

Tutorial: Securing an API by using OAuth 2.0

☰ Table of contents

Search in all products

Search in this product...



DataPower Gateway only

This tutorial shows you how to secure an API by using OAuth 2.0 so that an application can access the API on a user's behalf.

Before you begin

To complete this tutorial, you need an environment capable of sending HTTP requests and receiving HTTP responses. In these instructions, the **curl** command is used in a command line interface to demonstrate the OAuth flow without the need to write any application code. When you implement the OAuth flows for your application, make the same HTTP REST calls as used with **curl** in this tutorial.

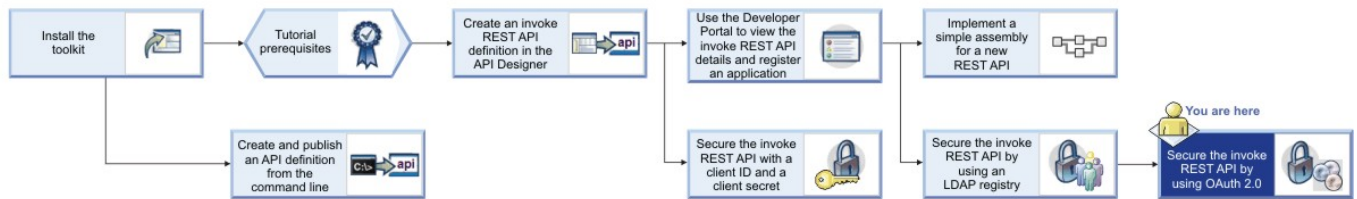
Note

The commands and example commands in this tutorial might need to be adapted for the syntax of your own command line interface.



The following diagram shows the sequential flow through the IBM® API Connect for IBM Cloud Developer toolkit tutorials for working with API definitions that call an existing endpoint. Before beginning a tutorial, ensure that you have completed the previous tutorials in the sequence. You can click a tutorial in the diagram to open the instructions for that tutorial.

sequence. You can click a tutorial in the diagram to open the instructions for that tutorial.



Note

You can secure your APIs with a third-party OAuth provider. For more information, see [Integrating third party OAuth provider](#).

About this tutorial

You will modify the security settings for the Branches API, which you created in the tutorial [Tutorial: Creating an invoke REST API definition](#), so that a calling application can make use of OAuth 2.0 to access the API on behalf of a user without requiring the user's password.

In this tutorial you will complete the following lessons:

- [Choosing your OAuth Scheme](#)
- [Creating an OAuth 2.0 provider API](#)
- [Configuring the API Security Scheme](#)
- [Acquiring Access Tokens for Individual Schemes](#)
- [Using the Access Token](#)

In OAuth 2.0, the following three parties are involved:

- The user, who possesses data that is accessed through the API and wants to allow the application to access it
- The application, which is to access the data through the API on the user's behalf
- The API, which controls and enables access to the user's data

Using OAuth 2.0, it is possible for the application to access the user's data without the disclosure of the user's credentials to the application.

The API will grant access only when it receives a valid access token from the application. How the application obtains an access token is dependent upon the OAuth scheme that is in use.

In this tutorial, you will be able to implement and test any of the following six OAuth schemes: implicit flow, application flow, confidential password flow, public password flow, confidential access code flow, and public access code flow. More information about these schemes is available in the following section.

Choosing your OAuth scheme

If you already know which OAuth scheme you intend to use, skip this section and proceed to [Creating an OAuth 2.0 provider API](#).

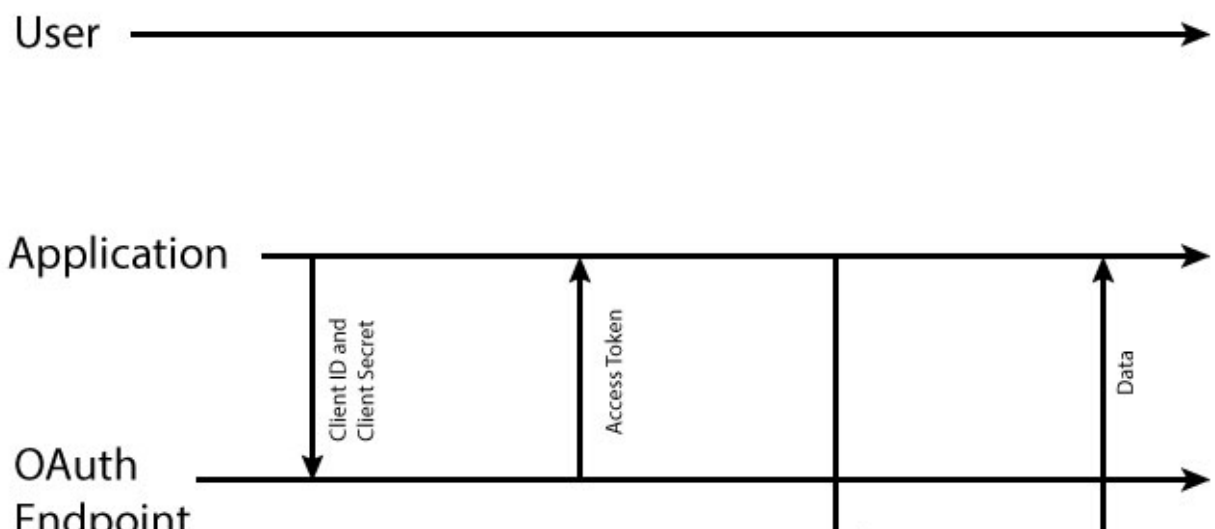
To choose an OAuth scheme, you must first establish whether your implementation is considered public or confidential. This will narrow your choices to three schemes. A brief outline of each scheme and the characteristics of the three public and three confidential schemes follows:

- Confidential

A confidential scheme is suitable when an application is capable of maintaining the secrecy of the client secret. This is usually the case when an application runs in a browser and accesses its own server when obtaining OAuth access tokens. As such, these schemes make use of the client secret.

- Application flow

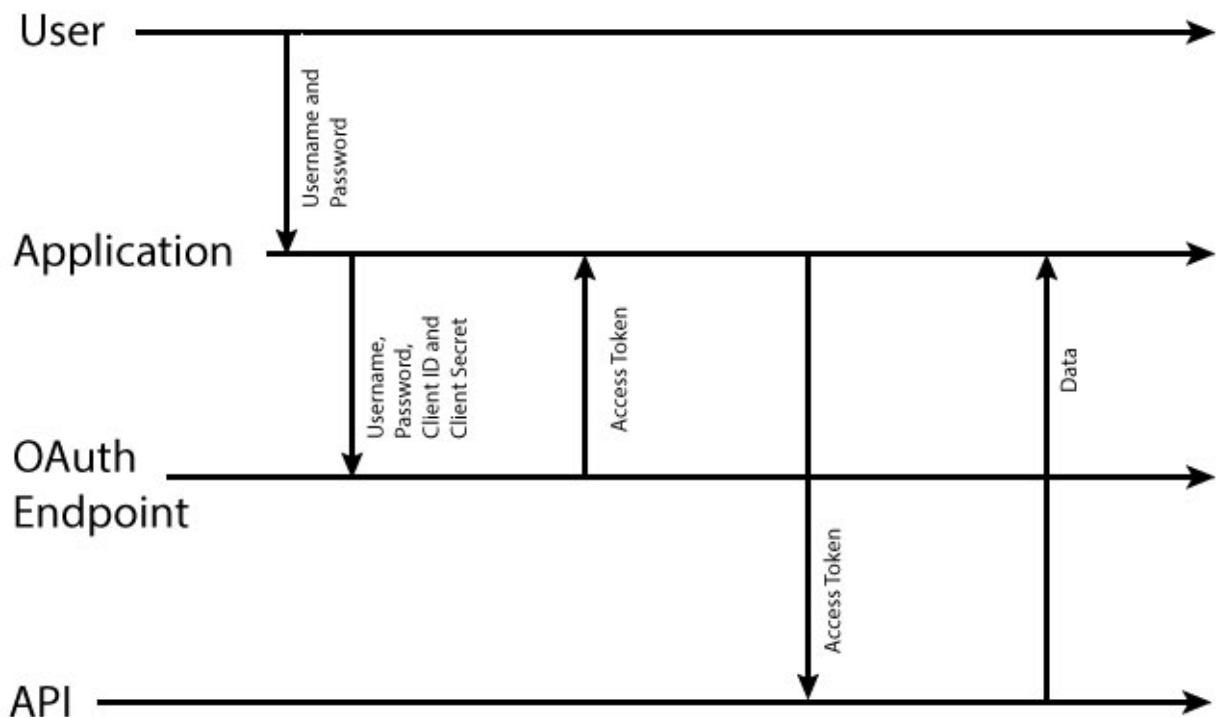
In the application flow scheme, the user is not required to provide authorization at any stage. Instead, the application uses its client secret to obtain an access token. In this case, it is critical that the client secret is kept safe.





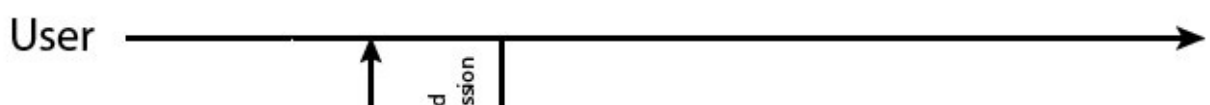
- Password flow

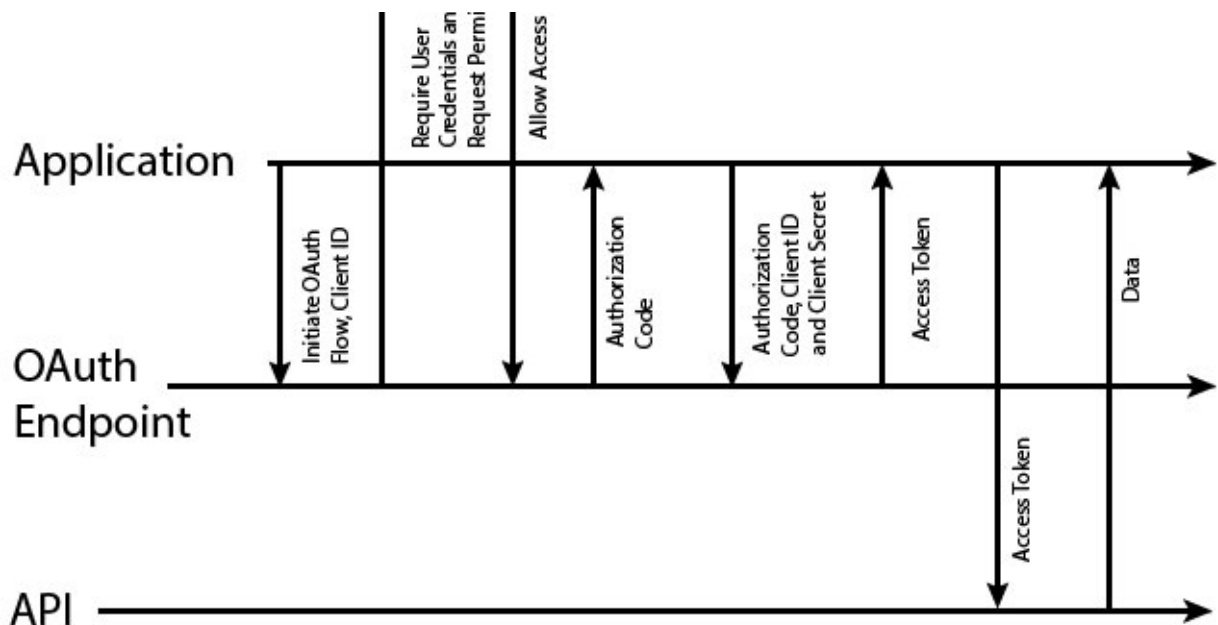
In the password flow scheme, the user provides the application with a user name and password that can be used to access the user's data. Following this, the client will directly contact the provider API to request an access token. In this case, trust must exist between user and application because the user's password is revealed to the application. However, this still has an advantage over the application using the password directly, because the validity of the access token or client ID can later be revoked without impacting other applications that do not need their access revoked. However, the application must be trusted to not store the user name and password.



- Access code flow

In the access code flow, the application has the user provide authorization through a form provided by the gateway server, which, if they grant authorization, provides an authorization code to the application. The application sends the authorization code to the provider API and is granted an access token in return.



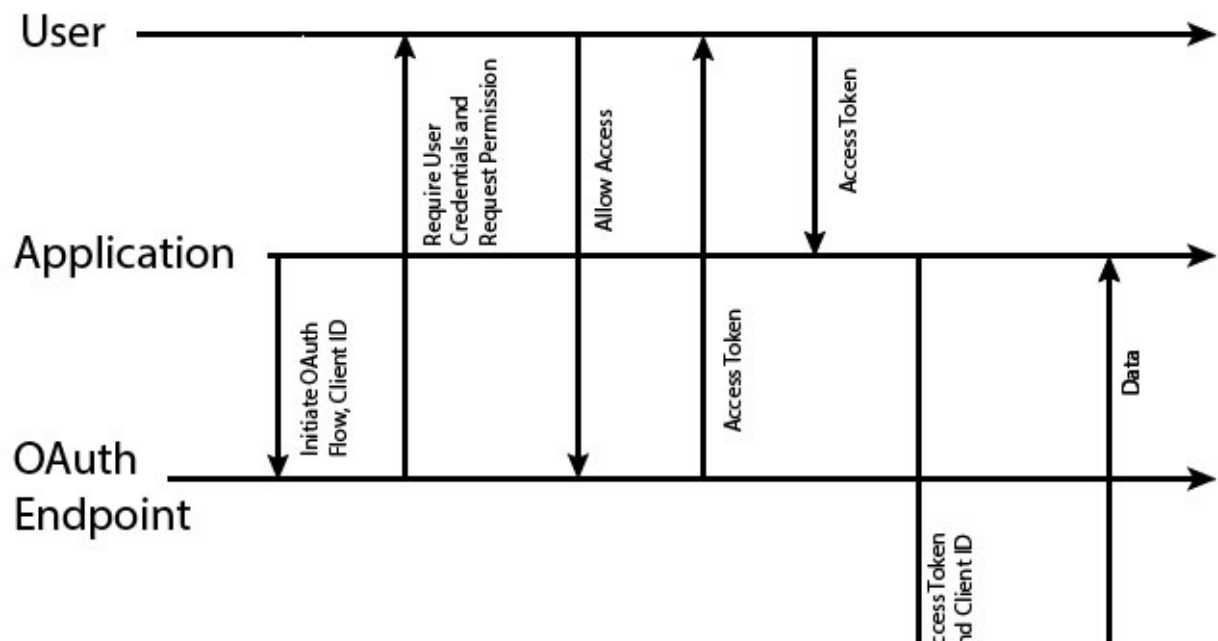


- Public

A public scheme is suitable when an application is incapable of maintaining the secrecy of the client secret. This is usually the case when the application is native on a computer or mobile where the secret would have to be stored on the user's device, likely inside the source code of the application. As such, these schemes do not make use of the client secret.

- Implicit flow

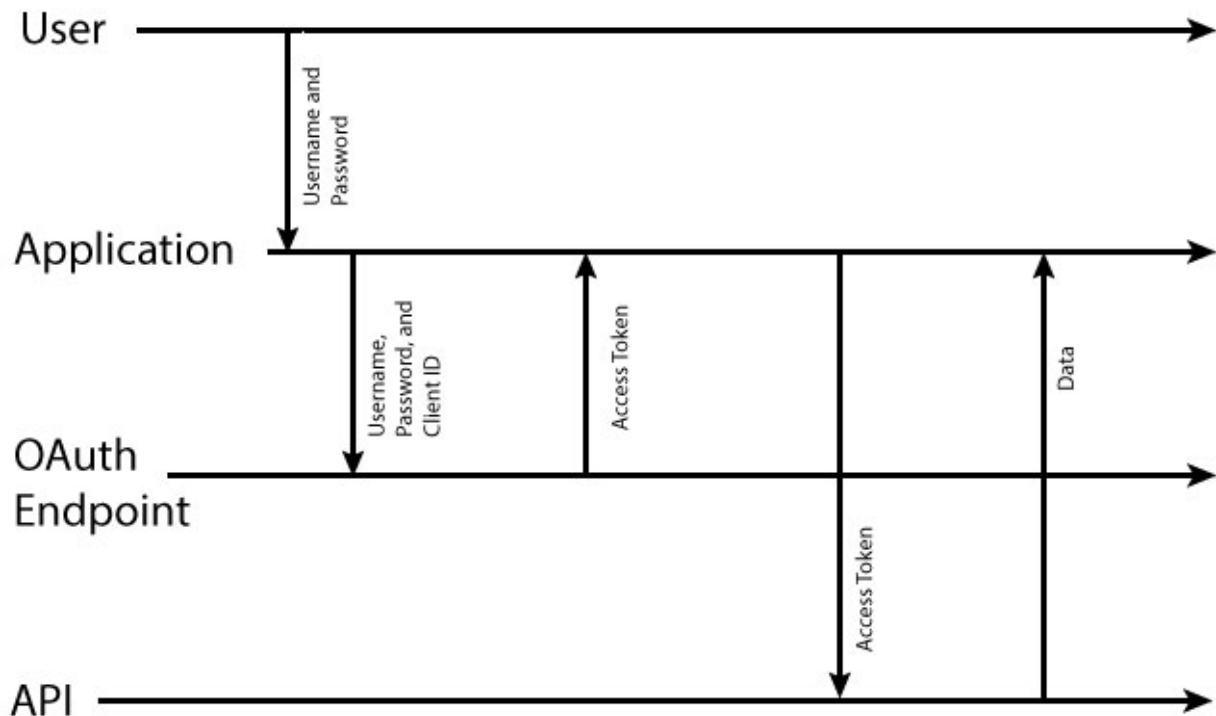
In the implicit flow scheme, the application requests an access token from the gateway server and the user grants permission, at which point an access token is provided to the user, who must then pass the token to the application





- Password flow

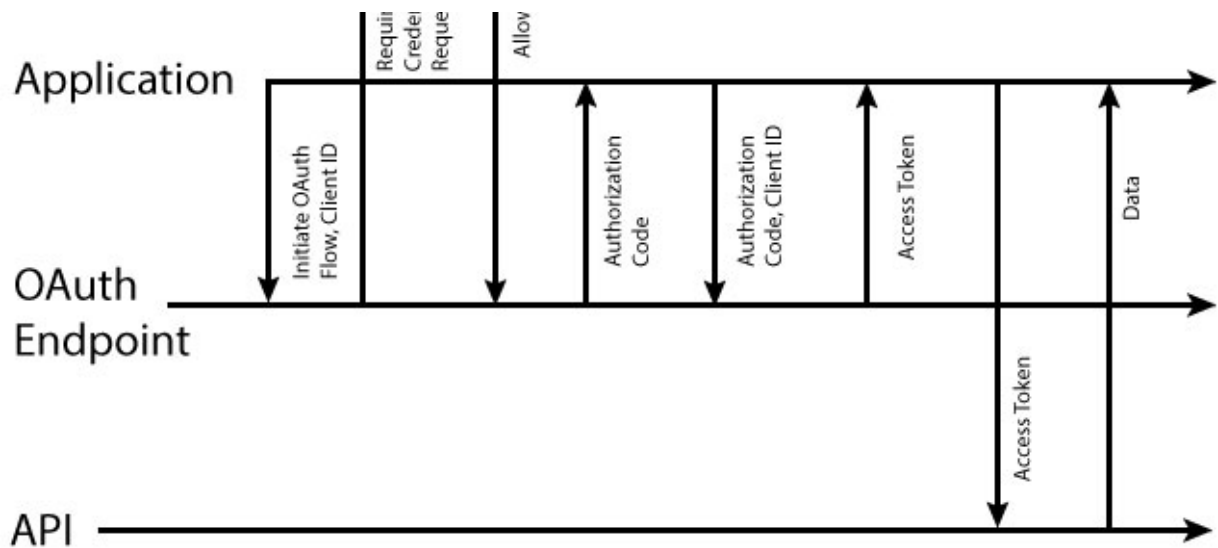
In the password flow scheme, the user provides the application with a user name and password that can be used to access the user's data. Following this, the client will directly contact the server to request an access token. In this case, trust must exist between user and application because the user's password is revealed to the application. However, this still has an advantage over the application using the password directly, because the validity of the access token or client ID can later be revoked without impacting other applications that do not need their access revoked. However, the application must be trusted to not store the user name and password.



- Access code flow

In the access code flow, the application has the user provide authorization through a form provided by the gateway server, which, if they grant authorization, provides an authorization code to the application. The application sends the authorization code to the provider API and is granted an access token in return.





Creating an OAuth 2.0 provider API

To create an OAuth 2.0 provider API, complete the following steps:

1. In a command window, change to the project folder that you created in the tutorial [Tutorial: Creating an invoke REST API definition](#).
2. Change directories to your LoopBack project and enter the following command:

```
apic edit
```



After a brief pause, the console displays this message:

```
Express server listening on http://127.0.0.1:9000
```



API Designer opens in your web browser, initially displaying the login page if you haven't logged in recently.

Note

The login page prompts you to **Sign in with IBM Cloud**. Enter your IBM Cloud credentials, which authenticates you on IBM Cloud and provides access to the API Manager features such as Publish, Explore, and Analytics. You will continue to work in API Designer locally to create APIs, models and data sources.

Note

If you need to run the editor on a different port, use the following command:

Linux

Mac OS X

```
PORT=port_number apic edit
```



Windows

```
set PORT=port_number && apic edit
```




where *port_number* is the port number to use.

3. In the API Designer, click the **APIs** tab.
4. Click **Add > OAuth 2.0 Provider API**.
5. Complete the fields according to the following table:

Table 1.

Field	Contents
Title	OAuth Endpoint API
Name	oauth-endpoint-api
Version	1.0.0
Base Path	/oauth-end

6. Click **Create API**.
7. In the **OAuth 2** section, configure the OAuth settings of your provider API.
 - a. Depending on your chosen scheme, select **Public** or **Confidential** in the **Client type** field.
 - b. Rename **Scope 1** to `view_branches` by using the text field.
 - c. In the **Description** field for **view_branches**, enter Allows access to branch details.
 - d. Rename **Scope 2** to `calculate_loans` by using the text field.
 - e. In the **Description** field for **calculate_loans**, enter Allows use of the loan calculator.
 - f. Delete **Scope 3** by clicking the **Remove scope** icon .
 - g. In the **Grants** section, clear the check box of any grant types you do not want to use.

Note

You must clear the check box of **Implicit** if you selected **Confidential** in step 7.a and you must clear the check box of **Application** if you selected **Public** in step 7.a.

- h. In the **Collect credentials using** field of **Identity extraction**, select **Basic**.
- i. In the **Authenticate application users using** field of **Authentication**, select **User registry** and, in the **User registry** field, enter `tutorialsldap` to reference the LDAP registry that you created in [Tutorial: Securing APIs by using an LDAP user registry](#).
- j. In the **Authorize application users using** field of **Authorization**, select **Default form**.
- k. Ensure that the **Enable refresh tokens** and **Enable revocation** switches of **Tokens** are in the **Off** position.

OAuth 2

Client type

Client type

Confidential

Scopes



Scope Name



view_branches

Description

Allows access to branch details

Scope Name



calculate_loans

Description

Allows use of the loan calculator

Grants

- ☐ Implicit
- ☒ Password
- ☒ Application
- ☒ Access Code

Identity extraction

Collect credentials using

Basic

Authentication

Authenticate application users using

User registry

User Registry

tutorialsldap

When using the Application grant type, authentication settings are not applicable and are ignored.

Authorization

Authorize application users using

Default form

Tokens

Access tokens

Time to live (seconds)

3600

- ☐ Enable refresh tokens
- ☐ Enable revocation
- ☐ Enable token introspection



Enabling this option inserts token introspect operation. It will also insert client ID and client secret header-based security definitions.

8. Click the **Save** icon  to save your changes.

Configuring the ADT security definition

Configuring the API Security Definition

To enable your chosen authentication scheme in API Designer, complete the following steps:

1. In the API Designer, click the **APIs** tab.
2. Click your **Branches** API definition.
3. In the **Security Definitions** section, click the **Add Security Definition** icon  and then click **OAuth**.
4. Scroll down to your newly created OAuth security definition.
5. In the **Name** field, rename your security definition as OAuth definition.
6. In the **Flow** field, select the type of flow you want to use.
7. In the **Scopes** section click the **Add scope** icon .
8. In the **Scope Name** field, rename the scope to view_branches.
9. If you are using a flow that uses an authorization URL, in the **Authorization URL** field, enter the following URL:

`https://Host_Address/Org_Segment/sb/oauth-end/oauth2/authorize`



where:

- *Host_Address* is the address of your gateway host.
- *Org_Segment* is the path segment of the organization that you want to use.

Note

If you do not know your host address or organization segment, you can find them by completing the following steps:

- a. Log in to the API Manager user interface.
- b. From the **Dashboard** page, click the **Sandbox** Catalog to display its details.
- c. Click the **Settings** tab, then click **Gateways**. You can obtain the host address and organization segment from the displayed **Base URL** value.

10. If you are using a flow that uses an authentication URL, in the **Token URL** field, enter the following URL:

`https://Host_Address/Org_Segment/sb/oauth-end/oauth2/token`



where:

- *Host_Address* is the address of your gateway host.
- *Org_Segment* is the path segment of the organization that you want to use.

Note




If you do not know your host address or organization segment, you can find them by completing the following steps:

- Log in to the API Manager user interface.
- From the **Dashboard** page, click the **Sandbox** Catalog to display its details.
- Click the **Settings** tab, then click **Gateways**. You can obtain the host address and organization segment from the displayed **Base URL** value.

- In the **Security** section, select the check box for your **OAuth definition** and clear the check box for your **LDAP (Basic)** security definition.

Note

You should have only your **OAuth definition** and **clientID** security definitions selected, together with the **view_branches** scope.

- Click the **Save** icon  to save your changes.
- Click **All APIs** and then click the **Products** tab.
- Click your **Banking Services** Product.
- In the **APIs** section, click the **Add API** icon .
- Select **Branches** and **OAuth Endpoint API** and then click **Apply**.
- Click your **Basic** Plan to expand it and ensure that **Branches** and **OAuth Endpoint API** are selected.
- Click the **Save** icon  to save your changes.
- Publish the **Banking Services** Product.
 - Click **Publish** and then click the **Sandbox** Catalog for the host address and organization that you want to use.
 - Select **Select specific products** and then select your **Banking Services** Product.

- b. Select **Select specific products** and then select your **Banking Services** Product.
- c. Click **Publish**.

Acquiring access tokens for individual schemes

In this section you will acquire an access token to access your API by using OAuth 2.0. The different schemes each use a different method to acquire an access token.

1. In your Developer Portal, sign in to your developer account.

Note

An administrator account cannot register applications and therefore cannot be used in this tutorial.

2. Click **Apps**
3. Click **Create new App**.
4. In the **Title** field, enter OAuth Application.
5. In the **OAuth Redirect URI** field, enter `https://example.com/redirect` and then click **Submit**.
6. Click **Show Client Secret** and click **Show** beside the **Client ID** field and then record your application's client secret and your application's client ID.

Note

The Client Secret can be viewed only once; if you have viewed it before and not recorded it you will need to reset it, which will invalidate uses of the previous secret elsewhere.

7. Click **API Products** and then click your **Banking Services** Product.
8. For your **Basic** plan, click **Subscribe**.
9. Select your **OAuth Application** and then click **Subscribe**.
10. Acquire the access token. The process for obtaining an access token differs for each

scheme. Click the link for your chosen scheme:

- Confidential
 - [Application flow](#)
 - [Password flow](#)
 - [Access code](#)
- Public
 - [Implicit flow](#)
 - [Password flow](#)
 - [Access code](#)

Using the access token

This section is common to all the OAuth schemes. You will access the Branches API and be authenticated.

1. From the API page of your Branches API in the Developer Portal, select the **GET /branches/details** operation.
2. Record the operation URL that is displayed.
3. Enter the following command into your command line interface (as one line):

```
curl -k -v -H "X-IBM-Client-Id: Client_ID" -H "Authorization: Bearer 'Operation_URL'
```



where:

- *Client_ID* is as recorded in step 6 of the [Acquiring access tokens for individual schemes](#) section of this tutorial.
- *Access-Token* is the access token that you recorded at the end of the scheme-specific section of this tutorial.
- *Operation_URL* is the URL you would use to call the operation if it were to be unsecured, as recorded in step 2.

For example:

```
curl -k -v -H "X-IBM-Client-Id: 6e3f115d-5220-48bd-a019-3c9680e657b7" -H "Authorization: Bearer AAEkNmUzZjExNWQtNTIyMC00OGJkLWEwMTktM2M5NjgwZTY1N2I3_3QEMOL240pXJTnuA4y0qvqM_7HgX7ImJ0Uy1_Jcq8Xx8CRHTBtPoBukuebfgDK3BtZjId7_Yeyjd01vqx8Xl00RdFxC0BF-grZLCG1BeaY" -X GET 'https://host.com/myorg/sb/branches/details'
```



The response includes the data from the operation.

```
[{"id": "0b3a8cf0-7e78-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "600 Anton Blvd.", "street2": "Floor 5", "city": "Costa Mesa", "state": "CA", "zip_code": "92626"}}, {"id": "79bf1c40-b2e9-11e5-9d3a-032ed6750760"}, {"id": "9d72ece0-7e7b-11e5-9038-55f9f9c08c06", "type": "atm", "address": {"street1": "4660 La Jolla Village Drive", "street2": "Suite 300", "city": "San Diego", "state": "CA", "zip_code": "92122"}}, {"id": "ae648760-7e77-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "New Orchard Road", "city": "Armonk", "state": "NY", "zip_code": "10504"}}, {"id": "c23397f0-7e76-11e5-8059-a1020f32cce5", "type": "branch", "phone": "512-286-5000", "address": {"street1": "11400 Burnet Rd.", "city": "Austin", "state": "TX", "zip_code": "78758-3415"}}, {"id": "ca841550-7e77-11e5-8059-a1020f32cce5", "type": "atm", "address": {"street1": "334 Route 9W", "city": "Palisades", "state": "NY", "zip_code": "10964"}}, {"id": "dc132eb0-7e7b-11e5-9038-55f9f9c08c06", "type": "branch", "phone": "978-899-3444", "address": {"street1": "550 King St.", "city": "Littleton", "state": "MA", "zip_code": "01460-1250"}}, {"id": "e1161670-7e76-11e5-8059-a1020f32cce5", "type": "branch", "phone": "561-893-7700", "address": {"street1": "5901 Broken Sound Pkwy. NW", "city": "Boca Raton", "state": "FL", "zip_code": "33487-2773"}}, {"id": "e9237f90-b2e9-11e5-9d3a-032ed6750760"}, {"id": "f9ca9ab0-7e7b-11e5-9038-55f9f9c08c06", "type": "atm", "address": {"street1": "1 Rogers Street", "city": "Cambridge", "state": "MA", "zip_code": "02142"}}, {"id": "string", "type": "string", "phone": "string", "address": {"street1": "string", "street2": "string", "city": "string", "state": "string", "zip_code": "string"}}]
```

4. To verify that access is granted only for the correct user ID and the correct access token, alter one or both in the previous command

Please note that DISQUS operates this forum. When you sign in to comment, IBM will provide your email, first name and last name to DISQUS. That information, along with your comments, will be governed by [DISQUS' privacy policy](#). By commenting, you are accepting the [DISQUS terms of service](#).

Sign In

0 Comments IBM Knowledge Center

Recommend

Share

Sort by Best ▾

Nothing in this discussion yet.

Subscribe Add Disqus to your siteAdd DisqusAdd

More topics

[Tutorial: Creating an invoke REST API definition](#)

[Tutorial: Implementing a simple assembly for a REST API](#)

[Tutorial: Securing an API with a client ID and client secret](#)

[Tutorial: Securing APIs by using an LDAP user registry](#)

[Tutorial: Securing an API by using OAuth 2.0](#)

[Tutorial: Creating and publishing an API definition from the command line](#)

 [Print this topic](#)

 [PDF download page \[Beta\]](#)

 [PDF download section \[Beta\]](#)

 [Help](#)

 [Select a product](#)

 [Open a ticket and download fixes at the IBM Support Portal](#)

[Find a technical tutorial in developerWorks](#)

[Find a best practice for integrating technologies in IBM Redbooks](#)

 [Explore, learn and succeed with training on the IBM Skills Gateway](#)

[Contact](#) [Privacy](#) [Terms of use](#) [Accessibility](#) [Feedback](#)

English

