Twitter Authenticate a Connector

Last Modified on 03/16/2020 10:16 pm EDT

On this page

You can authenticate with Twitter to create your own instance of the Twitter connector through the UI or through APIs. Once authenticated, you can use the connector instance to access the different functionality offered by the Twitter platform.

Authenticate Through the UI

Use the UI to authenticate with Twitter and create a connector instance. As you authenticate with Twitter via OAuth 1.0, all you need to do is add a name for the instance. After you create the instance, you'll log in to Twitter to authorize SAP Cloud Platform Open Connectors to access your account. For more information about authenticating a connector instance, see Authenticate a Connector Instance (UI).

After successfully authenticating, we give you several options for next steps. Make requests using the API docs associated with the instance, map the instance to a common resource, or use it in a formula template.

Authenticate Through API

Twitter Follows the OAuth1 Authorization protocol.

For more information about the Twitter API, please view their API documentation.

Step 1. Get Connectors OAuth Token

- HTTP Header: None
- HTTP Verb: GET
- Request URL: /elements/{keyOrId}/oauth/token
- Request Body: None

- Query Parameters:

- key - twitter
- apiKey– - the key obtained from registering your app with the provider
- apiSecret – the secret obtained from registering your app with the provider
- callbackUrl – the URL that you supplied to the provider when registering your app

Description: The result of this API invocation returns a requestToken and Secret from the endpoint, which are used to retrieve the redirect URL. The requestToken is used in the GET /elements/{keyOrId}/oauth/url call.

Each of the OAuth API calls will be shown below.

Example cURL Command:

```
curl -X GET
-H 'Content-Type: application/json'
'https://api.openconnectors.us2.ext.hana.ondemand.com/elements/api-v2/eleme
nts/twitter/oauth/token?apiKey=insert_fake_api_key&apiSecret=insert_fake_ap
i_secret&callbackUrl=https://www.mycoolapp.com/oauth&state=twitter'
```

Response:

```
{
  "token": "qyprd1Twij60MH06lKGUZTJwx7tbzpPQx6aZnvKe0xI7",
  "secret": "KKSbL0J2wLGMqhMXTEh1ERkSx9tsIbAHUevF"
}
```

Twitter expects a token and secret. These are contained in the response to the initial GET request. Please make note of the token and secret. The token is needed in the GET /elements/{keyOrId}/oauth/url call which is shown below.

Step 2. Get Connectors OAuth URL

- HTTP Header: None
- HTTP Verb: GET
- Request URL: /elements/{keyOrId}/oauth/url
- Request Body: None
- Query Parameters:
- key - twitter

- apiKey– - the key obtained from registering your app with the provider
- apiSecret – the secret obtained from registering your app with the provider
- callbackUrl – the URL that you supplied to the provider when registering your app,
- requestToken - the token obtained from the GET /elements/{keyOrId}/oauth/token call (previous step).

Description: The result of this API invocation is an OAuth redirect URL from the endpoint. Your application should now redirect to this URL, which in turn will present the OAuth authentication and authorization page to the user. When the provided callback URL is executed, a code value will be returned, which is required for the Create Instance API.

Example cURL Command:

```
curl -X GET
-H 'Content-Type: application/json'
'https://api.openconnectors.us2.ext.hana.ondemand.com/elements/api-v2/elements/twitter/oauth/url?apiKey=insert_fake_api_key&apiSecret=insert_fake_api_secret&callbackUrl=https://www.mycoolapp.com/oauth&requestToken=insert_fake_request_token&state=twitter'
```

Response:

```
{
    "element": "twitter",
    "oauthUrl": "https://appcenter.intuit.com/Connect/Begin?oauth_token=qyprdJHtIbwm3sGOoOCvXuv2Cs8fsQrZFjJWe4HEZAyb0&oauth_callback=http%3A%2F%2Fwww.cloud-elements.com%3Fstate%3Dtwitter"
}
```

Step 3. Create an Instance

Your application should now redirect to the oauthUrl returned in step 2, which in turn will present the OAuth authentication and authorization page to the user.

HANDLE CALLBACK FROM THE ENDPOINT

After the user successfully authenticates, the provided callback URL is executed. The callback URL will contain several parameters, listed below. These additional parameters, along with the original API key and API secret are required for the Create Instance API.

The parameters that you will need to parse from the callback URL are listed below, along with an example of what the callback URL should look like. **oauth_token oauth_verifier**

```
https://www.mycoolapp.com/oauth?state=twitter&oauth_token=qyprdlGChtClXwBpA
w1vm1fJSC3mQqS3dGX0PPphEzNEUI9s&oauth_verifier=br6qctk
```

These values will be used to create an Instance. An example of this process along with sample JSON will be shown in the next section.

To provision your Twitter connector, use the /instances API.

Below is an example of the provisioning API call.

- **HTTP Headers**: Authorization- User , Organization
- **HTTP Verb**: POST
- **Request URL**: /instances
- **Request Body**: Required – see below
- **Query Parameters**: none

Description: token is returned upon successful execution of this API. This token needs to be retained by the application for all subsequent requests involving this connector instance.

A sample request illustrating the /instances API is shown below.

HTTP Headers:

```
Authorization: User , Organization
```

This instance.json file must be included with your instance request. Please fill your information to provision. The “key” into SAP Cloud Platform Open Connectors Twitter is “twitter”. This will need to be entered in the “key” field below depending on which connector you wish to instantiate.

```
{
  "element": {
    "key": "twitter"
  },
  "providerData": {
    "oauth_token": "",
    "oauth_verifier": "",
    "secret": ""
  },
  "configuration": {
    "oauth.api.key": "",
    "oauth.api.secret": "",
    "oauth.callback.url": ""
  },
  "tags": [
    ""
  ],
  "name": ""
}
```

Here is an example cURL command to create an instance using /instances API.

Example Request:

```
curl -X POST
-H 'Authorization: User , Organization '
-H 'Content-Type: application/json'
-d @instance.json
'https://api.openconnectors.us2.ext.hana.ondemand.com/elements/api-v2/instances'
```

If the user does not specify a required config entry, an error will result notifying her of which entries she is missing.

Below is a successful JSON response:

```
{
  "id": 1234,
  "name": "test",
  "token": "3sU/S/kZC46BaABPS7EAuhT+ukiEHkI=",
  "element": {
    "id": 1359,
    "name": "Twitter",
    "key": "twitter",
    "description": "Add a Twitter Instance to connect your existing Twitter
account to the Social Hub, allowing you to manage statuses and followers ac
ross multiple Social connectors. You will need your Twitter account informa
tion to add an instance.",
    "image": "https://abs.twimg.com/a/1426096855/images/oauth_application.p
ng",
    "active": false,
    "deleted": false,
    "typeOauth": false,
    "trialAccount": false,
    "resources": [],
    "transformationsEnabled": true,
    "bulkDownloadEnabled": false,
    "bulkUploadEnabled": false,
    "cloneable": true,
    "authentication": {
      "type": "oauth1"
    },
    "hub": "social",
    "protocolType": "http",
    "parameters": [],
    "private": false
  },
  "provisionInteractions": [],
  "valid": true,
  "disabled": false,
  "maxCacheSize": 0,
  "cacheTimeToLive": 0,
  "configuration": {
    "base.url": "https://api.twitter.com/1.1",
    "oauth.api.secret": "API_SECRET",
    "oauth.token.url": "https://api.twitter.com/oauth/access_token",
    "oauth.user.token.secret": "API_SECRET",
    "pagination.max": "100",
    "event.vendor.type": "polling",
    "oauth.request.url": "https://api.twitter.com/oauth/request_token",
    "oauth.user.token": "794566698309472260-XXX",
    "oauth.authorization.url": "https://api.twitter.com/oauth/authorize",
    "pagination.type": "page",
    "event.poller.refresh_interval": "15",
    "event.notification.callback.url": null,
    "oauth.request.authorization.type": "query",
    "oauth.callback.url": "http://localhost:8080/elements/jsp/home.jsp",
    "oauth.api.key": "API_KEY",
    "pagination.page.startindex": "0",
    "event.notification.enabled": "false"
  },
  "eventsEnabled": false,
  "traceLoggingEnabled": false,
  "cachingEnabled": false,
  "externalAuthentication": "none",
    "user": {
      "id": 9723
    }
  }
```

Note: Make sure you have straight quotes in your JSON files and cURL commands. Please use plain text formatting in your code. Make sure you do not have spaces after the in the cURL command.

Instance Configuration

The content in the `configuration` section or nested object in the body posted to the `POST /instances` or `PUT /instances/{id}` APIs varies depending on which connector is being instantiated. However, some configuration properties are common to all connectors and available to be configured for all connectors. These properties are -

- `event.notification.enabled` : This property is a `boolean` property, and determines if event reception (via `webhook` or `polling`) is enabled for the connector instance. This property defaults to *false*.
- `event.vendor.type` : When `event.notification.enabled` property is set to *true*, this property determines the mechanism to use to receive or fetch changed events from the service endpoint. The supported values are `webhook` and `polling` . Most connectors support one mechanism or the other, but some like Salesforce.com support both mechanisms. This property is *optional*.
- `event.notification.type` : This property can be used to determine how an event notification should be sent to the consumer of the connector instance, in most cases your application. Currently, `webhook` is the only supported value for this property. This means that when an event is received by the connector instance, it will get forwarded to the provided `event.notification.callback.url` via a `webhook` to you. This property is *optional*.
- `event.notification.callback.url` : As mentioned above, the value of this property is an `http` or `https` URL to which we will post the event for consumption by your application. This property is *optional*.
- `filter.response.nulls` : This property defaults to *true*, i.e., it's `boolean` property, and determines if `null` values in the response `JSON` should or should not be filtered from the response returned to the consuming application. By default, all `null` values are filtered from the response before sending the response to the consuming application.