Creating an OAuth provider API

An OAuth provider API contains the authorization and token endpoints of an OAuth flow.

Before you begin

To use your DataPower® Gateway to manage the revocation and introspection of access tokens, you must be running IBM® DataPower version 7.5.1.1 or later.

About this task

An OAuth provider API can serve multiple APIs that are employing OAuth security definitions. It provides operations that are the authorization and token endpoints of an OAuth flow.

In API Connect, scopes are defined in the provider API and listed as requirements by the secured API. All scopes that are listed by the security definition of the secured API must be granted by the access token. You have the opportunity to override the value of your scope during each phase of the OAuth process. For more information, see Scope for API Connect.

Each token grants access to a specific site for specific resources for a defined duration. By using an OAuth token, a user can grant a third-party site access to a range of information, which is stored with another service provider, without needing to share their personal credentials. For more information, see Tokens.

A scope cannot be restricted to a single user, and instead, you should configure your secured API to behave differently by referencing the `oauth.resource-owner` context. For more information, see API Connect context variables.

If the user changes their mind and decides that they do not want a third-party site to continue to have access to their information, the user can revoke the token access. If the token revocation URL is specified, the token revocation list is always checked before access is granted to the user information. For more information, see OAuth revocation URL.

Your OAuth provider API can support multiple OAuth flows, each of which corresponds to an OAuth grant type. Security definitions, which are applied to other APIs to use OAuth, use only one of the flows. The different types of OAuth flow, and the corresponding OAuth grant types, are shown in the following table:

Table 1. OAuth security definition types

OAuth flow	Corresponding OAuth grant type
Implicit	Implicit
Password	Resource Owner Password Credentials
Application	Client Credentials
Access Code	Authorization Code

You can secure your APIs with a third-party OAuth provider. For more information, see Integrating third party OAuth provider.

Note

The OAuth provider API logs failure cases in Analytics data, but does not log successful cases. Activity log policies that call for logging of Analytics data upon success do not apply for the OAuth provider API.

Procedure

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

Note

The API Manager and API Designer user interfaces both include the ability to create and edit APIs. However, the preferred method for these tasks is by using the API Designer user interface, as described in the following steps. Any tasks that are specific to a particular user interface are marked with an icon.

1. API Designer only Click **APIs**. The **APIs** tab opens.
2. API Manager only If you have not previously pinned the UI navigation pane then click the **Navigate to** icon  . The API Manager UI navigation pane opens. To pin the UI navigation pane, click the **Pin menu** icon  .
3. API Manager only Click **Drafts** in the UI navigation pane, and then click **APIs**. The **APIs** tab opens.
4. Click **Add** and then click **New OAuth 2.0 Provider API**.
5. Complete the fields that are presented.
   - The title can include special characters but should be kept short so that it can be easily displayed in the user interface.
   - The name should be kept short and can contain only lowercase alphanumeric characters (a-z and 0-9), or hyphen characters (-). A hyphen cannot be used as the first or last character in the name.
   - V5.0.4+ The base path is the URL segment of the API and does not include the host name or any additional segments for paths or operations. The base path cannot include special characters and must begin with a "/" character even if it is otherwise empty.
   - The version corresponds to the value of the `info.version` property of the API's OpenAPI (Swagger 2.0) definition. The `version.release.modification` version numbering scheme is recommended; for example `1.0.0`.
6. Specify whether your provider API is included in a Product, and create the API.
   - To create a new Product and include your provider API in that Product, complete the following steps:
     a. Click **Add a product**.
     b. API Designer only In the **Product template** field, select **Default** if you want to use the template defined as the default, to create the Product definition. This can either be the default `.hbs` template file that is provided with the developer toolkit, or another template file that you configure as the default by using configuration variables. You can also select a custom template that you created. For information about template files and configuration variables, see Creating and using API and Product definitions templates.
     c. Accept the default values for the Product title, name, and version, or change them.
     d. To publish the Product to a target Catalog, ensure that the **Publish this product to a catalog** check box is selected and then select the Catalog. You can clear this check box and stage or publish the Product

later by using the API Designer UI and API Manager UI, as described in Staging a Product and Publishing a Product.

e. If Spaces have been enabled, select the Catalog and Space that you require.

f. Click **Create API**.

- To create your provider API without adding it to a Product, click **Create API**.

The **Design** tab for the draft of your provider API definition opens. You can skip to different sections of your API definition by using the page navigation in the side bar. You can view the OpenAPI (Swagger 2.0) definition of your API in the **Source** tab and after you create an assembly, view your policy assembly in the **Assemble** tab.

7. In the **Design** tab, edit the **Info** section.

   a. Optional: Edit any or all of **Title**, **Name**, **Version**, and **Description**.

   b. Optional: In the **Contact** section, provide details for any or all of **Name**, **Email**, and **URL**.

   c. In the **Terms and License** section, provide details for any or all of **Terms of Service**, **License Name**, and **License URL**.

   d. In the **External Documentation** section, provide a **Description** for any external documentation you want to refer users to and a **URL** for where the documentation can be accessed.

8. In the **Schemes** section, select which transfer protocols you want your provider API to use.

> **Note**
>
> If your provider API is enforced by the API Connect gateway, only the HTTPS protocol is supported. See Step 9 for instructions of how to enable enforcement.

9. If your provider API is to be enforced by a gateway other than API Connect, use the **Host** field in the **Host** section to define the gateway URL that is to be used.

10. Configure the OAuth 2 section.

    a. Use the **Client type** drop-down menu to specify whether the OAuth provider API uses a **Public** or **Confidential** flow.

    b. Define scopes by providing a **Scope Name** and an optional **Description**.
    You can create new scopes by clicking the **Add scope** icon ⊕ and you can delete scopes by clicking the **Remove scope** icon 🗑. A scope that is defined becomes an option in the request for an access token from the provider API. In the security definition of a secured API, describe the scopes for which a token must be valid to grant access to the secured API. When an access token is requested from the provider API, multiple scopes must be separated by spaces.

    In the Advanced Scope Check section, use the **Enable Application Scope Check** slider to enable or disable the option to provide additional scope verification, resulting in what the scope application is allowed to have. This process happens after API Connect successfully verifies the application credential, and before API Connect attempts to authenticate the resource user. For more information, see OAuth Scope.

    In the Advanced Scope Check section, use the **Enable Owner Scope Check** slider to enable or disable the option to provide additional scope verification, resulting in what scope the authenticated user is allowed to have. This process happens after user has been authenticated successfully during the OAuth protocol exchange. For more information, see OAuth Scope.

c. In the **Grants** section, select which OAuth 2.0 flows you want to use.

> **Note**
>
> If you are using a **Public** flow, you cannot have an **Application** grant type; and if you are using a **Confidential** flow, you cannot have an **Implicit** grant type. You must have at least one grant type to use the provider API.

d. In the **Identity extraction** section, use the **Collect credentials using** drop-down menu to specify how a user's credentials are collected during the authorization process.

- Select **Default form** where authentication is assumed.
- Select **Basic** to use basic authentication.
- Select **Custom form** to use a custom HTML form that you provide to API Connect. If you select this option, provide a URL at which your custom form can be found in the **Custom form** field. For more information, see Creating a custom sign-in form.
- Select **Redirect** to use an externally hosted service for authentication. If you select this option, provide the URL at which users start your authentication process in the **Redirect URL** field. For more information, see Authenticating and authorizing through a redirect URL.

e. In the **Authentication** section, use the **Authenticate application users using** drop-down menu to select how a user is authenticated.

- To authenticate users with an LDAP registry, select **User registry**, then complete the following action according to which user interface you are using:
  - API Manager only Select the registry from the **User registry** list.
  - API Designer only Enter the name of the registry.

> **Note**
>
> The LDAP user registry must be created on the Management server by using the API Manager user interface. For more information, see Creating an LDAP registry.

- Select **Authentication URL** to authenticate users by sending credentials that are collected by API Connect to this URL. When establishing authentication, API Connect makes a GET call to your authentication URL. When the call is made, it includes in its authorization header the user name and password it has collected from the user. Confirm that these are correct and respond with an HTTP success code such as `200 OK` if you want to allow the application access, or with an HTTP error code such as `401 Unauthorized` if you want to deny access.

> **Note**
>
> If you are authenticating and authorizing users through a redirect URL, you must supply an authentication URL. For more information, see Authentication URL.

To apply a TLS profile for communication with the authentication URL, complete one of the following actions according to the user interface you are using:

- API Manager only Select the TLS profile from the **TLS Profile** list.
- API Designer only Enter the name of the TLS profile.

> **Note**
>
> The TLS profile must be created on the Management server by using the API Manager user interface. For more information, see TLS profiles.

To change the value of the resource owner to differ from the value sent to the authentication URL, the authentication URL should return a header that is named `API-Authenticated-Credential`, with its value set to the new resource owner. For example:

```
--header "API-Authenticated-Credential: Resource_Owner"
```

where *Resource_Owner* is the value to which you want to set the `oauth.resource-owner` context.

f. In the **Authorization** section, use the **Authorize application users using** drop-down menu to select how authorization should be granted.

- Select **Default form** to use the default form that is provided by API Connect.
- Select **Custom form** to use your own custom HTML form. If you select this option, provide a URL at which your form can be found in the **Custom form** field. You can also select a TLS profile from the drop-down menu to use for communications with this URL. For more information, see Creating a custom authorization form.
- Select **Authenticated** to automatically grant authorization.

> **Note**
>
> If you are authenticating and authorizing users through a redirect URL, you must automatically grant authorization.

g. In the **Tokens** section, use the **Time to live (seconds)** field to specify for how many seconds an access token remains valid, for a minimum of 1 and a maximum of `63244800`.

h. In the **Tokens** section, use the **Enable refresh tokens** slider to enable or disable the use of refresh tokens by applications. If you enable refresh tokens, the following fields are available for you to use:

Count

Use the **Count** field to specify how many times a refresh token can be requested. You can request a refresh token up to `4096` times per *permission set*. (A permission set is an instance of application, owner, and permission)

Time to live (seconds)

Use the **Time to live** field to specify how many seconds a refresh token remains valid. The time range available is 2 to `252979200` seconds.

i. In the **Tokens** section, use the **Time to live (seconds)** field under Maximum Consent, to specify for how many

seconds the combination of any number of access and refresh token remain valid. The time range available is `0` to `2529792000` seconds. Note, this feature is only present if refresh tokens are enabled. Setting the value to `0` disables this feature.

j. In the **Tokens** section, use the **Enable revocation** slider to enable or disable the use of a list of blocked applications.

   - If you want to use your DataPower Gateway to manage revocation of access tokens, select **Use DataPower Gateway**. Use the **Enable users to view and revoke permissions** switch to specify whether more operations are made available to applications that can use the provider API so that they can view and revoke access tokens.
     The list of revoked applications is shared between all provider APIs. If you do not enable the additional operations, then to revoke an application you need to use a second provider API that has them enabled.

     In addition to revoking all tokens for an application, you can revoke a single token; use the **Allow application to revoke its token** switch to enable this capability. To revoke a single token for an application, send one or other of the following requests to the DataPower Gateway (the **`curl`** command is used by way of illustration):

```
curl -X POST \
  https://Datapower_Gateway_Hostname/oauth2/revoke \
  -H 'authorization: Basic base64_encoded_application_credential' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d "token_type_hint=access_token&token=Access_Token"
```

```
curl -X POST \
  https://Datapower_Gateway_Hostname/oauth2/revoke \
  -H 'authorization: Basic base64_encoded_application_credential' \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d "token_type_hint=refresh_token&token=Refresh_Token"
```

     Replace *Datapower_Gateway_Hostname*, *base64_encoded_application_credential*, *Access_Token*, and *Refresh_Token* with the appropriate values.

     For further information, see Managing tokens with the DataPower Gateway.

Restriction

The **Enable revocation** option is not supported if you are using API Connect with IBM Cloud public.

Important

If you have a cluster of DataPower Gateway servers, the OAuth data synchronization behavior across the servers depends on whether or not you enable revocation:

- If you enable revocation, API Connect uses the DataPower quota enforcement server, and OAuth data is synchronized across the servers. If an access token is obtained from one server, the OAuth data synchronization ensures that the same authorization code cannot be used to obtain an access code from another server. You must ensure that the DataPower quota enforcement server is configured.
- If you disable revocation, API Connect does **not** use the DataPower quota enforcement server and OAuth data does not synchronize across the cluster of DataPower Gateway servers.

Therefore, to prevent the same authorization code being used to obtain an access code from more than one server you must configure DataPower to synchronize OAuth data across the servers, by using the DataPower Gateway console.

To configure DataPower to synchronize OAuth data, complete the following steps:

i. Ensure that the DataPower quota enforcement server is configured. For more information, see Configuring the quota enforcement server.

- If you want to use a revocation URL, select **Revocation URL** and provide the URL in the **Revocation URL** field. You can also select a TLS profile from the drop-down menu to use for communications with this URL. For more information, see OAuth revocation URL.

k. Use the **Enable token introspection** slider to enable or disable token introspection. Enabling introspection creates a new operation that can be called, which returns all information about an access token that is passed to it, such as its scope and validity. For more information, see Integrating third party OAuth provider

l. Specify the metadata URL where request headers are sent to retrieve extra content for the OAuth transaction. For more information, see OAuth metadata URL and authentication URL.

11. Optional: In the **Consumes** section, select which types of media your provider API accepts when calls are made to it. Add other supported media types in addition to JSON and XML, by using the **Add Media Type** field.

Important

The provider API must accept `application/x-www-form-urlencoded` content and changes to this field do not affect the behavior of the OAuth flow, only the documentation available through the Developer Portal.

12. Optional: In the **Produces** section, select which types of media your provider API returns when calls are made to it. Add other supported media types in addition to JSON and XML, by using the **Add Media Type** field.

Important

The provider API always produces `application/json` content and changes made to this field do not affect the behavior of the OAuth flow, only the documentation is available through the Developer Portal.

13. Optional: Configure the **Lifecycle** section.

a. Optional: For **Phase**, use the drop-down menu to change the phase of the lifecycle that your provider API is in. The options are as follows.

Identified

The API is in the early conceptual phase and is neither fully designed nor implemented.

Specified

The API has been fully designed and passed an internal milestone but has not yet been implemented.

Realized

The API is in the implementation phase.

b. Optional: Set the **Testable** toggle to the **On** position to allow the provider APIs operations to be tested using the test tool in the Developer Portal.

> **Note**
>
> For the test tool to work, your provider API must be included in a Plan in a Product that is staged in a development Catalog.

c. Optional: Set the **Enforced** toggle to the **On** position to enforce your provider API by using the API Connect Gateway.

d. Optional: Set the **CORS** toggle to the **On** position to enable CORS access control.

14. Optional: If you want to perform transformations or other actions when the provider API is called, create an assembly by clicking **Create assembly** in the **Policy Assembly** section. For more information, see The assemble view.
15. Optional: In the **Security Definitions** section, manage any security definitions that might be used by the API or its operations. For more information, see Configuring API security
16. Optional: In the **Security** section, select any security definitions that you want to apply to your provider API. To be available in the **Security** section, definitions must have been defined in the **Security Definitions** section.
17. API Manager only In the **Extensions** section, add any vendor extensions you want to use with your API.
18. Optional: In the **Properties** section, define any API properties that you want to use. For more information, see API properties.
19. Optional: Add paths to your API. For more information, see Defining Paths for a REST API.

> **Important**
>
> Deleting the existing Paths in your provider API will interrupt the correct functioning of the OAuth flow.

20. Optional: In the **Parameters** section, add parameters that are shared by all Paths and operations in the API.

    a. Click the **Add Parameter** icon ⊕.

    b. In the **Name** field, provide a name for your parameter.

    c. In the **Located In** field, select where the parameter is found in the call of your operation.

    d. Optional: In the **Description** field, provide a description of your parameter.

    e. Use the **Required** check box to specify whether the parameter is required for a call to be valid.

    f. Optional: From the drop-down list for **Type**, select the type of data that the parameter is expected to have when the call is received by API Connect. If you have set **Located In** to **Body** then you can select a JSON schema that you have defined for your API. For more information about creating JSON schemas, see step 21.

21. Optional: In the **Definitions** section, you can create JSON schema definitions. You reference these definitions in an

operation to provide developers with information about the JSON request they should make or the JSON response they should expect to receive from the operation. Your schema definitions are made available to developers through the Developer Portal but are not enforced in any calls to the API unless a Validate (validate) policy is used.

> **Important**
>
> Deleting the existing definitions in your provider API interrupts the correct functioning of the OAuth flow.

a. Click the **Add Definition** icon ⊕. A new definition is created.

b. Click the newly created definition to expand its details.

c. Complete the **Name** field for your definition.

d. From the **Type** drop-down list, select the type of your definition.

e. Optional: In the **Description** field for your definition, provide a description of what is defined by the definition.

f. Complete the details of your definition's properties. Each property requires a name and type and can also have a description.

g. Optional: To specify that a property is required when the operation is called, select it using the check box in the **Required** * column.

h. Optional: You can add additional properties by clicking **Add Property**.

i. Optional: You can delete properties or definitions by clicking the **Delete** icon 🗑 next to the property or definition.

j. If you want to allow the inclusion of properties that are not included in the definition, so that validation will not fail when a validate-rest policy is used on the request, set **Allow additional properties** to the **On** position.

> **Note**
>
> - You can include more complex schema definitions in your API by using the **Source** tab and editing your API's OpenAPI (Swagger 2.0) definition directly. For more information, see the OpenAPI (Swagger 2.0) specification.

22. Optional: To add any tags, in the **Tags** section click the **Add Tag** icon ⊕. Tags added in this way appear in the OpenAPI (Swagger 2.0) definition of the provider API but are not used by API Connect for any indexing.
23. Click the **Save** icon 💾 to save your changes.

Results

You created and configured an OAuth 2.0 provider API.

→ **Integrating third party OAuth provider**

V5.0.7+ OAuth token validation can be offloaded to the third-party Open Authorization (OAuth)/Open ID Connect (OIDC) provider by using the Introspection URL. Clients can use a third-party OAuth or OIDC provider to obtain a token that is protected by IBM API Connect. API Connect can use this feature along with the

mentioned provider to protect access to the API.

→ **OAuth custom forms and protection against XSS attacks**

DataPower Gateway only When the OAuth security definition uses Implicit Flow, Password Flow, or Authorization flow, you can present HTML forms to users during the extract identity and authorization stages. Because HTML forms can include external and inline sources, , such as images or JavaScript, these sources can be the origin of cross-site scripting (XSS) attacks.

Parent topic:

→ Protecting an API with OAuth

Related reference:

→ API and Product definition template examples

Last updated: Tuesday, 4 September 2018