SharePoint Authenticate a Connector

Last Modified on 03/05/2026 12:39 pm EST

Notice: Starting April 2, 2026, Azure ACS–based authentication will be deprecated. After this date, only Microsoft Graph–based authentication will be supported. Customers should transition to Microsoft Graph authentication as soon as possible by setting 'Use graph authentication' to True.

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

Authenticate Through the UI

Use the UI to authenticate with Microsoft and create a connector instance. Because you authenticate with Microsoft via OAuth 2.0, all you need to do is add a name for the instance and provide the Sharepoint site address, API key, and API secret you identified in Sharepoint API Provider Setup. After you create the instance, you'll log in to Microsoft to authorize SAP Open Connectors access to 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

SharePoint is a Documents Platform. When you provision an instance, your app will have access to the different functionality offered by the SharePoint platform.

Step 1. Get Connectors OAuth Information

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

- Query Parameters:

- **key** - sharepoint

- **apiKey** - the key obtained from registering your app with the provider

- **apiSecret** – the secret obtained from registering your app with the provider

- **siteAddress** - Your SharePoint Site Address

- **callbackUrl** – the URL that you supplied to the provider when registering your app, state – any custom value that you want passed to the callback handler listening at the provided callback URL.

- **documentLibrary** - Your SharePoint Document Library

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/sharepoint/oauth/url?apiKey=fake_sharepoint_client_id&apiSecret=fake_sharepoint_client_secret&siteAddress=yoursharepointsiteaddress.sharepoint.com&callbackUrl=http://fake.oauth.callback/url&state=sharepoint'
```

Response:

```
{"oauthUrl":"https://yoursiteaddress.sharepoint.com/_layouts/15/OAuthAuthorize.aspx?client_id=fake_sharepoint_client_id&client_secret=fake_sharepoint_client_secret&redirect_uri=https%3A%2F%2Fwww.yourcallbackurl.com&response_type=code&scope=Web.Write%20Web.Read%20Web.Manage&state=sharepoint","element":"sharepoint"}
```

Handle Callback from the Endpoint: Upon successful authentication and authorization by the user, the endpoint will redirect to the callback URL you provided when you setup your application with the endpoint, in our example, https://www.mycoolapp.com/auth. The endpoint will also provide two query string parameters: “state” and “code”. The value for the “state” parameter will be the name of the endpoint, e.g., “sharepoint” in our example, and the value for the “code” parameter is the code required by SAP Open Connectors to retrieve the OAuth access and refresh tokens from the endpoint. If the user denies authentication and/or authorization, there will be a query string parameter called “error” instead of the “code” parameter. In this case, your application can handle the error gracefully.

Step 2. Create an Instance

To provision your SharePoint 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 <INSERT_USER_SECRET>, Organization <INSERT_ORGANIZATION_SECRET>
```

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

```
{
  "element": {
    "key": "sharepoint"
  },
  "providerData": {
    "code": "Code on the Return URL"
  },
  "configuration": {
    "oauth.callback.url": "https://www.yourcallbackurl.com",
    "oauth.api.key": "<INSERT_SHAREPOINT_CLIENT_ID>",
    "oauth.api.secret": "<INSERT_SHAREPOINT_CLIENT_SECRET>",
    "sharepoint.document.library": "<INSERT_SHAREPOINT_DOCUMENT_LIBRARY_NAME>",
    "oauth.scope": "Web.Write Web.Read Web.Manage",
    "sharepoint.site.address": "<INSERT_SHAREPOINT_SITE_ADDRESS_NAME>",
    "document.tagging": false
  },
  "tags": [
    "<INSERT_TAGS>"
  ],
  "name": "<INSERT_INSTANCE_NAME>"
}
```

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

Example Request:

```
curl -X POST
-H 'Authorization: User <INSERT_USER_SECRET>, Organization <INSERT_ORGANIZATION_SECRET>'
-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": 123,
  "name": "Test",
  "token": "5MOr3Sl/E4kww6mTjmjBYV/hAUAzz1g=",
  "element": {
    "id": 30,
    "name": "Sharepoint",
    "key": "sharepoint",
    "description": "Add a SharePoint Instance to connect your existing SharePoint account to the Documents Hub, allo
wing you to manage files and folders. You will need your SharePoint account information to add an instance.",
    "image": "elements/provider_sharepoint.png",
    "active": true,
    "deleted": false,
    "typeOauth": true,
    "trialAccount": false,
    "existingAccountDescription": "Give your application access to your existing
 SharePoint accountEnter your credentials and details for your SharePoint Account",
    "configDescription": "If you do not have an Sharepoint account, you can create one at Office 365 Signup",
    "transformationsEnabled": false,
    "authentication": {
      "type": "oauth2"
    },
    "hub": "documents"
  },
  "provisionInteractions": [],
  "valid": true,
  "disabled": false,
  "maxCacheSize": 0,
  "cacheTimeToLive": 0,
  "eventsEnabled": true,
  "eventsNotificationCallbackUrl": "https://console.cloud-elements.com/elements/api-v2/events/sharepoint",
  "cachingEnabled": false
}
```


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 for SAP Open Connectors 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 connectors, e.g., 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 SAP Open Connectors 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.