OpenID Connect and OAuth 2.0 configuration

This guide outlines the configuration of SecureAuth IdP as an OpenID Connect Provider and OAuth 2.0 Authorization Server.

Supported OpenID Connect flows...

Authorization code

- Obtains the authorization code from the authorization endpoint and all tokens are returned from the token endpoint
- Returns an authorization code that can be exchanged for an identity token and / or access token
- Requires client authentication using a client ID and secret to retrieve tokens from the back end
- Permits long lived access through the use of refresh tokens

Implicit

- Obtains all tokens from the authorization endpoint
- Requests tokens without explicit client authentication
- Uses the redirect URI to verify the client identity
- Is not suitable for long lived access tokens; refresh tokens not supported
- ‘token’, ‘id_token’, ‘id_token token’

Hybrid

- Obtains the authorization code and tokens from the authorization endpoint, and request tokens from the token endpoint
- Lets the client make immediate use of an identity token and optionally retrieve an authorization code via one round trip to the authentication server
- Used for long lived access via the use of refresh tokens
- Clients using this flow must be able to maintain a secret
- ‘code id_token’, ‘code id_token token’, ‘code token’

Supported OAuth 2.0 flows...

Authorization code

- Obtains access tokens and refresh tokens
- Optimized for server-side applications in which source code is not publicly exposed
- Is a redirection-based flow; application interacts with the user-agent (user’s web browser) and receives API authorization codes routed through the user-agent
- Proof Key for Code Exchange (PKCE) support added in SecureAuth IdP 9.3 to mitigate the threat of the authorization code interception attack

**Implicit**
- Obtains access tokens; refresh tokens not supported
- Optimized for public clients operating a particular redirection URI; client secret confidentiality not guaranteed
- Is a redirection-based flow; access token is given to the user-agent which passes it to the application (identity of the application is not authenticated)

**Resource owner password credentials**
- Used as an authorization grant to obtain an access token; user provides service credentials (username and password) directly to the application for an access token
-Eliminates the need for the client to store the resource owner credentials by exchanging credentials for a long-lived access token or refresh token
-Should only be used on an application trusted by the user (user's desktop OS, or an application owned by the service)

**Client credentials**
- Used as an authorization grant when the authorization scope is limited to protected resources under control of the client or the authorization server
- Tokens are requested on behalf of a client, not a user, and requests are sent to the token endpoint

**Refresh token**
- Used for obtaining a new access token after the original access token expires or becomes invalid
- Issued to the client by the authorization server
- Lets users request a fresh access token from the authorization server for long-lived access to APIs

---

**Supported endpoint types...**

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization endpoint</td>
<td>Resides on the authorization server on which the resource owner authenticates for access to the client application.</td>
<td>https://.../secureauth1/secureauth.aspx</td>
</tr>
<tr>
<td>Token endpoint</td>
<td>Resides on the authorization server on which the client application exchanges information for an access token.</td>
<td>https://.../secureauth1/oidctoken.aspx</td>
</tr>
<tr>
<td>Endpoint Type</td>
<td>Description</td>
<td>URL</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>UserInfo endpoint</td>
<td>Used to retrieve a user's identity information.</td>
<td>https://.../secureauth1/oidcuserinfo.aspx</td>
</tr>
<tr>
<td>End session endpoint</td>
<td>Used to trigger a user's single sign-out by redirecting the user's browser to the end session URL.</td>
<td>https://.../secureauth1/oidcendsession.aspx</td>
</tr>
<tr>
<td>Check session iFrame endpoint</td>
<td>Used to verify if the user's session on a server using iFrames has ended so necessary cleanup can be performed.</td>
<td>https://.../secureauth1/oidcchecksession.aspx</td>
</tr>
<tr>
<td>Discovery endpoint</td>
<td>Used to retrieve the discovery document containing information clients might use for token validation.</td>
<td>https://.../secureauth1/.well-known/openid-configuration</td>
</tr>
<tr>
<td>Token introspection endpoint</td>
<td>Lets the holder of an access token request token verification information from the issuing provider.</td>
<td>https://.../secureauth1/OAuthTokenIntrospect.aspx</td>
</tr>
<tr>
<td>Token revocation endpoint</td>
<td>Revokes user access tokens and refresh tokens.</td>
<td>https://.../secureauth1/OAuthTokenRevocate.aspx</td>
</tr>
</tbody>
</table>

What's new in SecureAuth IdP version 9.3

- All supported flows have been updated to current specifications.
- PKCE support for OAuth 2.0 authorization code added for native and mobile app security. The "Use with PKCE Protocol" check box appears to the right of the Authorization Code field in the Clients section.

Previous version of OpenID Connect and OAuth 2.0 guide

See OpenID Connect and OAuth 2.0 Guide (version 9.1 and 9.2) for the previous version of this guide.

Prerequisites

- SecureAuth IdP version 9.3 or later
- Knowledge of OpenID Connect and OAuth 2.0
- SecureAuth IdP realm or integrated application with the following configured:
  - Data tab / Directory integrations

On the New Experience user interface in version 9.3, you can configure an Active Directory integration or SQL Server integration to be applied to applications made from App onboarding library templates. Configure the remaining components – for example, Workflow, Multi-Factor Methods, and Adaptive Authentication tabs – on the Classic Experience user interface.
SecureAuth IdP Web Admin - Classic Experience configuration

1. Once the directory integration is successful, go to the **Data** tab.
2. In the **Profile Fields** section, map the attributes from the profile fields to the SecureAuth IdP Profile Properties that will be used as OpenID Connect Claims. The standard OpenID Connect Claims supported by SecureAuth IdP can be used as a reference.

3. **Save** the configuration.
4. Go to the **Post Authentication** tab.
5. In the **Post Authentication** section, set the following:

<table>
<thead>
<tr>
<th>Authenticated User Redirect</th>
<th>Set to OpenID Connect/OAuth2.</th>
</tr>
</thead>
</table>

6. In the **OpenID Connect / OAuth 2.0 - Settings** section, set the following:

<p>| Enabled | Set security enhancement to enable (True) or disable (False) OpenID Connect and OAuth2 endpoints. |</p>
<table>
<thead>
<tr>
<th><strong>Issuer</strong></th>
<th>Set the value used in the 'iss' claim.</th>
</tr>
</thead>
</table>
| **Signing Algorithm** | Set the signing algorithm used for signing JSON web tokens as one of the following:  
  - **RSA SHA256** uses the X.509 certificate selected for Signing Cert  
  - **HMAC SHA256** uses the client secret for signing |
| **Signing Cert** | Certificate used to sign JSON Web Tokens produced by SecureAuth IdP. |
| **Authorization Code Lifetime** | Length of time for Authorization Code lifetime in minutes. |
| **Access Token Lifetime** | Length of time for Access Token lifetime in hours.  
  The system is set with a default clock skew at 5 minutes (see https://docs.microsoft.com/en-us/dotnet/api/microsoft.identitymodel.tokens.tokenvalidationparameters.defaultclockskew?view=azure-dotnet).  
  For example, if you set the Access Token Lifetime with a value of .25 for an intended token expiration at 15 minutes, it will actually expire at 20 minutes due to the default clock skew. |
<p>| <strong>Refresh Token Lifetime</strong> | Length of time for Refresh Token lifetime in hours. |</p>
<table>
<thead>
<tr>
<th><strong>Auto Accept User Consent</strong></th>
<th>When set to <strong>True</strong>, users are not prompted to grant consent to a client for a given request. Acceptance is assumed granted and tokens are issued.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable User Consent Storage</strong></td>
<td>When set to <strong>True</strong>, user consent granted to a client is stored as an encrypted and compressed string value in the attribute specified at <strong>Consent Storage Attribute</strong>.</td>
</tr>
<tr>
<td><strong>Consent Storage Attribute</strong></td>
<td>Data store attribute mapped to profile Property which saves the user's consent. For example, the Aux ID 10 field from the profile Property corresponds with the value set in the Consent Storage Attribute field and support a varying character length.</td>
</tr>
</tbody>
</table>

Sample image from Profile Fields section of the Data tab...

For example, the Aux ID 10 field from the profile Property corresponds with the value set in the Consent Storage Attribute field.
7. In the **OpenID Connect / OAuth 2.0 - Scopes** section, by default, a set of OpenID Connect scopes are preconfigured and required in certain OpenID Connect flows. To add a scope, click **Add Scope** and set the following:

<table>
<thead>
<tr>
<th>Scope</th>
<th>Value passed to the endpoints during authorization requests. This value must be URL-safe and not include spaces.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>User-friendly display name in list of scopes from which access is requested on the user consent page.</td>
</tr>
<tr>
<td>Description</td>
<td>User-friendly description of scope in list of scopes from which access is requested on the user consent page.</td>
</tr>
<tr>
<td>Discoverable</td>
<td>When the check box is selected, the associated scope value is listed at the Discovery Configuration endpoint.</td>
</tr>
</tbody>
</table>

8. **Save** the configuration.
9. In the **OpenID Connect / OAuth 2.0 - Clients** section, click **Add Client**.

![Add Client](image)

10. In **OpenID Connect / OAuth 2.0 - Client Details**, set the following:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>Security enhancement to enable (True) or disable (False) the client.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>User-friendly name of the client requesting access to display on the user consent page.</td>
</tr>
<tr>
<td><strong>Client ID</strong></td>
<td>Automatically created unique identifier of the client.</td>
</tr>
<tr>
<td><strong>Client Secret</strong></td>
<td>Automatically created unique secret of the client.</td>
</tr>
<tr>
<td><strong>JSON Web Encryption</strong></td>
<td>Indicate whether the JSON web encryption settings for the client is Enabled or Disabled.</td>
</tr>
<tr>
<td><strong>JSON Web Key URI</strong></td>
<td>When JSON Web Encryption is enabled for the client, enter the web key URL.</td>
</tr>
</tbody>
</table>

**Allowed Flows** - To indicate which flows the client is allowed to use, set to True or False

- **Authorization Code**
  - Indicate whether client can use the Authorization Code flow.
  - With the True setting, select the **Use with PKCE Protocol** check box to enable secure access to native and mobile apps using an Authorization Code flow with PKCE.

- **Implicit**
  - Indicate whether client can use the Implicit flow.

- **Hybrid**
  - Indicate whether client can use the Hybrid flow.

- **Client Credentials**
  - Indicate whether client can use the Client Credentials flow.

- **Resource Owner**
  - Indicate whether client can use the Resource Owner flow.

- **Refresh Token**
  - Indicate whether client can use the Refresh Token flow.

- **Introspection**
  - Indicate whether client can use the Introspection flow.

- **Revocation**
  - Indicate whether client can use the Revocation flow.
11. Save the configuration.
12. OPTIONAL. In the OpenID Connect / OAuth 2.0 - Client Scope Restrictions section, to restrict a client from using one or more of the defined Scopes, click Add Restricted Scope and add the Scope value.
OpenID Connect / OAuth 2.0 - Client Redirect URIs section, to let a client use the Authorization Code or Implicit flows, click Add Redirect URI and add the allowed URI value (must include https)

14. Save your edits before leaving the Clients page.
15. In the OpenID Connect ID Token Claims section, do the following:
   a. Map the Claim to the Profile Property. A value of Unmapped indicates the claim will not be included in the produced JSON Web Token.
b. To list the associated Claim at the Discovery Configuration endpoint, select the Discoverable check box.

16. In the Open ID Connect ID Token Custom Claims section, to create Custom Claims, click Add Custom Claim and set the following:

<table>
<thead>
<tr>
<th>Claim</th>
<th>Provide unique name for the Custom Claim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Property</td>
<td>Select a Profile Property to complete the mapping.</td>
</tr>
</tbody>
</table>

SecureAuth supports creating Custom Claims to include in the 'id_token' for OpenID Connect flows. Claim names cannot include spaces.

17. Save the configuration.

Application X configuration steps

Use the Client ID and Client Secret from the SecureAuth IdP Web Admin to configure applications that are OpenID Connect / OAuth 2.0 ready.
on how many OI D C clients your organization expects to connect to.

Ex
a
m
ple:

"h
ou
sel
de
ntif
ier"

ha
s
a
m
axi
m
u
m
val
ue
of
32
76
8
byt
es.
so
it
could
th
eo
ret
ica
lly
sto
re
54
2
diff
er
en
t
t
co
ts
en
ts.
(3
27
68
mi
nu
s
27
7,
then
div
ide
d
by
60
=
54
2)