Configure OpenID Connect authentication

Last Modified on 01/12/2021 12:48 pm EST

Overview

OpenID Connect is an authentication layer on top of the OAuth 2.0 protocol. With OpenID Connect, clients verify the identity of end-users based on the authentication performed by an authorization server, and obtain basic profile information about the connecting end-user in an interoperable and REST-like manner.

Cora SeQuence supports the authorization code flow as a means of authentication.
For more details on the core OpenID Connect functionality, see this page.

Prerequisites

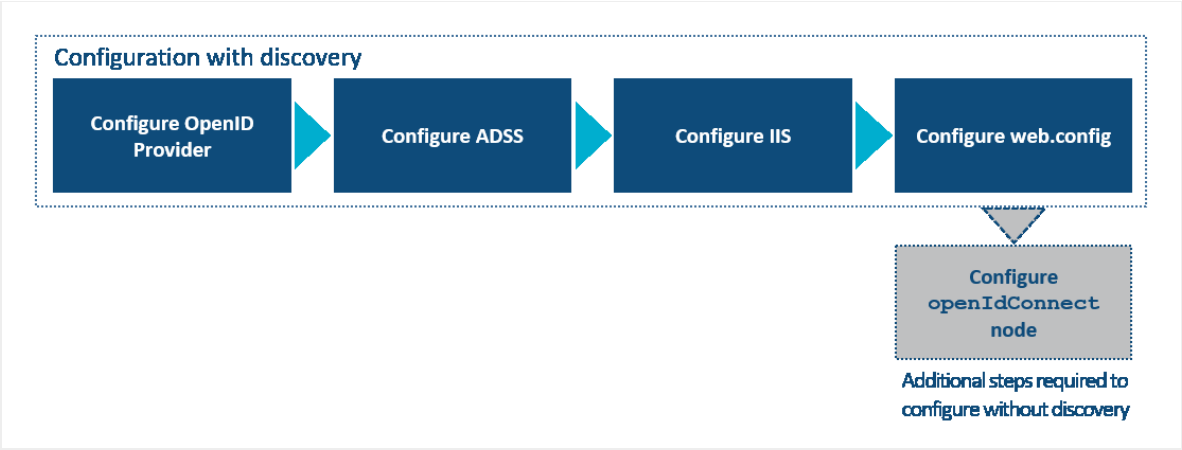
- Cora SeQuence Administration and Flowtime must be configured to use HTTPS.
  For details on configuring HTTPS for Cora SeQuence sites, see this article.
- Obtain your server's fully qualified domain name (FQDN) and Cora SeQuence's base URL.
- Determine whether users are identified by email or by name.

Supported flows

- Cora SeQuence only supports the Authorization Code Flow.
  - For the authentication request, Cora SeQuence supports HTTP-POST (by default) and HTTP-GET.
  - For the Token request, Cora SeQuence supports Basic-Authentication and POST-Authentication.

How to configure

Cora SeQuence supports automatic discovery of the OpenID provider's configuration, or manual configuration of the OpenID provider.



NOTE
If your Cora SeQuence installation does not use discovery, see **Appendix A** below for additional configuration instructions.

Configuring the OpenID Provider (OP)

1. Generate the following with your OpenID Provider:
    - Client ID
    - Client secret
2. Configure the Reply URLs by appending the following to the base part of the application:
   `oidcServices/signinresponse (i.e. )`
3. Do one of the following:
    - If your OP works with discovery, obtain the discovery URI.
      It will be used when configuring the web.config.
    - If your OP does not work with discovery, but has the token signing keys available from an end point, then obtain a copy of that end point.

Configuring ADSS

- If you use Windows Active Directory or Azure Active Directory, configure Cora SeQuence's Active Directory Synchronization Service to retrieve users.
- Validate that users are synchronized and that at least one of the authentication fields contains the unique user identifier.
  For more details on configuring active directory synchronization, see this article.

Configuring IIS

- Enable only anonymous authentication for the Administration site and Flowtime site.
- Make sure that the only default document is `default.aspx`.

Configuring the web.config file

1. Add the following section under the nodes:
    a. Directly under , add the following nodes:

    b. Under the , add the following nested :

2. Register the following modules:

```
...



...
```

3. Add the OpenID Connect configuration.

   Template of the node

```
...

"

clientId="your clientID"
clientSecret="Your ClientSecret"


...
```

NOTE
For versions earlier than V9.6, add the following to the above code, below clientSecret attribute:
`cookieEncryptionKey="a random choice of 30 alpha-numeric chars"`
`cookieEncryptionInitializationVector="another random choice of 30 alpha-numeric chars"/>`

Attributes

Name	Description	Required	Value
discoveryKind	How the application gets the additional OpenID Connect configuration from the OP.	Yes	OpenIdDiscoveryDocument (when using discovery)
discoveryUri	The URL of the OP's discovery document.	Yes (if using discovery)	To be obtained from the OP
clientID	The ID of the application in the OP.	Yes	To be obtained from the OP
clientSecret	The symmetric key of the application and the OP.	Yes	To be obtained from the OP
cookieEncryptionKey **NOTE** Required only for V9.5 and earlier.	The key Cora SeQuence uses to encrypt its cookie.	Yes	A randomly chosen string of alpha-numeric characters (30 characters)

Name	Description	Required	Value
cookieEncryptionInitializationVector **NOTE** Required only for V9.5 and earlier.	The initialization vector Cora SeQuence uses to encrypt its cookie. **NOTE** If your system uses load balancing, `cookieEncryptionKey` and `cookieEncryptionInitializationVector` must be identical.	Yes	A randomly chosen string of alpha-numeric characters (30 characters)
authorizationRequestHttpMethod	Determines whether the authorization request is sent to the OP as an HTTP-GET or HTTP-POST. **NOTE** From our tests, Azure AD works with POST, whereas Okta works with GET.	No	POST or GET

4. Configure the `SessionAuthenticationModule`.
   Anywhere under the root configuration node, add the following nodes, making sure that you replace **https://sequenceadmin.mycompany.com** with your application base URL.

```
...

      "https://sequenceadmin.mycompany.com"/>

...
```

5. Configure Cora SeQuence Services.

   NOTE
   Windows authentication must be removed from Cora SeQuence Web Services.

   a. Locate the node in the web.config file.

b. In the child node , change the `clientCredentialType` value from "Windows" to "None":

```
            None" />


            None" />
```

6. Configure the Cora SeQuence authentication provider.
   a. Add claims authentication under the section:

```
        >IdentityClaims>
          >add claimType=">claim type>" originalIssuer=">token issuer>" authenticationT
ype=">sequence authentication type>"/>
        >/IdentityClaims>
      >/claims>
```

   b. Replace the attribute values in this line:

`claimType="" originalIssuer="" authenticationType="` with the appropriate values.

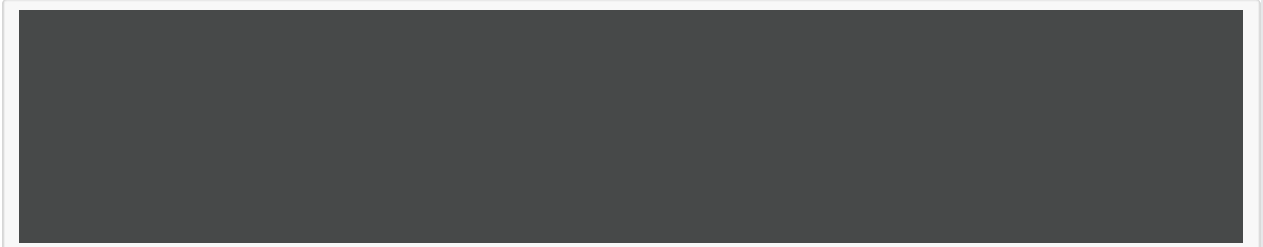
For additional information on claims-based authentication configuration, see this article.

Appendix A: Configure OpenID Connect without discovery

Perform the configuration procedures described above, and when configuring the node, add attributes according to the discovery kind used by your installation.

Example of the node with `discoveryKind="JsonWebKeySet"`:

Example of the node with `discoveryKind="Store"`:



NOTE

For versions earlier than V9.6, add `cookieEncryptionKey="yourChoiceOf30AlphanumericCharactersForKey"` and `cookieEncryptionInitializationVector="yourChoiceOf30AlphanumericCharactersForInitVector"` attributes.

`Configuration settings`

Item	Description	Value	Required
validIssuer	The name of the token issuer. The name will be validated against the token and signing certificate.	N/A	Yes
authority	The OpenID authorization endpoint.		Depends on discovery Kind
discoveryKind	Determines how the OpenID module discovers the identity provider's signing tokens.	See the *discoveryKind* table below	Yes
discoveryUri	• When the `discoveryKind` is set to `OpenIdDiscoveryDocument`: this is the URL of the OP's discovery document. • When the `discoveryKind` is set to `JsonWebKeySet` or `Federation`: this setting is used to locate the document containing the required information to retrieve the identity of the provider's signing keys. • When the `discoveryKind` is set to Store: this value is not required.	The default is an empty string.	Depends on discovery Kind
identityProviderCertificates	A collection of configuration elements that determine how to locate a certificate in the machine Certificate Store. This element is required only when the `discoveryKind` is set to 'Store'. **NOTE** identityProviderCertificates is a node, not an attribute.	Null (default) -- see *identityProviderCertificate options* table below.	No
clientId	The ID of the application in the OP.	To be obtained from the OP.	Yes

Item	Description	Value	Required
clientSecret	The symmetric key of the application and the OP.	To be obtained from the OP.	Yes
cookieEncryptionKey **NOTE** Required only for V9.5 and earlier.	The key Cora SeQuence uses to encrypt its cookie.	A randomly chosen string of alpha-numeric characters (30 characters).	Yes
cookieEncryptionInitializationVector **NOTE** Required only for V9.5 and earlier.	The initialization vector Cora SeQuence uses to encrypt its cookie.	A randomly chosen string of alpha-numeric characters (30 characters).	Yes
tokenEndpoint	The OpenID token endpoint.		Yes
logoutEndpoint	The OpenID logout endpoint.		Yes
tokenRequestAuthorizationMethod	The means by which the token request is authenticated by the OP.	"client_secret_basic" or "client_secret_post"	Yes
idTokenSigningAlgValues	Identifies the signing algorithm of the ID token.	Use one of the values in the left most column in the table in section A.1	Yes
authorizationRequestHttpMethod	Determines whether the authorization request is sent to the OP as an HTTP-GET or HTTP-POST. **NOTE** From our tests, Azure AD works with POST, whereas Okta works with GET.	POST	No

Item	Description	Value	Required
forceReauthenticate	**Available with Cora SeQuence V9.3** Forces the user to sign in again on the OIDC provider SSO sign-in page even if the user has a valid session (user has already signed in to the browser with the Identity Provider). If set to "true," the attribute “login=prompt” is added to the request.	true or false	No

`discoveryKind options`

`A token is usually signed by the OpenID Provider using a public certificate. Cora SeQuence should be able to obtain the public certificate to validate the OpenID Provider’s signature in the token.`

`There are four ways that  Cora SeQuence can discover the public certificate:`

discoveryKind attribute value	Description
OpenIdDiscoveryDocument	Indicates that the discovery is done using an OpenIdDiscoveryDocument . When this option is used, the `discoveryUri` attribute must be set to a Uri that contains this type of document.
JsonWebKeySet	Requires a Uri that returns a JsonWebKeySet result. This setting is a subsetting of the `OpenIdDiscoveryDocument` .
Federation	Requires a Uri that returns a WS-Federation metadata document. When this option is used, the `discoveryUri` attribute must be set to a Uri that contains this type of document.
Store	Use this option when the certificates are stored in the computer's local certificate store. When this option is used, the `IdentityProviderCertificates` element is required.

`identityProviderCertificate options`

`Under identityProviderCertificate, you configure where and how to retrieve certificates from the local certificate store. This is relevant only when setting Store under the discoveryKind attribute.`

Attribute	Description
name	A unique name for each certificate.

Attribute	Description
certificateFindType	Value based on the System.Security.Cryptography.X509Certificates.X509FindType enum.
certificateFindValue	The value to search for based on the `certificateFindType` attribute.
certificateLocation	Value based on the System.Security.Cryptography.X509Certificates.StoreLocation enum.
valid	Indicates that the certificate can be retrieved even if it is not valid (entire chain can not be validated).