Access Manager 5.0

OAuth Application Developer Guide

March 2021

Legal Notice

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

You can access other information resources in the library at the following locations:

Access Manager Product Documentation (https://www.microfocus.com/documentation/access-manager/index.html)

Access Manager Developer Resources (https://www.microfocus.com/documentation/access-manager/developer-documentation-5.0/)

NOTE: Contact namsdk@microfocus.com for any query related to Access Manager SDK.

---

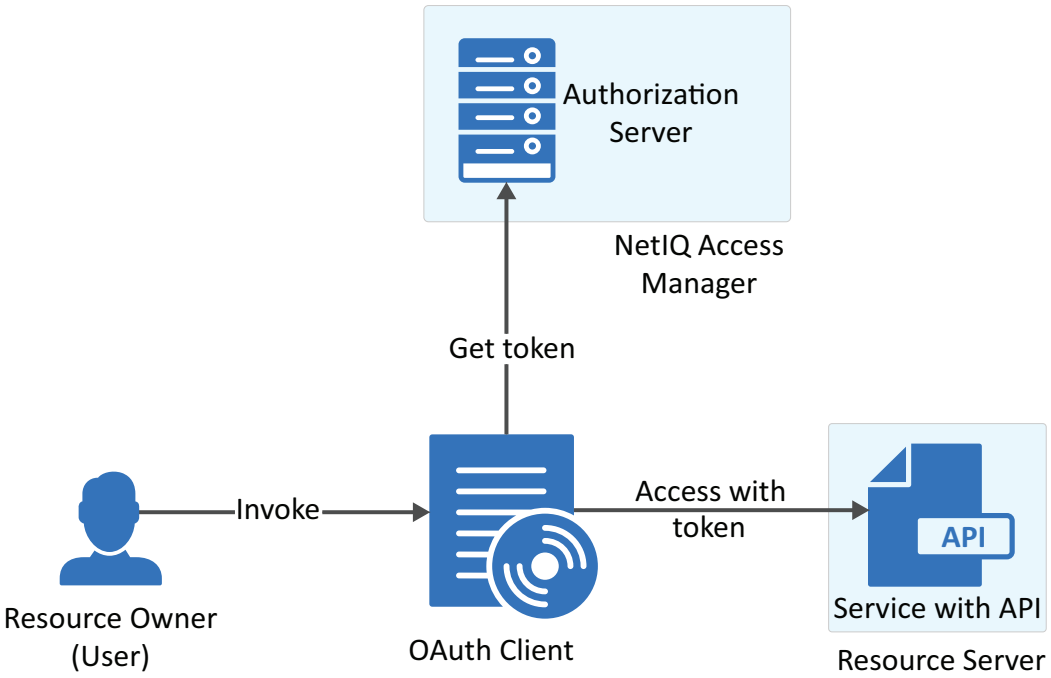
1 Getting Started

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.

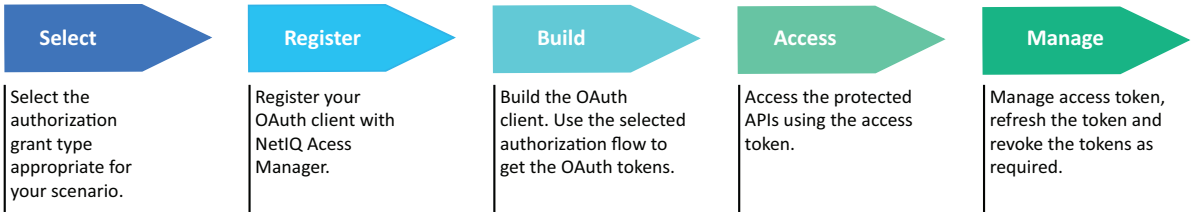
  Here, the authorization server is NetIQ Access Manager.
- **Client:** an application that requires to access the protected data on the resource server.

The simplified interaction between these participants is illustrated in the following figure:



2 Steps to Create an OAuth Client Application

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



2.1 Selecting the Authorization Grant Type

The tokens (access, refresh, and ID) 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. For more information about security requirements, see OAuth 2.0 Security Best Current Practice.

Authorization Grant	Type of Application
Authorization Code	Server-side Applications
Authorization Code with PKCE	◆ Native Applications (Mobile applications, Desktop applications) ◆ Web Applications
Implicit	◆ Single-page applications, where navigation between different screens of the website can be performed without loading different web page in the browser. For example, Gmail. ◆ Applications that run on the user's device, such as mobile apps. ◆ Web applications that do not require high security. Applications that require high-security use the authorization code flow.
Resource Owner Password Credentials	◆ Highly trusted applications or the applications that are owned by the service itself, such as a mobile application. ◆ Legacy application migrating to OAuth.

Authorization Grant	Type of Application
Client Credentials	◆ Headless clients ◆ Batch processing scripts
SAML 2.0 Bearer Grant	Applications that already have the SAML assertions and require to access the OAuth-protected resources.

2.2 Registering an OAuth Client Application

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

- Section 2.2.1, “Requirements,” on page 10
- Section 2.2.2, “Registering an OAuth Client Application,” on page 11

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 OAuth **Global Settings**, Identity Server does not issue the refresh token even when you register the application using the token type as a 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>`

  In addition, you must have one of the following roles:

  NAM_OAUTH2_DEVELOPER: This allows the user to view and modify the client registration details of the applications that the user has registered on the portal.

  NAM_OAUTH2_ADMIN: This allows the user to view and modify the client registration details of all the client applications that are registered with Access Manager.
- The client application redirection URIs, where Identity Server can send the tokens.

  NOTE: The `urn:ietf:wg:oauth:2.0:oob` redirect URI is supported only for the authorization code flow.

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

2.2.2 Registering an 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, "Selecting the Authorization Grant Type" on page 9.

     ---
  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

    The value, `urn:ietf:wg:oauth:2.0:oob` is supported only for the authorization code flow.
  - **grant_types:** select the grant types based on the OAuth flow that you have decided.

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

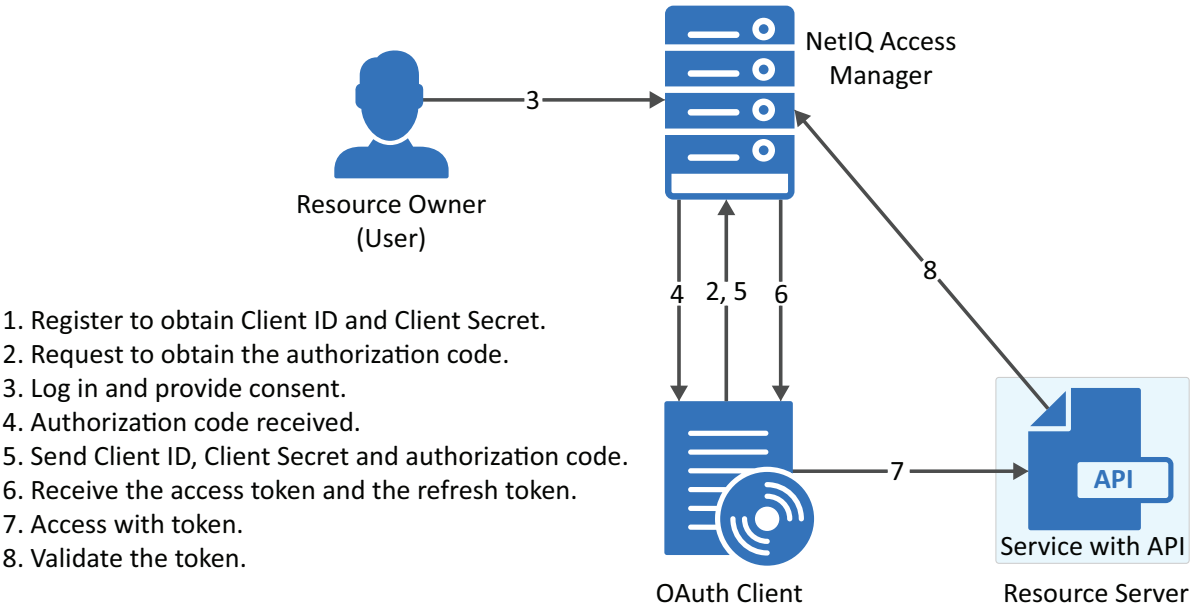
After the registration is successful, you will receive the client ID and the client secret. Using these, you can initiate the OAuth flow.

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.

In this Section

2.3.1 Authorization Code



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.

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 values are “none”, "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. 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. 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 49.

NOTE: The authorization code flow does not support the basic authorization header in an authorization request. However, this flow supports the basic authorization header in an access token request.

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.
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 52.

Sample Response

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

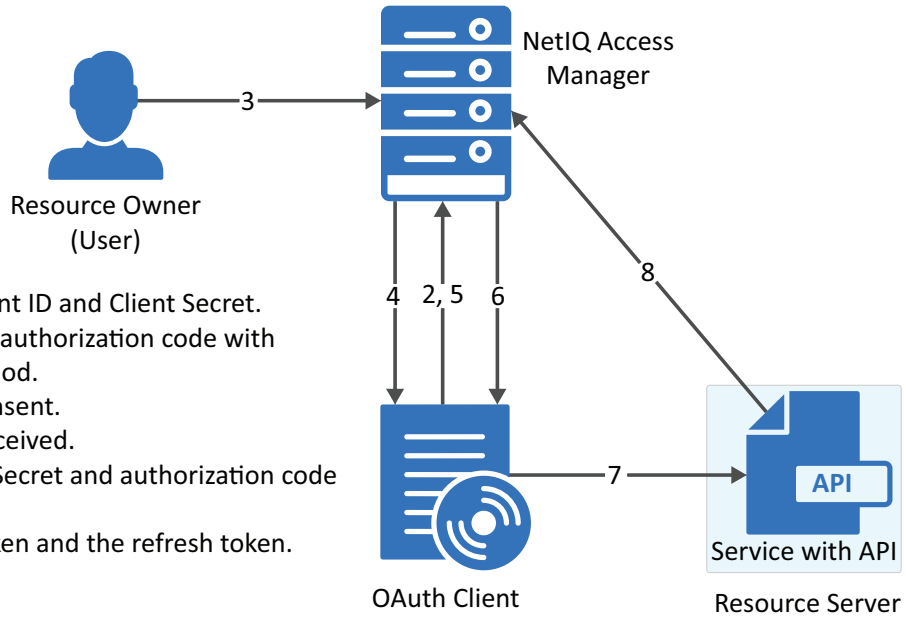
Response Parameter	Description
token_type	Authorization server currently supports only "Bearer" token type
access_token	Access token that can be used to invoke resource server APIs
id_token	When invoking authorization code request, if the client has sent OpenID in scope parameter, this response object 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 is sent 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 Developer Resources.

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 not registered, register the client application. 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

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

```
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`. You can specify the value as `plain` or `S256`.

This table lists the minimal set of parameters. For the complete list, see Authorization Endpoint.

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

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, Identity Server redirects 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 are 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.
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	This 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 52.

Sample Response

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

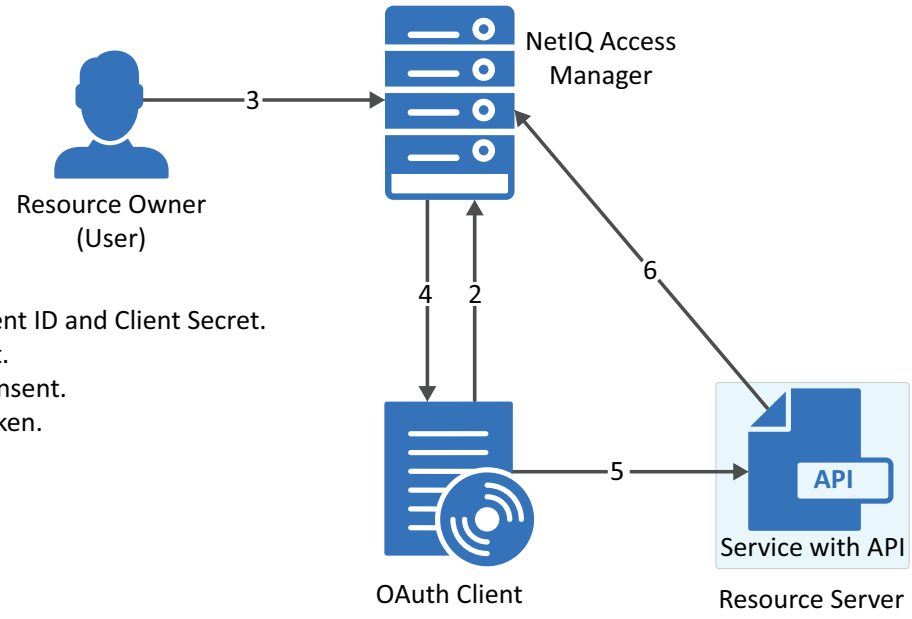
Response Parameter	Description
token_type	Authorization server currently supports only "Bearer" token type.
access_token	Access token that can be used to invoke resource server APIs.
id_token	When invoking authorization code request, if the client has sent OpenID in scope parameter, this response object 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 Developer Resources.

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 “none”, "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, Section 4.3.1, “Request Parameters,” on page 50.

NOTE: The Basic Authorization header is not supported for the implicit flow.

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.
.....&expires_in=3600&id_token=eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJo&scope=ema
il&state=s1234
```

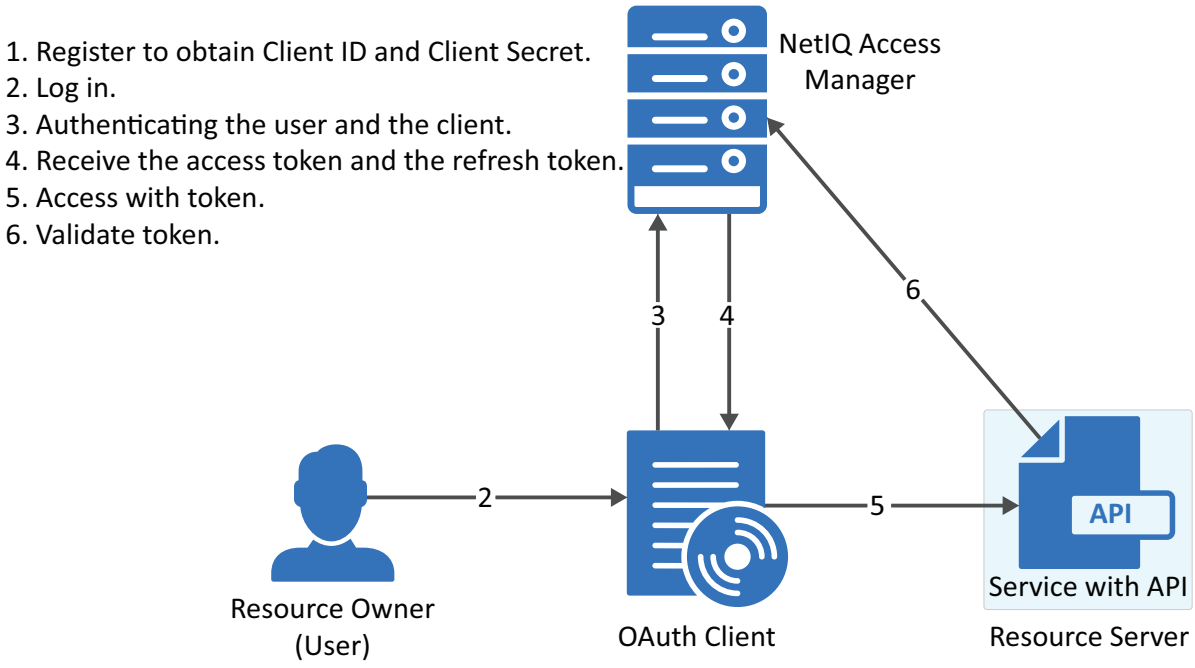
Response Parameter	Description
token_type	The type of the token. Authorization server currently supports only Bearer type
access_token	Based on response_type value sent in the request. If the response_type value is either "token" or "token id_token" then authorization server returns access token
id_token	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 Developer Resources.

2.3.4 Resource Owner



2.3.4.1 Register the Client Application

If not registered, register the client application. 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'
```

NOTE: ◆The authentication is done by using `client_id` and `client_secret` in the request body parameters (as mentioned in the preceding curl request), or send client credentials in basic authorization header (as mentioned in RFC 6749).

- The authentication contracts for Resource Owner Credential can be specified in the `acr_values` request parameter. The value of `acr_values` must be URI encoded and match exactly with the Access Manager Authentication contract URI. If no `acr_values` and no global RO authentication contracts are configured, only the Identity Server's default authentication contract is executed. For more information, see **Contracts for Resource Owner Credentials Authentication** in Defining Global Settings.

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 52.

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.

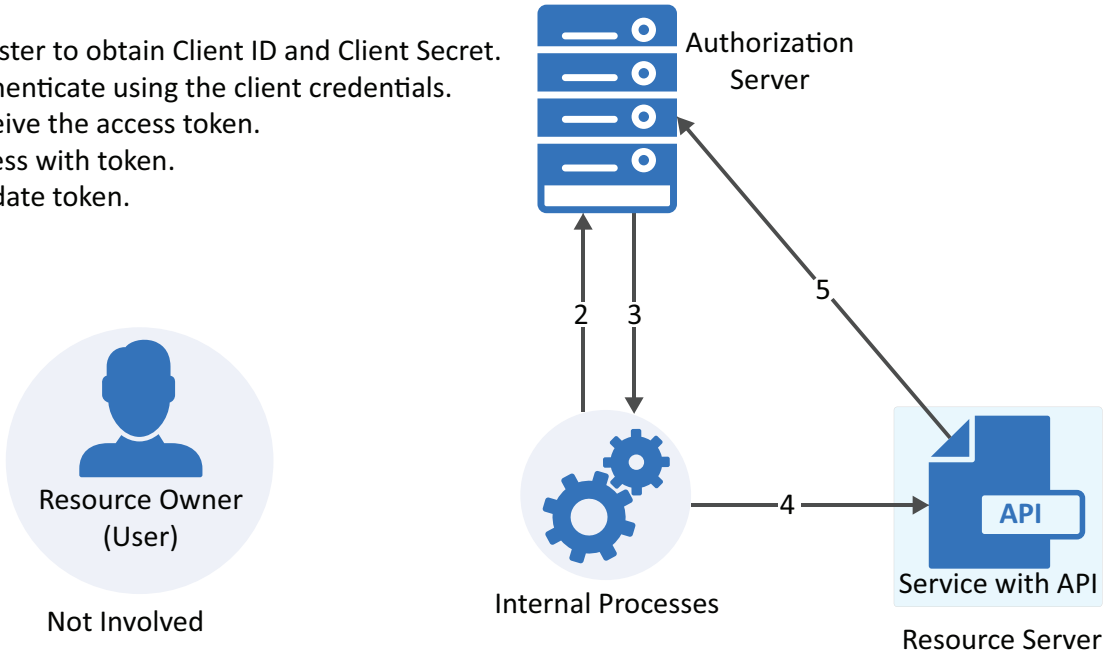
NOTE: If validation errors are occurred, HTTP Status 400 returned with the JSON response contains “error” and “error_description”.

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 63.

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.



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'
```

NOTE: The authentication is done by using `client_id` and `client_secret` in the request body parameters (as mentioned in the preceding curl request), or send client credentials in basic authorization header (as mentioned in RFC 6749).

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 Creating Custom OAuth2 Scope.

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 63.

2.3.6 OIDC Front-Channel Logout

Access Manager supports OIDC front-channel logout and is implemented as per OIDC specification. This feature enables the following two forms of logout request:

- **Identity Provider initiated logout request:** Allows a user to log out from all the client applications when the user logs out at Identity Server.
- **Relying Party (client application) initiated logout request:** When a user initiates logout from one client application, the user can authorize to log out from Identify Server and other logged-in applications. For more information, see OIDC RP-Initiated Logout.

Consider a scenario where multiple employees share a single device. A user named Alice logs in to the User Portal, and then accesses five client applications. At the end of her shift, she needs to log out of these applications so that Bob, another employee, can use the same device. Instead of logging out from each application individually, she can log out from the user portal. This logs her out from all five applications.

Alternatively, when she initiates logout from one client application, the client application initiates logout from Identity Server. Identity Server then triggers logout for the remaining four applications. A consent message is displayed to verify if she wants to log out from all applications. When she authorizes, she is logged out of Identify Server and the active applications.

The consent message is displayed when the administrator enables the **Require Logout Consent** option in **Global Settings**. By default, this option is enabled.

As a client application developer, you must register the `logout_redirect_uri` parameter for OIDC logout to enable this feature.

You must also have the details of the following endpoints. In the Administration Console Dashboard, click **Devices** > **Identity Servers** > **Edit** > **OAuth & OpenID Connect** > **EndPoint Summary** to view the details of the endpoints.

- Logout EndPoint
- OpenID Metadata EndPoint

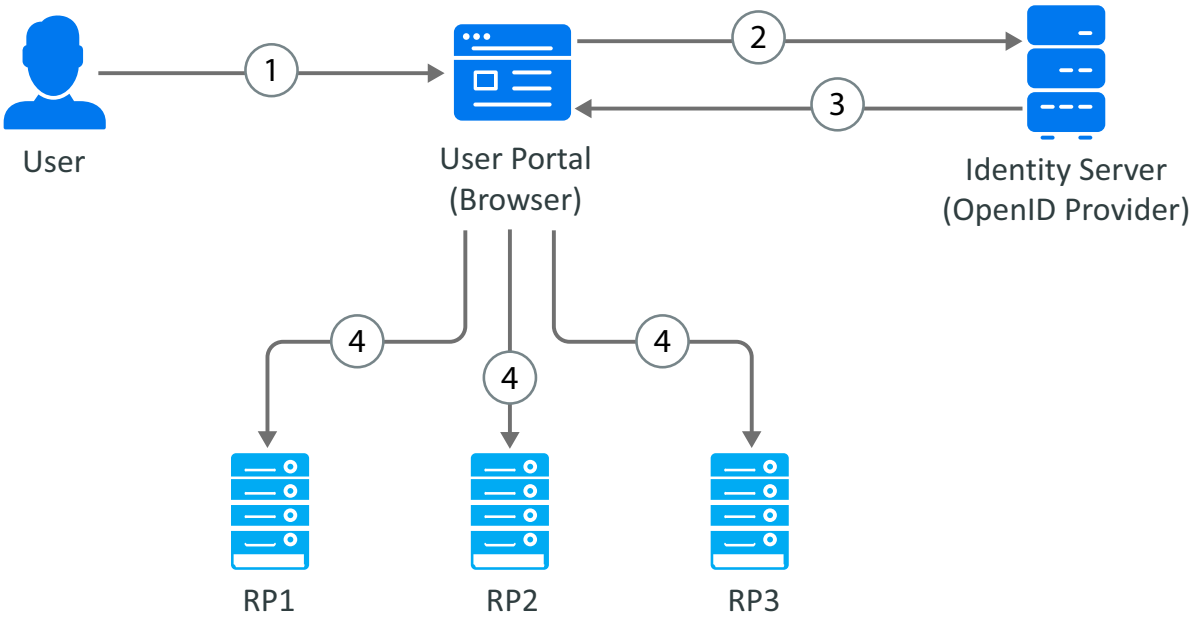
To configure the front-channel logout option, see Step 5 of Registering OAuth Client Applications (https://www.microfocus.com/documentation/access-manager/5.0/admin/b1dj6b2f.html#oauth_add_clients).

To enable logout from multiple browsers, you must enable the **Allow multiple browser session logout** option under **Devices** > **Identity Servers** > **Edit** > **General** > **Configuration** > **Limits**.

You can also revoke the refresh tokens issued during the front-channel logout for the user session using the **Logout to revoke tokens** option. By default, this option is disabled. For more information, see Defining Global Settings (https://wwwtest.microfocus.com/documentation/access-manager/5.0/admin/b1dj6b2f.html#oauth_global_settings).

The following diagram illustrates the workflow of the Identity Provider initiated logout request:

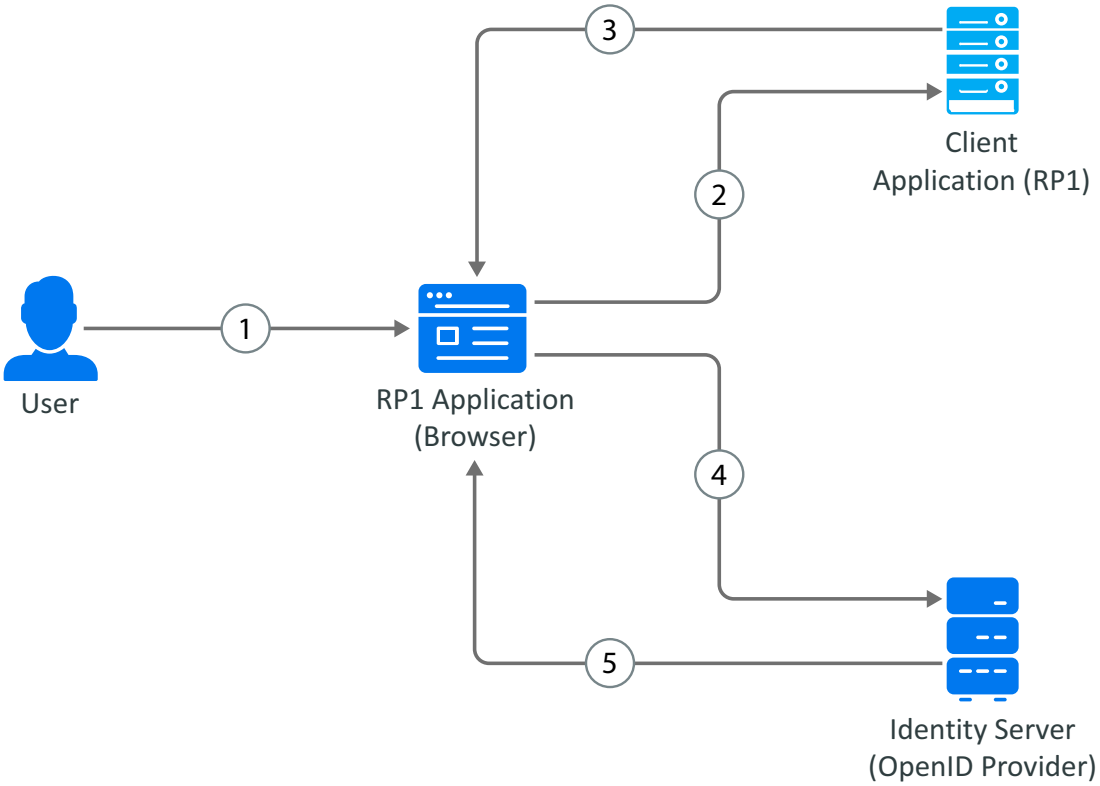
Figure 2-1 *Identity Provider Initiated Logout Request*



1. A user initiates logout at the Identity Server User portal.
2. The user agent sends the logout request to Identity Server.
3. Identity Server returns the logout response. The response contains iframes (logout URLs) for each relying party (RP1, RP2, and RP3). The relying party can be OAuth 2.0 or OIDC client application or SAML2 client application.
4. The browser agent generates responses and makes calls to each relying party's logout URL.

The following diagram illustrates the workflow of the Relying Party (client application) initiated logout request:

Figure 2-2 *Relying Party Initiated Logout Request*

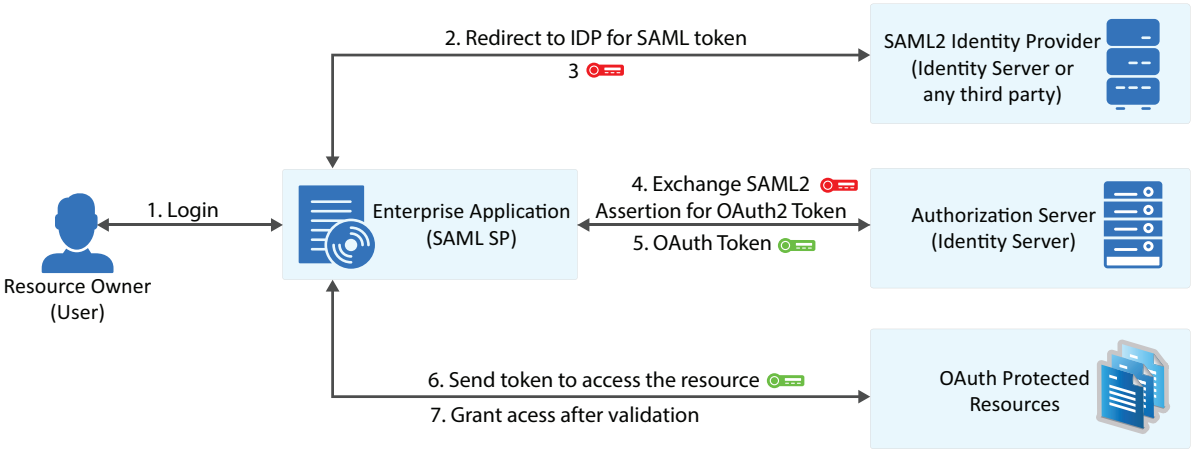


1. A user initiates logout at the client application.
2. The user agent sends the logout request to the client application.
3. The client application redirects the user's browser agent for Identity Server logout.
4. The browser agent sends the logout request to Identity Server.
5. After the user grants consent, Identity Server terminates the user session, sends logout requests to other client applications (OAuth 2.0 or OIDC client application or SAML2 client application), and optionally redirects to the client application's post-logout page.

Try Now

To view how the OIDC front-channel logout works, you can use the Access Manager OAuth sample scripts. For more information about the sample scripts, see Developer Resources.

2.3.7 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.

2.3.7.1 Register the Client Application

If not registered, register the client application 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.

2.3.7.2 Exchange SAML 2.0 Assertion for Access Token

SAML 2.0 assertions can be exchanged for access token. The consent page is 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'
```

NOTE: The authentication is done by using `client_id` and `client_secret` in the request body parameters (as mentioned in the preceding curl request), or send client credentials in basic authorization header (as mentioned in RFC 6749).

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 with %20 or +

2.3.7.3 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 must include the access token when invoking any OAuth protected API service. The API server validates this token and authorize the incoming API requests based on the scopes embedded in the access token. For information about validating the tokens, see Validating a JWT Token.

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 and developer can perform the following to manage the tokens:

- Section 2.5.1, “Using Refresh Token,” on page 30
- Section 2.5.2, “Revoking Tokens,” on page 31
- Section 2.5.3, “Revoking Token Issued to a Device,” on page 32
- Section 2.5.4, “Validating a JWT Token,” on page 33

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 Registering an OAuth Client Application.

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 52.

2.5.1.2 Receiving Access Token

Sample token response

```
{"access_token":
"eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0N
DMiLCJ0.....","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=eyJraWQiOiI0MjgzNzQyNDYxMjE5OT
M1ODU5OTYyOD'
```

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
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 the JWKS format (JSON Web Key Set) or JWKS URI to Access Manager Administrator for creating custom resource server using REST API or Administration Console.

- Section 2.5.4.1, “Validating Tokens Encrypted Using the Access Manager keys,” on page 33
- Section 2.5.4.2, “Validating Tokens Encrypted Using Resource Server Keys,” on page 35
- Section 2.5.4.3, “Validating Unencrypted Tokens,” on page 35

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 TokenIntrospect endpoint, `/introspect`.

Sample request

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

Authorization: `Bearer access_token`

```
curl --request POST \

--url https://<idphost:port>/nidp/oauth/v1/nam/introspect \
--header 'content-type: application/x-www-form-urlencoded;Authorization: Bearer
eyJraWQiOiI0MjgzNzQyNDYxMjE5OTM1ODU5OTYyODYwNzYyODAzMzEyNjI1MjUzMDQyMTk0NDMiLCJ0..
.. \'
--data 'token=eyJhbGciOiJBMT........_.xvpQmbtb-hzR6TAs3dOm7A
```

Sample response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 296
Date: Thu, 19 Mar 2015 15:47:25 GMT
{
"active": true,"exp": 1536731362,"nbf": 1536727762,"iat": 1536727762,"scope":
[],"token_type": "bearer","jti": "9b7d2252-cb1d-4e41-9155-97fe2f3dc85a","iss":
"https://<Identity Server hostname>/nidp/oauth/nam","aud": "21ae0625-f319-4b24-
bf52-b2fc7e2a866a","client_id": "21ae0625-f319-4b24-bf52-b2fc7e2a866a"
}
```

The response will have the following parameters

Parameter	Description
active	A boolean value indicating whether the mentioned token in the introspection request is currently active.
client_id	The client identifier for the OAuth 2.0 client that requested this token.
exp	The timestamp in seconds indicating when the token will expire.

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:

```
//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 sample code and tool for validating the JWT access token, see JWT Validation tool and Readme.

2.5.4.3 Validating Unencrypted Tokens

When a token is not encrypted, the resource server validates the token signature. In the Java code, use step 2 of the sample mentioned in Validating Tokens Encrypted Using Resource Server Keys.

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 an 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 Token Endpoint. 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

4.1.1 Registering a Client Application

To register a client application, the HTTP method value must be POST.

The client registration endpoint is protected by OAuth 2.0. Therefore, invoking this endpoint must include an access token as a bearer token or if you are using curl, log in using the username and password. Also, the user account must have the NAM_OAUTH2_DEVELOPER role. Contact the Access Manager administrator to get an OAuth 2 developer account. If you are registering a client using an access token, get client application details from the Access Manager administrator.

To obtain an access token for registering a client, you can use any one of the OAuth flows (resource owner, authorization code, or implicit flow). The endpoint of the resource owner flow is

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

This endpoint requires the followings parameters to provide an access token for registering a client application:

Parameter	Value
grant_type	password
username	Application developer's user name
password	Application developer's password
scope	urn:netiq.com:nam:scope:oauth:registration:full (This scope allows you to register, view, modify, and delete client applications.) urn:netiq.com:nam:scope:oauth:registration:read (This scope provides read-only access)
token endpoint	Identity Server URL: Port Number>/ nidp/oauth/nam/token

Identity Server uses the following endpoint for registering a client application:

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

Headers:

- Authorization bearer eyJhbGciOiJBMTI4S1ciL... (Login or access token in the authorization header is required.)
- Content-Type application/json

This endpoint requires the followings OAuth parameters for registering or modifying a client application:

Parameter	Required	Description
client_name	Required	The name of the client application.
redirect_uris	Required	The redirection URI values used by the client application
enabled	Optional	The boolean value true or false. The default value of this parameter is true. This parameter is optional, when the value is not specified, the default value would be considered.
application_type	Optional	Web or native
response_types	Optional	JSON string array of any of the following values are the supported response types: - code - code token - id_token Sample supported values: ["code","id_token","token"] or ["code","id_token"] or ["code"]

Parameter	Required	Description
tokenFormat	Optional	By default, the token format is set to default. **NOTE:** When an administrator changes the format, the changed format will be seen only for the newly issued tokens. The following are the supported token formats: ◆ **default:** The default format is controlled by an administrator. The administrator can set the default format globally for a specific Identity Server (Authorization server). If the administrator has not set the format, then JWT is the default format. Whenever the Access Manager administrator changes the token format globally for a specific Identity Server, the default format also changes to the same for the registered client applications. ◆ **binary:** The Binary option is recommended only if you have an existing client application that cannot use the jwt format because of the browser restrictions for the length of the parameter values. This value will not change when the Access Manager administrator changes the token format globally for a specific Identity Server. ◆ **jwt:** This is the recommended option. The token format will always be jwt even when the administrator changes the format in the global settings of Identity Server (authorization server).
grant_types 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 values for grant_types: ◆ authorization_code ◆ implicit ◆ refresh_token ◆ password (resource owner credentials) ◆ client_credentials ◆ saml2-bearer
alwaysIssueNewRefreshToken	Optional	Specify true as a value to issue a new refresh token on every refresh token request.
authzCodeTTL	Optional	Specify the duration in minutes after how long the authorization code becomes invalid.
accessTokenTTL	Optional	Specify the duration in minutes after how long the Access token and ID token become invalid.
refreshTokenTTL	Optional	Specify the duration in minutes 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. For example, [“beem://www.test.com”, “fb://app.local.url”, “https://namapp.com”]
logo_uri	Optional	Specify the URL of the logo to include in the consent page. For example, https://client.example.org/logo.png

Parameter	Required	Description
policy_uri	Optional	URL of the Relying Party Client's privacy policy. For example, https://client.example.org/privacypolicy
tos_uri	Optional	URL of the Relying Party's terms of service. For example, https://client.example.org/terms
contacts	Optional	Email addresses of people related to this client application
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 as a string. This algorithm is required for signing the ID token issued to the client. The following are supported values: ◆ none ◆ RS256 ◆ RS384 ◆ RS512
id_token_encrypted_response_alg	Optional	Specify the algorithm used to encrypt the key as a string. The following are supported values: ◆ RSA1_5 ◆ A128KW
id_token_encrypted_response_enc	Optional	Specify the algorithm used to encrypt the content as a string. The supported value is A128CBC-HS256.
frontchannel_logout_uri	Optional	Specify the client application's logout URL that will cause the client application to log itself out when rendered in an iframe by Identity Server (Authorization server).
frontchannel_logout_session_required	Optional	The boolean value. Specify true if the client application requires that a `sid` (Session ID) query parameter be included to identify the session with Identity Server (Authorization Server) when the frontchannel_logout_uri is used. If omitted, the default value is false. For example, register the `frontchannel_logout_uri` value as `https://client.example.org/fc_logout` and `frontchannel_logout_session_required` are false, then Identity Server causes the front-channel logout to occur by rendering the `https://client.example.org/fc_logout` URL in an iframe. Similarly, when `frontchannel_logout_session_required` is true, Issuer and Session ID values are included as query parameters: `https://client.example.org/fc_logout?iss=https://idp.server.com&sid=LDtAIRsTGdW6Fyhdi`

Parameter	Required	Description
post_logout_redirect_uris	Optional	Specify an array of URLs supported by the client application to which the end user's browser will be redirected after a successful logout at Identity Server (Authorization Server). For example, `["https://client.example.org/callback","https://client.example.org/callback2"]` **NOTE:** Identity Server redirects the end-user to the registered logout redirect URI only when the client application sends the front-channel logout request with an `id_token_hint` parameter. If `id_token_hint` is omitted in the logout request, an error response is returned.
contracts	Optional	Specify an array of contract URIs assigned to a client application. For example, `["name/password/uri","basic/name/password/uri"]` **NOTE:** When you configure authentication contracts for a client application, the ACR value in the request is ignored and the first contract in the list will be used for authentication with the Authorization endpoint. In the case of Resource Owner Credentials flow, it will consider the first Resource Owner flow supported contract to authenticate by traversing the list of contracts.
scope	Optional	Specify an array of scope assigned to a client application. For example, `["profile","email"]` **NOTE:** When you configure scopes for a client application, only those scope value in the authorization/token endpoint request would be accepted if they are configured for requested client application. To know the list of scopes supported by the NAM, the openid connect metadata endpoint (`https://<IDPhost:port>/nidp/oauth/nam/.well-known/OpenID-configuration`) can be used.

Sample Request and Response

- Sample Request
- Sample Response

Sample Request

```
POST /nidp/oauth/nam/clients HTTP/1.1
Host: www.idp.com:8443
Content-Type: application/json
Authorization: Bearer
eyJhbGciOiJBMTI4S1ciL.....ihoypluZPQwvE0VcehrubnTh_NjCCcJuqNh2OQ.n1FHCm7WF
kTrOI6oMsLqnQ
Cookie: JSESSIONID=AA47446E669A51A686F01A5D44F662A9
Content-Length: 1804
```

```
{
  "application_type": "web",
  "logo_uri": "https://client.example.org/logo.png",
  "registration_client_uri": "https://www.idp.com:8443/nidp/oauth/nam/
clients/abc7893f-2d6d-4db7-ad4a-4612c033f8aa",
  "enabled": true,
  "token_endpoint_auth_method": "client_secret_basic",
  "client_id": "abc7893f-2d6d-4db7-ad4a-4612c033f8aa",
  "corsdomains": [
    "insurance.novell.com",
    "client.example.org"
  ],
  "Version": "5.0",
  "token_format": "jwt",
  "authzCodeTTL": 3,
  "accessTokenTTL": 10,
  "client_secret":
"6VzPz9beQiswUHTs22TgRbEhSg89Pjgk2p6LJEKYvyTFT4LuuchbkkaKbnNInFrSRGXbivorX
qvcQzZkqZIjtw",
  "client_id_issued_at": 1625571581465,
  "client_name": "InsuranceMobileApp",
  "policy_uri": "https://client.example.org/privacypolicy",
  "id_token_signed_response_alg": "RS256",
  "post_logout_redirect_uris": [
    "https://insuranceapp.novell.com/"
  ],
  "subject_type": "pairwise",
  "grant_types": [
    "authorization_code"
  ],
  "redirect_uris": [
    "https://client.example.org/callback",
    "https://client.example.org/callback2"
  ],
  "frontchannel_logout_session_required": true,
  "frontchannel_logout_uri": "https://insuranceapp.novell.com/logout",
  "id_token_encrypted_response_alg": "RSA1_5",
  "refreshTokenTTL": 1440,
  "id_token_encrypted_response_enc": "A128CBC-HS256",
  "client_secret_expires_at": 1625657981465,
  "jwks_uri": "https://client.example.org/my_public_keys.jwks",
  "tos_uri": "https://client.example.org/terms",
  "contacts": [
    "ve7jtb@example.org",
    "mary@example.org"
  ],
  "response_types": [
    "code",
    "id_token",
    "token"
  ]
}
```

Sample Response

```
HTTP/1.1 201
Content-Type: application/json
Status Code: 201 201

{
  "application_type": "web",
  "logo_uri": "https://client.example.org/logo.png",
  "registration_client_uri": "https://sangeetha1.blr.novell.com/nidp/
oauth/nam/clients/3c0beb03-f370-4b07-9ae2-31282b75948e",
  "enabled": true,
  "token_endpoint_auth_method": "client_secret_basic",
  "client_id": "3c0beb03-f370-4b07-9ae2-31282b75948e",
  "corsdomains": [
    "insurance.novell.com",
    "client.example.org"
  ],
  "Version": "5.0",
  "token_format": "jwt",
  "authzCodeTTL": 3,
  "accessTokenTTL": 10,
  "client_secret":
"I5GZ5n9rRxMDHJ3VKHFEFl8uBC0Q8RXvHZqhjvRP287t5PY1KovC0e2Si2KrNmCY1uZ2mIM3o
4kTz3TIEaT6fQ",
  "client_id_issued_at": 1625572906389,
  "client_name": "InsuranceMobileApp",
  "policy_uri": "https://client.example.org/privacypolicy",
  "id_token_signed_response_alg": "RS256",
  "developerDn": "admin",
  "post_logout_redirect_uris": [
    "https://insuranceapp.novell.com/"
  ],
  "subject_type": "pairwise",
  "grant_types": [
    "authorization_code"
  ],
  "redirect_uris": [
    "https://client.example.org/callback",
    "https://client.example.org/callback2"
  ],
  "frontchannel_logout_session_required": true,
  "frontchannel_logout_uri": "https://insuranceapp.novell.com/logout",
  "id_token_encrypted_response_alg": "RSA1_5",
  "refreshTokenTTL": 1440,
```

```
  "id_token_encrypted_response_enc": "A128CBC-HS256",
  "client_secret_expires_at": 1625659306389,
  "jwks_uri": "https://client.example.org/my_public_keys.jwks",
  "tos_uri": "https://client.example.org/terms",
  "contacts": [
    "ve7jtb@example.org",
    "mary@example.org"
  ],
  "response_types": [
    "code",
    "id_token",
    "token"
  ]
}
```

NOTE: The response will contain client_id and client_secret. Make a note of these values to use in the token request.

Error Response

- When a client name is already registered with Identity Server, trying to register it again throws the HTTP 400 error with the following response:

  {"error_description":"The client already exists. Please contact administrator","error":"invalid_client"}

- When the request does not to include a mandatory parameter, redirect_uri throws the HTTP 400 error with the following response:

  {"error_description":"Could not find client metadata field redirect_uris","error":"invalid_client_metadata"}

- When an unsupported ID token signing algorithm is sent in the request, it throws the HTTP 400 error with the following response:

  {"error_description":"Unsupported ID token signing algorithm. Supported algorithms are {none / RS256 / RS384 / RS512}","error":"unsupported_idtoken_signing_algo"}

4.1.2 Modifying a Client Application

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

Headers:

- Authorization bearer eyJhbGciOiJBMTI4S1ciL... (A login or access token in authorization header is required.)
- Content-Type application/json

Perform the following steps:

1. Retrieve the client details from the `https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients/<client ID>` endpoint. In the request for retrieving client details, use GET as the HTTP method value.
2. Send the update request. In the update request, use POST as the HTTP method value. Identity Server uses the following endpoint for modifying a registered client application:

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

   For the list of parameters that this endpoint requires for a client application modification, see the table in Registering a Client Application.

NOTE: For updating a client application, you must send the complete XML with all parameters during the update request. If you do not include a parameter in the update XML, the server does not initialize this parameter. For example, if you want to update the response_types parameter, send the updated value for this parameter and existing values for other parameters in the request.

4.1.3 Viewing a Client Application

To view a client application, use GET as the HTTP method value. A login or access token in the authorization header is required to view a client application.

You can view a registered client application by using the following two endpoints:

- https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients/: To view all clients applications registered by a developer.
- https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients/<client ID>: To view a specific client application registered by a developer.

4.1.4 Deleting a Client Application

To delete a client application, the HTTP method value must be DELETE.

Identity Server uses the following endpoint for deleting a registered client application:

Endpoint URL: `https://<Identity Server URL: Port Number>/nidp/oauth/nam/clients/<client ID>`Headers:

Authorization bearer eyJhbGciOiJBMTI4S1ciL... (A login or access token in the authorization header is required.)

4.2 Metadata Endpoint

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

`https://<Identity Server URL: Port Number>/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
- Supported response_types
- Supported response_modes
- Supported token_endpoint_auth_methods
- Supported revocation_endpoint_auth_methods
- Supported introspection_endpoint_auth_methods_supported
- Supported Front Channel Logout

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",
"end_session_endpoint": "https://am-test.lab.novell.com/nidp/oauth/v1/nam/
end_session",
"revocation_endpoint": "https://am-test.lab.novell.com/nidp/oauth/nam/revoke",
"introspection_endpoint": "https://am-test.lab.novell.com/nidp/oauth/v1/nam/
introspect",
"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",
"none"
],
"frontchannel_logout_supported": true,
"frontchannel_logout_session_supported": true,
"response_modes_supported": [
"query",
"fragment",
"form_post"
],
"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",
"client_secret_basic"
],
"revocation_endpoint_auth_methods_supported": [
"client_secret_post",
"client_secret_basic"
],
"introspection_endpoint_auth_methods_supported": [
"client_secret_post",
"client_secret_basic",
"bearer"
]
}
```


4.3 Authorization Endpoint

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

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

Request method: Request method: GET and POST

NOTE: The basic authorization header is not supported for this endpoint.

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		The following values are supported: ◆ code ◆ token ◆ id_token ◆ code token ◆ code token id_token
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. The scopes are case-sensitive. The request must contain the scope "openid" to get id_token, otherwise the scope parameter is optional. 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. For more information, see Creating Custom Resource Server.
state	Recommended		An opaque value used by the client to maintain the 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.

Parameter	Required	Value	Description
prompt	Optional	none or login or consent	The values can be "none", "login", or "consent". With none, no user interface will be shown to the user if the 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's application. This is useful if the client want to detect whether the user has an existing session with Identity Server or not and has necessary consents. For more information, see Defining Scopes for a Resource Server.
login_hint	Optional		This parameter takes the username from the client application and auto-populates the Access Manager login form. For example, if you specify `login_hint=abc@xyz.com`, then when the user is redirected to the Access Manager login page, the username field is populated as `abc@xyz.com`.
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. This contract is also sent in the ID Token after authentication.
device_id	Optional		Specify the device ID that token to be associated with the device.
response_mode	Optional	query/ fragment/ form_post	Specify `response_mode` to receive response parameters that are required for your client applications. For more information about this parameter and its values, see Response Modes in OAuth 2.0 Multiple Response Type Encoding Practices (https://openid.net/specs/oauth-v2-multiple-response-types-1_0.html). **NOTE:** If `response_mode` parameter is unavailable, default response_mode is used, which is based on the response_type parameter value as available in the Response Modes specs.

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 the variable length field. The 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.

NOTE: If you are using CURL, add `-H 'Expect:'` in the header of the token request.

4.4 Token Endpoint

The 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 URL: Port Number>/nidp/oauth/nam/token`

Request method: POST

NOTE: The authentication is done by using `client_id` and `client_secret` in the request body parameters (as mentioned in the preceding curl request), or send client credentials in the basic authorization header (as mentioned in RFC 6749).

4.4.1 Request Parameters

The token request must 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	Use 'authorization_code' for authorization code exchange for a token. Use 'urn:ietf:params:oauth:grant-type:saml2-bearer' as value to exchange SAML token for OAuth token (SAML bearer grant) Use 'password' for Resource Owner Password Credential grant.
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.

Parameter	Required	Description
client_secret	Optional	Client Secret of the registered client. It is optional for a native application and mandatory for a web application.
code	Yes	Code received in the Authorization code flow.
redirect_uri	Yes	This should be the same as the one sent during the authorization code request.
device_id	Optional	Specify the device ID that token to be associated with the 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 +).
acr_values	Optional	This parameter is supported only for grant_type=password (Resource Owner Password Credentials Grant). The value of acr_values should be URI encoded and must match exactly with the Access Manager Authentication contract URI. If no acr_values and no global ROPC authentication contracts are configured, then only the default authentication contract of Identity Server is executed. For more information, see the 'Contracts for Resource Owner Credentials Authentication'. Sample value can be `'/name/password/uri'`.

4.4.2 Response Values

Parameter	Required	Description
token_type	Yes	The type of the token. Authorization server supports only Bearer type.
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 to not use the Expect: 100-Continue header in the request when using a multi-node Identity Server 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 (Deprecated)

The TokenInfo Endpoint is used for validating refresh tokens 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 URL: Port Number>/nidp/oauth/nam/tokeninfo`

Request method: GET and POST

NOTE: This endpoint is deprecated. It is recommended to use the Token Introspect endpoint, `/introspect`. For information about `/introspect`, see Token Introspect Endpoint.

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 Token Introspect Endpoint

The token introspect endpoint is used for determining the active state and the meta-information of an OAuth 2.0 token. Only an authorized protected resource can query an OAuth 2.0 authorization server for token introspection.

This endpoint returns a JSON response, which is implemented based on RFC 7662.

Endpoint URL: `https://<Identity Server URL: Port Number>/nidp/oauth/v1/nam/introspect`

Request method: POST

4.6.1 Request Headers

You must use any one of the following methods to authenticate a request:

- Authorization header with any of the following values:
  - **For `client_secret_basic`:** Basic `Base64Encode of(<client_id>:<client_secret>)`
  - **For `bearer`:** Bearer <*access_token_generated_using_client_credential_flow*>
- client_secret_post (using `client_id` and `client_secret` in request body parameters)

The following is the priority list in which token introspect considers these methods:

1. `client_secret_basic`
2. `bearer`
3. `client_secret_post`

If you send a request to token introspect endpoint with `client_secret_post` (client_id and client_secret in request body) and `client_secret_basic` (basic authorization header), the endpoint validates the request based on the credentials that are provided through `client_secret_basic`.

If values in `client_secret_basic` are invalid, the response displays `401 exception`. The endpoint does not consider `client_id` and `client_secret` in `client_secret_post` when `client_secret_basic` is available in the request.

NOTE: If you provide more than one method for authentication, the introspect endpoint returns the response based on the priority.

4.6.2 Request Parameters

The token introspect request should have the following parameters:

Parameter	Required	Description
token	Yes	The token that requires to be introspected. (only the access tokens or the refresh tokens can be introspected).
token_type_hint	Optional	The type of the token submitted for introspection.

4.6.3 Response Values

The response parameters are sent in JSON format with the following parameters:

Parameter	Required	Description
active	Yes	A boolean value indicating whether the mentioned token in the introspection request is currently active.
scope	Optional	A list of scopes associated with the token.
client_id	Optional	The client identifier for the OAuth 2.0 client that requested this token.

Parameter	Required	Description
username	Optional	An identifier for the resource owner who authorized the token.
token_type	Optional	The type of the token.
exp	Optional	The timestamp in seconds indicating when the token will expire
iat	Optional	The timestamp in seconds indicating when the token was issued.
nbf	Optional	The timestamp indicating till when the token cannot be used.
sub	Optional	A machine-readable identifier of the resource owner who authorized the token.
aud	Optional	The intended audience of the token.
iss	Optional	The issuer of the token.
jti	Optional	A string identifier of the token.

4.7 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 URL: Port Number>/nidp/oauth/nam/userinfo`

Request method: GET and POST

4.7.1 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

4.7.2 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.

Parameter	Description
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. For example, if the client has requested "email" scope, the UserInfo endpoint will return following value: "email" : "alice@c.com" along with the "sub" field.	

4.7.3 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"
}
```

4.8 Revocation Endpoint

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

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

Request method: POST

NOTE: The authentication is done by using `client_id` and `client_secret` in the request body parameters (as mentioned in the preceding curl request), or send client credentials in the basic authorization header (as mentioned in RFC 6749).

4.8.1 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	The client application ID that is obtained at the time of client application registration.
client_secret	Optional	The secret that is obtained at the time of client application registration. The client secret is optional for a native application and mandatory for a web application.
token	Yes	refresh_token that is obtained during authorization grant, resource owner credentials, client credentials flow

4.8.2 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.

4.8.3 Revoking a Token Issued to a Device

When Mobile Access SDK is not used for on-boarding and off-boarding devices, the token can be manually associated with a device. This can be done by providing additional parameter device_id while requesting an access token. Such manually associated tokens can be revoked by using the revocation endpoint. Send a request to revoke all tokens that are issued to a device.

Endpoint URL: https://idpbaseurl.com/nidp/oauth/nam/revoke/<device_id>

Request method: POST

Content-Type: application/x-www-form-urlencoded (Optional)

Request Parameters:

Parameter	Required	Description
userstore_name	Yes	Specify the name of the user store.
user_dn	Yes	Specify the user's DN to whom the token is issued.

Response Values

```
HTTP 200 OK
{
"status": "Successfully revoked token(s) issued to this device."
}
```

Sample Request and Response

A sample request and response (with the line breaks for better readability):

Request

```
HTTP/1.1 POST /nidp/oauth/nam/revoke/andriodtest_1401
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/41.0.2228.0 Safari/537.36
Host: www.idp.com:8443
'userstore_name=namsignboxuserstore
& user_dn=cn%3Dharry%2Co%3Dnovell'
```

Response

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

Error Response

When an invalid device ID is specified or a device had not been associated with any token, the HTTP 404 NOT FOUND error occurs with the following response:

```
{
"error": "invalid_request",
"error_description": "Invalid device ID or no tokens to revoke for this
device."
}
```

4.9 Logout Endpoint

The logout endpoint is used by client application to end user session at Identity Server. The client application sends request the Identity Server to logout the user session by redirecting the user to the Identity Server's logout endpoint via browser agent. This way of initiating logout is called Relying Party (RP) Initiated Logout.

Endpoint URL: `https://idpbaseurl.com/nidp/oauth/v1/nam/end_session`

Request method: HTTP GET and POST

4.9.1 Request Parameters

To initiate logout at Identity Server, the client application should invoke a GET or a POST request to Identity Server's logout endpoint with the following request query string parameters:

Parameter	Required	Description
id_token_hint	No	A valid ID token that was issued by the Identity Server for the user.
post_logout_redirect_uri	No	The redirect URI where the user is redirected after the logout by the Identity Server. This URI must match with the value provided at the time of client registration. The `id_token_hint` parameter is required to redirect the user to this URI. If the logout request is sent without `id_token_hint`, the user will not be redirected to the `post_logout_redirect_uri`.
state	No	An optional value. The same value is returned as a query parameter when the user is redirected to the `post_logout_redirect_uri`.

NOTE: ◆If the user session is logged out or expired at Identity Server when the logout request is received, the logout is considered to have succeeded.

- When the logout request does not contain `post_logout_redirect_uri`, the user is redirected to the Identity Provider's logout success page.
- If the registered `post_logout_redirect_uri` is specified in the logout request, the user will be redirected to the `post_logout_redirect_uri`.
- If the logout request contains the `id_token_hint` parameter, and its value is a valid ID token, but the token has expired, and the subject matches the current user session. In that case, the user session will be logged out at Identity Server.

4.9.2 Response Values

Identity Server renders one or more `<iframe src="frontchannel_logout_uri">` in the logout success page with the registered logout URI as the source to trigger the logout actions by the client application(s). Upon receiving a request to render the logout URI in an iframe, the client application should clear the state associated with the logged-in user session, including any cookies

Identity Server might include the following query parameters to `frontchannel_logout_uri` for the client application that are registered with **Enable Session Token** UI option. If the client is registered using REST API, refer to Registration Endpoint.

Parameter	Required	Description
iss	No	Issuer Identifier for the Identity Server issuing the front-channel logout request.
sid	No	Unique String identifier for a user session. The client application can use this parameter to identify the unique user sessions established at Identity Server.

The client application must verify the `iss` and `sid` parameters with `iss` and `sid` claims in `idtoken` issued for the current user session.

Success Response

- The client application redirects the user to the Identity Server logout endpoint via browser agent without `post_logout_redirect_uri`. After a successful logout, the user is redirected to the Identity Server's logout success page.

  Sample redirected logout URL without any request parameter:

  `https://idpbaseurl.com/nidp/oauth/v1/nam/end_session`

- The client application redirects the user to the Identity Server logout endpoint via browser agent with `post_logout_redirect_uri`. After a successful logout, the user is redirected to the registered `post_logout_redirect_uri`.

  Sample redirected logout URL with request parameter:

  `https://idpbaseurl.com/nidp/oauth/v1/nam/end_session?`

  `id_token_hint={id_token}&`

  `post_logout_redirect_uri={post_logout_redirect_uri}&`

  `state={state}`

  Request: `https://idpbaseurl.com/nidp/oauth/v1/nam/end_session?id_token_hint=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0NNIiw……&post_logout_redirect_uri=https://client.example.org/logoutRedirect&state=JaysvoMyK71YfVG5`

  Response: `HTTP 302 Found`

  Location: The user will be redirect to the `https://client.example.org/logoutRequest&state=JaysvoMyK71YfVG5` URI

Error Response

- If the parameter `id_token_hint` contains either an invalid or encrypted ID Token, the HTTP 401 error response is returned with the `Invalid ID Token` error message.
- When the `post_logout_redirect_uri` parameter value does not match the registered client application, the HTTP 401 error response is returned.

5 Developer Resources

This chapter includes the information about the Access Manager OAuth Samples that is available on the Developer Documentation page. You can use the sample scripts that are included in the OAuth Samples zip file to 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-configuring Access Manager by creating scopes and resource server, and registering an OAuth client.
- Execute various OAuth flows and return the access token, refresh token, and ID token as per requirement.
- Manage tokens by refreshing a token or revoking a token.

5.2 Try Now

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

Prerequisite

Perform the following tasks to use the sample scripts:

- ❒ Enable the OAuth protocol.
- ❒ Enable `NAM_OAUTH2_DEVELOPER` and `NAM_OAUTH2_ADMIN` roles for developers.
- ❒ Extend the user store schema and add LDAP attribute to store user's consent and refresh token information.

Access Manager Administrator Tasks

1. Log in to 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 options.
   - **Token Type(s):** Select all options.
   - **Require Logout Consent:** Select this option when you have configured OIDC front-channel logout. This displays a consent message to users seeking their permission to log out from all logged-in applications.
   - **Signing Certificate:** Add the signing certificate.
4 Update Identity Server.

Client Developer Tasks

1 Download OAuth Samples from the 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`.
authorizationFlow-accessToken-with-pkceS256.sh	Retrieve access token using pkce with the `code_challenge_method` as `s256`.

Sample	Use Case
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
IDP-initiated-FrontChannelLogout.sh	Identity Provider initiates the front-channel logout of a logged-in user through the client application.
RP-initiated-FrontChannelLogout.sh	The client application initiates the front-channel logout of a logged-in user.
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

5.3 Access Manager OAuth Playground

To try out the OAuth flows, see OpenID Connect with Access Manager Identity Server and OAuth 2.0 Playground.

5.4 OAuth Sample Client Applications

These sample client applications aim to provide users with information on how to configure Access Manager for OAuth and how to get an OAuth token from Access Manager. There are four sample items provided:

- Insurance Application: It is an android application that can be deployed to view insurance details. It demonstrates resource owner flow and is developed using ionic framework.
- Fitness Application: It is an android application that can be deployed to view accessible services. It demonstrates authorization code flow and the use of token refresh and token revocation. It is developed using ionic framework.
- Fitness Website: It is a website that can be accessed using a browser to view the accessible services. It demonstrates implicit flow and the use of ID token. It is developed using angular framework.
- Resource Server: It is a springboot application that protects the details of customers and allows access to it through the OAuth protocol.

To access the sample applications, see Sample Applications for OAuth Use Cases in the Developer Documentation.