# OAuth Application Developer Guide

## Access Manager 4.4

**November 2018**

## Legal Notice

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see https://www.microfocus.com/about/legal/.

**License Grant** Licenses purchased for PlateSpin Forge 11 or later versions cannot be used for PlateSpin Forge 3.3 or prior versions.

# Contents

# About This Guide

The *Developer Guide for OAuth Applications* includes the information to help you build an OAuth-based application with Access Manager as the authorizing server.

## Intended Audience

This book is intended for the OAuth application developers, who are creating an OAuth application using Access Manager as the authorization server. It is assumed that you have the basic knowledge of OAuth and its authorization flows.

## Other Information in the Library

REST API Guide

OAuth Samples

# 1 Getting Started

In the OAuth authorization flows, the following are the OAuth 2 defined participants:

- **Resource Owner:** a user who owns the resource.
- **Resource Server:** a server that provides APIs to access user data and perform user functions.
- **Authorization Server:** a server that protects the resource server APIs.

  Here, the authorization server is NetIQ Access Manager.
- **Client:** an application that requires to access the protected data on the resource server.

The simplified interaction between these participants is illustrated in the following figure:
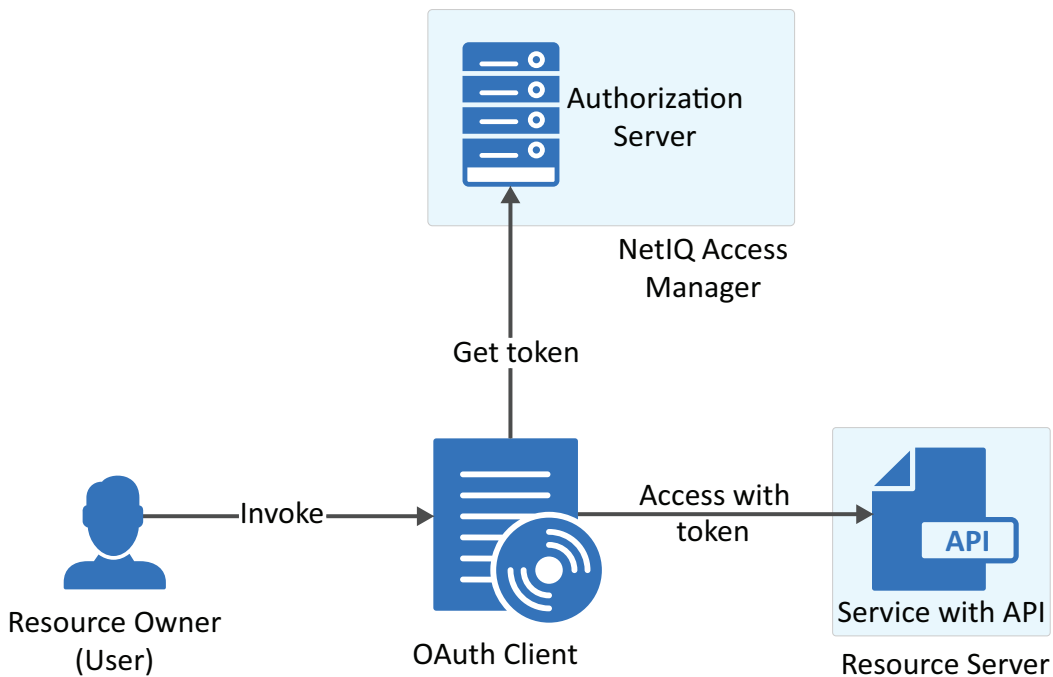
# 2 Steps to Create an OAuth Client Application

The following image displays the basic steps required to build an OAuth client that can access the protected APIs:

| Select | Register | Build | Access | Manage |
|---|---|---|---|---|
| Select the authorization grant type appropriate for your scenario. | Register your OAuth client with NetIQ Acess Manager. | Build the OAuth client. Use the selected authorization flow to get the OAuth tokens. | Access the protected APIs using the access token. | Manage access token, refresh the token and revoke the tokens as required. |
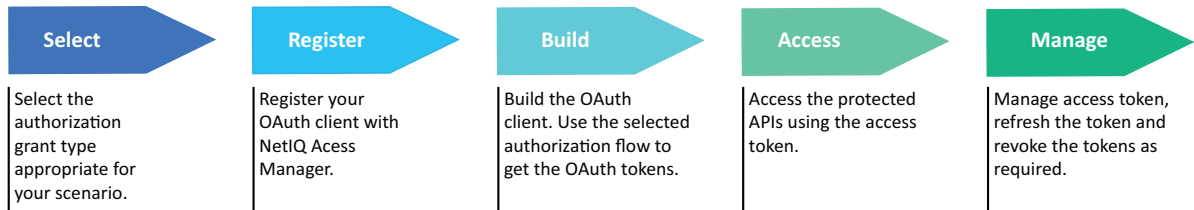
## 2.1 Selecting the Authorization Grant Type

The tokens (access, refresh, and ID tokens) are the key to use OAuth 2.0 and OpenID Connect.

The OAuth protocol provides different ways to obtain these tokens. You can use the appropriate authorization grant type based on the business requirements.

| Authorization Grant | Type of Application |
|---|---|
| Authorization Code | Server-side Applications |
| Authorization Code with PKCE | Native apps |
| Implicit | <ul><li>Single page applications, where navigation between different screens of the website can be performed without loading different webpage in the browser. For example, Gmail.</li><li>Applications that run on the user's device such as, mobile apps</li><li>Web applications that do not require high security</li></ul> |
| Resource Owner Password Credentials | <ul><li>Highly trusted applications or the applications that are owned by the service itself, such as a mobile application</li><li>Legacy application migrating to OAuth</li></ul> |
| Client Credentials | <ul><li>Headless clients</li><li>Batch processing scripts</li></ul> |

| Authorization Grant | Type of Application |
|---|---|
| SAML 2.0 Bearer Grant | Applications that already have the SAML assertions and require to access the OAuth protected resources. |

# 2.2 Registering the OAuth Client Application

Access Manager issues token to the confidential clients. To get the token, you must register the OAuth client application with Access Manager Identity Server (authorization server).

## 2.2.1 Requirements

To register, ensure that you have the information about the following requirements:

- A valid account is created in the Access Manager Identity Server (authorization server).
- The account is enabled with the OAuth developer role.
- (Conditional)

  If you require Identity Server to issue the refresh token, you must inform the Access Manager administrator to enable the **Refresh Token** option in the OAuth global settings.

  If the administrator has not enabled the **Refresh Token** option in the OAuth **Global Settings**, Identity Server will not issue the refresh token even when you register the application using the token type as refresh token.

  ---

  **NOTE:** Refresh tokens are issued for authorization code flow and resource owner flow.

  ---

- The credentials to access the user portal of Access Manager.

  To get the credentials, check with the Access Manager administrator. Also, get the URL and port of the Access Manager Identity Server to log in to the user portal.

  The following URL is a sample URL that you will get from the administrator:

  `https://<IDPServer>:<port>`

- The client application redirection URIs, where Identity Server can send the tokens.

Also, to get the endpoint details of Identity Server, see OAuth Metadata Endpoint.

## 2.2.2 Registering the OAuth Client Application

You can register the client application by using any one of the following options:

- Register the client application using the Access Manager user portal.
    1. Log in to the Access Manager user portal.

       Sample URL: `https://<IDPServer:port>/nidp/portal`
    2. Navigate to **User > Administer OAuth Apps > Register New Client**.
    3. Specify the client configuration details.

       For more information about each field, see the context sensitive help.

> **NOTE:** Select **Grants Required** based on the OAuth flow that you have decided using the first step, .

    4. For refresh tokens, select **Refresh Token** in **Token Types**.

- Register using Rest API.

  Send an API request to the registration endpoint (`https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients`) and include the following required OAuth parameters:

  - **client_name:** Name of the application
  - **redirect_uris:** Redirection URI values
  - **grant_types:** select the grant types based on the OAuth flow that you have decided.

    The grant_type can be `refresh_token` when using authorization code flow or resource owner flow.

  For more information about sending request to the registration endpoint, see .

After the registration is successful, you will receive the client id and the client secret, using this you can initiate the OAuth flow.

# 2.3 Building an OAuth Client

The OAuth client uses APIs to retrieve, manage and use the access tokens for accessing the OAuth protected resource. The steps for these operations may vary depending on the selected OAuth authorization grant type.

This section includes steps for implementing the following OAuth flows:

## 2.3.1    Authorization Code



Resource Owner
(User)

NetIQ Access
Manager

1. Register to obtain Client ID and Client Secret.
2. Request to obtain the authorization code.
3. Log in and provide consent.
4. Authorization code received.
5. Send Client ID, Client Secret and authorization code.
6. Receive the access token and the refresh token.
7. Access with token.
8. Validate the token.

OAuth Client

API

Service with API
Resource Server
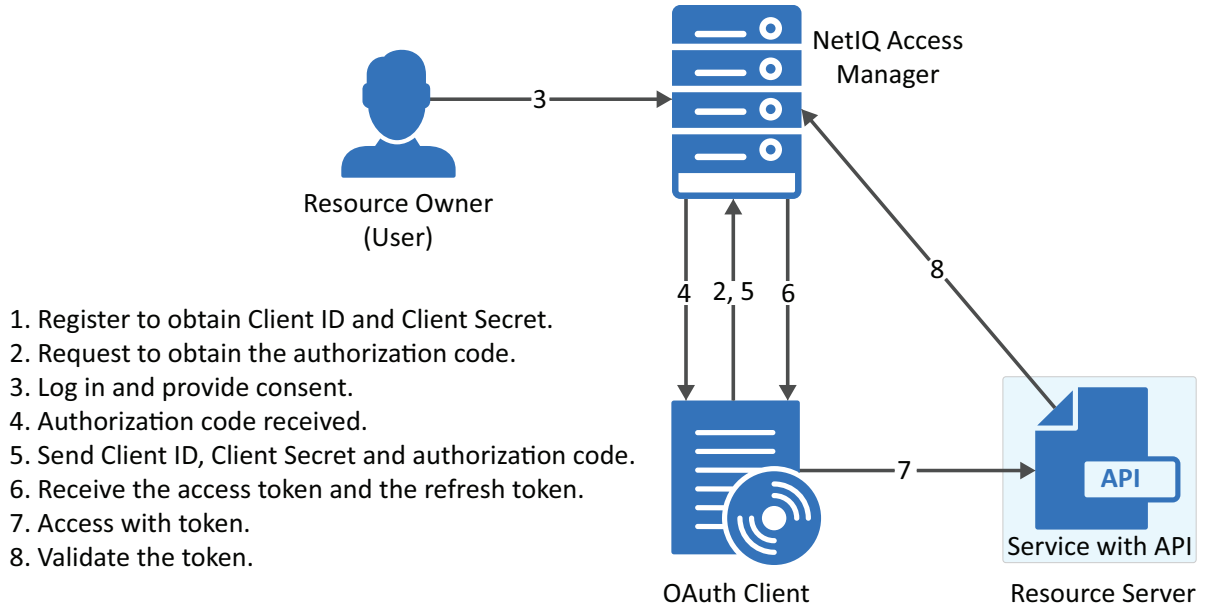
## 2.3.1.1    Register the Client Application

If you have not registered the client application, register it and ensure that the grant type is authorization code.

**Using user portal page:** Set **Grants Required** field as **Authorization Code**

**Using REST API:** Specify the value of `grant_types` parameter as `authorization_code`

After registering the client application you must save the client ID and the secret securely.

## 2.3.1.2    Request for Authorization Code

To get the authorization code, send an HTTPS GET request to the Authorization Endpoint with the appropriate query parameters.

---

**NOTE:** HTTP connections are denied. Therefore, use HTTPS.

---

### Sample request:

```
https://<idphost:port>/nidp/oauth/nam/authz?response_type=code&client_id=bb775b12-
bbd4-423b-83d9-647aeb98608d&redirect_uri=https://client.oauth.com/
callback&scope=profile+email&nonce=ab8932b6&state=AB32623HS
```

| Request Parameter | Description |
|---|---|
| client_id | Client application ID obtained during client registration. |
| response_type | Set it to "code", to indicate Authorization Code flow. OpenID Connect Hybrid flow is supported. The supported response_type are "code", "code id_token", "code token" and "code id_token token". |
| redirect_uri | Access Manager will redirect with the authorization code only to URIs specified during registration. Specify this parameter if you have pre-registered multiple redirect URIs and want to select the specific one to redirect to. Ensure that this exactly matches one of the pre-registered URI.<br><br>If not specified, it defaults to any one of the registered URIs. |
| nonce | String value used to associate a Client session with an ID Token, and to mitigate replay attacks.<br><br>This is required when you have specified the response_type as id_token or you have sent the request with the token and id_token along with the code. |

The preceding table lists the minimal set of parameters. For the complete list of parameters, see Section 4.3, "Authorization Endpoint," on page 39.

## 2.3.1.3    User Authenticates and Authorizes the Client Application

Access Manager Identity Server prompts the user to log in if not already logged in. After a successful login, the user is redirected to the consent screen to authorize the application. The consent screen can include permissions that require user authorization.

## 2.3.1.4    Client Receives the Authorization Code

The Identity Server responds with an HTTP 302 redirect message leading to the redirect_uri specified in the authorization request. If the request does not contain the `redirect_uri` parameter, Identity Server will redirect to one of the registered redirect_uri.

### Sample Response

```
Response HTTP/1.1 302 Found Cache-Control: no-cache, no-store, no-transform
Location: https://client.oauth.com/callback?
code=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0Niwi......&scope=email
```

**NOTE:** If `response_type=code id_token token`, access token and ID token will be included in the response.

| Response Data | Description |
|---|---|
| code | An opaque JWT token. Variable length field. Application should not assume the size of the code and should allocate sufficient space for reading the code. |
| state | Contains the state parameter sent in the authentication request above |
| id_token | Based on response_type sent in the request. If the response_type value is either "code id_token", " code id_token token" then authorization server sends id_token. |

| Response Data | Description |
|---|---|
| access_token | Based on response_type sent in the request. If the response_type value is either "code token" or "code id_token token" then authorization server sends access token. |
| expires_in | Based on response_type sent in the request. Expiration time of the code in seconds since the response was generated. |

## 2.3.1.5  Get Access Token

The code can be exchanged for tokens by sending an HTTPS POST request to the Token Endpoint with the required parameters. The only supported web protocol is HTTPS.

**NOTE:** You can exchange an authorization code only once.

### Sample Request

```
curl --request POST \--url https://<idphost:port> /nidp/oauth/nam/token \--header
'content-type: application/x-www-form-urlencoded' \--data
'grant_type=authorization_code&client_id=b017c96c-b16a-4d80-a5fa-
68f5050abc58&client_secret=ZDDwbuuWPdV_e5quAf7f0Jkg_iJJ7g&redirect_uri=https://
client.oauth.com/
callback&code=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0NNIiwi.......'
```

| Request Parameter | Description |
|---|---|
| grant_type | Set to "authorization_code" |
| client_id | Client ID of the registered client |
| client_secret | Client Secret of the registered client. It is optional for native applications and is mandatory for web applications. |
| code | Code received in the Authorization code flow |
| redirect_uri | This should be same as the one sent during the authorization code request |

The preceding table lists the minimal set of parameters. For the complete list of parameters, see Section 4.4, "Token Endpoint," on page 41.

### Sample Response

```
{"access_token":
"eyJhbGciOiJSU0ExXzU......",
"token_type": "bearer","expires_in": 179,
"refresh_token": "eyJhbGcidHlwIjoiSldUIiwiemlwjo..........",
"scope": "email"}
```

| Response Parameter | Description |
|---|---|
| token_type | Authorization server currently supports only "Bearer" token type |
| access_token | Access token that can be used to invoke resource server APIs |
| id_token | When invoking authorization code request, if the client has sent OpenID in scope parameter, this response object will contain an ID Token. IDToken is signed and encrypted based on the client's registration. |
| Scope | The list of scopes that user has authorized. Hence this may not contain all the scopes that the client requested. |
| State | If the "state" parameter was present in the client authorization request, the same state value sends in response |

**NOTE:** If there are validation errors, the JSON response returns `HTTP Status 400` with additional fields `error` and `error_description`.

### Try Now

To view how the authorization code flow works, you can use the Access Manager OAuth sample scripts. For more information about the sample scripts, see Chapter 5, "Developer Resources," on page 45.

## 2.3.2   Authorization Code with PKCE



1. Register to obtain Client ID and Client Secret.
2. Request to obtain the authorization code with a code challenge method.
3. Log in and provide consent.
4. Authorization code received.
5. Send Client ID, Client Secret and authorization code with *code_verifier*.
6. Receive the access token and the refresh token.
7. Access with token.
8. Validate the token.
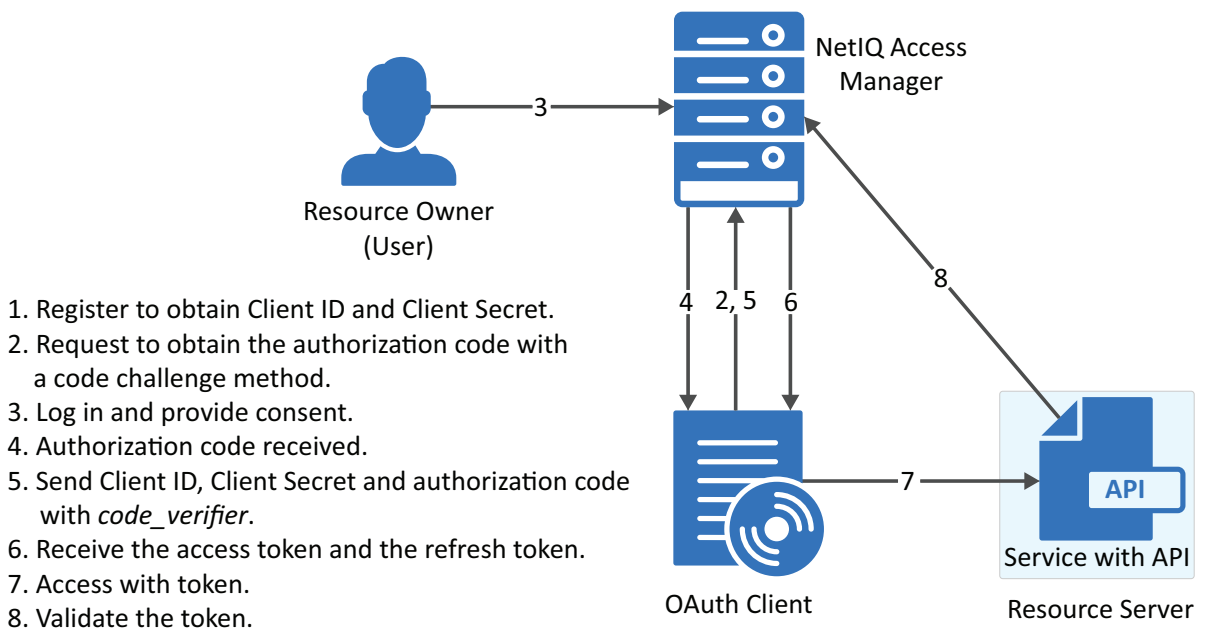
### 2.3.2.1   Register the Client Application

If you have not registered the client application, register it and the grant type must be authorization code.

**Using user portal page:** Set Grants Required field as Authorization Code

**Using REST API:** Specify the value of `grant_types` parameter as `authorization_code`

After registering the client application you must save the client ID and the secret securely.

## 2.3.2.2 Request for Authorization Code with PKCE

Client sends the code challenge as part of the OAuth 2.0 Authorization request with following additional parameters:

### Sample request:

```
https://<<IDP>>:8443/nidp/oauth/nam/authz?code_challenge=WsEH2Rr4lWdciBEb
CuHVlH_UIBUGFPRbDXcPsbPl74&code_challenge_method=S256&scope=profile&response_type=
code&redirect_uri=<<Redirect_URI>>&client_id=484fd33f-12b0-44c4-bbf5-82bae803b71d
```

| Request Parameter | Description |
|---|---|
| code_challenge | Specify the code challenge parameter to initiate the PKCE flow. |
| code_challenge_method | This is an optional parameter. The default value is `plain`.<br><br>You can specify the value as `plain` or `S256`. |

The preceding table lists the minimal set of parameters. For the complete list of parameters, see Section 4.3, "Authorization Endpoint," on page 39.

## 2.3.2.3 User Authenticates and Authorizes the Client Application

Access Manager Identity Server prompts the user to log in if not already logged in. After a successful login, the user is redirected to the consent screen to authorize the application. The consent screen can include permissions that require user authorization.

## 2.3.2.4 Client Receives the Authorization Code

The Identity Server responds with an HTTP 302 redirect message to the redirect_uri specified in the Authorization request. If the request does not contain redirect_uri parameter, the Identity Server will redirect to one of the registered redirect_uri.

### Sample Response

```
Response HTTP/1.1 302 Found Cache-Control: no-cache, no-store, no-transform
Location: https://client.oauth.com/callback?
code=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0Niwi......&scope=email
```

**NOTE:** If you specify `response_type=code id_token token`, access token and ID token will be included in the response.

| Response Parameter | Description |
|---|---|
| *code* | An opaque binary token. Variable length field. Application should not assume the size of the code and should allocate sufficient space for reading the code. |

| Response Parameter | Description |
| --- | --- |
| *state* | Contains the state parameter sent in the authentication request above |
| id_token | Based on `response_type` sent in the request. If the response_type value is either `code id_token`, `code id_token token` then authorization server sends ID token. |
| access_token | Based on `response_type` sent in the request. If the response_type value is either `code id_token`, `code id_token token`, then authorization server sends access token. |
| expires_in | Based on `response_type` sent in the request. Expiration time of the access token in seconds since the response was generated. |

## 2.3.2.5  Get Access Token

The code can be exchanged for tokens by sending an HTTPS POST request to Token Endpoint with the required parameters. The token request must contain the `code_verifier` parameter. The only supported web protocol is HTTPS.

**NOTE:** You can exchange an authorization code only once.

### Sample Request

```
curl --request POST \--url https://<idphost:port> /nidp/oauth/nam/token \--header
'content-type: application/x-www-form-urlencoded' \--data
'grant_type=authorization_code&redirect_uri=<Redirect_URI>&client_id=484fd33f-
12b0-44c4-bbf5-
82bae803b71d&code_verifier=0ak1mD3loHOy1ZksmyoO1fQEhRBEuzGYbkQqKFe1N
y0'
```

| Request Parameter | Description |
| --- | --- |
| grant_type | Set to "authorization_code" |
| client_id | Client ID of the registered client |
| client_secret | Client Secret of the registered client. It is optional for native applications and is mandatory for web applications. |
| code | Code received in the Authorization code flow |
| redirect_uri | This should be same as the one sent during the authorization code request |
| code_verifier | Code verifier parameter is required if authorization code is requested using the PKCE flow. The server verifies `code_verifier` before returning the token. |

The preceding table lists the minimal set of parameters. For information about the complete list of parameters, see Section 4.4, "Token Endpoint," on page 41.

**Sample Response**

```
{"access_token":
"eyJhbGciOiJSU0ExXzU......",
"token_type": "bearer","expires_in": 179,
"refresh_token": "eyJhbGcidHlwIjoiSldUIiwiemlwjo..........",
"scope": "email"}
```

| Response Parameter | Description |
|---|---|
| token_type | Authorization server currently supports only "Bearer" token type. |
| access_token | Access token that can be used to invoke resource server APIs. |
| id_token | When invoking authorization code request, if the client has sent OpenID in scope parameter, this response object will contain an ID Token. IDToken is signed and encrypted based on the client's registration. |
| Scope | The list of scopes that user has authorized. Hence this may not contain all the scopes that the client requested. |
| State | If the "state" parameter was present in the client authorization request, the same state value sends in response. |

**NOTE:** If validation errors occurred, HTTP Status 400 is returned with additional fields "error" and "error_description" in the JSON response.

**Try Now**

To view how the authorization code with PKCE flow works, you can use the Access Manager OAuth sample script. For more information about the OAuth samples, see Chapter 5, "Developer Resources," on page 45.

## 2.3.3 Implicit



1. Register to obtain Client ID and Client Secret.
2. Implicit token request.
3. Log in and provide consent.
4. Receive the access token.
5. Access with token.
6. Validate token.
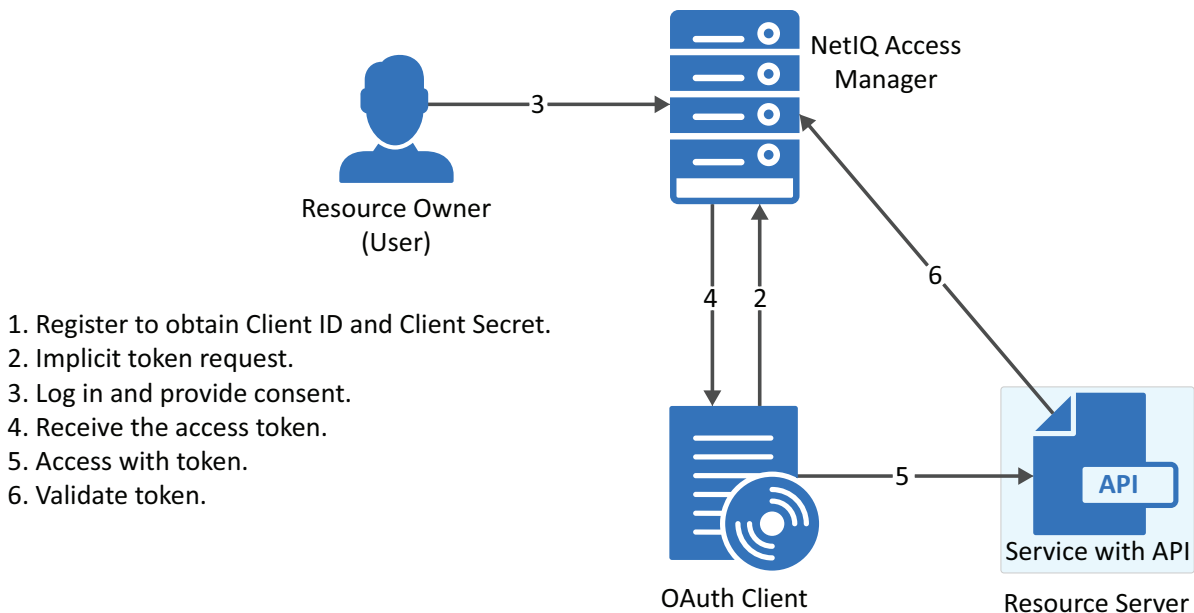
The implicit flow is the simplest to integrate but it is also considered as insecure because the access token is sent in the browser request. To secure this token, Access Manager sends these tokens in the fragment mode.

## 2.3.3.1 Register the Client Application

If you have not registered the client application, register it and the grant type must be implicit.

**Using user portal page:** Set **Grants Required** field as **Implicit**

**Using REST API:** Specify the value of `grant_types` parameter as `implicit`

After registering the client application you must save the client ID and the secret securely.

## 2.3.3.2 Application sends Implicit token request

To initiate the Implicit Grant flow, make an HTTPS GET request to the Authorization endpoint with required parameters. The only supported web protocol is HTTPS.

### Sample Request

```
https://<idphost:port>/nidp/oauth/nam/
authz?response_type=token+id_token&client_id=4e4ae330-1215-4fc8-9aa7-
79df8325451c&redirect_uri=https://client.oauth.com/
callback&scope=profile+email+OpenID&state=s1234&nonce=n123
```

| Request Parameter | Description |
|---|---|
| *client_id* | Client application ID obtained during client registration |
| *response_type* | Set to 'token". OpenID Connect Hybrid flow is supported. The supported response_type are "token", "id_token", and "token id_token". |
| *redirect_uri* | If provided, the value of this must exactly match one of URIs of the registered application. |
| *state* | An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when the response sends to the client. The parameter is used to prevent cross-site forgery requests. |
| *nonce* | String value used to associate a Client session with an ID Token. It is used to mitigate replay attacks. This is a mandatory parameter when the `response_type` includes `id_token`. |

The preceding table lists the minimal set of parameters. For the complete list of parameters see, .

## 2.3.3.3 User Authenticates and Authorizes Application

Access Manager Identity Server prompts the user to log in if not already logged in. After a successful login, the user is redirected to the consent screen to authorize the application. The consent screen can include permissions that require user authorization.

## 2.3.3.4 Application receives token response

The Authorization Endpoint sends an HTTP 302 redirect response with `token` and/or `id_token` as a fragment to the registered redirect_uri.

### Sample Response

```
https://client.oauth.com/
callback#token_type=bearer&access_token=eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1O......&ex
pires_in=3600&id_token=eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJo&scope=email&state=s1234
```

| Response Parameter | Description |
|---|---|
| token_type | The type of the token. Authorization server currently supports only Bearer type |
| access_token | Based on response_type value sent in the request. If the response_type value is either "token" or "token id_token" then authorization server returns access token |
| id_token | Based on response_type sent in the request. If the response_type value is either "id_token" or "token id_token" then authorization server returns id_token. |
| scope | The list of scopes that user has authorized. This can contain all the scopes the client requested or lesser. |
| state | if the "state" parameter was present in the client authorization request, the same state value sends in response. |

**NOTE:** If validation errors occurred, HTTP Status 400 is returned with additional fields "error" and "error_description" in the JSON response.

## 2.3.3.5 Try Now

To view how the implicit flow works, you can use the Access Manager OAuth samples. For more information about the samples, see Chapter 5, "Developer Resources," on page 45.

## 2.3.4 Resource Owner

1. Register to obtain Client ID and Client Secret.
2. Log in.
3. Authenticating the user and the client.
4. Receive the access token and the refresh token.
5. Access with token.
6. Validate token.
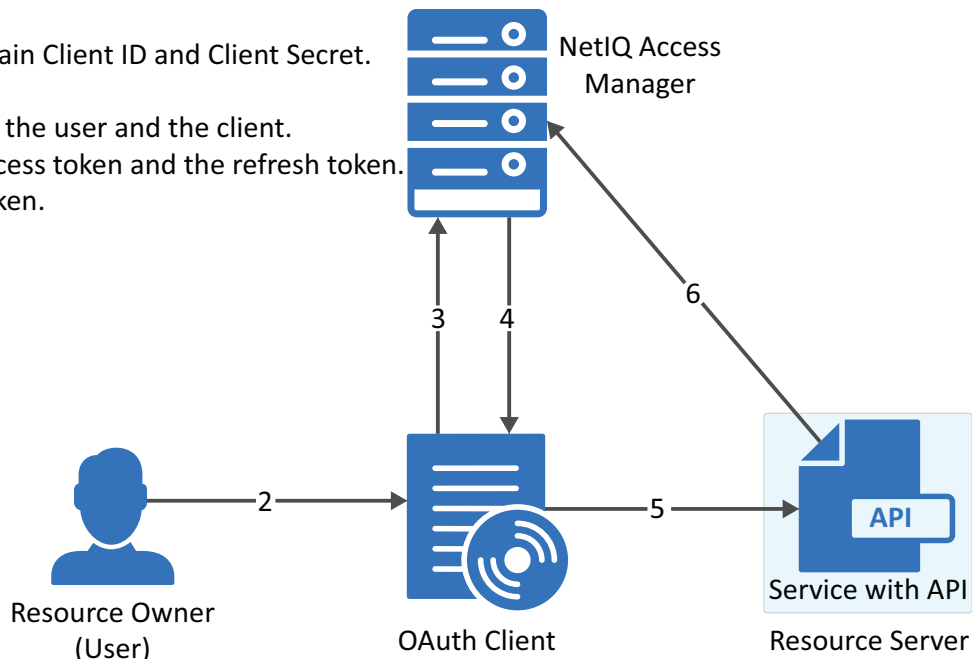


### 2.3.4.1 Register the Client Application

If you have not registered the client application, register it and the grant type must be resource owner credentials.

**Using user portal page:** Set **Grants Required** field as **Resource Owner Credentials**

**Using REST API:** Specify the value of `grant_types` parameter as `password`

After registering the client application you must save the client ID and the secret securely.

### 2.3.4.2 Application Requests for Access Token

The client application should collect the user credentials and make HTTPS POST request to the token endpoint for an access token.

**NOTE:** The HTTP connections are refused, use HTTPS.

#### Sample CURL request

```
curl --request POST \
 --url https://<idphost:port>/nidp/oauth/nam/token \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data 'grant_type=password&client_id=bb775b12-bbd4-423b-83d9-
647aeb98608d&client_secret=bBbE-
4mNO_kWWAnEeOL1CLTyuPhNLhHkTThArEckyrdLmRLn3GhnxjsKI2mEijCSlPjftxHod05dp-
uGs6wA&username=user1&password=pass@123&scope=email%20profile'
```

| Request Parameter | Description |
| --- | --- |
| grant_type | Value should be "password'" |
| client_id | Client ID of the registered client |
| client_secret | Client Secret of the registered client. It is optional for native application, for web application secret is mandatory. |
| username | The user login name |
| password | The user login password |

The preceding table lists the minimal set of parameters. For the complete list of parameters see, Section 4.4.1, "Request Parameters," on page 41.

## 2.3.4.3 Application Receives Access Token

### Sample token response

```
{
"access_token":
"eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0NDMiLCJ0.
....",
"token_type" : "bearer",
"expires_in" : 3599,
"scope" : "profile email"
}
```

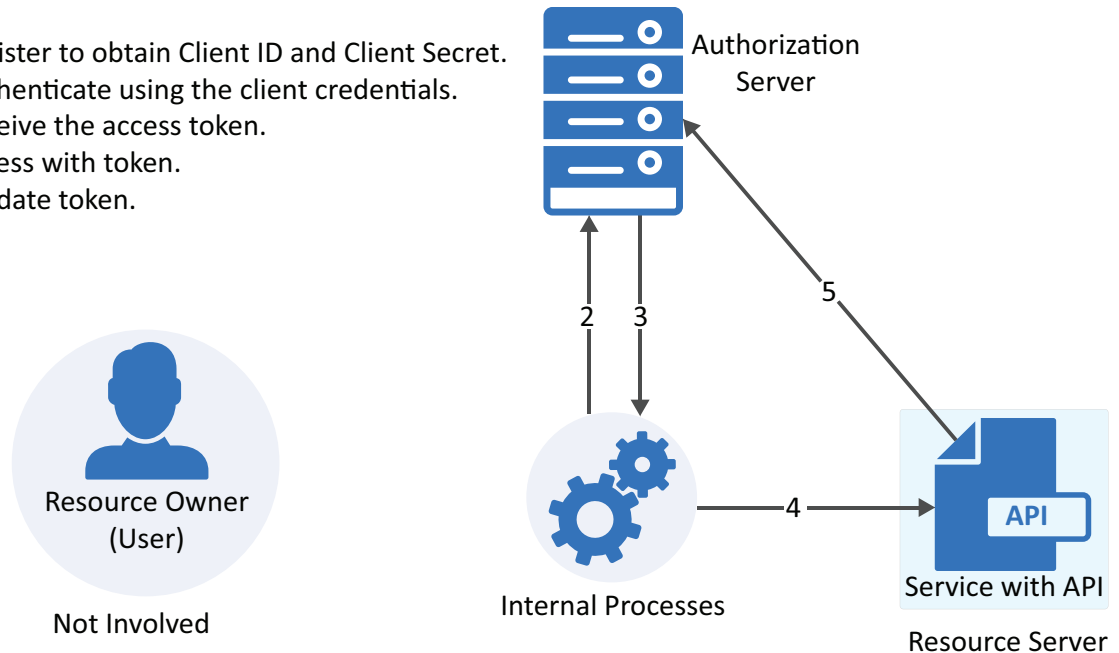| Response Parameter | Description |
| --- | --- |
| token_type | The type of the token. Authorization server currently supports only Bearer type |
| access_token | Access token that can be used to invoke resource server APIs |
| expires_in | The remaining lifetime of the access token. |
| scope | Scopes requested. |

**NOTE:** If validation errors are occurred, HTTP Status 400 returned with the JSON response contains "error" and "error_description".

### Try Now

To view how resource owner flow works, you can use the Access Manager OAuth samples. For more information about the samples, see Chapter 5, "Developer Resources," on page 45.

## 2.3.5    Client Credentials

1. Register to obtain Client ID and Client Secret.
2. Authenticate using the client credentials.
3. Receive the access token.
4. Access with token.
5. Validate token.



Resource Owner (User) — Not Involved

Internal Processes

Service with API — Resource Server

Authorization Server

## 2.3.5.1    Register the Client Application

If you have not registered the client application, register it and the grant type must be client credentials.

**Using user portal page:** Set **Grants Required** field as **Client Credentials**

**Using REST API:** Specify the value of `grant_types` parameter as `client_credentials`

After registering the client application you must save the client ID and the secret securely.

## 2.3.5.2    Get Access Token

To get access token make an HTTPS POST request to Token endpoint.

### Sample Request

```
curl --request POST \
--url https://<idphost:port>/nidp/oauth/nam/token \
--header 'content-type: application/x-www-form-urlencoded' \
--data 'grant_type=client_credentials&client_id=bb775b12-bbd4-423b-83d9-
647aeb98608d&client_secret=bBbE-
4mNO_kWWAnEeOL1CLTyuPhNLhHkTThArEckyrdLmRLn3GhnxjsKI2mEijCSlPjftxHod05dp-
uGs6wA&scope=read%20write'
```

| Request Parameter | Description |
| --- | --- |
| *grant_type* | Set to "client_credentials'" |
| *client_id* | Client ID of the registered client |
| *client_secret* | Client Secret of the registered client. It is optional for native application and is mandatory for web applications. |
| *scope* | List of scopes the application requires. The scope read and write are custom scopes, For creating custom scopes, see Section 3.3, "Creating Custom OAuth2 Scope," on page 33. |

### Sample Response

```
{"access_token": "eyJraWQiOiI0MjgzNzQyNDYxDYwNzYyODAzMz ........","token_type" :
"bearer","expires_in" : 3599,"scope" : "read write"}
```

| Response Parameter | Description |
| --- | --- |
| *token_type* | Authorization server currently supports only "Bearer" token type |
| *access_token* | Access token that can be used to invoke resource server APIs |
| *expires_in* | The remaining lifetime of the access token. |
| *scope* | Scopes requested. |

**NOTE:** If validation errors occurred, HTTP Status 400 is returned with additional fields "error" and "error_description" in the JSON response.

## 2.3.5.3    Try Now

To view how the client credentials flow works, you can use the Access Manager OAuth samples. For more information about the samples, see Chapter 5, "Developer Resources," on page 45.

## 2.3.6 Security Assertion Markup Language (SAML) 2.0 Bearer Grant
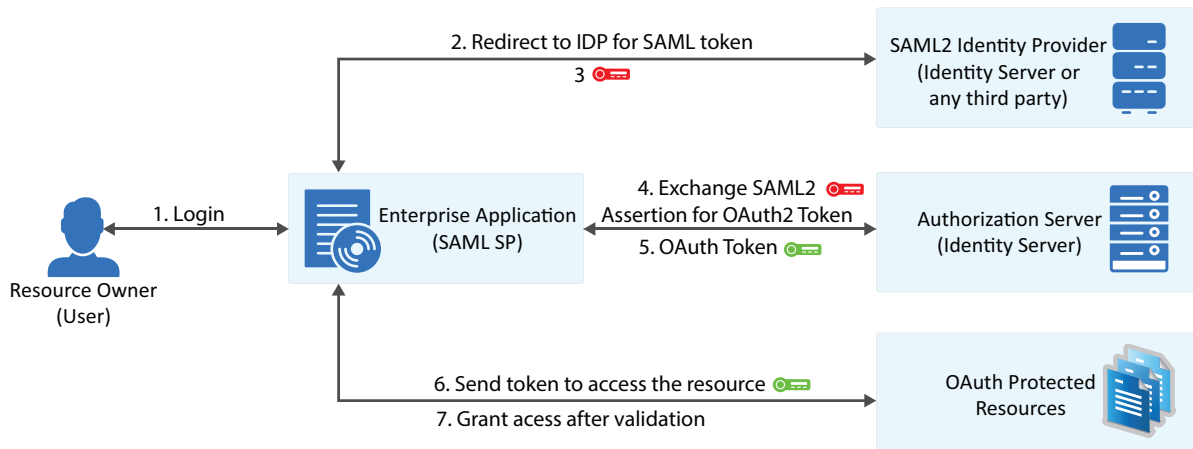


This flow requires trust relationship between Identity Providers and Service Provider. The Access Manager administrator needs to configure the assertion issuer details in Administration Console.

### 2.3.6.1 Register the Client Application

If you have not registered the client application, register it and the grant type must be SAML 2.0 assertion.

**Using user portal page:** Set **Grants Required** field as **SAML 2.0 Assertion**

**Using REST API:** Specify the value of grant_types parameter as saml2_assertion

After registering the client application you must save the client ID and the secret securely.

### 2.3.6.2 Exchange SAML 2.0 Assertion for Access Token

The SAML 2.0 assertions can be exchanged for access token. The consent page will not be displayed to the user for authorizing scopes. The access token will have only the scopes that are previously approved by the user.

To get access token send an HTTPS POST request to the token endpoint.

---

**NOTE:** The HTTP connections are denied, use HTTPS.

---

**Sample CURL request**

```
curl --request POST \
 --url https://<idphost:port>/nidp/oauth/nam/token \
 --header 'content-type: application/x-www-form-urlencoded' \
 --data 'grant_type=urn:ietf:params:oauth:grant-type:saml2-
bearer&client_id=bb775b12-bbd4-423b-83d9-
647aeb98608d&assertion=MPHnbWxv01….SY2&scope=email%20profile'
```

| Request Parameter | Description |
| --- | --- |
| grant_type | Value should be "urn:ietf:params:oauth:grant-type:saml2-bearer'" |
| client_id | Client ID of the registered client |
| assertion | A single base64url encoded SAML2.0 Assertion as value for this parameter. |
| client_secret | Optional. The client secret value |
| scope | List of scopes the application requires. Scope values should be space separated %20 or + |

### 2.3.6.3 Application Receives Access Token

#### Sample token response

```
{
"access_token":
"eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0NDMiLCJ0.
....",
"token_type" : "bearer",
"expires_in" : 3599,
"scope" : "profile email"
}
```

| Response Parameter | Description |
| --- | --- |
| access_token | Access token that can be used to invoke resource server APIs |
| scope | Requested scopes that are pre-approved by the user. |
| expires_in | The remaining lifetime of the access token. |
| token_type | The type of the token. Authorization server currently supports only Bearer type |

**NOTE:** If validation errors are occurred, HTTP Status 400 returned with the JSON response contains "error" and "error_description".

## 2.4 Accessing Protected APIs

The client has to include the access token when invoking any OAuth protected API service. The API server will validate this access token and authorize the incoming API requests based on the scopes embedded in the access token. For information about validating the tokens, see Section 2.5.4, "Validating a JWT Token," on page 30.

### 2.4.1 Sample API request with access token using curl

```
 curl -X POST -H "Authorization: Bearer eyJhbGciOiJSU0ExXzU....."""https://
api.oauth.apiserver.com/v1/resource"
```

# 2.5 Managing Tokens

The administrator/developer can perform the following to manage the tokens:

## 2.5.1 Using Refresh Token

The access token has shorter lifetime. The refresh tokens have longer life time. Hence, to use access token for a longer duration you can use the refresh token. When the original access token is expired or going to expire, you can send a request with the refresh token to get a fresh access token. The refresh token is useful in a scenario where an application does not require frequent usage of authentication from user.

Identity Server issues refresh token for authorization code grant and resource owner grant.

---

**NOTE:** You can request for refresh token if the administrator has enabled the use of refresh token and if you have registered the application with token type as refresh token. For more information about registering the application, see Section 2.2, "Registering the OAuth Client Application," on page 10.

---

### 2.5.1.1 Requesting Access Token Using Refresh Token

The refresh token request should be sent to the Token Endpoint. The client application makes an HTTPS POST request to the /token endpoint for getting a fresh access token by using the refresh token.

**Sample CURL request**

```
curl --request POST \
--url https://<idphost:port>/nidp/oauth/nam/token \
--header 'content-type: application/x-www-form-urlencoded' \
--data 'grant_type=refresh_token&client_id=4e4ae330-1215-4fc8-9aa7-
79df8325451c&client_secret=Rxl5pvgL80DBzbIcLPVnH17FehZA8L&refresh_token=eyJraWQiOi
I0MjgzNzQyNDYxMjE5OTM1.....'
```

| Request Parameter | Description |
|---|---|
| grant_type | Value should be "refresh_token'" |
| client_id | Client ID of the registered client |
| client_secret | Client Secret of the registered client. It is optional for native application, for web application secret is mandatory. |
| refresh_token | refresh_token that is obtained during authorization code, resource owner credentials flow |

For information about other request parameters, see Section 4.4, "Token Endpoint," on page 41.

### 2.5.1.2 Receiving Access Token

**Sample token response**

```
{"access_token":
"eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0NDMiLCJ0.
....","token_type" : "bearer","expires_in" : 3599, "scope" : "profile email"}
```

| Response Parameter | Description |
|---|---|
| token_type | The type of the token. Authorization server currently supports only Bearer type |
| access_token | Fresh access token that can be used to invoke resource server APIs |
| refresh_token | Optionally re-issues a refresh token if configured in client application |
| expires_in | The remaining lifetime of the access token. |
| scope | The granted scopes to the client. |

**NOTE:** If validation errors are occurred, HTTP Status 400 returned with the JSON response contains "error" and "error_description".

## 2.5.2 Revoking Tokens

**Prerequisite:** Inform the Access Manager administrator to enable token revocation in the OAuth global settings of Identity Server.

The client application can revoke the refresh token in the following scenarios:

- The end-user logs out
- The application is uninstalled
- To notify the Identity Server that a previously obtained refresh token is no longer required.

The refresh token received in earlier flows can be revoked by sending an HTTPS Post request to the revocation endpoint, /revoke of Identity Server.

**NOTE:** Only the refresh tokens that are in JWT format can be revoked. Therefore, You can revoke only the refresh tokens that are issued from Access Manager 4.4 or later versions.

### 2.5.2.1 Application Request to Revoke Refresh Token

**Sample request**

```
curl --request POST \
--url https://<idphost:port>/nidp/oauth/nam/revoke\
--header 'content-type: application/x-www-form-urlencoded' \
--data 'client_id=bb775b12-bbd4-423b-83d9-647aeb98608d&client_secret=bBbE-
4mNO_kWWAnEeOL1CLTyuPhNLhHkTThArEckyr&token=eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OT
YyOD'
```

| Request Parameter | Description |
|---|---|
| client_id | Client ID of the registered client |
| client_secret | Client Secret of the registered client. It is optional for native application, for web application secret is mandatory. |
| token | refresh_token that is obtained during authorization grant, resource owner credentials flow |

## 2.5.2.2 Application Receives Response for Revoking Refresh Token

- The Identity Server responds with HTTP status code 200 OK if the token has been revoked successfully or if the client submitted an invalid token.
- The error code unsupported_token_type is returned by the Identity Server when the given token is not a refresh token.
- If the Identity Server responds with HTTP status code 503, the client must assume the token still exists and may retry revoking the refresh token after a reasonable delay.

## 2.5.3 Revoking Token Issued to a Device

You can ask the Access Manager administrator to revoke refresh tokens that are issued to a device. This feature is helpful in following scenarios:

- If the device is stolen or user lost the device
- Token is given to user for multiple devices and one device is compromised.
- User revokes by revoking the consent.

  When consent is removed, user is required to grant permission again to use the token.

The token can be manually associated with a device by providing additional parameter "device_id" while requesting for access token. Such manually associated tokens can be revoked using the revocation endpoint.

## 2.5.3.1 Application Requests for Revoke Token issued to a Device

The client application makes HTTPS POST request to `/revoke` endpoint with device_id as path parameter.

### Sample CURL request

```
curl --request POST \
--url https://<idphost:port>/nidp/oauth/nam/revoke/<device_id>\
--header 'content-type: application/x-www-form-urlencoded' \
--data 'userstore_name=namsignboxuserstore&user_dn=cn%3Dharry%2Co%3Dnovell'
```

### The request parameters that are used

| Parameter | Required | Description |
|---|---|---|
| userstore_name | Yes | Specify the name of the user store |

| Parameter | Required | Description |
|-----------|----------|-------------|
| user_dn | Yes | Specify the user's dn to whom the token issued. |

## 2.5.3.2 Application Receives Response

### Sample successful response

HTTP/1.1 200 OKCache-Control: no-cache, no-store, no-transformContent-Length: 0Date: Tue, 03 Mar 2015 18:12:55 GMT{"status": "Successfully revoked token(s) issued to this device."}

**NOTE:** If validation errors are occurred, HTTP Status 4xx returned with the JSON response contains "error" and "error_description".

# 2.5.4 Validating a JWT Token

The tokens are validated based on the configuration of the resource server that is managed through Access Manager Administration Console. To validate access tokens, you require to understand the encryption method used for encrypting the access tokens.

By default token encryption is done by using Access Manager keys. To encrypt access token using resource server keys share the encryption keys details in JWKS format (JSON Web Key Set) or JWKS URI to Access Manager Administrator for creating custom resource server using either REST API or Administration Console.

Refresh tokens are always encrypted by using Access Manager Encryption keys.

## 2.5.4.1 Validating Tokens Encrypted Using the Access Manager keys

To validate access tokens and refresh token that are signed and encrypted by using Access Manager encrypted keys, send a request to TokenInfo endpoint, `/tokeninfo`.

### Sample request

The request should contain the token in the Authorization header as follows: Authorization: Bearer access_token

```
HTTP/1.1 GET /nidp/oauth/nam/tokeninfo Host: www.idp.com:8443 Accept: */
*Authorization: Bearer
eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0NDMiLCJ0..
...
```

## Sample response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 48
Date: Thu, 19 Mar 2015 15:47:25 GMT
{
"expires_in": 145, "user_id": "alice", "scope": ["profile","email"]
}
```

## The response will have the following parameters

| Parameter | Description |
|-----------|-------------|
| expires_in | number of seconds the token is valid from now |
| user_id | user to whom the token was issued to |
| scope | list of scope values the token holds |

**NOTE:** If validation errors are occurred, HTTP Status 400 returned with the JSON response contains "error" and "error_description".

## 2.5.4.2   Validating Tokens Encrypted Using Resource Server Keys

When the access token is encrypted using the resource server keys, the resource server can validate the token without contacting Access Manager's token verification endpoint.

A resource server can validate using the following sample code written in JAVA:

### Sample Java code to validate a token

```
//Step1: decrypt the JWT Token (JWE Standard)
String jwtAccessToken =
"eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0NNIiwidHlwIjoiSldUIiwia2lkIjoibmFtLTEifQ.Zj
E0jRb5oh3suQZHFmaB-m....";
JsonWebEncryption jwe = new JsonWebEncryption();
jwe.setCompactSerialization(jwtAccessToken);
JsonWebKeySet jsonWebKeySet = new JsonWebKeySet(jwks);
List<JsonWebKey> jsonWebKeys = jsonWebKeySet.getJsonWebKeys();
JsonWebKey jsonWebkey = jsonWebKeys.stream().filter( p -
>p.getKeyId().equalsIgnoreCase(jwe.getKeyIdHeaderValue())).findFirst().orElse(json
WebKeys.get(0));
if(jsonWebkey instanceof RsaJsonWebKey){RsaJsonWebKey rsa = (RsaJsonWebKey)
jsonWebkey;
jwe.setKey(rsa.getPrivateKey());
```

```
}
else
{
jwe.setKey(jsonWebkey.getKey());
}String decryptedToken = jwe.getPlaintextString();
//Step 2: Verify the JWT Signature (JWS Standard)JsonWebKeySet jsonWebKeySet = new
JsonWebKeySet(jwks);JsonWebKey jsonWebkey =
jsonWebKeySet.getJsonWebKeys().get(0);JsonWebSignature jws = new
JsonWebSignature();
jws.setKey(jsonWebkey.getKey());
jws.setCompactSerialization(decryptedToken);
if(true == jws.verifySignature()){System.out.println("Signature is valid.");String
payload = jws.getPayload();
//}
```

For detailed sample code and tool for validating the JWT access token, refer JWT Validation tool and
Readme under Additional Resources of Access Manager documentation page.

## 2.5.4.3  Validating Unencrypted Tokens

When the token is not encrypted, the resource server can validate the token signature. In the Java
code, you can use the step 2 of the sample that is mentioned in the preceding section.

# 3 Customizing the Access Token

This section includes information about what you can perform as a resource server administrator and what you can request the Access Manager administrator for customizing the access token.

## 3.1 Adding Attributes to Token

The user's LDAP attribute or virtual attributes (LDAP attribute or constant values) can be included in the JWT access token. Talk to the Access Manager administrator to configure the access token to include these attributes by using Access Manager Administration Console.

## 3.2 Creating Custom Resource Server

You can ask the Access Manager administrator to create a custom resource server in Access Manager Administration Console to get more control on what crypto keys you require to use for encrypting the token. Access Manager provides the option to encrypt the access token as per your requirement.

The access token can be encrypted by using any of the following options:

- Encrypt using the resource server key
- Encrypt using Access manager key
- No encryption (not recommended because it may cause security issues).

After the Access Manager administrator creates the custom resource server, you can specify the resource server name in the token request for encrypting the access token using the encryption mechanism configured for that resource server. For more details about the request parameter, see Section 4.4, "Token Endpoint," on page 41. This helps in avoiding the need for contacting Identity Server's TokenInfo or UserInfo endpoints for token validation or for claims.

Only the Access Manager administrator can register the resource server.

## 3.3 Creating Custom OAuth2 Scope

Creating custom OAuth2 scope can be done using Access Manager Administrator console or using REST API. To create a scope by using REST API, you must have NAM_OAUTH2_ADMIN role. Contact the Access Manager Administrator to create required scopes.

# 4 Available Endpoints

## 4.1 Registration Endpoint

To register a client application, the HTTP method value must be POST. Identity Server uses the following endpoint for registering a client application:

```
https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients
```

The endpoint requires the following OAuth parameters for client registration or modification:

| Parameter | Required | Description |
| --- | --- | --- |
| client_name | Required | The Name of the client application |
| redirect_uris | Required | The redirection URI values used by the client application |
| application_type | Optional | Web or native |
| response_types | Optional | The following are the supported response types: <ul><li>code</li><li>code token</li><li>code id_token token</li><li>id_token</li><li>id_token token</li><li>access_token</li><li>refresh_token</li></ul> |

| Parameter | Required | Description |
| --- | --- | --- |
| grant_types<br><br>If you do not specify a grant type, the default grant type is used. The default value is authorization_code. | Optional | The following are the supported grant types:<br><br>◆ authorization_code<br>◆ implicit<br>◆ refresh_token<br>◆ password (resource owner credentials)<br>◆ client_credentials<br>◆ saml2_assertion |
| alwaysIssueNewRefreshToken | Optional | Specify true as a value to issue a new refresh token on every refresh token request. |
| authzCodeTTL | Optional | Specify the duration in minute, after how long the authorization code becomes invalid. |
| accessTokenTTL | Optional | Specify the duration in minute, after how long the Access token and ID token become invalid. |
| refreshTokenTTL | Optional | Specify the duration in minute after how long the Refresh token becomes invalid. |
| corsdomains | Optional | If you want to allow access for requests from only selected domains. Specify the domain(s) as JSON array.<br><br>For example: ["beem://www.test.com", "fb://app.local.url", "https://namapp.com"] |
| logo_uri | Optional | Specify the URL of the logo that you want to include in the consent page.<br><br>For example: https://client.example.org/logo.png |
| policy_uri | Optional | URL of the Relying Party Client's privacy policy.<br><br>For example: https://client.example.org/privacypolicy |
| tos_uri | Optional | URL of the Relying Party's terms of service.<br><br>For example: https://client.example.org/terms |
| contacts | Optional | Email addresses of people related to this client application |

| Parameter | Required | Description |
|---|---|---|
| jwks_uri | Optional | Specify the URI of the JSON file containing the json web keys. |
| | | This key set contains signing keys that the relying party uses to validate signatures from the OpenID provider. |
| | | For example: https://client.example.org/my_public_keys.jwks |
| id_token_signed_response_alg | Optional | Specify the ID Token Signed Response Algorithm. This algorithm is required for signing the ID token issued to the client |
| id_token_encrypted_response_alg | Optional | Specify the algorithm used to encrypt the key |
| id_token_encrypted_response_enc | Optional | Specify the algorithm used to encrypt the content. |

## 4.2 Metadata Endpoint

The metadata endpoint exposes the basic services and options available at Identity Server for OAuth 2.0 and OpenID Connect. This also contains URLs for the endpoints. This endpoint is in the following format:

```
https://<IDPhost:port>/nidp/oauth/nam/.well-known/OpenID-configuration
```

Invoking the endpoint URL responds with a JSON document that contains the following information:

- OAuth2.0 Endpoints
- ID Token supported algorithms
- JWKS Keys which can be used for verifying Access Token and ID token
- Client Registration Endpoint
- Scope and Resource Server registration Endpoint
- JSON Web Key Set Endpoint

**Sample Metadata Endpoint:**

```
{
  "issuer": "https://example.netiq.com/nidp/oauth/nam",
  "authorization_endpoint": "https://am-test.lab.novell.com/nidp/oauth/nam/authz",
  "token_endpoint": "https://am-test.lab.novell.com/nidp/oauth/nam/token",
  "userinfo_endpoint": "https://am-test.lab.novell.com/nidp/oauth/nam/userinfo",
  "jwks_uri": "https://am-test.lab.novell.com/nidp/oauth/nam/keys",
  "registration_endpoint": "https://am-test.lab.novell.com/nidp/oauth/nam/clients",
  "scopes_supported": [
  "phone",
  "urn:netiq.com:nam:scope:oauth:registration:read",
  "address",
  "urn:netiq.com:nam:scope:oauth:registration:full",
  "email",
  "profile",
  "openid"
  ],
  "response_types_supported": [
  "token",
  "id_token",
  "code",
  "token id_token",
  "code token",
  "code id_token token",
  "code id_token"
  ],
  "response_modes_supported": [
  "query",
  "fragment"
  ],
  "grant_types_supported": [
  "authorization_code",
  "implicit",
  "password",
  "client_credentials",
  "saml2-bearer",
  "refresh_token"
  ],
  "id_token_signing_alg_values_supported": [
  "RS256"
  ],
  "claims_supported": [
  "phone_number_verified",
  "phone_number",
  "read",
  "address",
  "add",
  "modify",
  "delete",
  "email_verified",
  "email",
  "website",
  "birthdate",
  "gender",
  "profile",
  "preferred_username",
  "given_name",
  "middle_name",
  "locale",
  "picture",
  "zone_info",
```

```
"updated_at",
"nickname",
"name",
"family_name"
],
"code_challenge_methods_supported": [
"plain",
"S256"
],
"subject_types_supported": [
"public"
],
"token_endpoint_auth_methods_supported": [
"client_secret_post"
]
}
```

# 4.3   Authorization Endpoint

Authorization endpoint is always contacted through a browser. This endpoint requires that user has existing browser session with the Identity Server. If no session exists at the time of request, the authorization endpoint redirects the user to login. This endpoint is used when the client uses the Authorization Code flow or Implicit flow.

**Endpoint URL:** https://<identity Server hostname>/nidp/oauth/nam/authz

**Request method:** Request method: GET and POST

## 4.3.1   Request Parameters

To get an authorization code, the client application should invoke a GET or a POST request to Identity Server's authorization endpoint with the following request query string parameters:

| Parameter | Required | Value | Description |
|---|---|---|---|
| client_id | Yes | | Client application ID, which is obtained at the time of client application registration. |
| response_type | Yes | code | Should be set to "code". |
| redirect_uri | Optional | | If provided, the value of this must exactly match one of the registered URIs during application registration. If not provided, the browser will be redirected to any of the registered redirect URIs registered during application registration. |
| scope | Yes | OpenID | List of scopes the application requires. It should contain "OpenID". You can get all "scopes_supported" at the authorization server's OpenID Metadata Endpoint. Scope values should be space separated %20 or +. |
| resourceServer | Optional | | Specify the registered resource server name. If this parameter is present, the authorization server will use the respective configured way to encrypt the access token. |

| Parameter | Required | Value | Description |
|---|---|---|---|
| state | Recommended | | An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter should be used to prevent cross- site forgery requests. |
| prompt | Optional | none<br><br>or<br><br>login<br><br>or<br><br>consent | The values can be "none", "login", or "consent".<br><br>With none, no user interface will be shown to user if user is not already authenticated. If not authenticated, an error message in one of "login_required", "interaction_required" or other will be sent back to client application. This is useful if client want to detect whether the user has an existing session with Identity Server or not and has necessary consents. |
| max_age | Optional | 300 | Maximum authentication age at Identity Server in seconds. If the user has not logged in within this elapsed time, the user will be re-prompted for authentication. |
| acr_values | Optional | /name/ password/ uri | If client request contains acr_values parameter, Identity Server maps the value to configured contracts in Identity Server and prompts the user with the contract if the user is not already authenticated with the contract. The contract is not sent in ID token in this release. |
| device_id | Optional | | Specify device id that token to be associated with device. |
| response_mode | Optional | form_post | Specify response_mode as form_post to receive the response parameters as HTML form values that are automatically submitted to the redirect_uri. |

## 4.3.2  Response Values

The Identity Server responds an HTTP 302 redirect message to the requested redirect_uri in the authorization request. If the request does not contain the redirect_uri parameter, Identity Server will redirect to one of the registered redirect_uri.

| Parameter | Description |
|---|---|
| code | An opaque binary token with variable length field. Application should not assume the size of the code but allocate sufficient space for reading the code. |
| state | Contains the state parameter sent in the authentication request above. |

# 4.4 Token Endpoint

Token endpoint is used directly by the client without involving the browser. Hence, it is possible to get an access token offline when the user is not connected via a browser. This endpoint can issue an access token when the client provides either a valid authorization code, SAML2 bearer profile for authorization grant flow, resource owner credentials, or client credentials.

**Endpoint URL:** https://<Identity Server hostname>/nidp/oauth/nam/token

**Request method:** POST

## 4.4.1 Request Parameters

The token request should have the following parameters:

| Parameter | Required | Description |
| --- | --- | --- |
| resourceServer | No | Registered resource server name.If this parameter is present, the authorization server will use the respective configured way to encrypt the access token. |
| grant_type | Yes | authorization_code<br><br>Use 'urn:ietf:params:oauth:grant-type:saml2-bearer' as value for this parameter. |
| assertion | Yes | (For SAML bearer grant) A single base64url encoded SAML2.0 Assertion as value for this parameter. |
| client_id | Yes | Client ID of the registered client. |
| client_secret | Optional | Client Secret of the registered client.<br><br>It is optional for native application, for web application secret is mandatory. |
| code | Yes | Code received in the Authorization code flow. |
| redirect_uri | Yes | This should be same as the one sent during the authorization code request. |
| device_id | Optional | Specify device id that token to be associated with device. |
| refresh_token | Yes | refresh_token that is obtained during authorization grant, resource owner credentials. |
| scope | Optional | List of scopes the application requires. Scope values should be separated using space (%20 or +). |

## 4.4.2 Response Values

| Parameter | Required | Description |
| --- | --- | --- |
| token_type | Yes | The type of the token. Authorization server currently supports only Bearer type. |

| Parameter | Required | Description |
| --- | --- | --- |
| access_token | Yes | Access token that can be used to invoke resource server APIs. |
| id_token | Optional if scope contains "OpenID" | When invoking authorization code request, if the client has sent OpenID, this response object will contain an ID Token. |
| scope | Optional | The list of scopes that user has authorized. This can contain all the scopes the client requested or lesser. |
| state | Optional | if the "state" parameter was present in the client authorization request, the same state value sends in response. |

**NOTE:** Ensure that you do not use the Expect: 100-Continue header in the request when using a multi-node IDP cluster setup. If the request contains this header you may experience HTTP 400 Bad Request. If you are using CURL, use "-H 'Expect:'" or do not include IDP cookies.

# 4.5 TokenInfo Endpoint

TokenInfo Endpoint is used for validating refresh token and access tokens issued in OAuth 2.0 Authorization flows. Clients can send the access token via Authorization Header. This endpoint returns a JSON response stating whether the token is valid.

**Endpoint URL:** https://<Identity Server hostname>/nidp/oauth/nam/tokeninfo

**Request method:** GET and POST

## 4.5.1 Request Parameters

The request should contain the token in the Authorization header as follows:

Authorization: Bearer access_token

## 4.5.2 Response Parameters

The response to the TokenInfo endpoint will contain the following values in JSON format:

| Parameter | Required | Description |
| --- | --- | --- |
| expires_in | Yes | number of seconds the token is valid from now |
| user_id | Yes | user to whom the token was issued to |
| scope | Yes | list of scope values the token holds |

# 4.6 UserInfo Endpoint

UserInfo Endpoint is used for getting Resource Owner's claims. A client can send a request to UserInfo endpoint with a valid access token and get the claims that are authorized by Resource Owner to share.

**Endpoint URL:** "https://<Identity Server hostname>/nidp/oauth/nam/userinfo

**Request method:** GET and POST

## 4.6.1  Request Parameters

The clients or resource servers can invoke the request to UserInfo Endpoint by including the access token in the authorization header as given below:

Authorization: Bearer access_token

## 4.6.2  Response Values

The UserInfo endpoint returns the claims associated with the access token in a JSON object as given in the response values.

| Parameter | Description |
|---|---|
| sub | Unique ID identifying the subject. This will be GUID of the user. |
| The other claims are included as values in JSON object if the access token contains the necessary scope and user has authorized the client to access the claim.<br><br>For example, if the client has requested "email" scope, the UserInfo endpoint will return following value:<br><br>"email" : "alice@c.com" along with the "sub" field. | |

## 4.6.3  Sample Request and Response

Request:

```
GET /nidp/oauth/nam/userinfo HTTP/1.1
User-Agent: curl/7.41.0
Host: www.idp.com:8443
 Accept: / Authorization:
 Bearer /wEBAA.............DSDG
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
 Content-Type: application/json
Content-Length: 73 Date:
 Thu, 19 Mar 2018 16:14:52 GMT
{
 "sub": "6adb7ca411d5a14c94946adb7ca411d5",
 "email": "alice@a.com"
}
```

## 4.7    Revocation Endpoint

Revocation endpoint is used for revoking refresh tokens and its corresponding access token.

**Endpoint URL:** https://idpbaseurl.com/nidp/oauth/nam/revoke

**Request method:** POST

### 4.7.1    Request Parameters

The request should contain the refresh token and client credentials in HTTP request parameters as mentioned in the following table:

| Parameter | Required | Description |
| --- | --- | --- |
| client_id | Yes | Client application ID that is obtained at the time of client application registration. |
| client_secret | Optional | Client secret that is obtained at the time of client application registration.<br><br>Client secret is optional for native application, for web application it is mandatory. |
| token | Yes | refresh_token that is obtained during authorization grant, resource owner credentials, client credentials flow |

### 4.7.2    Response Values

- The Identity Server responds with HTTP status code 200 OK if the token has been revoked successfully or if the client submitted an invalid token.
- The error code unsupported_token_type is returned by the Identity Server when the given token is not a refresh token.
- If the Identity Server responds with HTTP status code 503, the client must assume the token still exists and may retry revoking the refresh token after a reasonable delay.

# 5 Developer Resources

This chapter includes the information about the Access Manager OAuth Samples (https://www.netiq.com/documentation/access-manager-44-developer-documentation/resources/microfocus-am-oauth-samples.zip) that is available at Developer Documentation (https://www.netiq.com/documentation/access-manager-44-developer-documentation/) page.

The OAuth Samples zip file includes the sample scripts that can be used as a quick trial to help you understand the implementation flow.

---

**NOTE:** The use cases covered using the samples require completing tasks from the Access Manager administrator and the client developer.

---

## 5.1 APIs in Action

You can use the OAuth samples file to try various APIs to solve the OAuth use cases. These samples cover a wide range of functionalities:

- Auto configuration of Access Manager by creating scopes, resource server, registering an OAuth client
- Execute various OAuth flows and return the access token, the refresh token, and the ID token as per requirement.
- Manage tokens by refreshing a token or revoking a token.

### 5.1.1 Try Now

You can use the sample scripts as a trial to understand how the OAuth flow works with Access Manager.

#### 5.1.1.1 Prerequisite

The Access Manager administrator must perform the following tasks to enable using the sample scripts:

1. Enable OAuth protocol for Identity Server.
2. Enable `NAM_OAUTH2_DEVELOPER` and `NAM_OAUTH2_ADMIN` roles for developers.
3. Extend the user store schema and add LDAP attribute to store user's consent and refresh token information.
4. Perform the following tasks:

**Access Manager Administrator Tasks**

1 Log in to the Access Manager Administration console.

**2** Navigate to **OAuth & OpenID Connect > Global Settings**.

**3** Specify the following values:

- ◆ **Authorization Grant LDAP Attribute:** The LDAP attribute that can be used for storing the token information and the user consent.

- ◆ **CORS Domain:** Select **Allow All**

- ◆ **Grant Type(s):** Select all the options

- ◆ **Token Type(s):** Select all the options

- ◆ **Signing Certificate:** Add the signing certificate

**4** Update Identity Server.

**Client Developer Tasks**

**1** Download OAuth Samples (https://www.netiq.com/documentation/access-manager-44-developer-documentation/resources/microfocus-am-oauth-samples.zip) from the Developer Documentation (https://www.netiq.com/documentation/access-manager-44-developer-documentation/) page.

**2** From the `OAuth Samples` folder, go to `try-now-scripts`, then `sampleScripts`.

**3** Open the `config.txt` file, then specify the values for the following parameters that are mentioned within the System Settings section:

- ◆ username: OAuth developer username

- ◆ password: OAuth developer password

- ◆ user_email: OAuth developer email

- ◆ userstore: Name of the user store that is configured for the IDP cluster (check with NAM administrator)

- ◆ user_dn: Full domain name such as, `cn=admin, o=novell`.

  Change the domain as per requirement.

- ◆ idpurl: Identity server's host and port

- ◆ scope_username: Non-developer user name.

- ◆ scope_password: Password for the user mentioned in `scope_username`.

- ◆ scope_email: Email of the user mentioned in `scope_username`.

**4** Run the required script to see it in action or run sample `Scripts.sh` to run all the scripts as a batch.

| Sample | Use Case |
|---|---|
| Authorization Flow | |
| authorizationFlow-accessToken-using-defaultresourceServer.sh | Retrieve access token by using the default resource server that is configured in Access Manager. |
| authorizationFlow-accessToken-usingRefreshtoken.sh | Retrieve access token by using refresh token. |
| authorizationFlow-accessToken-using-resourceServerKey.sh | Retrieve access token that is encrypted using resource server key. |
| authorizationFlow-accessToken-with-pkcePlain.sh | Retrieve access token using pkce with the `code_challenge_method` as `plain`. |

| Sample | Use Case |
| --- | --- |
| authorizationFlow-accessToken-with-pkceS256.sh | Retrieve access token using pkce with the `code_challenge_method` as `s256`. |
| authorizationFlow-revoke-refreshToken.sh | Revoke the refresh token |
| authorizationFlow-revoke-refreshToken-using-globalDeviceID.sh | Revoke the refresh token issued to a mobile device. |
| idtoken-using-authorizationFlow.sh | Retrieve ID token |
| Client Credentials Flow | |
| clientCredentialFlow-accessToken-using-defaultResourceServer.sh | Retrieve access token by using the default resource server that is configured in Access Manager. |
| clientCredentialFlow-accessToken-using-resourceServerKey.sh | Retrieve access token encrypted using resource server key. |
| Implicit Flow | |
| idtoken-using-implicitFlow.sh | Retrieve ID token |
| implicitFlow-accessToken-using-defaultResourceServer.sh | Retrieve access token by using the default resource server that is configured in Access Manager. |
| implicitFlow-accessToken-using-resourceServerKey.sh | Retrieve access token encrypted using the resource server key. |
| Resource Owner Flow | |
| resourceOwnerFlow-accessToken-using-defaultResourceServer.sh | Retrieve access token by using the default resource server that is configured in Access Manager. |
| resourceOwnerFlow-accessToken-using-resourceServerKey.sh | Retrieve access token encrypted using resource server key. |
| resourceOwnerFlow-revoke-refreshToken.sh | Revoke refresh token. |
| resourceOwnerFlow-revoke-refreshToken-using-globalDeviceID.sh | Revoke refresh token using the device ID. |
| Manage and Validate | |
| deleteOauthResourceServer.sh | Delete the resource server configured in Access Manager |
| manageOAuthScope.sh | Manage the OAuth scopes |
| manageResourceServer.sh | Manage the resource server configured in Access manager. |
| accessToken-verification.sh | Verify access token |

# 5.2 Access Manager OAuth Playground

The User Interface to try out the OAuth flows is https://www.netiq.com/communities/cool-solutions/openid-connect-nam-identity-server-oauth2-playground/