ONE IDENTITY™

Cloud Access Manager 8.1.3

How to Develop OpenID Connect Apps

Legend

WARNING: A WARNING icon indicates a potential for property damage, personal injury, or death.

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

IMPORTANT, NOTE, TIP, MOBILE, or VIDEO: An information icon indicates supporting information.

Cloud Access Manager How to Develop OpenID Connect Apps
Updated - October 2017
Version - 8.1.3

Contents

1

Introduction

This guide describes how to develop OpenID Connect applications for use with Cloud Access Manager.

What is OAuth v2.0?

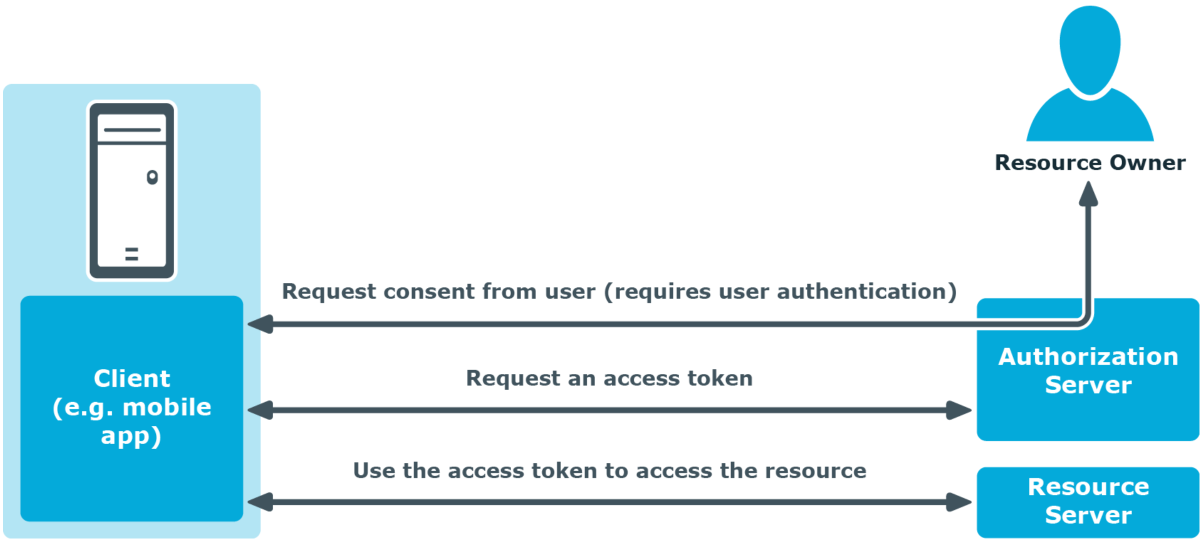
OAuth v2.0 is a standard for securely granting access to a web resource. With OAuth v2.0, an application (the client) can ask a service (the authorization server) for permission to access a private resource hosted on a resource server, and owned by an end-user (the resource owner). To grant permission to access the resource, the authorization server must authenticate the resource owner and obtain his consent.

The specification for OAuth v2.0 is available to view online at https://tools.ietf.org/html/rfc6749 which contains this example:

An end-user (resource owner) can grant a printing service (client) access to her protected photos stored at a photo-sharing service (resource server), without sharing her username and password with the printing service. Instead, she authenticates directly with a server trusted by the photo-sharing service (authorization server), which issues the printing service delegation-specific credentials (access token).

OAuth v2.0 is often loosely regarded as an authentication protocol in its own right; however the specification does not prescribe the means by which credentials are collected from the end-user.

Figure 1: Conceptual overview of OAuth v2.0


Resource Owner
Request consent from user (requires user authentication)
Client (e.g. mobile app)
Authorization Server
Request an access token
Use the access token to access the resource
Resource Server

OAuth v2.0 flows

OAuth v2.0 presents four different flows which are appropriate to different scenarios:

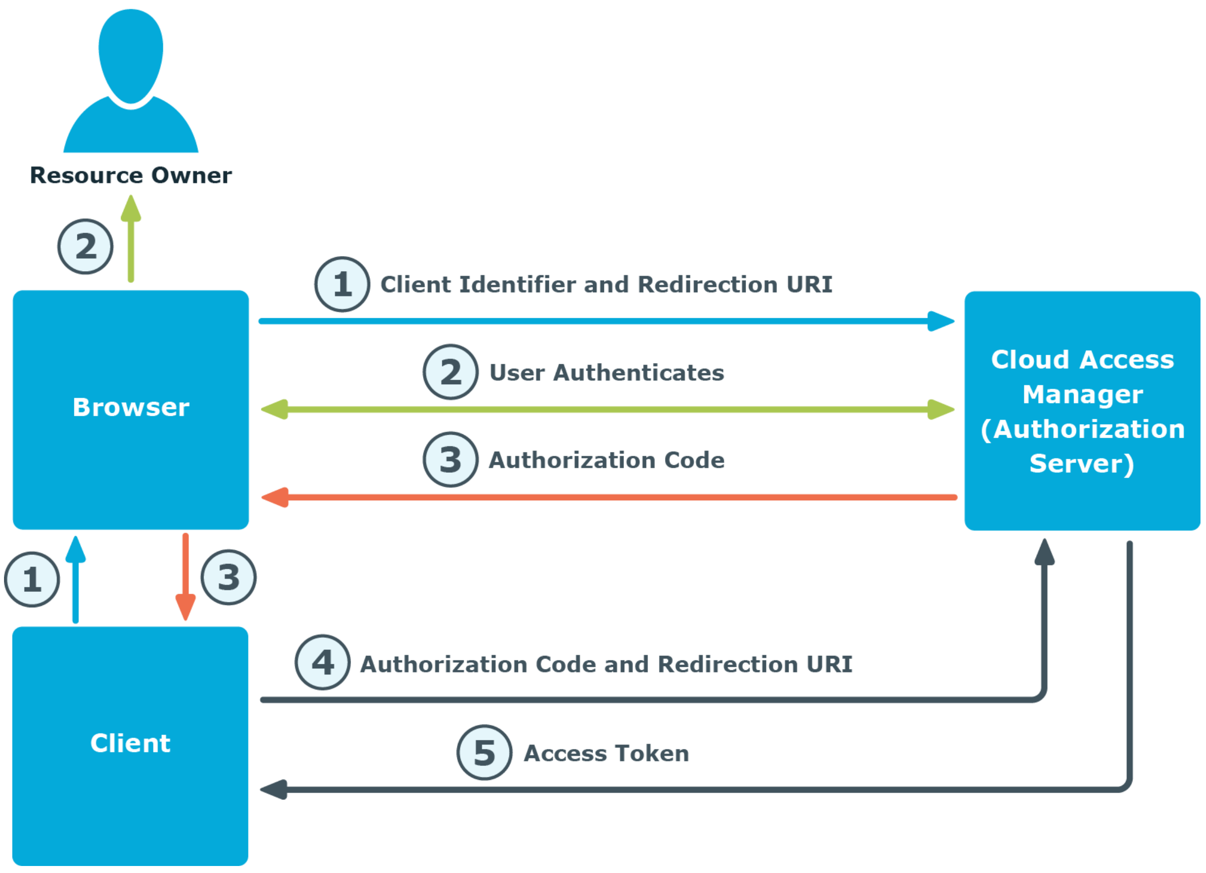
- Authorization Code Flow
- Implicit Flow
- Approval
- OAuth v2.0 client types

Cloud Access Manager provides the Authorization Server function for Authorization Code Flow and Implicit Flow. It does not support Resource Owner Password Credentials Flow or Client Credentials Flow.

Authorization Code Flow

This is the best choice for web applications which run on a web server, because they can be reliably authenticated. This flow requires a browser, because it relies on HTTP redirects, but the browser can be embedded into the client application. The client invokes the browser, directing it to the authorization server in order to authenticate the user and obtain consent (in the form of an authorization code). With this authorization code, the client app can contact the authorization server directly (not through the browser) in order to obtain an access token, which can then be used to access the required resource.

Figure 2: Authorization Code Flow


Resource Owner
2
1 Client Identifier and Redirection URI
2 User Authenticates
3 Authorization Code
Browser
Cloud Access Manager (Authorization Server)
1
3
4 Authorization Code and Redirection URI
5 Access Token
Client

1. The client initiates the flow by directing the user's browser to the authorization endpoint, adding querystrings to the URI as follows:

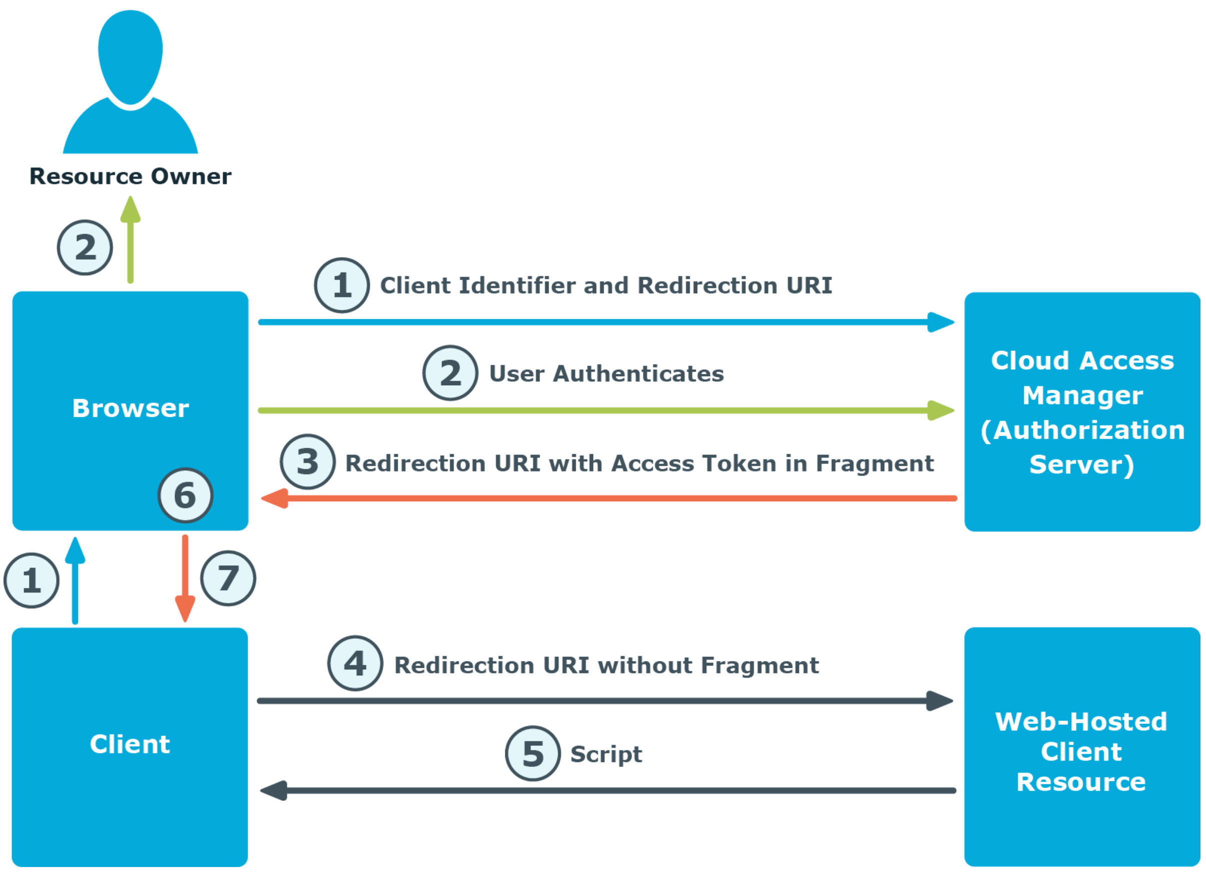
Table 1: Authorization Code Flow querystrings

Querystring	Description
response_type:	Set to **"code"** to request that the Authorization Server initiate an Authorization Code flow.
Client_id:	A unique identifier generated by Cloud Access Manager for the client when the application definition is configured.
redirect_uri:	The URI which Cloud Access Manager will redirect the user's browser to, when authorization processing is complete.
scope (optional):	Used to determine what resources are being requested from the **Resource Server**.
state (optional):	A value which the client can use to maintain state between request and callback. This can be used to prevent cross-site request forgery.

2. Cloud Access Manager authenticates the user (using the browser) and establishes whether the user grants or denies the client's access request.
3. Assuming the user grants access, Cloud Access Manager redirects the browser back to the client using the redirection URI provided earlier. The redirection URI includes an authorization code and any local state provided by the client.
4. The client requests an access token from Cloud Access Manager's token endpoint by including the authorization code received in the previous step. When making the request, if it is a **confidential** client (see below) the client authenticates with Cloud Access Manager using HTTP basic authentication (with the Client ID as the username and the Shared Secret as the password). The client includes the redirection URI used to obtain the authorization code for verification.
5. Cloud Access Manager authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client in step 3. If valid, Cloud Access Manager responds with an access token. The access token can then be used to access the required resource.

Implicit Flow

You would use this method if your OAuth client was a web application written in JavaScript code which runs in the browser, rather than on a server. In this case, the client app cannot be reliably authenticated (because the client credentials would be accessible to the resource owner and other applications running on the same device as the client). Like the Authorization Code Flow, it too relies on a browser.


Resource Owner
2
1 Client Identifier and Redirection URI
2 User Authenticates
3 Redirection URI with Access Token in Fragment
Browser
6
Cloud Access Manager (Authorization Server)
1
7
Client
4 Redirection URI without Fragment
5 Script
Web-Hosted Client Resource

1. The client initiates the flow by directing the user's browser to the authorization endpoint, adding querystrings to the URI as follows:

   Table 2: Implicit Flow querystrings

Querystrings	Description
response_ type:	Set to "**token**" to request that the Authorization Server initiate an Implicit Flow.
client_id:	A unique identifier generated by Cloud Access Manager for the client when the application definition is configured.
redirect_uri:	The URI which Cloud Access Manager will redirect the user's browser to, when authorization processing is complete.
scope (optional):	Used to determine what resources are being requested from the **Resource Server**.
state (optional):	A value which the client can use to maintain state between request and callback. This can be used to prevent cross-site request forgery.

2. Cloud Access Manager authenticates the user (using the browser) and establishes whether the user grants or denies the client's access request.

3. Assuming the user grants access, Cloud Access Manager redirects the browser back to the client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment, for example.

   http://example.com/cb#**access_token=2YotnFZFEjr1zCsicMWpAA**&state=xyz&token_type=example&expires_in=3600

4. The browser follows the redirection instructions by making a request to the web-hosted client resource (which does not include the fragment). The browser retains the fragment information locally.
5. The web-hosted client resource returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by the browser, and extracting the access token (and other parameters) contained in the fragment.
6. The browser executes the script provided by the web-hosted client resource locally, which extracts the access token.
7. The browser passes the access token to the client. The access token can then be used to access the required resource.

Approval

Cloud Access Manager does not explicitly prompt the resource owner to approve the client’s request to access a resource. The user is assumed to have given his/her consent in an enterprise context, where resources are not owned by an individual, but by the organization of which that individual is a member.

OAuth v2.0 client types

When configuring your OAuth v2.0 application in Cloud Access Manager, you are required to select its Client Type. OAuth v2.0 specifies two client types: public and confidential. Your choice determines whether Cloud Access Manager will enforce authentication on the client connection.

public

Applications which run in an environment in which the confidentiality of client credentials cannot reliably be protected. Generally, this would include JavaScript applications which execute in the browser, and native applications which run on the end-user’s device. For public client types, Cloud Access Manager does not authenticate the client.

confidential

Server-based applications (typically web applications) are capable of maintaining the confidentiality of secret information. Applications which run in this environment should be configured as confidential. Cloud Access Manager does authenticate confidential clients.

When you configure your application as a confidential client, you must use the authorization code flow; Cloud Access Manager will return an error if an attempt is made to invoke Implicit flow from a confidential client.

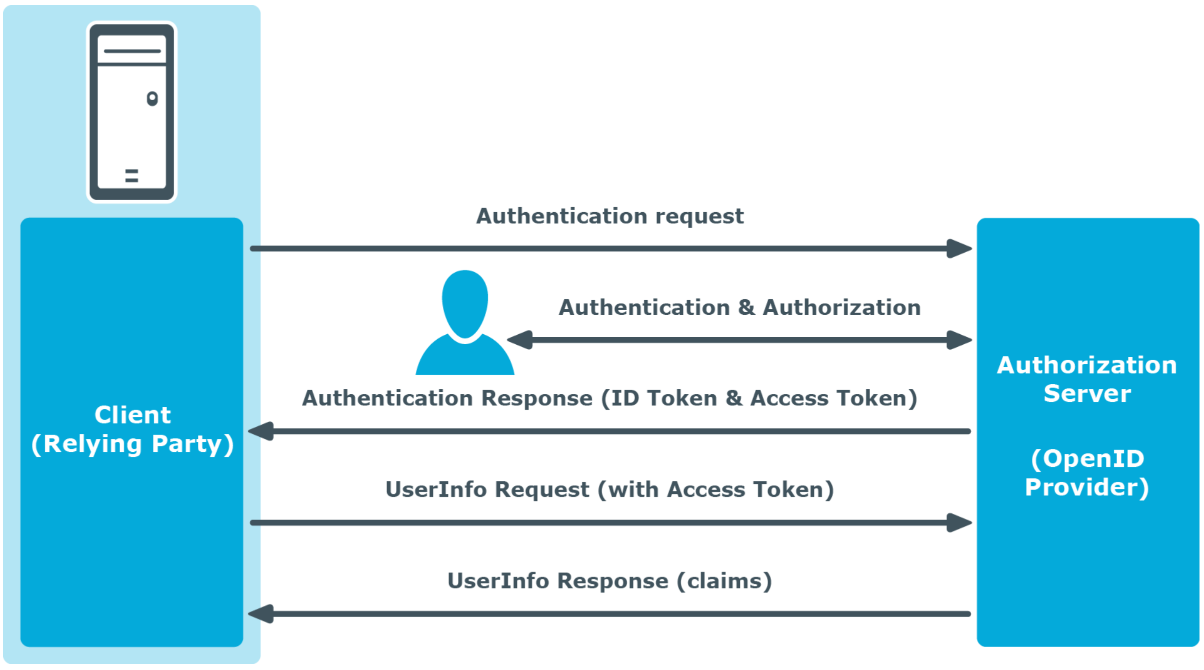
What is OpenID Connect?

OpenID Connect is a standard which defines how the identity of an end-user can be verified, and how claims about a user can be obtained. It is an extension to the OAuth v2.0 protocol.

OpenID Connect deals with scenarios where a user is authenticated using a browser (for example the **authorization code** and **implicit code** flows of the OAuth v2.0 specification). Clients signal to the authorization server that they wish to perform OpenID Connect authentication by setting the scope in the initial authorization request to **openid**.

In OpenID Connect, the client is referred to as the **Relying Party (RP)** and the Authorization Server is known as the **OpenID Provider (OP)**. The core specification for OpenID Connect can be found online at http://openid.net/specs/openid-connect-core-1_0.html

Figure 3: Conceptual view of OpenID Connect


Authentication request
Authentication & Authorization
Authentication Response (ID Token & Access Token)
UserInfo Request (with Access Token)
UserInfo Response (claims)
Client
(Relying Party)
Authorization Server
(OpenID Provider)

OpenID Connect ID Token

As a result of a successful authentication request, the OpenID Provider returns an ID Token in the form of a **JSON Web Token**, the specification for the JSON Web Token format can be found online at https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32. The ID Token contains claims about the authentication of an end-user as follows:

iss

Issuer. A URL that identifies the OP.

sub

Subject. Uniquely identifies the end-user

aud

Audience. Contains the OAuth 2.0 client_id of the Relying Party.

exp

Expiration time. Set by Cloud Access Manager by adding 30 minutes to the time the ID token was issued.

iat

Time at which the ID Token was issued.

auth_time

Time when the end-user authentication occurred.

nonce

If a nonce is present in the authentication request, the same value will be echoed to the client in this claim. The nonce value can be used by the client to mitigate replay attacks.

acr

Not returned.

amr

Not returned.

azp

Not returned.

OpenID Connect Token signing

OpenID Connect requires that the ID Token must be digitally signed. Cloud Access Manager offers two ways to sign the ID Token. You can sign the ID Token:

- Using HMAC SHA-256 with an automatically-generated pseudo-random shared secret
- Using RSA SHA-1 with a private key from a self-signed X.509 certificate

OAuth v2.0 and OpenID Connect response types

Cloud Access Manager supports the following response types:

Table 3: Supported response types

Response_type	Meaning	Comments
code	OAuth v2.0 author-ization code flow	
token	OAuth v2.0 implicit flow	Not used by OpenID Connect (no ID token returned)
id_token	OpenID Connect implicit flow	No access token returned. Claims are included in the ID Token
token id-token	OpenID Connect implicit flow	Claims are accessed by invoking UserInfo endpoint

Obtain claims from Cloud Access Manager

Cloud Access Manager does not use the scope values contained in the authentication request to determine what claims are to be returned. Instead, it returns the claims defined by the application configuration’s claim mappings. Thus the client cannot control what claims are returned to it.

OpenID Connect discovery

The optional OpenID Connect specification (available online at http://openid.net/specs/openid-connect-discovery-1_0.html) defines how Relying Parties can automatically query the End-User's OpenID Provider for information needed to interact with it, including its OAuth 2.0 endpoint locations. Cloud Access Manager does not provide this facility.

When would you use OAuth v2.0 and OpenID Connect?

The purpose of the OAuth v2.0 framework is to allow client applications to gain authorization to access information held on a resource server with the permission of the owner, without the client needing to know the owner's password. The key difference between it and other federation protocols such as SAML or WS-Federation is that the request for access can be limited to a single resource or a group of resources (rather than an entire web application) and the resource owner is invited to approve the access request.

While OAuth v2.0 can be used to provide the framework for authentication and authorization to any web application, it has become the *de facto* standard for mobile apps which communicate with an on-line resource server.

In summary, there are three main use cases in which you would consider using OAuth v2.0 or OpenID Connect to provide authentication and authorization services from Cloud Access Manager:

- If your organization has purchased software, or subscribes to an online service, which supports use of OAuth v2.0 or OpenID Connect.
- If your organization is developing a web-based application for your enterprise where users will interface with the application using a mobile app, then you may consider OAuth v2.0 and OpenID Connect instead of SAML.
- If your organization is developing a web-based application for your enterprise whereby other systems require access to resources hosted by your new application.

Cloud Access Manager supports both OAuth v2.0 and OpenID Connect applications through the same back-end SSO method.

Using the OpenID Connect Flow Test Tool

The OpenID Connect Flow Test Tool allows you to invoke Cloud Access Manager OpenID Connect calls, just as an application would. This can help your understanding of Cloud Access Manager's implementation of OpenID Connect and be used as a diagnostic tool when developing your own application.

To get started with the OpenID Connect Flow Test Tool, follow the steps below. Ensure that you have already set up a front-end authenticator in Cloud Access Manager to authenticate a test user:

Client machine

1. The OpenID Connect Flow Test tool can run on any machine with .NET Framework 4.5 installed which has network access to your Cloud Access Manager proxy.
2. From your Cloud Access Manager installation media, open the **Tools** folder and extract the **OIDCFlowTestTool.zip** to a suitable location on a machine.
3. Run **OIDCFlowTestTool.exe** in the extracted folder. This will open a Windows application.
4. In the **CAM Proxy Hostname** field, supply the hostname of your Cloud Access Manager proxy.

Cloud Access Manager administration

1. Log in to Cloud Access Manager as an administrator.
2. Under **Applications**, click **Add New**.
3. Click **Configure Manually**, click **Next**.
4. Choose **Using OpenID Connect / OAuth 2.0**, click **Next**.
5. Select the required **Client Type**, the tool can be configured as public or confidential.

Exchanging set up information between Cloud Access Manager and client machine

1. Copy the **Redirect URI** from the test tool to the **Redirect URI** input field of the application configuration in Cloud Access Manager.
2. Copy the **Client ID** from the application configuration in Cloud Access Manager to the **Client ID** field of the test tool.
3. In the test tool, click the **Details** tab.
4. Copy the **Shared Secret** value from the application configuration in Cloud Access Manager to the **Shared Secret** field in the test tool.

Cloud Access Manager administration

1. Click **Next**.
2. Choose **Do not proxy this application**, click **Next**.
3. Select the roles which are permitted to use this application and click **Allow Role Access**, click **Next**.
4. Type in a name for your application, click **Next**.
5. In the **URL** field, type https://*cloudaccessmanager.proxy.com/myapp* where *cloudaccessmanager.proxy.com* is the hostname of your proxy, and myapp is a name of your choice. The URL is not important for this exercise, since no users will be launching the application from the portal, but the wizard requires that you enter a valid URL in that field.
6. Click **Finish**.

Client machine

1. Click the **Flows** tab.
2. To invoke an Implicit Flow, click the **GET Token (Implicit flow)** button.

   NOTE: If your application was set up as a confidential client, then a request for Implicit Flow will be rejected with an authorization error response.

3. To invoke Authorization Code Flow, check the **Authenticate** box (this causes the client to authenticate to the token endpoint), then click the **1. GET AZCode (AuthZ Code flow)** button. To obtain an access token, click **2. POST AZCode for Token (AuthZ Code flow)**.
4. Use the **Show Id Token**, **Show Access Token**, and **GET UserInfo** buttons to view information returned from Cloud Access Manager.

Building your OpenID Connect client

There are libraries available which can help you to build an OAuth or OpenID Connect client.

NOTE: The following links are for information only, and One Identity does not assume any responsibility for their content.

- https://www.nuget.org/packages/Thinktecture.IdentityModel.Client/

  Includes .NET OAuth2 and OpenID Connect client and helpers for parsing token and authorize responses, Epoch Time helpers, and extensions methods for HttpClient.

- https://bitbucket.org/b_c/jose4j/wiki/Home

  Open source (Apache 2.0) implementation of JWT and the JOSE specification suite, written in Java.

Example: Providing OpenID Connect SSO to a Salesforce.com Auth Provider

This example will guide you through the steps required to configure single sign-on for Salesforce.com using OpenID Connect.

To configure single sign-on for Salesforce using OpenID Connect

1. Log in to the Administration Console using the desktop shortcut **Cloud Access Manager Application Portal**, then select **Add New** from the **Applications** section on the home page.

Cloud Access Manager provides a set of application templates to automatically configure common applications. This example describes how to configure an application manually, rather than using a template.

2. Click **Configure Manually**.
3. Select **Using OpenID Connect / OAuth 2.0**, then click **Next**.
4. Create a new **Auth. Provider** in Salesforce of type **Open ID Connect** and enter the endpoint, issuer and client information displayed on the Cloud Access Manager **OpenID Connect / OAuth 2.0 Settings** page, see Step 3.
5. After creating the **Auth Provider** in Salesforce, copy the **Callback URL** into the **Redirect URI** text box on the Cloud Access Manager **Using OpenID Connect / OAuth 2.0 Settings** page.
6. Select **Confidential** as the **Client Type**.
7. Select **Sign token with shared secret** as the **Token Signing** method.
8. Click **Next** to continue.
9. Select **Do not proxy this application**, then click **Next**.
10. You will now see the **Permissions** page which enables you to control the users who can access the application. By default, all Cloud Access Manager users have access to the application. You can restrict access to the application to users who belong to a specific Cloud Access Manager role, but for this demonstration deployment, simply click **Next** to allow all Cloud Access Manager users to access the application.
11. Enter an **Application Name**, for example, Salesforce, then click **Next**.
12. Enter the URL that you want your users to be initially logged into, for example, https://login.salesforce.com/services/auth/sso/<provider_id>/CAM
13. Enter can now configure how the application is displayed on the Cloud Access Manager Portal. Enter the **Title** and **Description** you want to display on the Cloud Access Manager Portal.
14. Click **Fetch icon from application** to locate and display the application icon.

    NOTE: In addition the **Add application to application portal home** and **Allow user to remove application from application portal home** options allow you to specify whether the application should automatically appear on each user's portal page and how the user can manage the application from the application portal.

Table 4: Application portal options

Add application to application portal home	Allow user to remove application from application portal home	Functionality
✓	✗	application is added to the portal and it cannot be

Add application to application portal home	Allow user to remove application from application portal home	Functionality
		removed by the user through the application catalog.
✓	✓	application is added to the portal and it can be removed by the user through the application catalog
✗	✗	application is not automatically added to the portal. The user can add or remove the application to/from the portal through the application catalog.

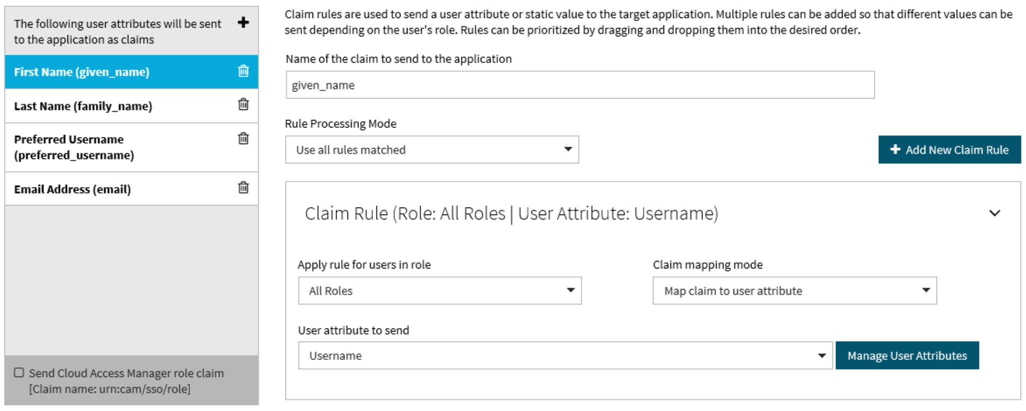
To access the application catalog from the application portal, the user simply needs to click their username, then select Application Catalog.

Depending on the settings in the Add application to application portal home and Allow user to remove application from application portal home options, the user can add or remove applications to/from the application portal.

15. Click **Finish** to complete the configuration of the application.

NOTE: Some claims need to be set up manually. To do this, edit the application and select the **Claim Mapping** tab. Add claims for **given_name**, **family_name**, **preferred_username** and **email**.

Claim Mapping



The following user attributes will be sent to the application as claims

- First Name (given_name)
- Last Name (family_name)
- Preferred Username (preferred_username)
- Email Address (email)

☐ Send Cloud Access Manager role claim [Claim name: urn:cam/sso/role]

Claim rules are used to send a user attribute or static value to the target application. Multiple rules can be added so that different values can be sent depending on the user's role. Rules can be prioritized by dragging and dropping them into the desired order.

Name of the claim to send to the application: given_name

Rule Processing Mode: Use all rules matched

Claim Rule (Role: All Roles | User Attribute: Username)

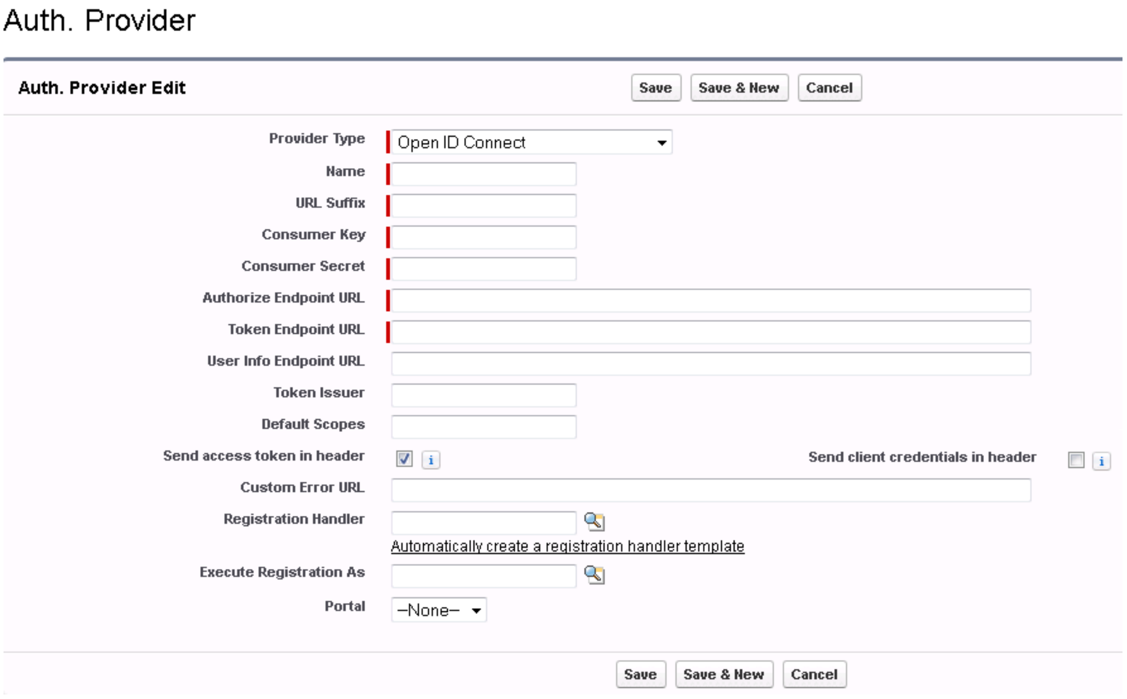
Apply rule for users in role: All Roles

Claim mapping mode: Map claim to user attribute

User attribute to send: Username

To configure your Salesforce account to authenticate your users with Cloud Access Manager OpenID Connect

1. Login to Salesforce with administration rights.
2. Go to **Setup**.
3. Go to **Security | Auth Provider**.


Auth. Provider

Auth. Provider Edit Save | Save & New | Cancel

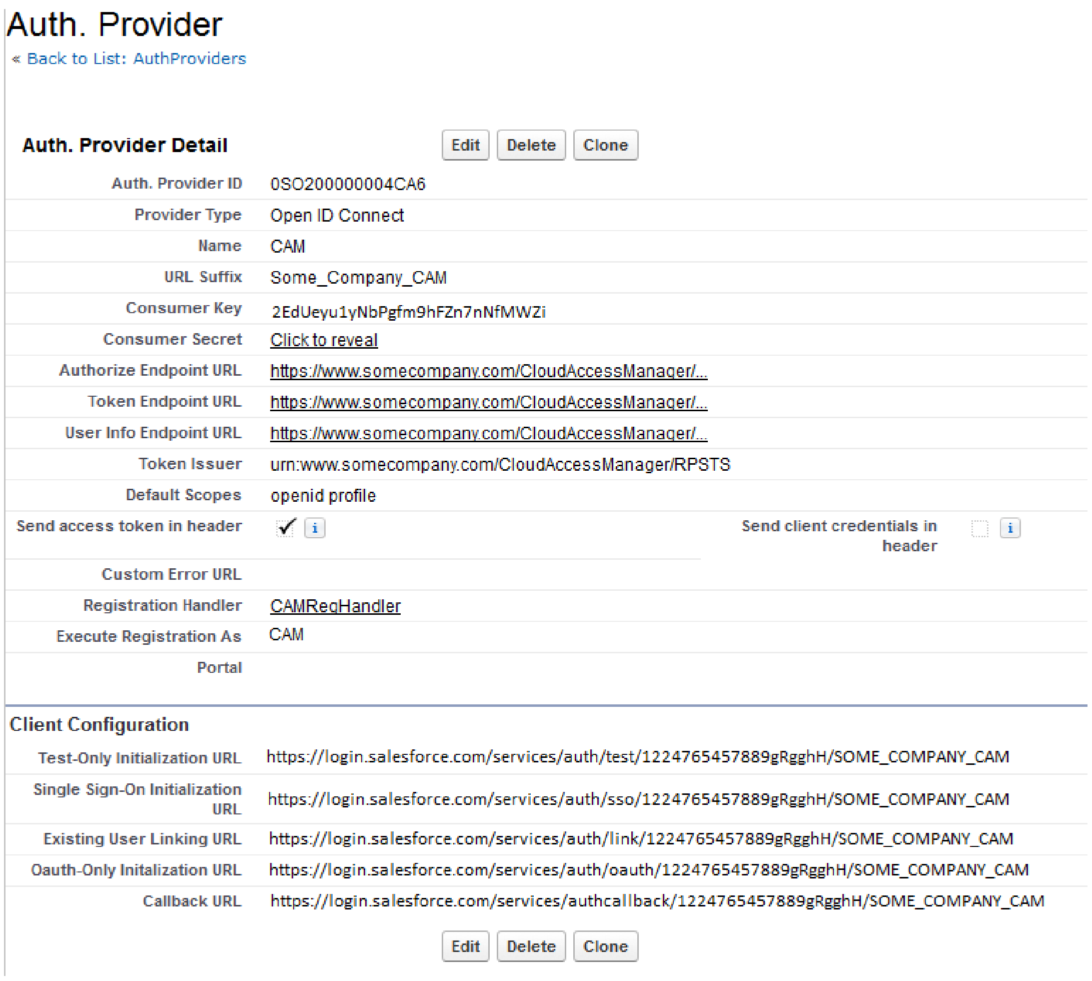
Provider Type	Open ID Connect
Name	
URL Suffix	
Consumer Key	
Consumer Secret	
Authorize Endpoint URL	
Token Endpoint URL	
User Info Endpoint URL	
Token Issuer	
Default Scopes	
Send access token in header	✓
Send client credentials in header	☐
Custom Error URL	
Registration Handler	Automatically create a registration handler template
Execute Registration As	
Portal	--None--

Save | Save & New | Cancel

4. Click **New**.
5. In **Provider Type** select **Open ID Connect** from the list.
6. Enter a **Name** for the provider.
7. Copy the corresponding values from your Cloud Access Manager OpenID Connect / OAuth 2.0 Settings into the appropriate fields, **Consumer Key**, **Consumer Secret**, **Authorize Endpoint URL**, **Token Endpoint URL**, **User Info Endpoint URL**, **Token Issuer**.

   NOTE: You should verify that your Cloud Access Manager sends a full certificate chain with its SSL certificate or the Salesforce server may return an error and refuse to connect to the required **Endpoints**; the error returned is not explanatory.

8. Click **Automatically create a registration handler template**.
9. Select a privileged user for **Execute Registration As**.



Auth. Provider

« Back to List: AuthProviders

Auth. Provider Detail

Auth. Provider ID	0SO200000004CA6
Provider Type	Open ID Connect
Name	CAM
URL Suffix	Some_Company_CAM
Consumer Key	2EdUeyu1yNbPgfm9hFZn7nNfMWZi
Consumer Secret	Click to reveal
Authorize Endpoint URL	https://www.somecompany.com/CloudAccessManager/...
Token Endpoint URL	https://www.somecompany.com/CloudAccessManager/...
User Info Endpoint URL	https://www.somecompany.com/CloudAccessManager/...
Token Issuer	urn:www.somecompany.com/CloudAccessManager/RPSTS
Default Scopes	openid profile
Send access token in header	✓
Send client credentials in header	
Custom Error URL	
Registration Handler	CAMRegHandler
Execute Registration As	CAM
Portal	

Client Configuration

Test-Only Initialization URL	https://login.salesforce.com/services/auth/test/1224765457889gRgghH/SOME_COMPANY_CAM
Single Sign-On Initialization URL	https://login.salesforce.com/services/auth/sso/1224765457889gRgghH/SOME_COMPANY_CAM
Existing User Linking URL	https://login.salesforce.com/services/auth/link/1224765457889gRgghH/SOME_COMPANY_CAM
Oauth-Only Intialization URL	https://login.salesforce.com/services/auth/oauth/1224765457889gRgghH/SOME_COMPANY_CAM
Callback URL	https://login.salesforce.com/services/authcallback/1224765457889gRgghH/SOME_COMPANY_CAM

10. Click **Save**.
11. Copy the **Callback URL** into Cloud Access Manager for the **Redirect URI**.

    Single Sign-On Initialization URL — This is used to SSO into Salesforce using the Registration Handler. The default handler created above will require the Salesforce account to be already linked to a Cloud Access Manager account. However, it is possible to write your own handler which would automatically provision a user and link it to a Cloud Access Manager account.

    Existing User Linking URL — This is used to link Cloud Access Manager accounts to existing Salesforce accounts. The user is prompted to log into Cloud Access Manager and to then select the Salesforce account to link to.

Configuration of Salesforce for OpenID Connect/OAuth 2.0 is now complete.

The following example will guide you through the steps required to configure single sign-on for a thick client or mobile application.

To configure single sign-on for an application using OpenID Connect

1. Log in to the Administration Console using the desktop shortcut **Cloud Access Manager Application Portal**, then select **Add New** from the **Applications** section on the home page.

   Cloud Access Manager provides a set of application templates to automatically configure common applications. This example describes how to configure an application manually, rather than using a template.

2. Click **Configure Manually**.
3. Select **Using OpenID Connect / OAuth 2.0**, then click **Next**.
4. Configure the application with the endpoint, issuer and client information displayed on the Cloud Access Manager **OpenID Connect / Oauth 2.0 Settings** page displayed in Step 3.
5. Enter the **Redirect URI** on the Cloud Access Manager **OpenID Connect / Oauth 2.0 Settings** page.

   NOTE: Some applications require the authorization code to be returned in the web page title, if this is the case, use urn:InstalledApplication as the **Redirect URI**.

6. Select **Public** as the Client Type.
7. Select the **Token Signing** method as appropriate. Click **Next** to continue.
8. Select **Do not proxy this application**, then click **Next**.
9. You will now see the **Permissions** page, which enables you to control which users can access the application. By default, all Cloud Access Manager users have access to the application. You can restrict access to the application to users who belong to a specific role, but for this example, simply click **Next** to allow all users to access the application.
10. Enter an **Application Name**, then click **Next**.

    NOTE: It is not possible to launch this type of application from the Application Portal, therefore no portal settings are required.

11. Click **Finish** to complete the configuration of the application.

    NOTE: If the application requires additional claims. You will need to set these up manually. To do this, edit the application and select the **Claim Mapping** tab.

About us

Contacting us

For sales or other inquiries, visit https://www.oneidentity.com/company/contact-us.aspx or call +1-800-306-9329.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at https://support.oneidentity.com/.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to-videos
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product