Revoking an OAuth Token and Reauthenticating an Connector Instance

Last Modified on 09/21/2020 12:12 am EDT

There are a handful of connector-specific articles available that discuss how to reauthenticate through the API as opposed to through the SAP Cloud Platform Open Connectors user interface, but this article's purpose is to provide guidelines on reauthenticating connector instances through an example. Revoking an OAuth token can be useful when testing your application and handling response codes other than 200 when an instance needs to be re-authenticated. As there can be specific requirements between each API provider (such as Box, Salesforce, or Quickbooks) this article will not address how to create a cloud service OAuth application and the discussion relies on OAuth credentials having already been obtained.

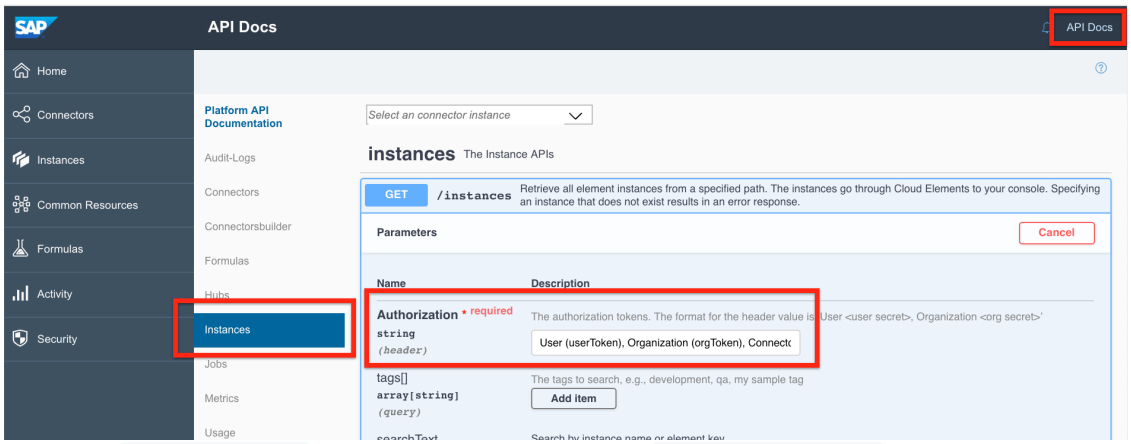
The first question that likely crosses your mind when thinking about how to reauthenticate instances is "how can I simulate invalidated/expired/revoked tokens for my instance?" Each software vendor and cloud service endpoint that you interact with can have their own unique ways of allowing you to revoke user tokens through their UI or through API calls to `revoke` endpoints, but here are a few examples of calls to revoke OAuth tokens through API:

```
Box:
curl -X POST \
  https://api.box.com/oauth2/revoke \
  -H 'Cache-Control: no-cache' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'client_id=MY_CLIENT_ID&client_secret=MY_CLIENT_SECRET&token=MY_TOKEN'

Salesforce:
curl -X POST \
  'https://na54.salesforce.com/services/oauth2/revoke?token=MY_TOKEN' \
  -H 'Cache-Control: no-cache' \
  -H 'Content-Type: application/json' \

Adobe Sign:
curl -X POST \
  'https://secure.na1.echosign.com/oauth/revoke?token=MY_TOKEN' \
  -H 'Cache-Control: no-cache' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
```

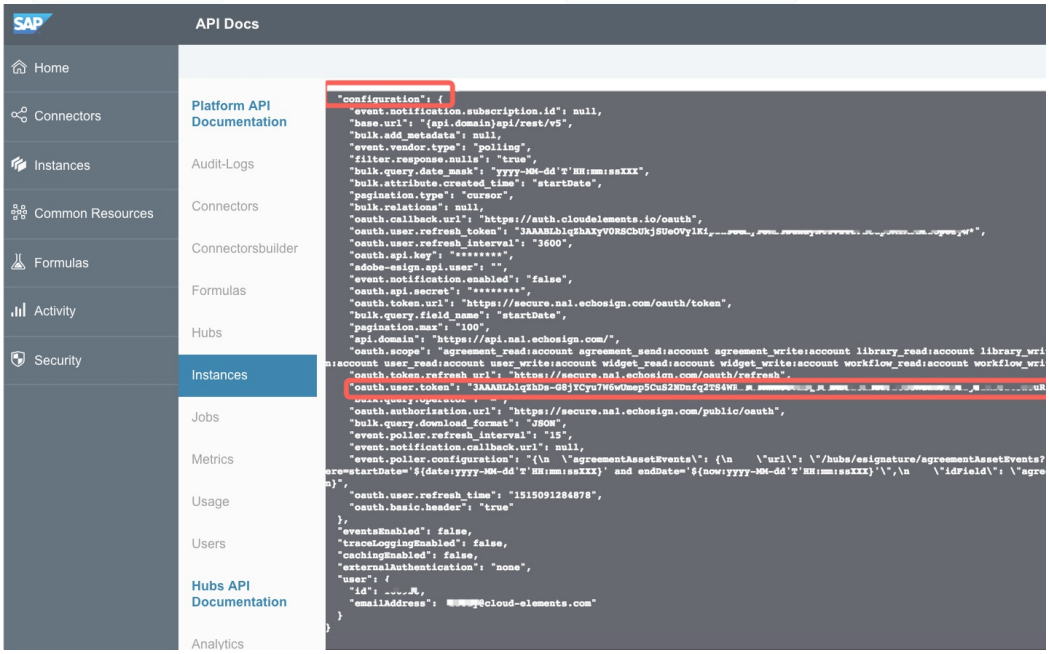
To revoke a user's OAuth token you must first determine the existing token value. One way to do this is with `GET /instances` (with the full Authorization header of your instance user/organization/SAP Cloud Platform Open Connectors tokens) and retrieve the `oauth.user.token` value from the configuration object. You can get this value through other methods as well, but this API GET call is used in this example. By hitting the above endpoints with the appropriate OAuth Token you can invalidate the authorization for that SAP Cloud Platform Open Connectors instance, then go through the OAuth flow to retrieve a new Provider Code and simply pass the new code to `PATCH /instances` (again with the full SAP Cloud Platform Open Connectors Authorization header for your existing connector instance) to reauthenticate it.

To illustrate, let's take the SAP Cloud Platform Open Connectors Adobe Sign for example:

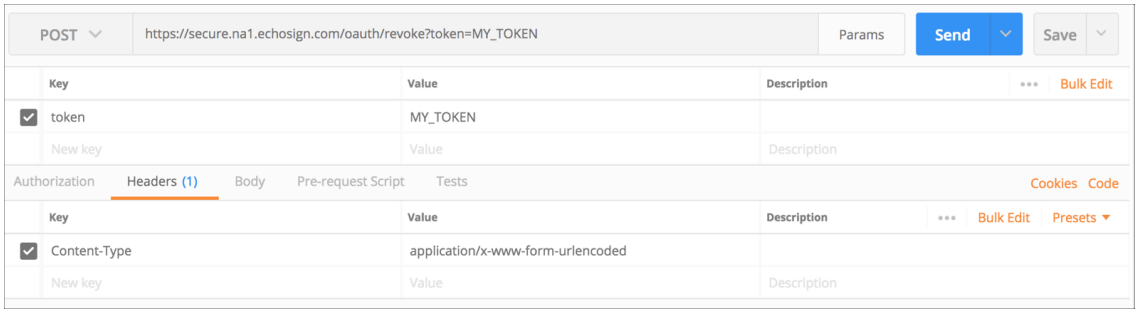
1. `GET /instances` with the full authorization header (User, Organization, and Connector tokens).



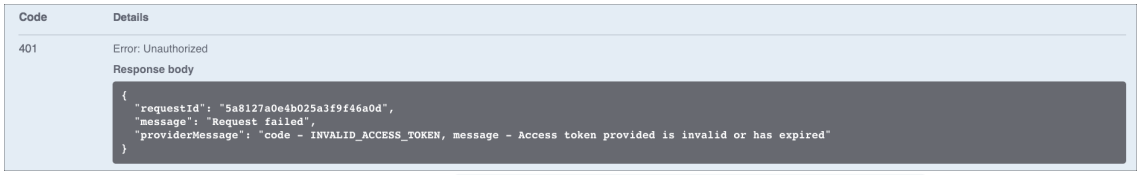
2. Locate the `configuration` object, and retrieve the current `oauth.user.token` value.



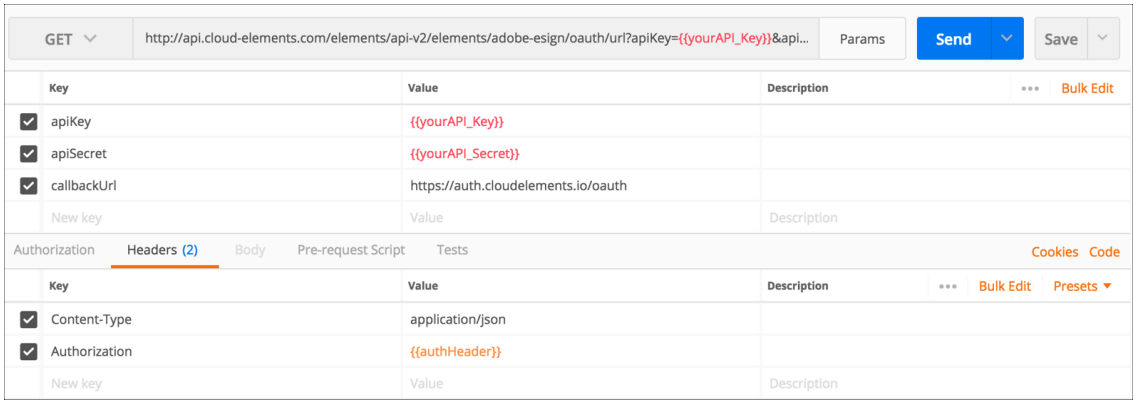
3. Make an API call directly against the API provider's endpoint to revoke the OAuth token, and supply the required parameters/payload. Confirm that a successful 200 response is returned indicating that the revocation was successful.



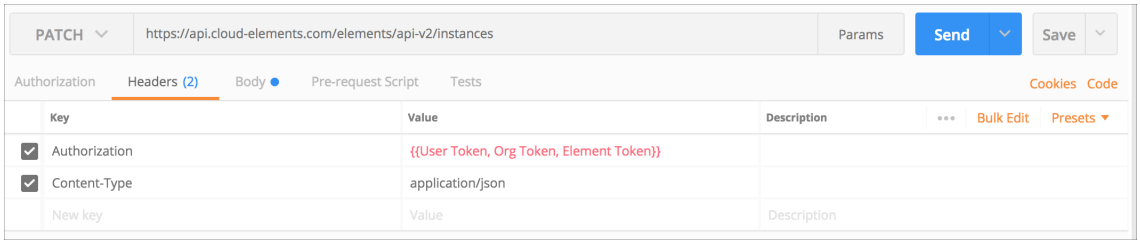
4. Attempt to make an API request to any endpoint of the SAP Cloud Platform Open Connectors instance, and confirm that a response is returned such as **401** "Request failed" due to INVALID_ACCESS_TOKEN.

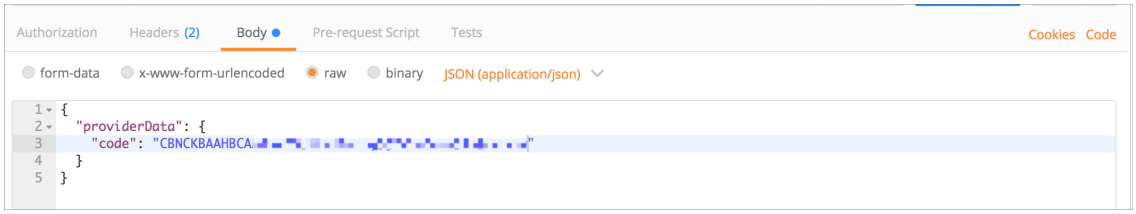


5. Generate an OAuth URL by making a call to `GET elements/{elementID}/oauth/url` .



6. Enter the resulting OAuth URL into a browser, authenticate with the API provider, and intercept the oauth provider code that is returned.
7. Make a call to `PATCH /instances` with the full Authorization header and the `providerData.code` value.





8. Confirm that the user is successfully re-authenticated, and that API calls can be made successfully against the SAP Cloud Platform Open Connectors instance.