Aras OAuth Registry 32

Administration Guide

Document #: D-009160

Last Modified: *8/22/2024*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Phone: 978-806-9400

Notice of Rights

Notice of Liability

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
TechDocs@aras.com
Subject: Aras Product Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Technical Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support/

1 Introduction

1.1 Purpose

The purpose of this guide is to enable Aras Innovator administrators to configure clients for the OAuth Server without modifying code tree configuration files.

1.2 Scope

This document provides instructions and information on configuring clients and managing user roles for the Aras Innovator OAuth Server using the OAuth Registry platform component.

1.3 Target Audience

This document is intended for Aras Innovator administrators and IT support.

1.4 Permissions

Aras Innovator administrators can create and edit OAuth Clients by default. For more information on permissions, please see the Configuring OAuth Permissions section.

1.5 Feature Terms and Definitions

The following Feature terms and definitions are used in this document:

Term	Definition
Access Token	A string that an OAuth Client uses to make requests to the resource server (from OAuth 2)
API	Application Programming Interface
Authorization Code Grant	The client requests an authorization code from the Aras Innovator OAuth server. The client can then exchange the authorization code for an access token.
CRUD	Set of capabilities to manipulate data: create, read, update, delete
CWS	Configurable Web Service
Grant Type	Describes the method an application uses to retrieve an access token from the OAuth Server
Identity Token	Represents the outcome of an authentication process. It contains an identifier for the user (see OpenID)
Impersonate Grant	The client uses a certificate to send a request as any user in the system. This is typically used for integrations and server-to-server use cases without end-user interaction.
Implicit Grant	The client redirects the user to the Aras Innovator login page. After logging in on the Aras Innovator login page, the user is redirected to the client application.
OAR	OAuth Registry platform component
OAuth 2	Authentication Protocol. For more information, see: https://oauth.net/2/

Term	Definition
OAuth Client	An application that requests access or identity tokens from the OAuth Server (from OAuth 2)
OpenID	Identity Management Protocol. For more information, see: https://openid.net/
Password Grant	The client exchanges a user's credentials for an access token. This grant type is typically not recommended because the client application must collect the user's password and send it to the authorization server.
Platform Component	A Platform Component is a separately installable platform service that provides forward and backward compatibility with platform releases.
UI	User Interface
URI	Uniform Resource Identifier. A string that identifies a logical or physical resource. A URL (Uniform Resource Locator) is one type of URI.

2 Introduction to OAuth Registry

2.1 OAuth Registry Overview

The OAuth Registry platform component enables administrators to configure OAuth Clients without leaving the Aras Innovator UI. This eliminates the need for administrators to access the code tree and manually modify configuration files.

2.2 Purpose

The Aras OAuth Registry application can define OAuth Clients for authenticating requests from applications and integrations.

2.3 Process Overview

The process of configuring and managing the OAuth Registry is outlined below:

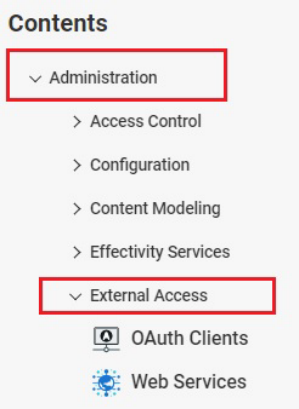
- Create an OAuth Client
- Request an OAuth access token
- Request with an OAuth access token

3 OAuth Clients

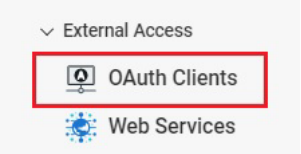
3.1 Creating an OAuth Client

The following steps outline the process of creating an **OAuth Client**:

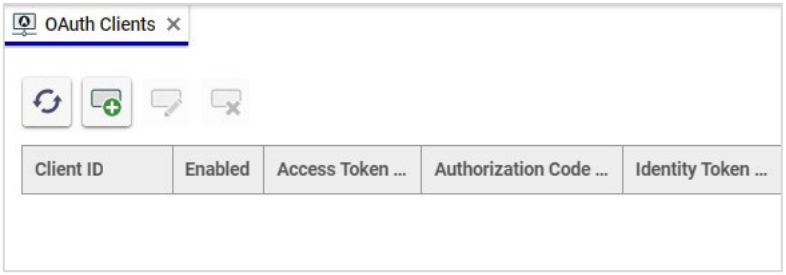
1. From the **Table of Contents**, expand **Administration** and **External Access**.



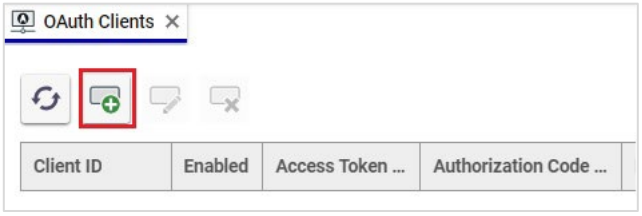
2. Select **OAuth Clients**.



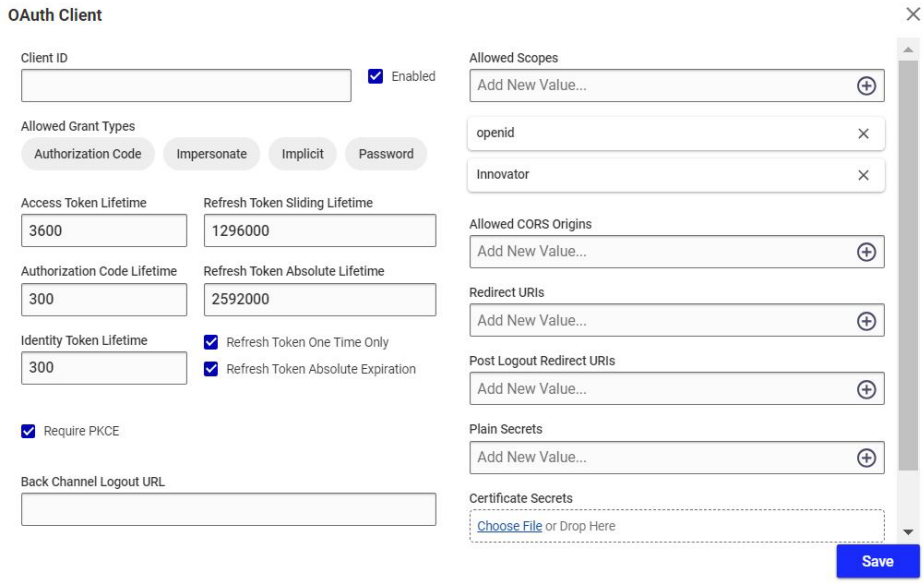
The **OAuth Clients** grid appears:



3. Click **Create New Client**.



The **OAuth Client** form appears:



See the table below for the description of the parameters to create an **OAuth Client**:

Parameters	Definitions	Details
Client ID	A unique identifier for a client application The **Client ID** can contain only Latin alphabet letters, digits, and symbols such as '-' and '_.' **Note**: The following are reserved values used by the platform and cannot be modified through the Aras OAuth Registry UI: • OAuthServer • InnovatorServer • InnovatorClientServer • VaultServer • ConversionServer • AgentService • InnovatorClient	Required
Enabled	Indicates whether the client can authenticate users.	Default: Enabled
Allowed Grant Types	Identifies the grant type(s) for this OAuth Client The following grant types are available: **Authorization Code**: The client requests an authorization code from the Aras Innovator OAuth server. The client can then exchange the authorization code for an access token.	Required

Parameters	Definitions	Details
	Impersonate: The client uses a certificate to send a request as any user in the system. This is typically used for integrations and server-to-server use cases without end-user interaction. **Implicit**: The client redirects the user to the Aras Innovator login page. After logging in, the user is redirected to the client application. **Password**: The client exchanges a user's credentials for an access token. This grant type is typically not recommended because the client application must collect the user's password and send it to the authorization server. **Note:** An OAuth Client may not combine the Authorization Code and Implicit grant types.	
Access Token Lifetime	Lifetime of an access token for this client in seconds The value must be greater than zero.	Required Default: 3600
Refresh Token Sliding Lifetime	Sliding the lifetime of a refresh token for this client in seconds To enable this feature, the value must be greater than or equal to zero. Zero indicates that the sliding lifetime is disabled.	Default: 1296000
Authorization Code Lifetime	The lifetime of an authorization code for this client in seconds	Default: 300
Refresh Token Absolute Lifetime	Maximum lifetime of a refresh token for this client in seconds To enable this feature, the value must be greater than or equal to zero. Zero indicates the refresh token will have an unlimited lifetime.	Default: 2592000
Identity Token Lifetime	Specifies the Lifetime of the identity token for this client in seconds The value must be greater than zero.	Default: 300
Refresh Token One Time Only	Indicates whether the refresh token handle will remain the same when refreshing tokens. **True**: The refresh token handle will remain the same when tokens are refreshed. **False**: The refresh token handle will be updated when tokens are refreshed.	Default: True
Refresh Token Absolute Expiration	Indicates whether the refresh token expiration is absolute. **True**: The refresh token will expire on a fixed point in time specified by the “Refresh Token Absolute Lifetime.” **False**: When refreshing the token, the lifetime of the refresh token will be renewed by the amount specified in “Refresh Token Sliding Lifetime.” The lifetime will not exceed the “Refresh Token Absolute Lifetime” value.	Default: True
Require PKCE	Indicates whether a proof key is required for authorization code-based token requests.	Default: True

Parameters	Definitions	Details
Back Channel Logout URI	Specifies the client's logout URI for HTTP back-channel-based logout.	Optional
Front Channel Logout URI	Specifies the client's logout URI for HTTP front-channel-based logout.	Optional
Allowed Scopes	Specifies the API scopes that the client is allowed to request. If empty, the client cannot access any scope. The most common scopes are **openid**, **offline_access**, and **Innovator**.	
Allowed CORS Origins	Specifies a list of domains this client may use to send requests. This is necessary to enable CORS origins for JavaScript clients.	Required
Redirect URIs	Specifies the URIs that may receive authorization codes or tokens for this client.	Required when the grant type is Authorization Code or Implicit
Post Logout Redirect URIs	Specifies which URIs to redirect to after logout. Empty or null values are not valid.	Required
Plain Secrets	A Plain Secret is a string shared between the client and the server.	Optional
Certificate Secrets	A Certificate Secret is a Base64 encoded certificate containing the client's public key.	Optional

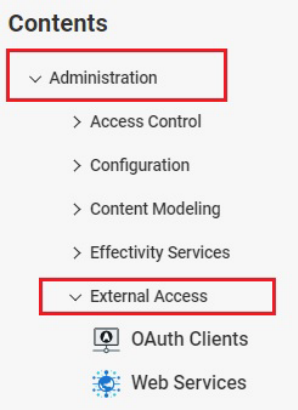
4. Fill out the form, as applicable.
5. Click **Save**.

3.2 Editing an OAuth Client

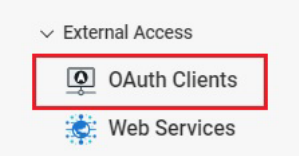
The Aras Innovator OAuth server includes several preconfigured clients to support the platform. These cannot be viewed, edited, or deleted using the below UI.

The following steps outline the process of editing an OAuth Client.

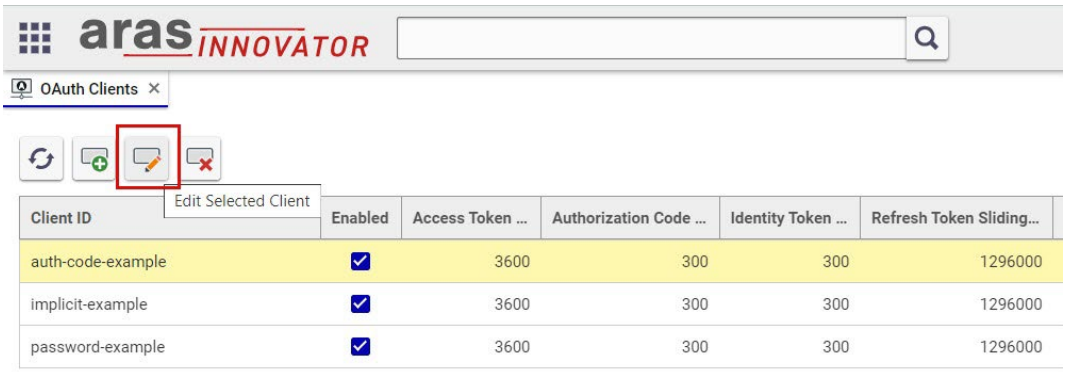
1. From the **Table of Contents**, expand **Administration** and **External Access**.



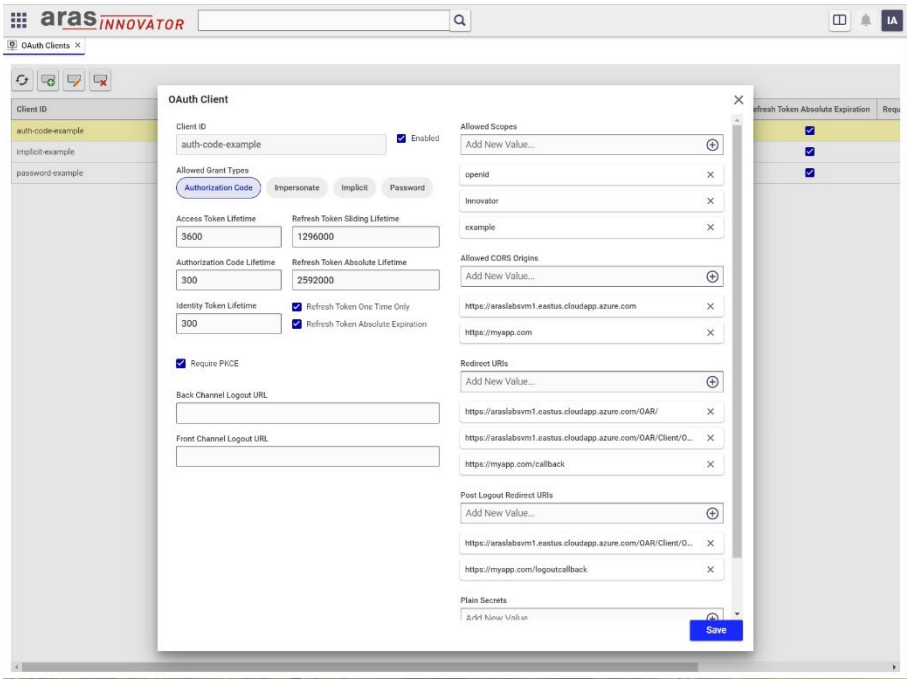
2. Select **OAuth Clients**.



3. Select an **OAuth Client** in the grid and click **Edit**.



4. Edit any properties as needed in the **OAuth Client** form. The section **Creating an OAuth Client** provides detailed descriptions of each property.



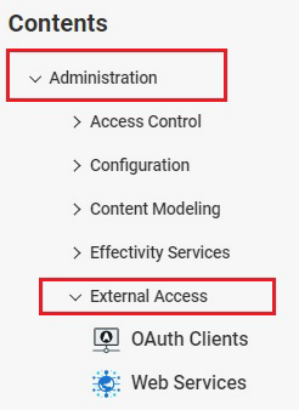
5. Click **Save**.

3.3 Deleting an OAuth Client

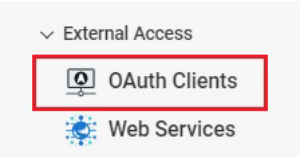
The following steps outline the process of editing an OAuth Client.

Warning Deleting an OAuth Client will prevent applications or integrations from using that client to access data in Aras Innovator.

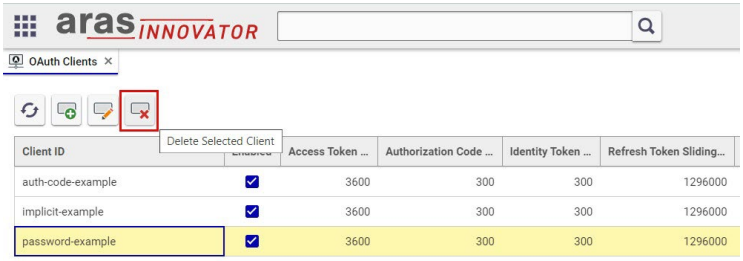
1. From the Table of Contents, expand **Administration** and **External Access**.



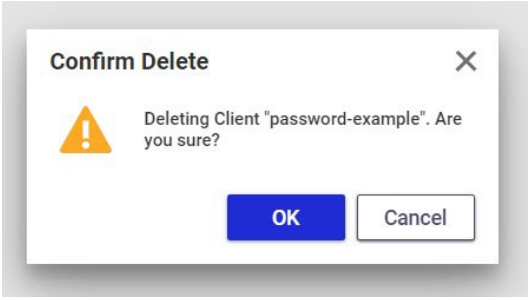
2. Select **OAuth Clients**.



3. Select an OAuth Client in the grid and click **Delete**.



4. Click **OK** in the Confirm Delete dialog to delete the client.



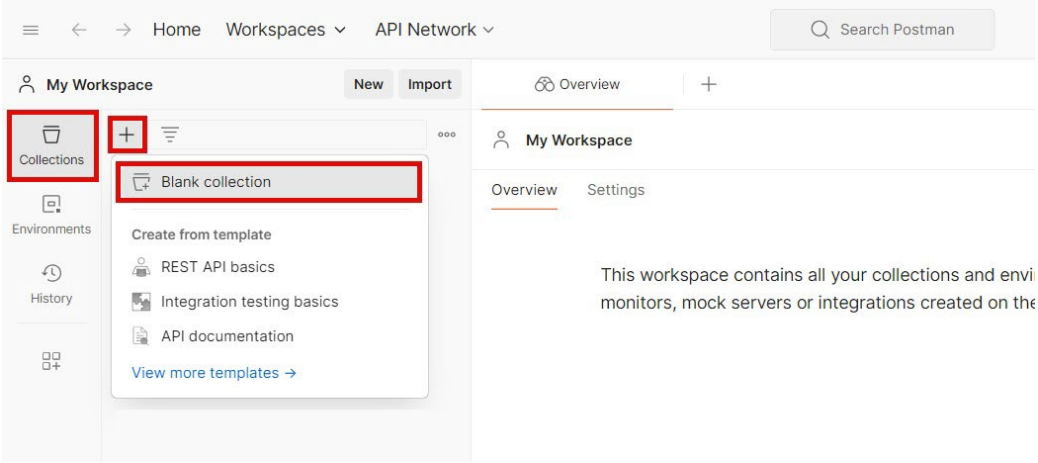
3.4 Using an OAuth Client

This section provides examples demonstrating how to authorize REST requests with each type of client. Each example uses the Postman HTTP client, though the exact requests can be sent with any API platform or custom code.

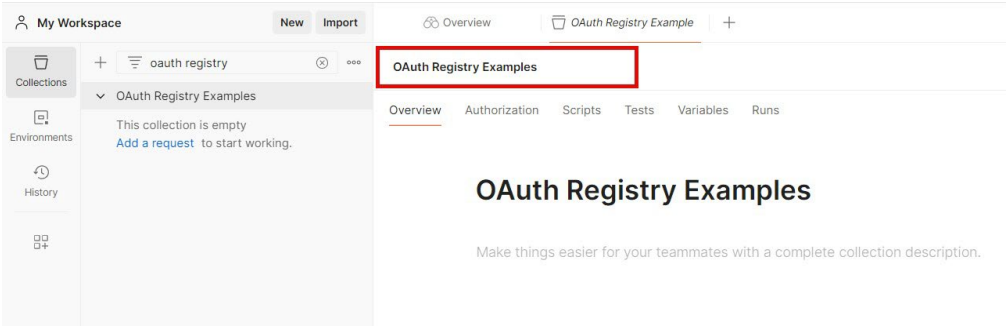
3.4.1 Setup

The steps below describe the setup assumed by the examples in the following sections.

1. Download and install Postman.
2. Under **My Workspace**, select **Collections**, click **Create New Collection**, and then **Blank Collection**.



3. Provide a name for the new Collection — for example, **OAuth Registry Examples**.



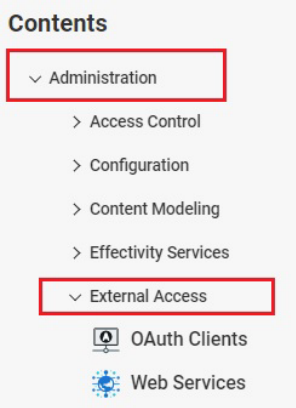
The examples in the following sections will add new requests to this collection.

3.4.2 User Login Flow Example

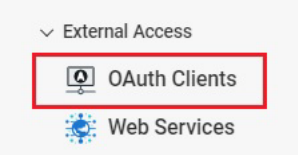
This example demonstrates using an OAuth Client with the Implicit grant type.

3.4.2.1 Configuring the OAuth Client

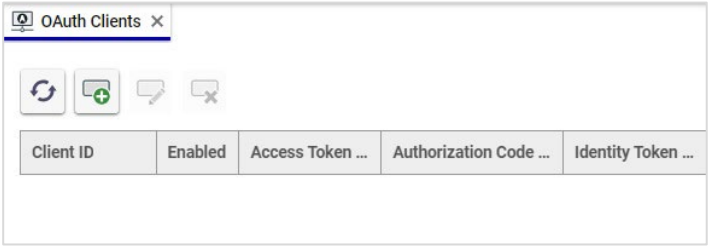
1. Log into Aras Innovator as administrator.
2. From the Table of Contents, expand **Administration** and **External Access**.



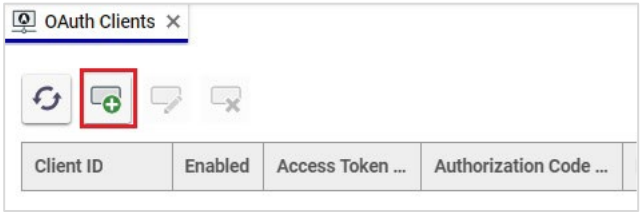
3. Select **OAuth Clients**.



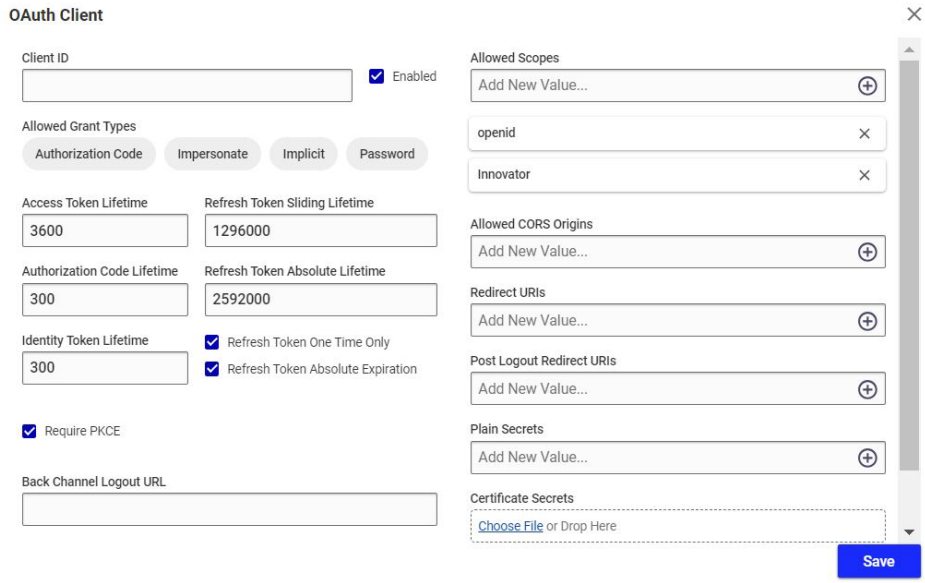
The **OAuth Clients** grid appears:



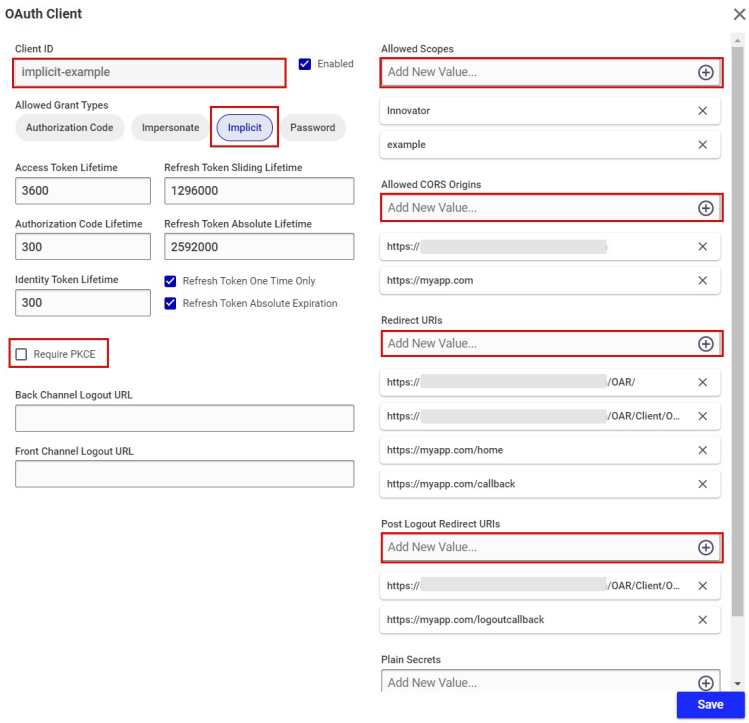
4. Click **Create New Client**.



The **OAuth Client** form appears:



5. Fill out the form with the values below, substituting the environment's details for the highlighted placeholders.



- **Client ID**: implicit-example
- **Allowed Grant Types**: Implicit
- **Require PKCE**: false
- **Allowed Scopes**: (Optional) Innovator, example
- **Allowed CORS Origins**:
  - https://server/webalias
  - Any other valid domain that should be allowed to send requests with this client id

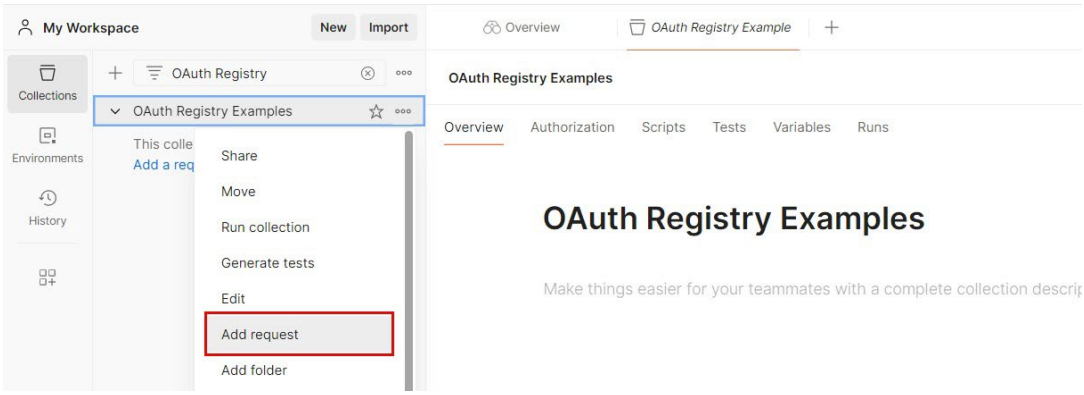
Note: CORS domains should not include a trailing slash at the end.

- Redirect URIs:
  - https://server/webalias/
  - https://server/webalias/Client/OAuth/PopupCallback
  - Any other valid URI to redirect to after the user logs in, such as the URL of a custom application
- Post Logout Redirect URIs:
  - https://server/webalias/Client/OAuth/PostLogoutCallback
  - Any other valid URI to redirect to after the user logs out, such as the URL of a custom application

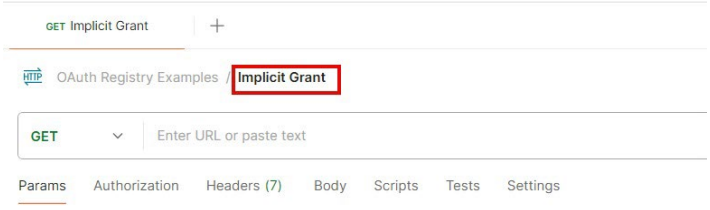
6. Click **Save**.

3.4.2.2 Requesting an Access Token

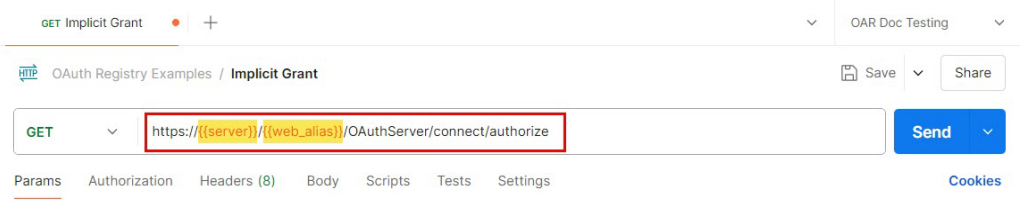
1. In Postman, right-click the **OAuth Registry Examples** collection and select **Add Request**.



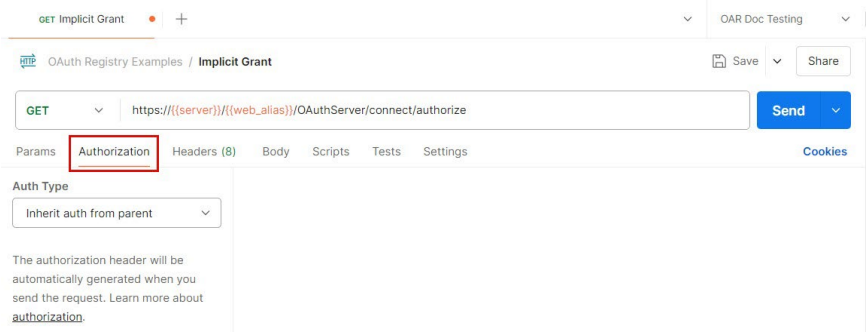
2. Give the request a name like **Implicit Grant**.



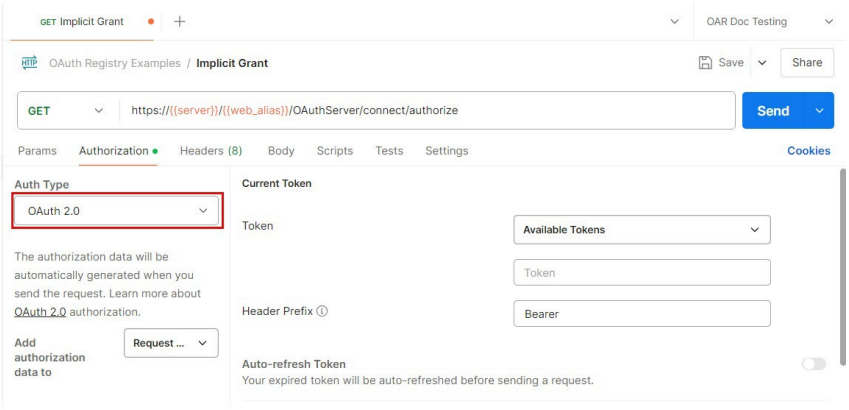
3. Enter the **URL** of the OAuth Server's `authorize` endpoint as shown below.



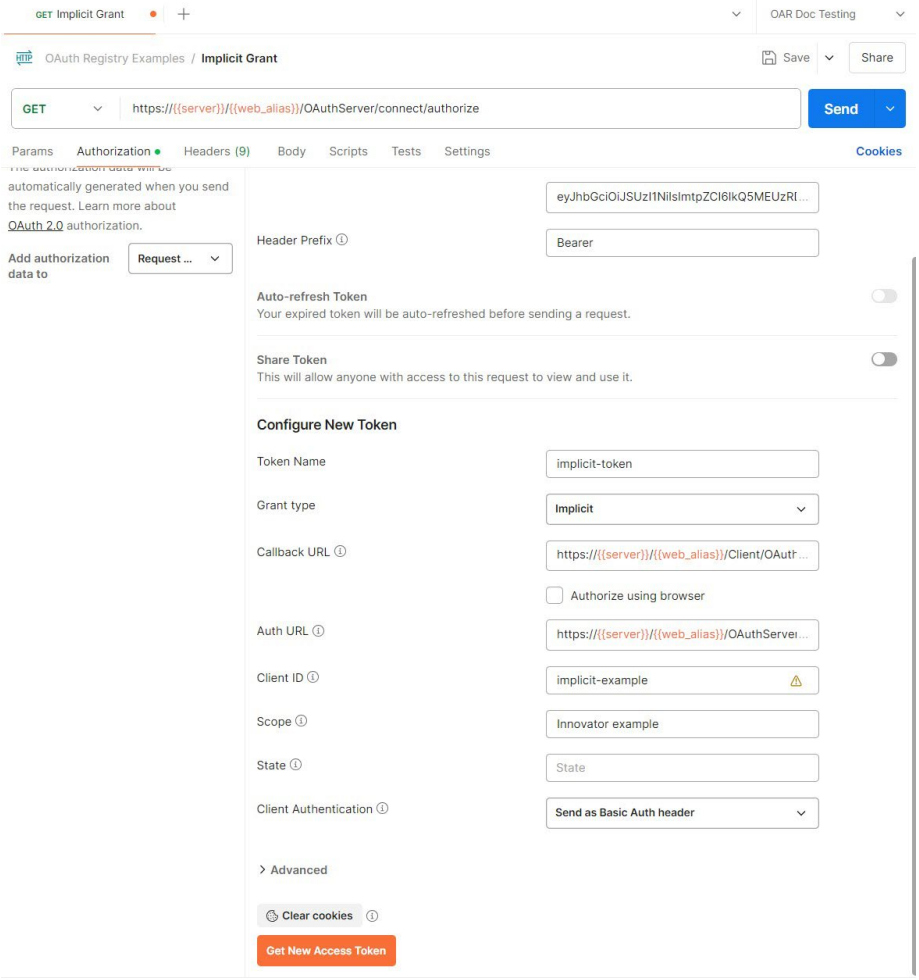
4. Click the **Authorization** tab below the URL bar.



5. Select **OAuth 2.0** from the Auth Type field.



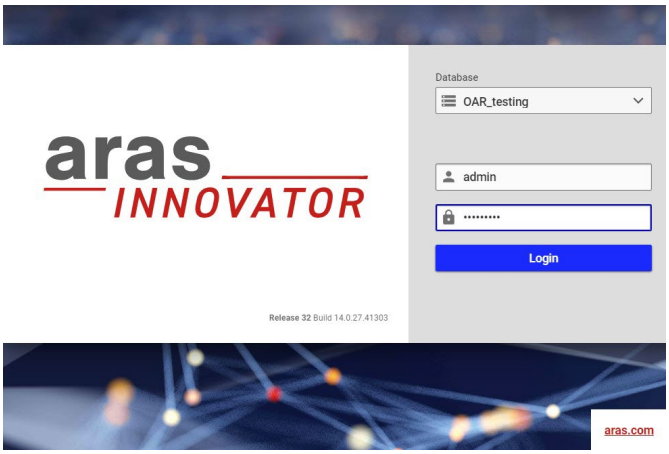
6. Scroll down to **Configure New Token** section and set the configuration options as follows:



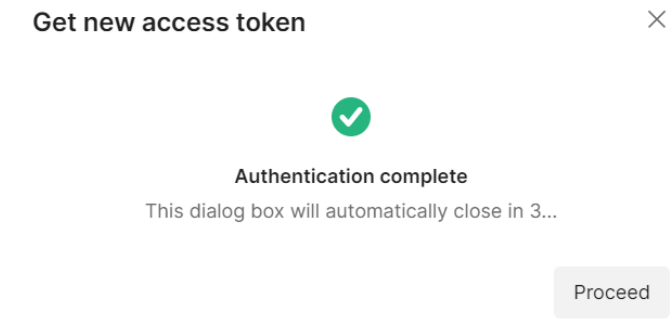
Field	Value
Token Name	Name that Postman will use for the token
Grant Type	Implicit
Callback URL	Enter a valid Redirect URI for the client. Example: https://<servername>/<web alias>/Client/OAuth/PopupCallback
Auth URL	https://<servername>/<web alias>/oauthserver/connect/authorize
Access Token URL	https://<servername>/<web alias>/oauthserver/connect/token
Client ID	implicit-example
Scope	Innovator example

7. Click Get New Access Token.

8. **Log in** to the popup Aras Innovator window with your credentials.

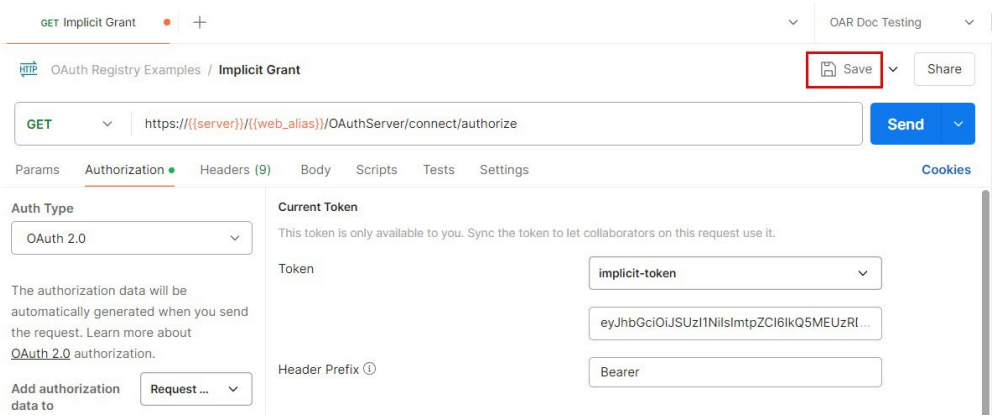


9. In the **Get new access token** dialog box, click **Proceed**.



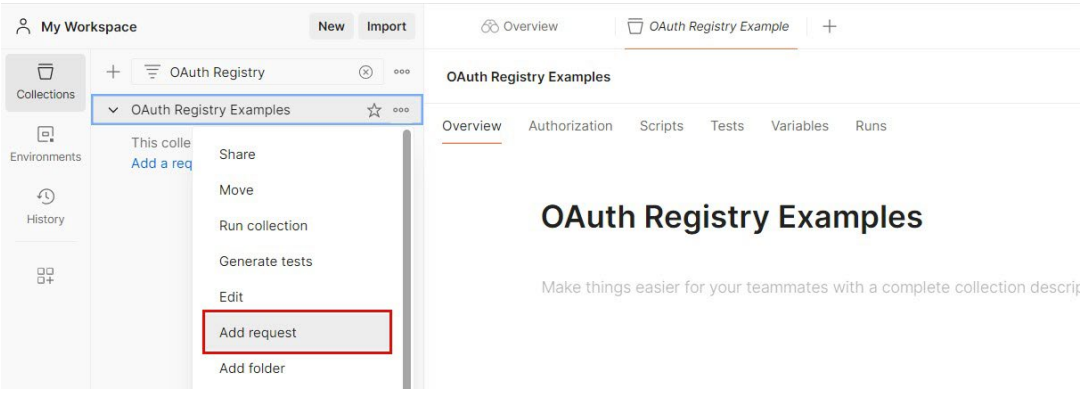
10. In the Manage Access Token dialog box, click Use Token.

11. Click **Save** to save the changes.

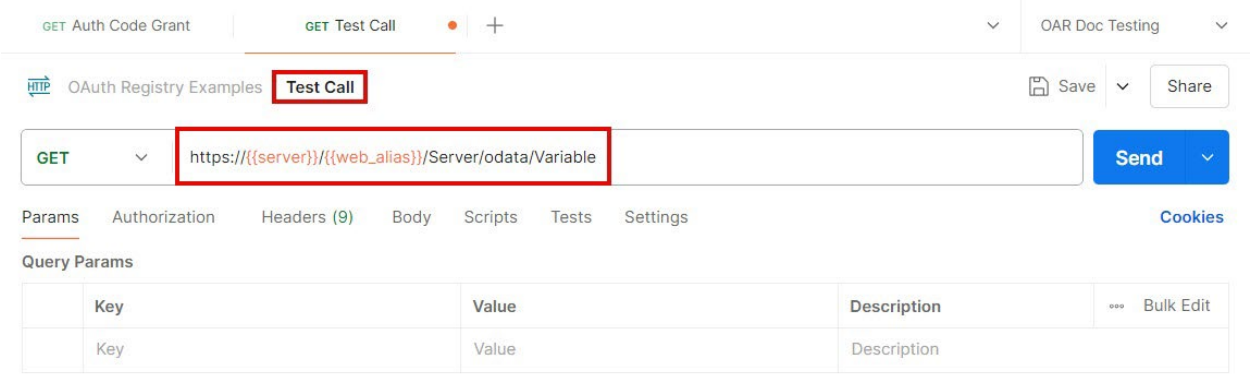


3.4.2.3 *Using the Access Token*

1. In Postman, right-click the **OAuth Registry Examples** collection and select **Add Request**.

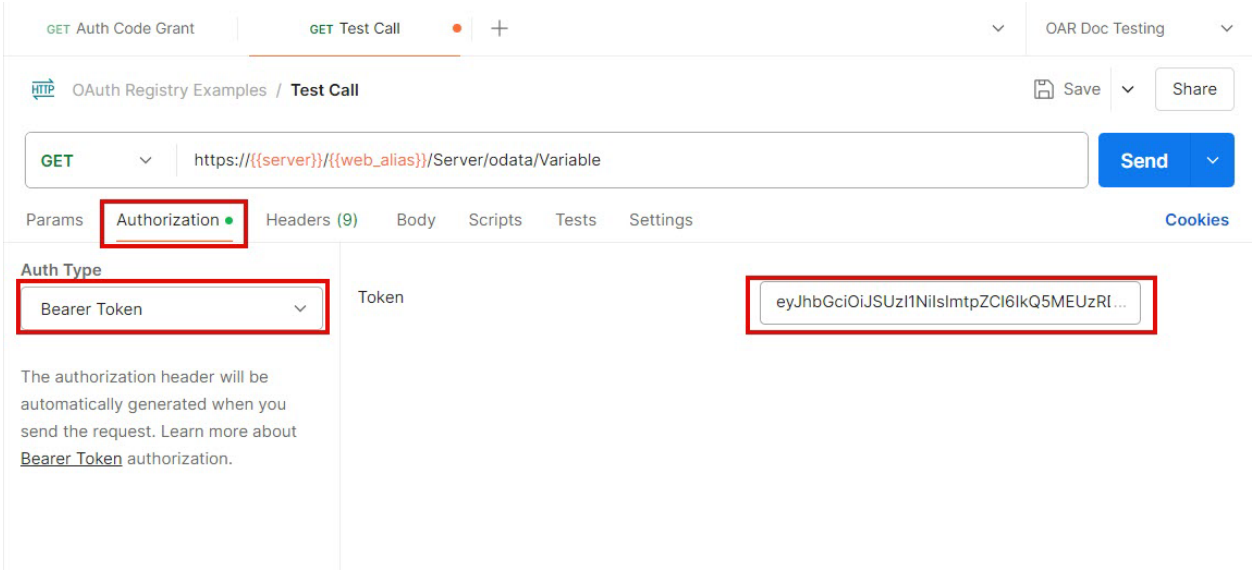


2. Name the request **Test Call** and enter the URL shown below.

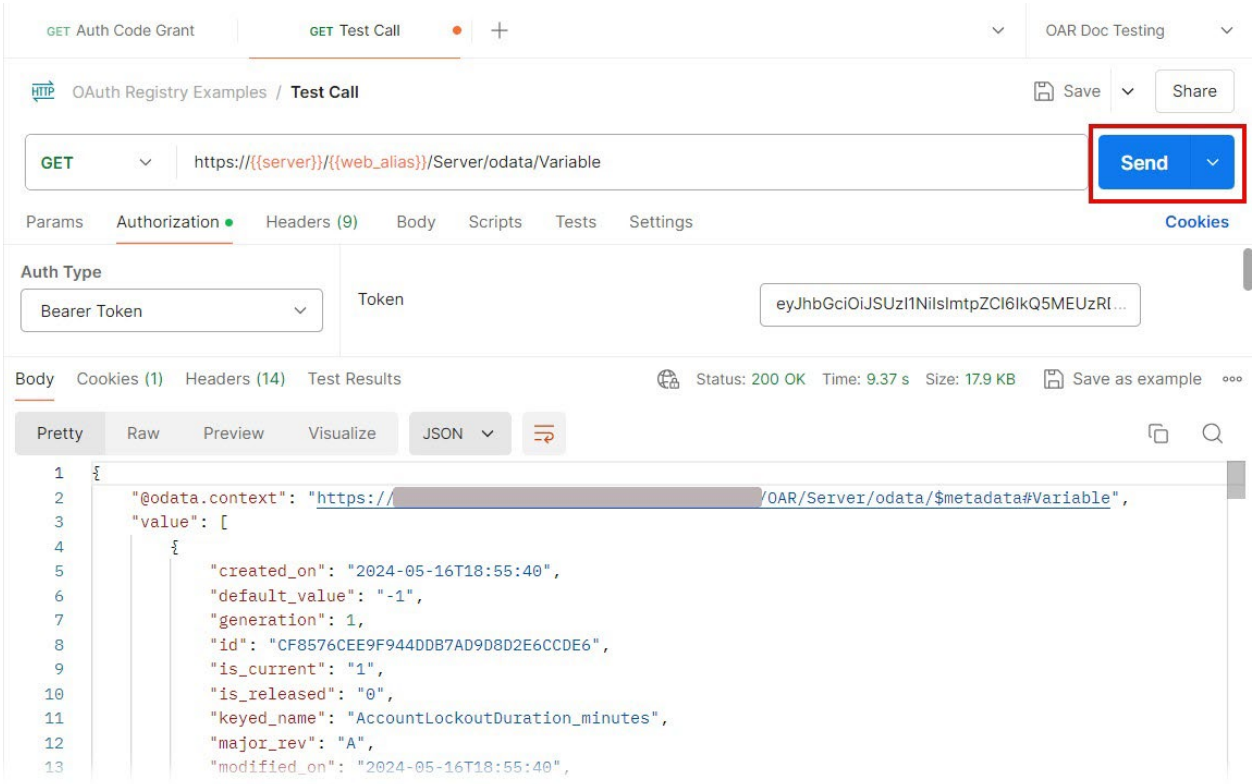


This request will retrieve the Variable items from the Aras Innovator database, confirming the access token is valid.

3. Select the **Authorization** tab, choose the **Bearer Token** Auth Type, and enter the token requested in the last section.

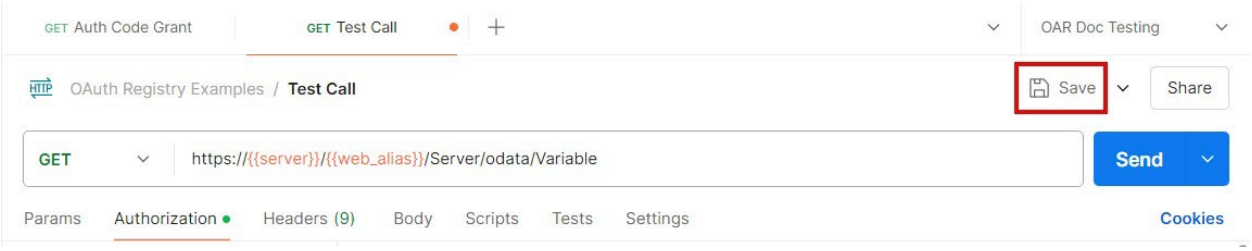


4. Click **Send** to submit the request to the server.



The server should return a 200 OK status code, and the response body should include the Variable items from the Aras Innovator database.

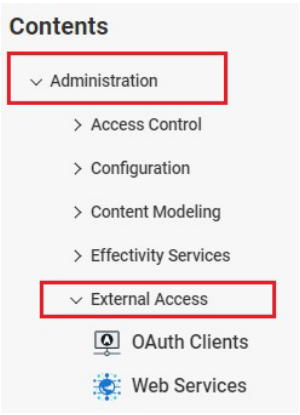
5. Click **Save**.



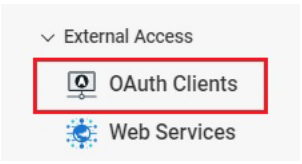
3.4.3 Authorization Code Example

3.4.3.1 Configuring the OAuth Client

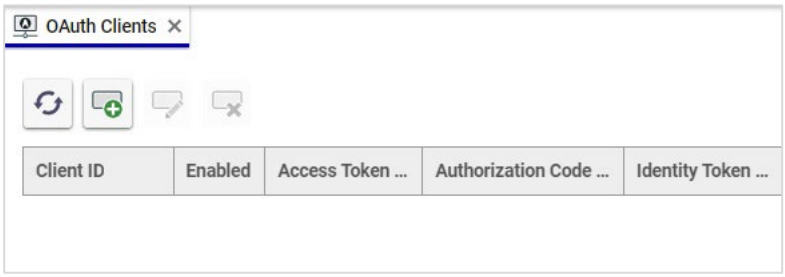
1. Login to Aras Innovator as admin.
2. From the Table of Contents, expand **Administration** and **External Access**.



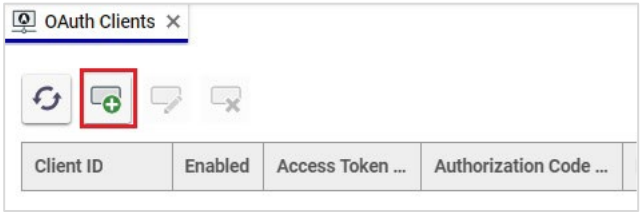
3. Select OAuth Clients.



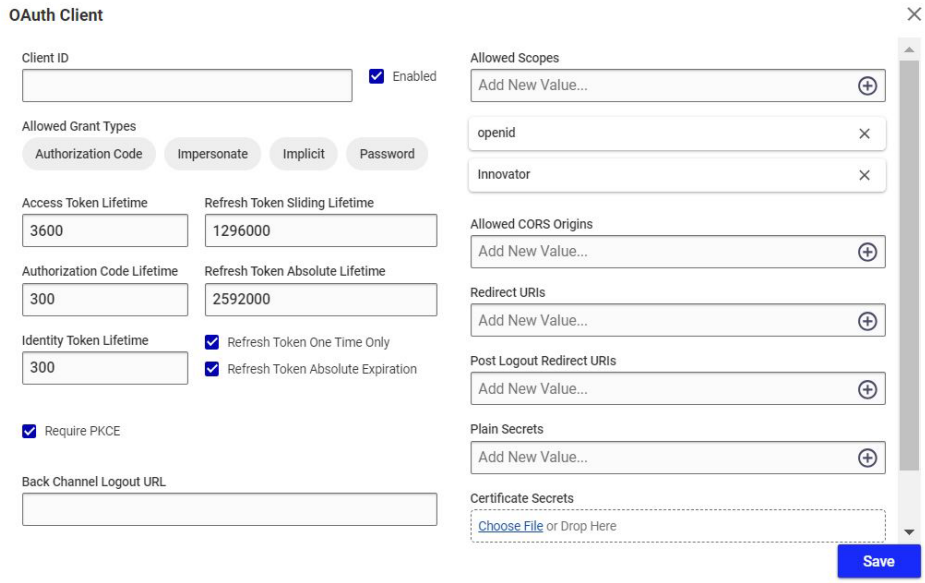
The OAuth Clients grid appears:



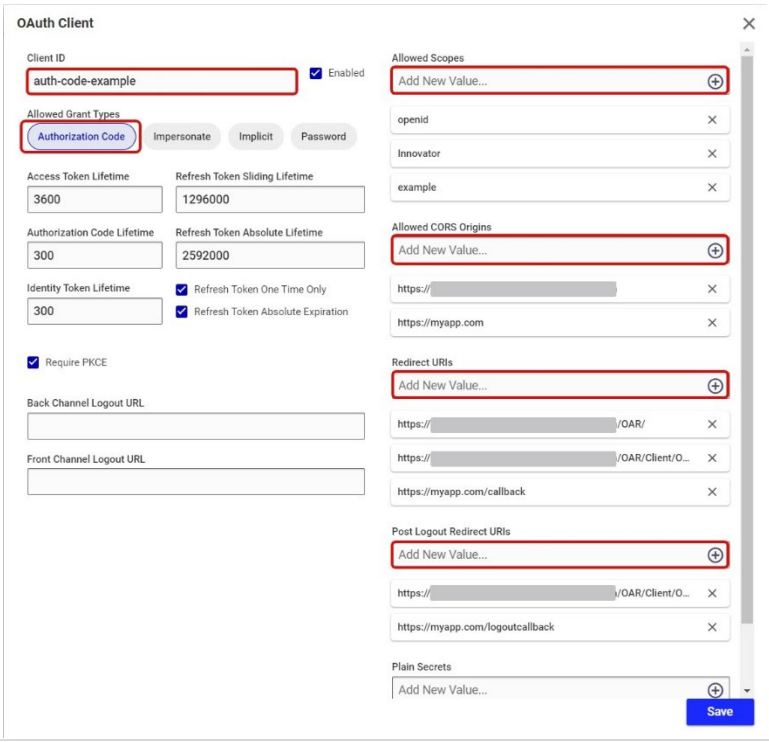
4. Click Create New Client.



The **OAuth Client** form appears:



5. Fill out the form with the values below, substituting the environment's details for the highlighted placeholders.



- **Client ID**: auth-code-example
- **Allowed Grant Types**: Authorization Code
- **Allowed Scopes**: (Optional) openid, Innovator, example
- **Allowed CORS Origins**:
- https://server/webalias

  Any other valid domain that should be allowed to send requests with this client id

  CORS domains should not include a trailing slash at the end.
- **Redirect URIs:**

  https://server/webalias/

  https://server/webalias/Client/OAuth/PopupCallback

  Any other valid URI to redirect to after the user logs in, such as the URL of a custom application
- **Post Logout Redirect URIs:**

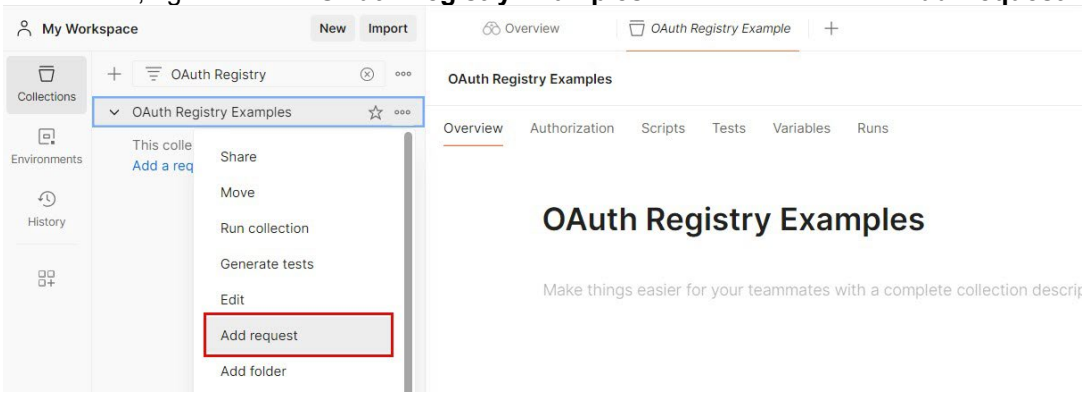
  https://server/webalias/Client/OAuth/PostLogoutCallback

  Any other valid URI to redirect to after the user logs out, such as the URL of a custom application

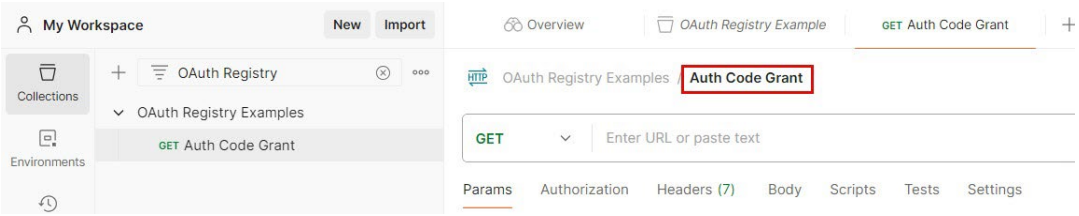
6. Click **Save**.

3.4.3.2 Requesting an Access Token

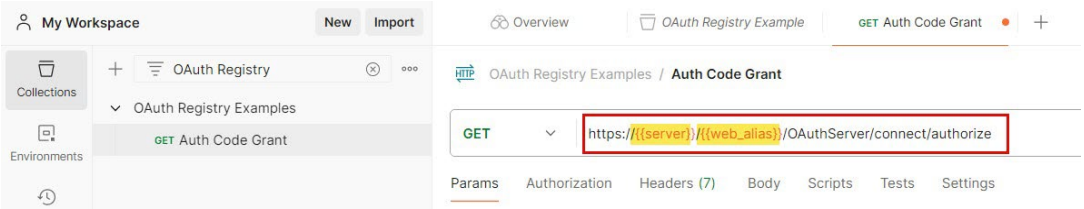
1. In Postman, right-click the **OAuth Registry Examples** collection and select **Add Request**.



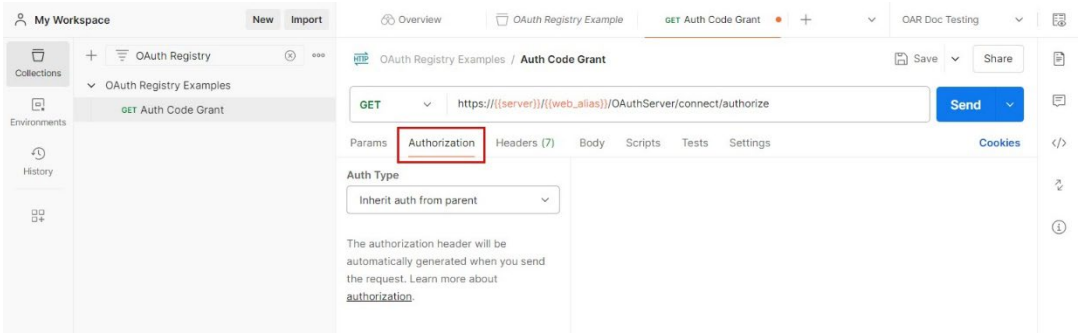
2. Give the request a name like **Auth Code Grant**.



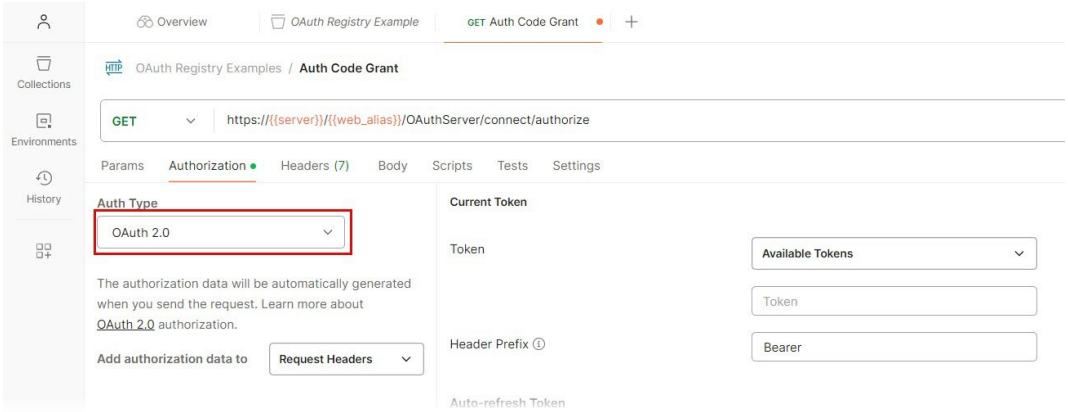
3. Enter the **URL** of the OAuth Server's `authorize` endpoint as shown below.



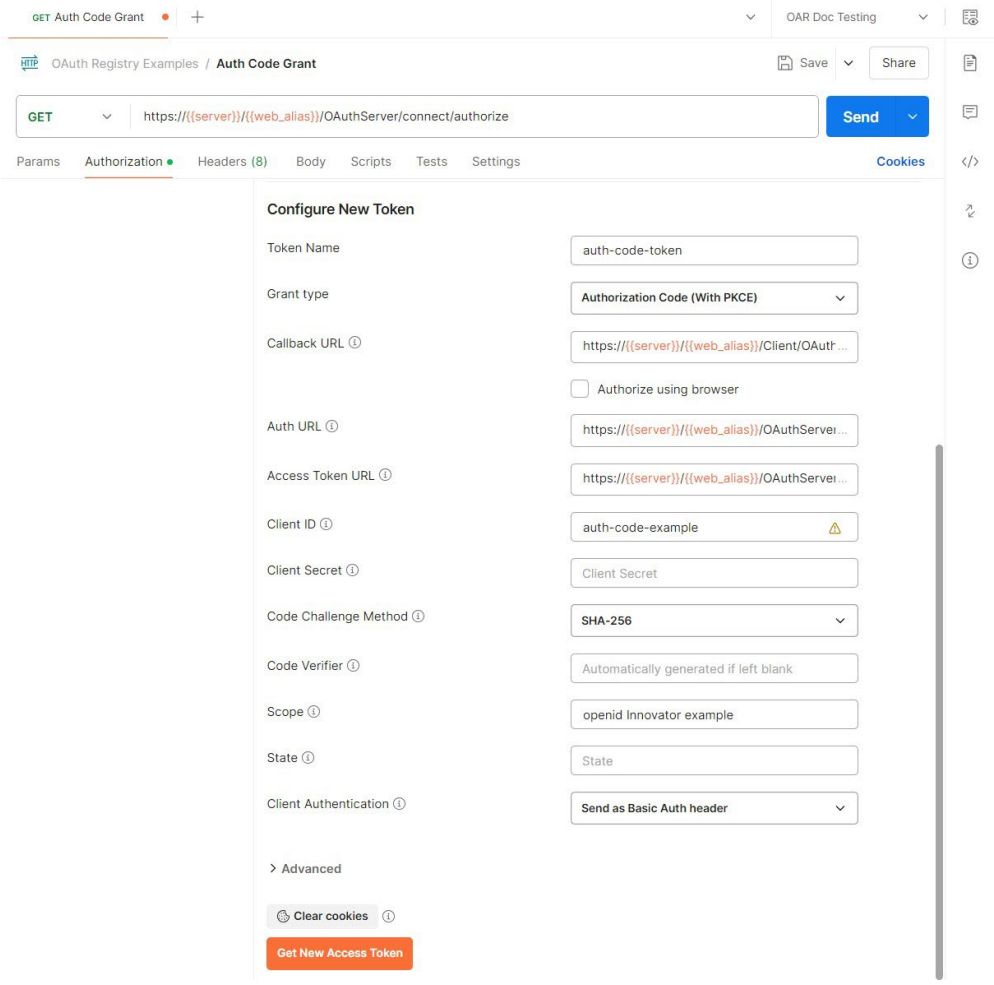
4. Click the **Authorization** tab below the URL bar.



5. Select **OAuth 2.0** from the Auth Type field.



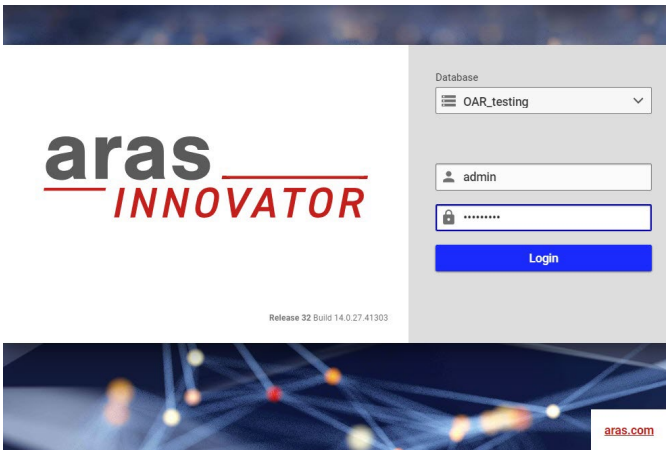
6. Scroll down to **Configure New Token** section and set the configuration options as follows:



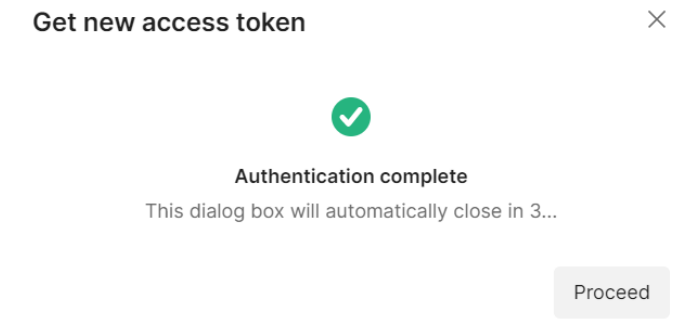
Field	Value
Token Name	Name that Postman will use for the token
Grant Type	Authorization Code (With PKCE)
Callback URL	Enter a valid Redirect URI for the client. Example: https://<servername>/<web alias>/Client/OAuth/PopupCallback
Auth URL	https://<servername>/<web alias>/oauthserver/connect/authorize
Access Token URL	https://<servername>/<web alias>/oauthserver/connect/token
Client ID	auth-code-example
Scope	openid Innovator example

7. Click Get New Access Token.

8. **Log in** to the popup Aras Innovator window with your credentials.

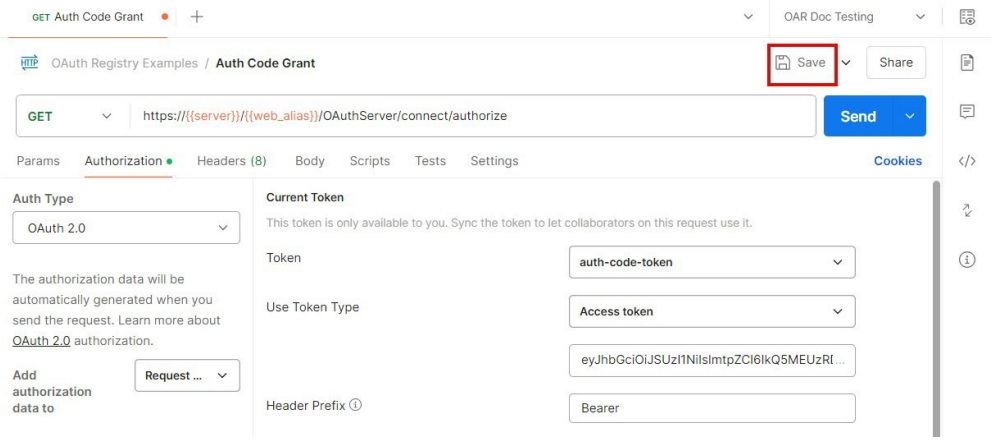


9. In the **Get new access token** dialog box, click **Proceed**.



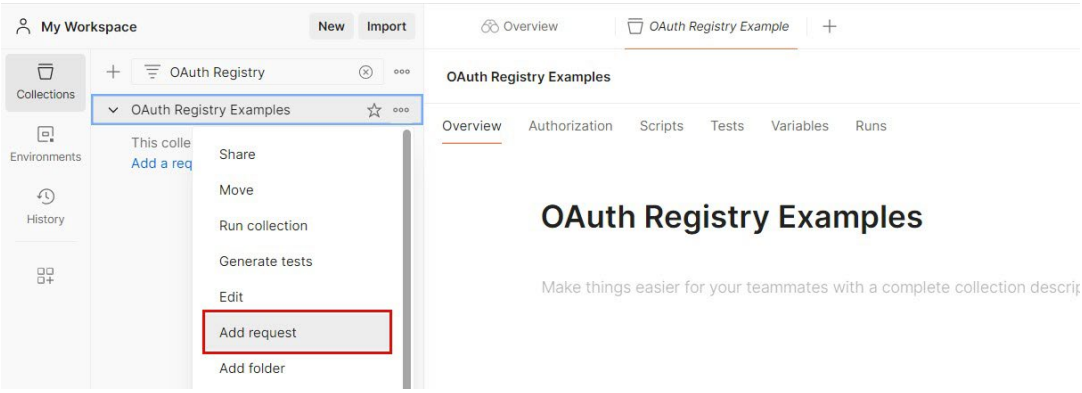
10. In the Manage Access Token dialog box, click Use Token.

11. Click **Save** to save the changes.

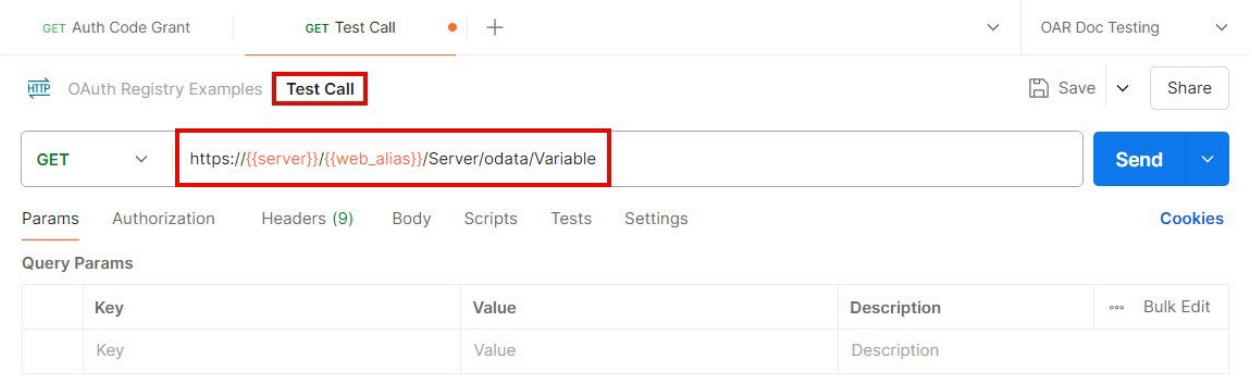


3.4.3.3 *Using the Access Token*

1. In Postman, right-click the **OAuth Registry Examples** collection and select **Add Request**.

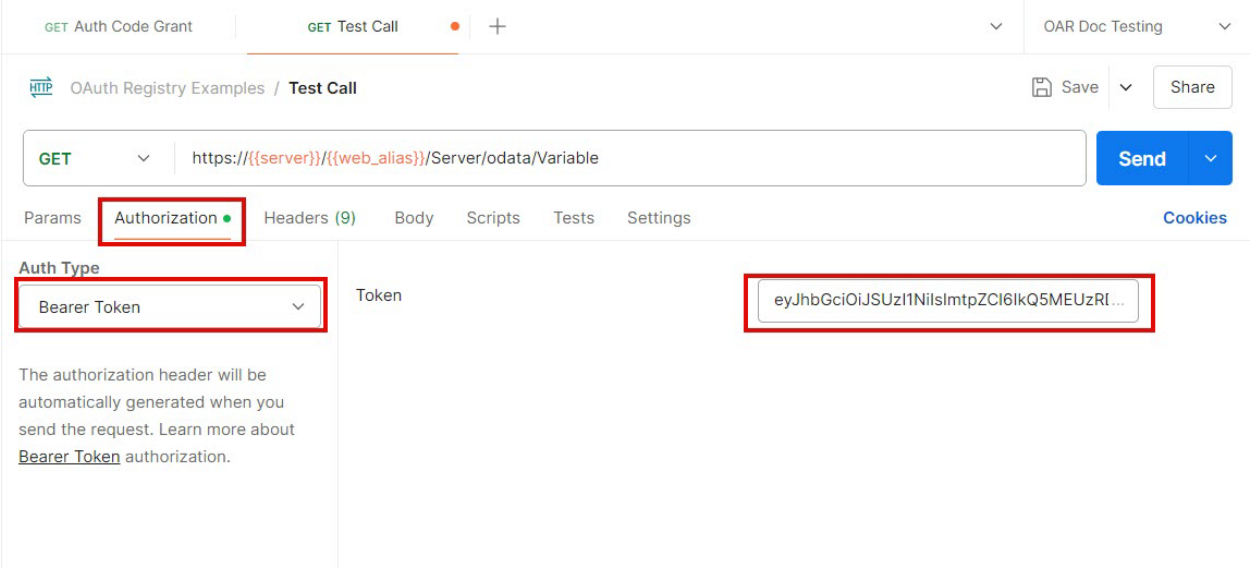


2. Name the request **Test Call** and enter the URL shown below.

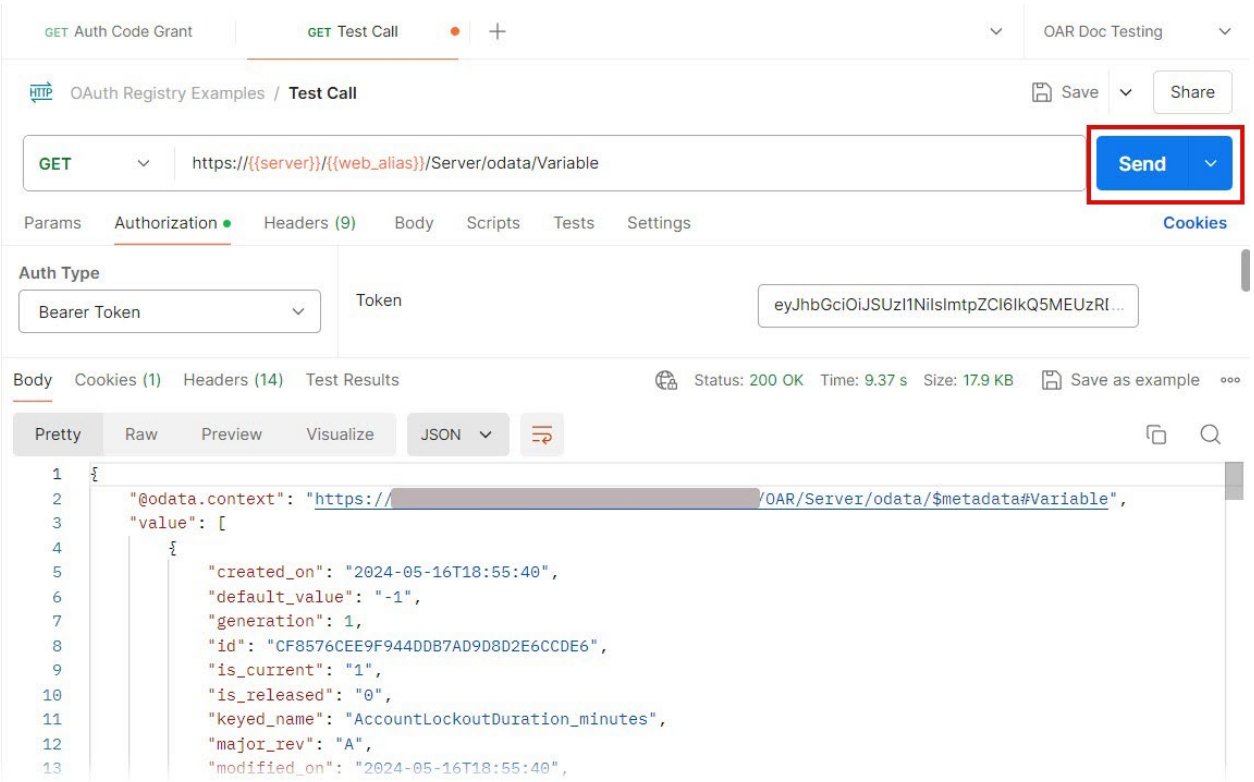


This request will retrieve the Variable items from the Aras Innovator database, confirming the access token is valid.

Select the **Authorization** tab, choose the **Bearer Token** Auth Type, and enter the token requested in the last section.

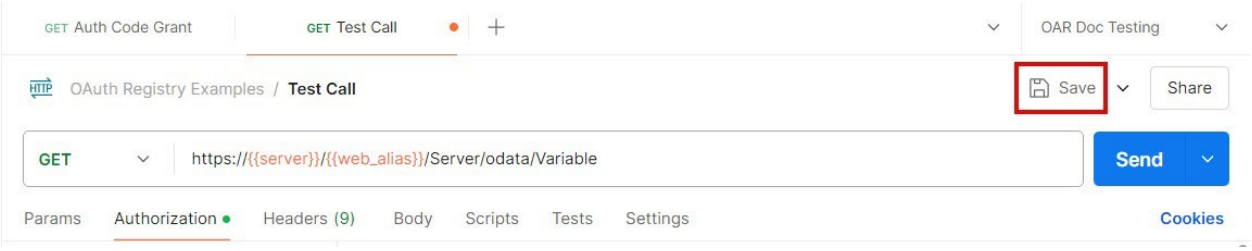


3. Click **Send** to submit the request to the server.



The server should return a 200 OK status code, and the response body should include the Variable items from the Aras Innovator database.

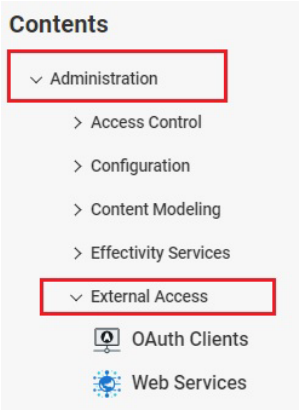
4. Click **Save**.



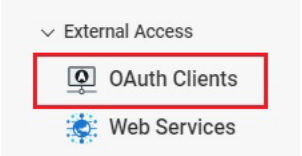
3.4.4 Impersonation Example

3.4.4.1 Configuring an OAuth Client with the Impersonation grant type

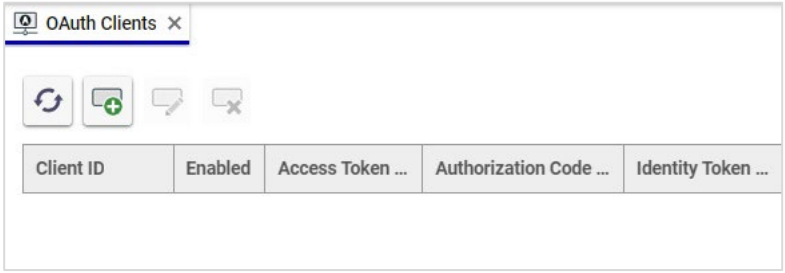
1. Login to Aras Innovator as admin.
2. From the Table of Contents, expand **Administration** and **External Access**.



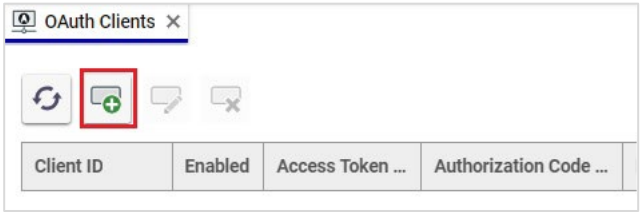
3. Select OAuth Clients.



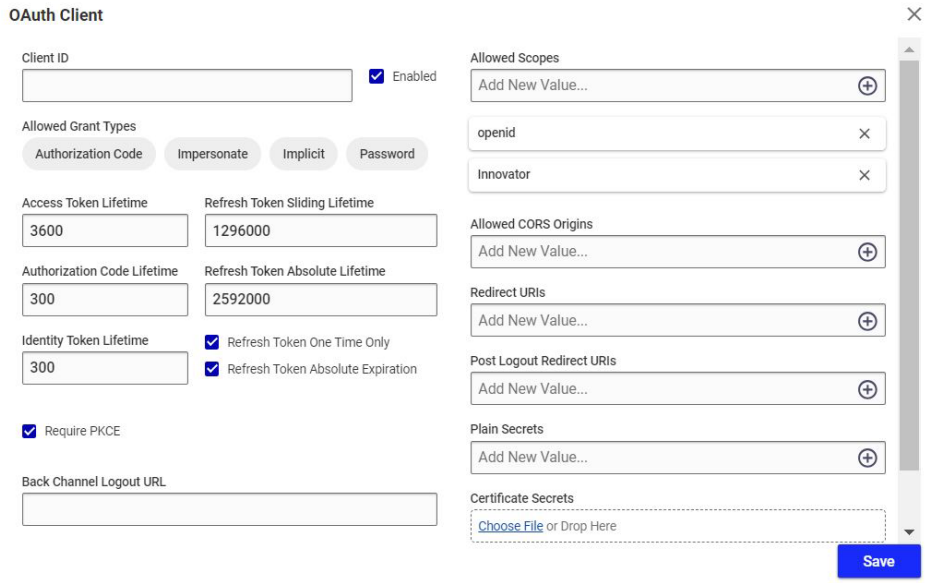
The OAuth Clients grid appears:



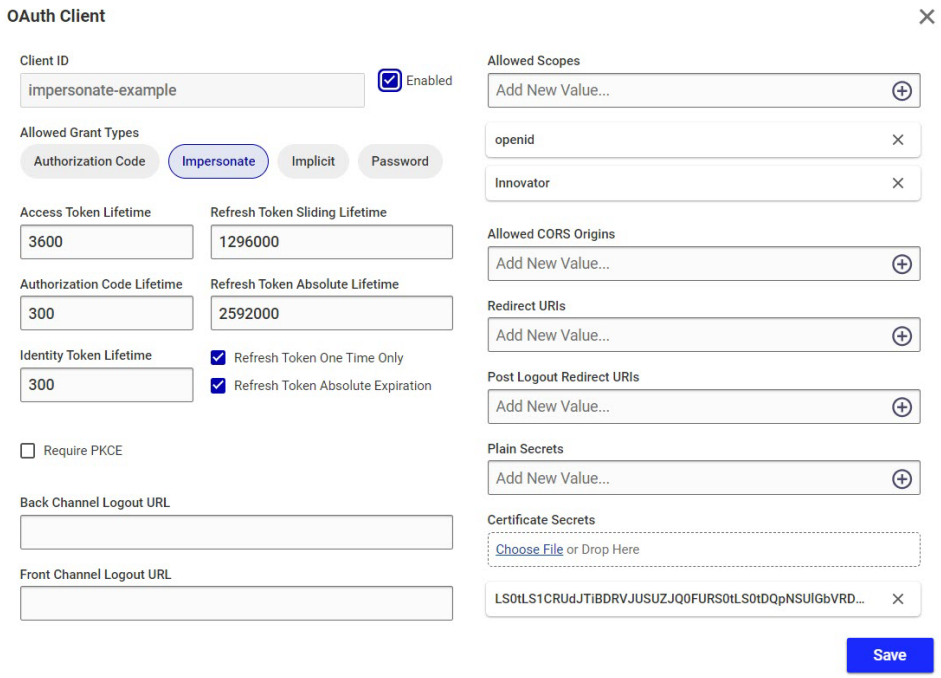
4. Click Create New Client.



The **OAuth Client** form appears:



5. Fill out the form with the values below, substituting the environment's details for the highlighted placeholders.



- **Client ID**: impersonate-example
- **Allowed Grant Types**: Impersonate
- **Certificate Secrets**: Upload the public key (.cer certificate file) that corresponds to the private key on the application server

6. Click **Save**.

3.4.4.2 Using the OAuth Client

Refer to section Obtaining an Access Token in the Aras Innovator Programmer's Guide for information on using an impersonate grant OAuth client with the IOM API.

4 OAuth Registry Permissions

This section describes the permission model for the Aras OAuth Registry and how to configure those permissions.

4.1 Roles

The Aras OAuth Registry uses a role-based permission model to determine which users can configure OAuth Clients.

- **Clients Admin:** Users with this role have permission to create and edit OAuth Clients
- **Users Admin:** Users with this role have permission to control which users have the Clients Admin role. By default, only Super User (root) has this role.

4.2 Users API

The Aras OAuth Registry provides a REST API for managing admin user roles. The API is based on RESTful principles. It uses standard HTTP methods and status codes and is available at /api/v1/users.

Note: This API does not create, edit, or delete User items in the Aras Innovator database. It is solely used to map OAuth Registry roles onto Aras Innovator users. All data this API affects is stored in the MongoDB database installed with the OAuth Registry installation package.

4.2.1 Create a User

Creates a new User with the details provided in the request body.

4.2.1.1 Request

```
HTTP POST /api/v1/users
```

The request body should contain a JSON object with the username and roles to assign.

```
{
    "userName": "user_name",
    "roles": ["ClientsAdmin"]
}
```

4.2.1.2 Responses

- 201 Created

User is created successfully. The response body contains the created user details.

```
{
  "userName": "user_name",
  "roles": ["ClientsAdmin"]
}
```

- 400 Bad Request

The request body is invalid. The response body contains the error details.

```
{
  "userName": [
    "The userName field is required."
  ]
}
```

- 401 Unauthorized

The requester is not authorized to create a new user.

- 403 Forbidden

The requester is not allowed to create a new user.

- 409 Conflict

The user with the same userName already exists.

User with name user_name already exists

4.2.2 Get Users

Retrieves all users.

4.2.2.1 Request

```
HTTP GET /api/v1/users
```

4.2.2.2 Responses

- 200 OK

Users are retrieved successfully. The response body contains the list of users. Returns an empty array [] if no users are found.

```
[{
   "userName": "user_name",
   "roles": ["ClientsAdmin"]
}]
```

- 401 Unauthorized

The request is not authorized to get the users.

- 403 Forbidden

The request is not allowed to get the users.

4.2.3 Get User by Name

Retrieves the user by userName.

4.2.3.1 Request

```
HTTP GET /api/v1/users/{userName}
HTTP GET /api/v1/users?userName={userName}
```

4.2.3.2 Responses

- 200 OK

User is retrieved successfully. The response body contains the user details.

```
{
   "userName": "user_name",
   "roles": ["ClientsAdmin"]
}
```

- 401 Unauthorized

The client is not authorized to get the user.

- 403 Forbidden

The client is not allowed to get the user.

- 404 Not Found

The request with the specified userName does not exist.

4.2.4 Update a User

Updates the user by userName.

4.2.4.1 *Request*

```
HTTP PUT /api/v1/users/{userName}
```

The request body should contain a JSON object with the user details.

```
{
  "userName": "user_name",
  "roles": ["ClientsAdmin"]
}
```

HTTP PUT /api/v1/users?userName={userName}

The request body should contain a JSON object with the user details.

```
{
  "userName": "user_name",
  "roles": ["ClientsAdmin"]
}
```

4.2.4.2 *Responses*

- 200 OK

User is updated successfully. The response body contains the updated user details.

```
{
  "userName": "user_name",
  "roles": ["ClientsAdmin"]
}
```

- 400 Bad Request

The request is invalid:

- The userName in route is not the same with user_name in the request body.
- The request body is invalid. The response body contains the error details.

```
{
  "userName": [
    "The userName field is required."
  ]
}
```

- 401 Unauthorized

The request is not authorized to update the user.

- 403 Forbidden

The request is not allowed to update the user.

- 404 Not Found

The user with the specified userName does not exist.

4.2.5 Delete a User

Deletes the user by userName.

4.2.5.1 *Request*

```
HTTP DELETE /api/v1/users/{userName}
HTTP DELETE /api/v1/users?userName={userName}
```

4.2.5.2 Responses

- 204 No Content

User is deleted successfully.

- 401 Unauthorized

The request is not authorized to delete the user.

- 403 Forbidden

The request is not allowed to delete the user.

- 404 Not Found

The user with the specified userName does not exist.

4.3 API Authentication and Authorization

Only requests with a valid Bearer Token are allowed to access the API. The token should be obtained from the OAuth Server and passed in the Authorization header of the API request.

Only users granted the *UsersAdmin* role may access the Users API.

Note: Super User (root) is a system user that always has access to the Users API.