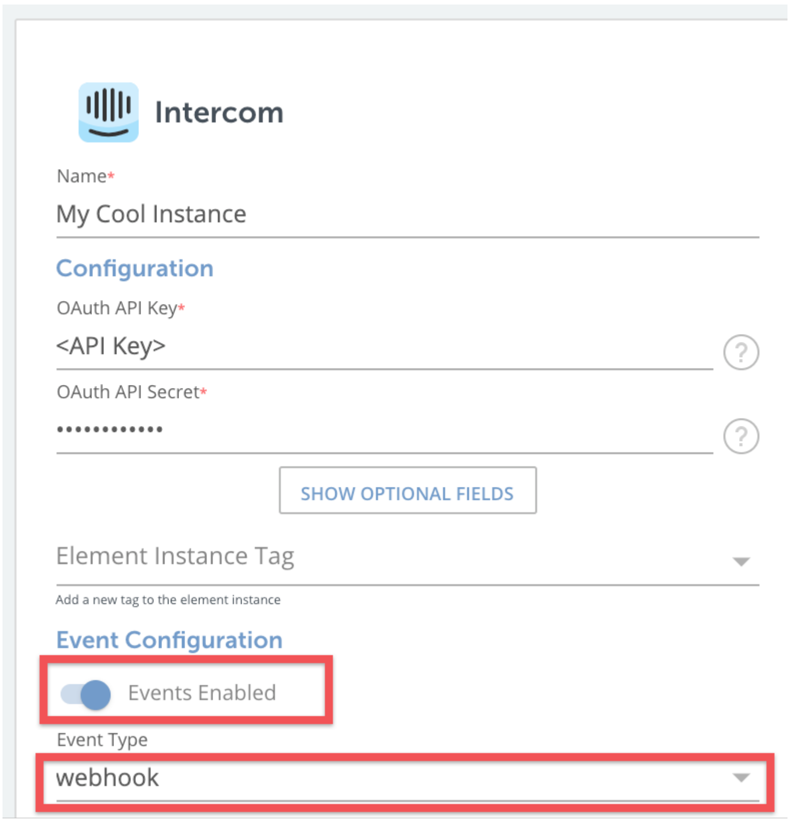
Follow these steps to set up your Intercom application with the endpoint.

1. Via a web browser, sign in to your Intercom developer account at https://app.intercom.com.
2. In the developer app dashboard, select the app that you'd like to send webhooks for.
3. Select **Webhooks** from the navigation items displayed on the left.



The Webhooks page appears, where you see a field to enter your request endpoint URL.

4. In order to enter this URL, create an instance of the Intercom connector. While creating an instance, enable **Events** under **Events Configuration** and select **Webhook** as the **Event Type**.



Name*
My Cool Instance

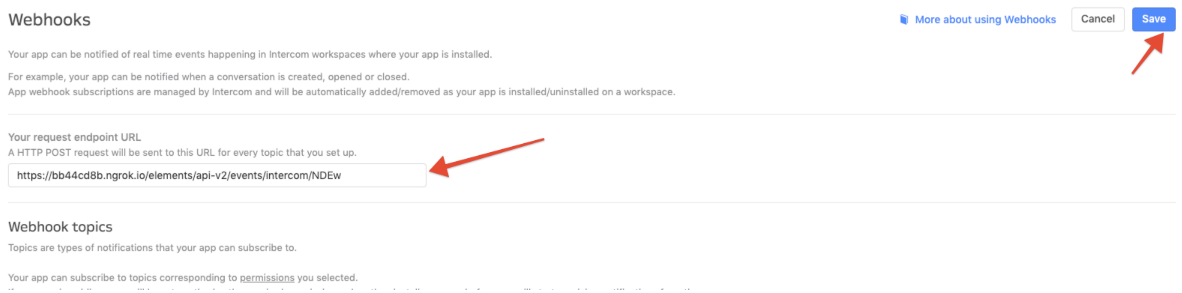
OAuth API Key*
<API Key>

OAuth API Secret*
••••••••••••

Events Enabled

Event Type
webhook

5. Click **Create Instance.**
6. Choose an account through which you'd like to create an instance.
7. You're now taken back to SAP Open Connectors. Select **Instances** on the navigation pane on your left and edit your instance.
8. You will see a field named **Webhook URL.** Copy the URL in that field.
9. Back on the Intercom page, click **Edit**, complete step 4 and click **Save.**



Webhooks

Your app can be notified of real time events happening in Intercom workspaces where your app is installed.

For example, your app can be notified when a conversation is created, opened or closed.
App webhook subscriptions are managed by Intercom and will be automatically added/removed as your app is installed/uninstalled on a workspace.

Your request endpoint URL
A HTTP POST request will be sent to this URL for every topic that you set up.
https://bb44cd8b.ngrok.io/elements/api-v2/events/intercom/NDEw

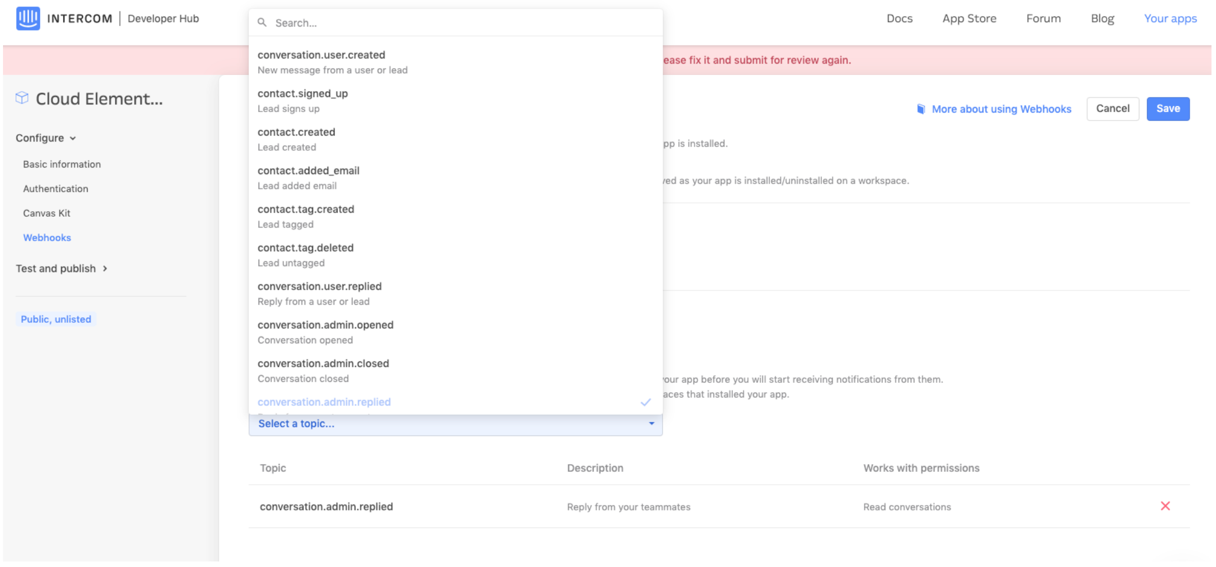
Webhook topics
Topics are types of notifications that your app can subscribe to.

Your app can subscribe to topics corresponding to permissions you selected.

10. Next, you'll need to set up individual subscriptions for each object and event type for which you'd like to receive a notification. Do this by selecting **Webhook Topics** according to your needs. **Webhook Topics** are types of

notifications that your app can subscribe to.

11. Click **Edit** on the top right corner and select topic/s to configure the subscription/s as shown below.


INTERCOM | Developer Hub
Search...
Docs
App Store
Forum
Blog
Your apps
ease fix it and submit for review again.
Cloud Element...
Configure
Basic information
Authentication
Canvas Kit
Webhooks
Test and publish
Public, unlisted
More about using Webhooks
Cancel
Save
pp is installed.
ved as your app is installed/uninstalled on a workspace.
your app before you will start receiving notifications from them.
aces that installed your app.
Select a topic...

conversation.user.created
New message from a user or lead
contact.signed_up
Lead signs up
contact.created
Lead created
contact.added_email
Lead added email
contact.tag.created
Lead tagged
contact.tag.deleted
Lead untagged
conversation.user.replied
Reply from a user or lead
conversation.admin.opened
Conversation opened
conversation.admin.closed
Conversation closed
conversation.admin.replied ✓

Topic	Description	Works with permissions
conversation.admin.replied	Reply from your teammates	Read conversations

Configure Webhooks Through the UI

To configure webhooks through the UI, follow the same steps to authenticate a connector instance, and then turn on events. For more information, see Authenticate an Connector Instance with Events (UI) or the connector-specific authentication topic.

Configure Webhooks Through API

Use the `/instances` endpoint to authenticate with Intercom and create a connector instance with webhooks enabled.

> **Note:** The endpoint returns a connector instance token and id upon successful completion. Retain the token and id for all subsequent requests involving this connector instance.

To authenticate a connector instance with webhooks:

1. Get an authorization grant code by completing the steps in Getting a redirect URL and Authenticating users and receiving the authorization grant code.

2. Construct a JSON body as shown below (see Parameters):

```
{
  "element": {
    "key": "intercom"
  },
  "providerData": {
    "code": ""
  },
  "configuration": {
    "oauth.api.key": "",
    "oauth.api.secret": "",
    "oauth.callback.url": "",
    "event.notification.enabled": true,
    "event.vendor.type": "webhooks",
    "event.notification.callback.url": "",
    "event.notification.signature.key": ""
  },
  "tags": [
    ""
  ],
  "name": ""
}
```

3. Call the following, including the JSON body you constructed in the previous step:

```
POST /instances
```

> **Note:** Make sure that you include the User and Organization keys in the header. For more information, see Authorization Headers, Organization Secret, and User Secret.

4. Locate the `token` and `id` in the response and save them for all future requests using the connector instance.

Example cURL

```
curl -X POST \
 https://api.openconnectors.us2.ext.hana.ondemand.com/elements/api-v2/instances \
  -H 'authorization: User , Organization ' \
  -H 'content-type: application/json' \
  -d '{
  "element": {
    "key": "intercom"
  },
  "providerData": {
    "code": "xoz8AFqScK2ngM04kSSM"
  },
  "configuration": {
    "oauth.api.key": "Rand0MAP1-key",
    "oauth.api.secret": "fak3AP1-s3Cr3t",
    "oauth.callback.url": "https://mycoolapp.com",
    "event.notification.enabled": true,
    "event.vendor.type": "webhooks",
    "event.notification.callback.url": "https://mycoolapp.com/events",
    "event.notification.signature.key": "xxxxxxxxxxxxxxxxxxxxxxxxxx"
  },
  "tags": [
    "Marketing"
  ],
  "name": "API Instance"
}'
```

Parameters

API parameters are in `code formatting` .

Parameter	Description	Data Type
`key`	The connector key. intercom	string
`code`	The authorization grant code returned from the API provider in an OAuth2 authentication workflow.	string
Name `name`	The name for the connector instance created during authentication.	string
`authentication.type`	Identifies how you are authenticating with Intercom.	string
`oauth.callback.url`	OAuth 2.0 authentication only. The URL where you want to redirect users after they grant access. This is `https://auth.cloudelements.io/oauth` , the **Callback URL** that you noted in API Provider Setup.	string
`oauth.api.key`	OAuth 2.0 authentication only. The Client ID from Intercom. This is the **Client ID** that you noted in API Provider Setup	string
`oauth.api.secret`	OAuth 2.0 authentication only. The Client Secret from Intercom. This is the **Client Secret** that you noted in API Provider Setup.	string
Events Enabled `event.notification.enabled`	*Optional*. Identifies that events are enabled for the connector instance. Default: `false`	boolean
Event Type `event.vendor.type`	*Optional*. Identifies the type of events enabled for the instance, either `webhook` or `polling` .	string
Event Notification Callback URL `event.notification.callback.url`	*For webhooks and polling.* The URL where your app can receive events.	string
Callback Notification Signature Key `event.notification.signature.key`	*For webhooks and polling.* *Optional* A user-defined key for added security to show that events have not been tampered with. This can be any custom value that you want passed to the callback handler listening at the provided Event Notification Callback URL.	string
Objects to Monitor for Changes `event.objects`	*For webhooks and polling.* *Optional* Comma separated list of objects to monitor for changes.	string
Event poller refresh interval (mins) `event.poller.refresh_interval`	*For polling only.* A number in minutes to identify how often the poller should check for changes.	number
Configure Polling `event.poller.configuration`	Optional*. Configuration parameters for polling.	JSON object
resource name e.g., contact, account, etc.	The configuration of an individual resource.	JSON object
URL `url`	The url to query for updates.	String

Parameter	Description	Data Type
ID Field `idField`	The field that is used to uniquely identify an object.	String
Advanced Filtering `datesConfiguration`	Configuration parameters for dates in polling	JSON Object
Updated Date Field `updatedDateField`	The field that identifies an updated object.	String
Updated Date Format `updatedDateFormat`	The date format of the field that identifies an updated object.	String
Created Date Field `createdDateField`	The field that identifies a created object.	String
Created Date Format `createdDateFormat`	The date format of the field that identifies a created object.	String
tags	*Optional*. User-defined tags to further identify the instance.	string