

Box Events

Last Modified on 02/06/2020 11:59 am EST

SAP Cloud Platform 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 Cloud Platform Open Connectors supports webhook events for Box. You can configure webhooks through UI or through API. Both methods are discussed in detail further below. For more information about webhooks at Box including the currently available webhooks, see [their webhooks documentation](#).

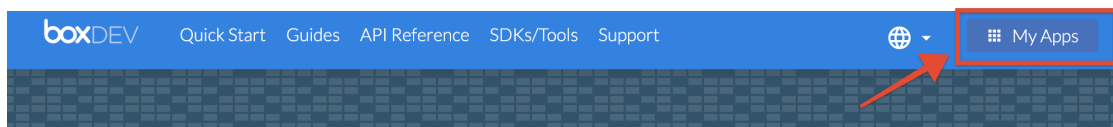
Webhooks

Box provides two versions for webhooks - Webhooks v1 and Webhooks v2. Depending on your needs you can opt for v1 or v2. Webhooks v1 offers limited events on all objects in your profile. Webhooks v2 offers all events on specific objects (*files or folders*) in your profile. To enable Webhooks v1, you must set up webhooks in Box first and then provision an instance with the Box . Webhooks v2, on the other hand, offers APIs which are enabled as normal APIs in SAP Cloud Platform Open Connectors platform.

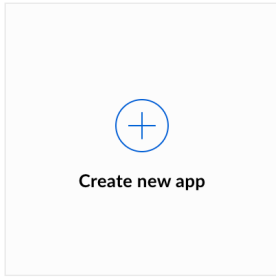
Webhooks v1

To enable webhooks v1, follow the steps below to set up your Box application.

1. On [Box](#) website, click on the **My Apps** button on the top.






2. Log in using your credentials for Box. If you don't already have an account with them, create an account.
3. Once you're logged in, you need to create an app. If you already have an app, jump to step 8.
4. To create a new app, click on the **Create new app** button.



5. Choose what type of app you wish to build and click **Next**. In the example below, we are building a Custom app.

CREATE A NEW BOX APP

Let's get started. What type of app are you building?

 <p>Custom app</p> <p>Build a standalone app with Box's content services, such as managing and rendering files and enabling end-user collaboration.</p> <p><i>For developers using Box's content services without requiring Box user accounts.</i></p>	 <p>Enterprise integration</p> <p>Extend your Box instance with programmatic processes and backend integrations, such as user, group and event management.</p> <p><i>For Box admins and developers building an integration for their existing Box users.</i></p>	 <p>Partner integration</p> <p>Allow users to access, edit and save their Box content within your third-party app, such as an e-signature or project management service.</p> <p><i>For developers building an integration for existing Box users.</i></p>
--	--	--

6. Next, select the **Standard OAuth 2.0 (User Authentication)** type for your app and click **Next**.

Authentication method

We've recommended an authentication method based on the type of app you've chosen.
You may change the authentication method below. [Learn more.](#)

RECOMMENDED FOR CUSTOM APP:

OAuth 2.0 with JWT (Server Authentication)
Allows your app to authenticate directly to Box using a digitally signed JSON Web Token instead of user credentials. For use with service accounts and app users.

OTHER AVAILABLE AUTHENTICATION METHODS:

App token (server authentication)
Provides API functionality scoped to previewing content in your application. This will NOT allow you to create and manage users in Box. Please read our documentation on New Box view and make sure that it fits your application's requirements before proceeding.

Standard OAuth 2.0 (User Authentication)
Requires Box users to log in with a username and password to authorize your app to access content in their account.

Back

Next

7. Give your app a name and click on the **Create app** button.

What would you like to name your app?

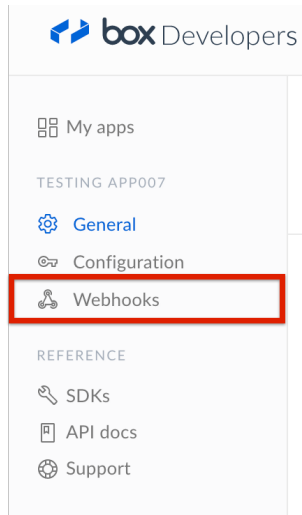
Don't worry – you can change this later.

By clicking 'Create app', you agree to the terms of the [Box Developer Agreement](#) and the [Box Privacy Policy](#).

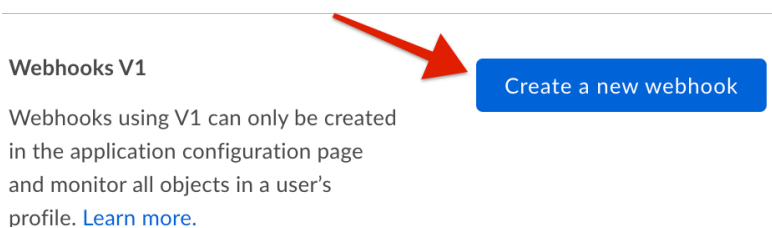
Back

Create app

8. If you have an app that already exists, click on the app and on the navigation panel to your left, click **Webhooks**.



9. In the Webhooks V1 section, click **Create a new webhook**.



10. Configure it based on your requirements and click **Save Webhook**.
11. If the application is for a production account, as opposed to a sandbox account, then you will need to get in contact with Box support to enable webhooks for that application.

After you set up webhooks on the Box website, provision an instance with the Box connector with events enabled to get webhooks working.

Webhooks v2

Box Webhooks v2 enables webhooks by offering APIs. These APIs are offered as normal APIs by SAP Cloud Platform Open Connectors.

Here is what you need to do to enable webhooks v2.

1. Provision an instance with the Box connector with Events enabled.

2. Call our normalized API `POST/webhooks` and mention the `target`, that is, the folder/file you want events to be enable for and the `triggers`.

You have now enabled webhooks v2.

Note: There are a few limitations affecting webhooks v2 in Box, which apply to the above steps. You can find them [here](#).

Configure Webhooks Through UI

To configure webhooks through the UI, follow the same steps to authenticate a connector instance, and then turn on events. Enter the webhook information, and then click **Create Instance**. For more information, see [Authenticate an Connector Instance with Events \(UI\)](#) or the connector-specific authentication topic. You also need to set up webhooks on Box.com

Configure Webhooks Through API

Use the `/instances` endpoint to authenticate with Box 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": "box"
  },
  "providerData": {
    "code": ""
  },
  "configuration": {
    "oauth.callback.url": "",
    "oauth.api.key": "",
    "oauth.api.secret": "",
    "event.notification.enabled": true,
    "event.notification.callback.url": "",
    "events.list.ids": ""
  },
  "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": "box"
  },
  "providerData": {
    "code": "xoz8AFqScK2ngM04kSSM"
  },
  "configuration": {
    "oauth.callback.url": "https://mycoolapp.com",
    "oauth.api.key": "xxxxxxxxxxxxxxxxxxxx",
    "oauth.api.secret": "xxxxxxxxxxxxxxxxxxxx"
    "event.notification.enabled": true,
    "event.notification.callback.url": "https://mycoolapp.com/events",
    "event.notification.signature.key": "xxxxxxxxxxxxxxxxxxxx"
  },
  "tags": [
    "Docs"
  ],
  "name": "API Instance"
}'

```

Parameters

API parameters not shown in the SAP Cloud Platform Open Connectors are in

`code formatting` .

Parameter	Description	Data Type
<code>key</code>	The connector key.	string
<code>code</code>	The authorization grant code returned from the API provider in an OAuth2 authentication workflow.	string
Name <code>name</code>	The name for the connector instance created during authentication.	Body
<code>oauth.callback.url</code>	The URL where you want to redirect users after they grant access. This is the Callback URL that you noted in	

Parameter	Description	Data Type
<code>oauth.api.key</code>	The Client ID from Box. This is the Client ID that you noted in the API Provider Setup section .	string
<code>oauth.api.secret</code>	The Client Secret from Box. This is the Client Secret that you noted in the API Provider Setup section .	string
Events Enabled <code>event.notification.enabled</code>	<i>Optional.</i> Identifies that events are enabled for the connector instance. Default: <code>false</code> .	boolean
Event Notification Callback URL <code>event.notification.callback.url</code>	The URL where you want SAP Cloud Platform Open Connectors to send the events.	string
Callback Notification Signature Key <code>event.notification.signature.key</code>	<i>Optional.</i> A user-defined key for added security to show that events have not been tampered with.	string
tags	<i>Optional.</i> User-defined tags to further identify the instance.	string