Identity Cloud Guide

ON THIS PAGE

Identity Cloud Guide

ForgeRock® Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

This guide is for ForgeRock Identity Cloud evaluators, administrators, and architects. It provides examples of how to integrate your business application and APIs with Identity Cloud for Single Sign-On and API Security, with ForgeRock Identity Gateway.

Example Installation for This Guide

Unless otherwise stated, the examples in this guide assume the following installation:

- Identity Gateway installed in standalone mode, on `http://openig.example.com:8080`, as described in Download and Start IG in Standalone Mode.
- Sample application installed on `http://app.example.com:8081`, as described in Downloading and Starting the Sample Application.
- The ForgeRock Identity Cloud, with the default configuration, as described in the ForgeRock Identity Cloud Docs.

  When you are using the ForgeRock Identity Cloud, you need to know the value of the following properties:

  - The root URL of your ForgeRock Identity Cloud. For example, `https://myTenant.forgeblocks.com`.

The URL of the Access Management component of the ForgeRock Identity Cloud is the root URL of your Identity Cloud followed by `/am`. For example, `https://myTenant.forgeblocks.com/am`.

- The realm where you work. The examples in this document use `alpha`.

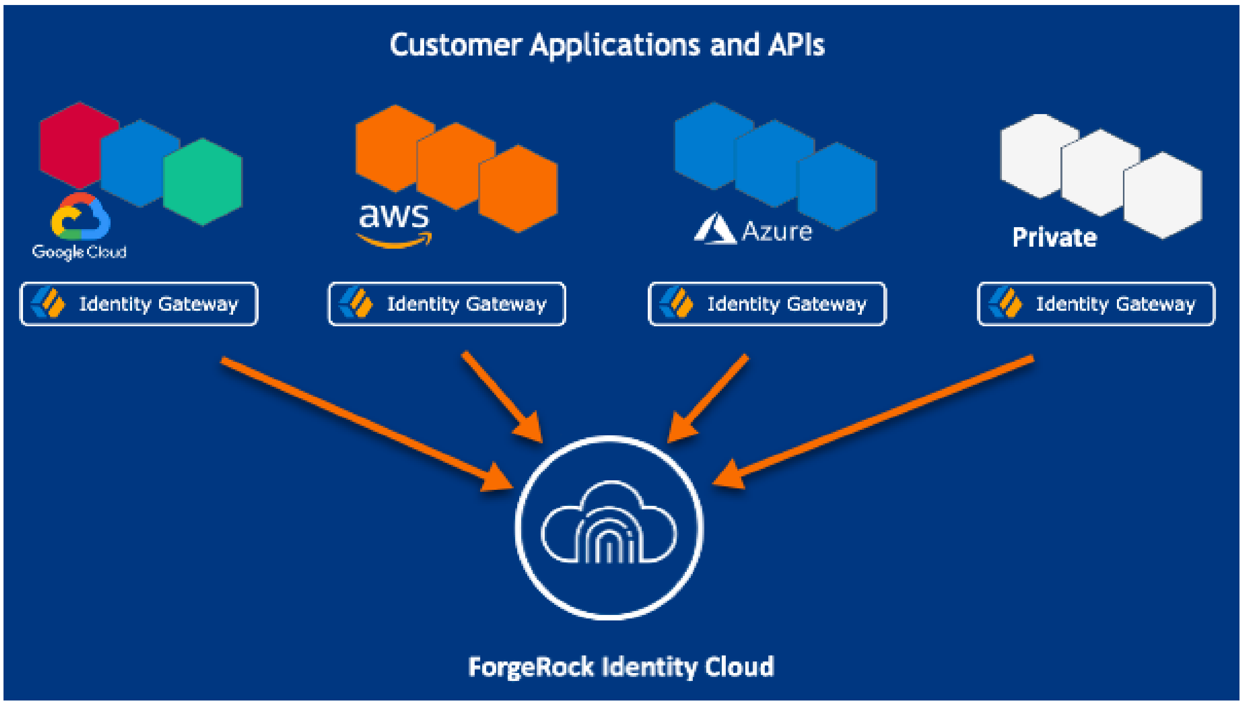
  Prefix each realm in the hierarchy with the `realms` keyword. For example, `/realms/root/realms/alpha`.

If you use a different configuration, substitute in the procedures accordingly.

About Identity Gateway and the ForgeRock Identity Cloud

ForgeRock Identity Cloud simplifies the consumption of ForgeRock as an Identity Platform. However, many organizations have business web applications and APIs deployed across multiple clouds, or on-premise.

Identity Gateway facilitates non-intrusive integration of your web applications and APIs with the Identity Cloud, for SSO and API Security. The following image illustrates how Identity Gateway bridges your business to the ForgeRock Identity Cloud:



For information about the ForgeRock Identity Cloud, see the ForgeRock Identity Cloud Docs.

API Security With OAuth 2.0 and the ForgeRock Identity Cloud

This example sets up OAuth 2.0, using the standard introspection endpoint, where ForgeRock Identity Cloud is the authorization server, and Identity Gateway is the resource server.

For more information about Identity Gateway as an OAuth 2.0 resource server, see Validate Access_Tokens Through the Introspection Endpoint.

Before you start, prepare Identity Cloud and Identity Gateway as described in Example Installation for This Guide.

1. Set up Identity Cloud:
   a. Log in to the ForgeRock Identity Cloud as an administrator.
   b. In the platform console, go to Identities > Manage > Alpha realm - Users, and add a new user with the following values:
      - **Username** : `demo`
      - **First name** : `demo`
      - **Last name** : `user`
      - **Email Address** : `demo@example.com`
      - **Password** : `Ch4ng3!t`
   c. Make sure that you are managing the `alpha` realm. If not, click the current realm at the top of the screen, and switch realm.
   d. Add a web application:
      i. In the platform console, click Applications > + Add Application > Web, and add a web application with the following values:
         - **Client ID** : `oauth2-client`
         - **Client Secret** : `password`
      ii. On the application page, add the following general settings:
         - **Grant Types** : `Resource owner Password Credentials`
         - **Scopes** : `mail`
   e. Click Gateways & Agents, and add an agent profile with the following values:
      - **ID** : `ig_agent`
      - **Password** : `password`

      By default, the agent can introspect OAuth 2.0 tokens issued to any client, in the realm and subrealm where it is created. To change the introspection, click Native Consoles > Access Management, and update the agent in the AM console.

2. Set up Identity Gateway:

   a. Set an environment variable for the IG agent password, and then restart IG:

   ```
   $ export AGENT_SECRET_ID='cGFzc3dvcmQ='
   ```

   The password is retrieved by a SystemAndEnvSecretStore, and must be base64-encoded.

   b. Add the following route to IG, to serve .css and other static resources for the sample application:

      1. Linux

      2. Windows

   ```
   $HOME/.openig/config/routes/static-resources.json
   ```

   ```
   %appdata%\OpenIG\config\routes\static-resources.json
   ```

   ```
   {
     "name" : "sampleapp-resources",
     "baseURI" : "http://app.example.com:8081",
     "condition": "${find(request.uri.path,'^/css')}",
     "handler": "ReverseProxyHandler"
   }
   ```

   c. Add the following route to Identity Gateway, replacing the value for the property `amInstanceUrl`:

      1. Linux

      2. Windows

   ```
   $HOME/.openig/config/routes/oauth2rs-idc.json
   ```

   ```
   %appdata%\OpenIG\config\routes\oauth2rs-idc.json
   ```

   ```
   {
     "name": "oauth2rs-idc",
     "baseURI": "http://app.example.com:8081",
     "condition": "${find(request.uri.path, '^/oauth2rs-
   idc')}",
     "properties": {
       "amInstanceUrl": "<myIdentityCloudUrl/am>"
   ```

```
  },
  "heap": [
    {
      "name": "SystemAndEnvSecretStore-1",
      "type": "SystemAndEnvSecretStore"
    },
    {
      "name": "AmService-1",
      "type": "AmService",
      "config": {
        "url": "&{amInstanceUrl}",
        "realm": "/alpha",
        "version": "7.1",
        "agent": {
          "username": "ig_agent",
          "passwordSecretId": "agent.secret.id"
        },
        "secretsProvider": "SystemAndEnvSecretStore-1"
      }
    }
  ],
  "handler": {
    "type": "Chain",
    "config": {
      "filters": [
        {
          "name": "OAuth2ResourceServerFilter-1",
          "type": "OAuth2ResourceServerFilter",
          "config": {
            "scopes": [
              "mail"
            ],
            "requireHttps": false,
            "realm": "OpenIG",
            "accessTokenResolver": {
              "name":
"TokenIntrospectionAccessTokenResolver-1",
              "type":
"TokenIntrospectionAccessTokenResolver",
              "config": {
                "amService": "AmService-1",
                "providerHandler": {
                  "type": "Chain",
                  "config": {
                    "filters": [
                    {
                      "type":
"HttpBasicAuthenticationClientFilter",
                      "config": {
                        "username": "ig_agent",
                        "passwordSecretId":
"agent.secret.id",
                        "secretsProvider":
"SystemAndEnvSecretStore-1"
                      }
                    }
                  ],
                  "handler": "ForgeRockClientHandler"
                }
              }
            }
          }
        }
      }
    ],
    "handler": {
      "type": "StaticResponseHandler",
      "config": {
        "status": 200,
        "headers": {
          "Content-Type": [ "text/html" ]
        },
        "entity": "<html><body><h2>Decoded
access_token: ${contexts.oauth2.accessToken.info}</h2>
</body></html>"
      }
    }
  }
}
```

Notice the following features of the route compared to `rs-introspect.json` in Validate Access Tokens Through the Introspection Endpoint, where a local Access Management instance is the authorization server:

- The AmService `URL` points to Access Management in the Identity Cloud.
- The AmService `realm` points to the realm where you have configured your web application and the IG agent.

3. Test the setup:

   a. In a terminal, export an environment variable for URL of Access Management in the Identity Cloud:

   ```
   $ export amInstanceUrl='myAmInstanceUrl'
   ```

   b. Use a **`curl`** command similar to the following to retrieve an access_token:

   ```
   $ mytoken=$(curl -s \
   --user "oauth2-client:password" \
   --data
   'grant_type=password&username=demo&password=Ch4ng3!t&sc
   ope=mail' \
   $amInstanceUrl/oauth2/realms/alpha/access_token | jq -r
   ".access_token")
   ```

   c. Validate the access_token returned in the previous step:

   ```
   $ curl -v http://openig.example.com:8080/oauth2rs-idc -
   -header "Authorization: Bearer ${mytoken}"

   {
     active = true,
     scope = mail,
     realm = /alpha,
     client_id = oauth2-client,
     ...
   }
   ```

Single Sign-On With OpenID Connect and the ForgeRock Identity Cloud

This example sets up ForgeRock Identity Cloud as an OpenID Connect identity provider, and Identity Gateway as a relying party.

For more information about Identity Gateway and OpenID Connect, see Act As an OpenID Connect Relying Party.

Before you start, prepare Identity Cloud, Identity Gateway, and the sample application as described in Example Installation for This Guide.

1. Set up Identity Cloud:
   a. Log in to the ForgeRock Identity Cloud as an administrator.
   b. In the platform console, go to Identities > Manage > Alpha realm - Users, and add a new user with the following values:
      - **Username** : `demo`
      - **First name** : `demo`
      - **Last name** : `user`
      - **Email Address** : `demo@example.com`
      - **Password** : `Ch4ng3!t`
   c. Make sure that you are managing the `alpha` realm. If not, click the current realm at the top of the screen, and switch realm.
   d. Add a web application:
      i. In the platform console, click Applications > ✚ Add Application > Web, and add a web application with the following values:
         - **Client ID** : `oidc-client`
         - **Client Secret** : `password`
      ii. In **General Settings** on the application page, add the following values:
         - **Sign-in URLs** : `http://openig.example.com:8080/home/id_token/callback`
         - **Grant Types** : `Authorization Code` , `Resource owner Password Credentials`
         - **Scopes** : `openid` , `profile` , `mail`
      iii. Click **Show advanced settings** > **Authentication**, and click **Implied Consent** :

         The resource owner is not asked for consent during authorization flows.
2. Set up Identity Gateway:
   a. Set an environment variable for the `oidc-client` password, and then restart IG:

```
$ export CLIENT_SECRET_ID='cGFzc3dvcmQ='
```

   a. Add the following route to IG, to serve .css and other static resources for the sample application:
      1. Linux
      2. Windows

```
$HOME/.openig/config/routes/static-resources.json
```

```
%appdata%\OpenIG\config\routes\static-resources.json
```

```
{
  "name" : "sampleapp-resources",
  "baseURI" : "http://app.example.com:8081",
  "condition": "${find(request.uri.path,'^/css')}",
  "handler": "ReverseProxyHandler"
}
```

b. Add the following route to Identity Gateway, replacing the value for the property `amInstanceUrl`:

   1. Linux

   2. Windows

```
$HOME/.openig/config/routes/oidc-idc.json
```

```
%appdata%\OpenIG\config\routes\oidc-idc.json
```

```
{
  "name": "oidc-idc",
  "baseURI": "http://app.example.com:8081",
  "condition": "${find(request.uri.path,
'^/home/id_token')}",
  "properties": {
    "amInstanceUrl": "<myIdentityCloudUrl/am>"
  },
  "heap": [
    {
      "name": "SystemAndEnvSecretStore-1",
      "type": "SystemAndEnvSecretStore"
    }
  ],
  "handler": {
    "type": "Chain",
    "config": {
      "filters": [
        {
          "name": "OAuth2ClientFilter-1",
          "type": "OAuth2ClientFilter",
          "config": {
            "clientEndpoint": "/home/id_token",
            "failureHandler": {
              "type": "StaticResponseHandler",
              "config": {
                "status": 500,
                "headers": {
                  "Content-Type": [
                    "text/plain"
                  ]
                },
                "entity": "Error in OAuth 2.0 setup."
              }
            },
            "registrations": [
              {
                "name": "oauth2-client",
                "type": "ClientRegistration",
                "config": {
                  "clientId": "oidc-client",
                  "clientSecretId": "client.secret.id",
                  "issuer": {
                    "name": "Issuer",
                    "type": "Issuer",
                    "config": {
                      "wellKnownEndpoint": "&
{amInstanceUrl}/oauth2/realms/alpha/.well-known/openid-
configuration"
                    }
                  },
                  "scopes": [
                    "openid",
                    "profile",
                    "mail"
                  ],
                  "secretsProvider":
"SystemAndEnvSecretStore-1",
                  "tokenEndpointAuthMethod":
"client_secret_basic"
                }
              }
            ],
            "requireHttps": false,
            "cacheExpiration": "disabled"
          }
          }
        ],
        "handler": "ReverseProxyHandler"
      }
    }
  }
```

Notice the following features of the route compared to `07-openid.json` in Use AM As a Single OpenID Connect Provider, where Access Management is running locally:

- The ClientRegistration `wellKnownEndpoint` points to the Identity Cloud.

3. Test the setup:
   a. Go to http://openig.example.com:8080/home/id_token. The Identity Cloud login page is displayed.
   b. Log in to Identity Cloud as user `demo`, password `Ch4ng3!t`. The home page of the sample application is displayed.

Cross-Domain Single Sign-On With the ForgeRock Identity Cloud

For organizations relying on AM's session and policy services with SSO, consider cross-Domain Single Sign-On (CDSSO) as an alternative to SSO through OpenID Connect.

This example sets up ForgeRock Identity Cloud as an SSO authentication server for requests processed by Identity Gateway. For more information about about Identity Gateway and CDSSO, see Authenticate With CDSSO.

Before you start, prepare AM, IG, and the sample application, as described in Download and Start IG in Standalone Mode.

1. Set up Identity Cloud:
   a. Log in to the ForgeRock Identity Cloud as an administrator.
   b. In the platform console, go to Identities > Manage > Alpha realm - Users, and add a new user with the following values:
      - **Username** : `demo`
      - **First name** : `demo`
      - **Last name** : `user`
      - **Email Address** : `demo@example.com`

        - **Password** : `Ch4ng3!t`

    c. Make sure that you are managing the `alpha` realm. If not, click the current realm at the top of the screen, and switch realm.

    d. Select **Applications** > **Agents** > **Identity Gateway**, add an agent with the following values:

        - **Agent ID**: `ig_agent_cdsso`
        - **Password**: `password`
        - **Redirect URL for CDSSO**: `https://openig.ext.com:8443/home/cdsso/redirect`

2. Set up Identity Gateway:

    a. Set up IG for HTTPS, as described in Configure IG For HTTPS (Server-Side) in Standalone Mode.

    b. Add the following `session` configuration to `admin.json`, to ensure that the browser passes the session cookie in the form-POST to the redirect endpoint (step 6 of Information Flow During CDSSO):

```
{
  "connectors": […],
  "session": {
    "cookie": {
      "sameSite": "none",
      "secure": true
    }
  },
  "heap": […]
}
```

    This step is required for the following reasons:

        - When `sameSite` is `strict` or `lax`, the browser rejects the session cookie, which contains the nonce used in validation. If IG doesn't find the nonce, it assumes that it didn't originate the authorization request.
        - When `secure` is `false`, the browser is likely to reject the session cookie.

          For more information, see admin.json.

    c. Set an environment variable for the IG agent password, and then restart IG:

```
$ export AGENT_SECRET_ID='cGFzc3dvcmQ='
```

    The password is retrieved by a SystemAndEnvSecretStore, and must be base64-encoded.

d. Add the following route to IG, to serve .css and other static resources for the sample application:

   1. Linux

   2. Windows

```
$HOME/.openig/config/routes/static-resources.json
```

```
%appdata%\OpenIG\config\routes\static-resources.json
```

```
{
  "name" : "sampleapp-resources",
  "baseURI" : "http://app.example.com:8081",
  "condition": "${find(request.uri.path,'^/css')}",
  "handler": "ReverseProxyHandler"
}
```

e. Add the following route to Identity Gateway, replacing the value for the property `amInstanceUrl`:

   1. Linux

   2. Windows

```
$HOME/.openig/config/routes/cdsso-idc.json
```

```
%appdata%\OpenIG\config\routes\cdsso-idc.json
```

```
{
  "name": "cdsso-idc",
  "baseURI": "http://app.example.com:8081",
  "condition": "${find(request.uri.path,
'^/home/cdsso')}",
  "properties": {
    "amInstanceUrl": "<myIdentityCloudUrl/am>"
  },
  "heap": [
    {
      "name": "SystemAndEnvSecretStore-1",
      "type": "SystemAndEnvSecretStore"
    },
    {
      "name": "AmService-1",
        "type": "AmService",
        "config": {
          "url": "&{amInstanceUrl}",
          "realm": "/alpha",
          "version": "7.1",
          "agent": {
            "username": "ig_agent_cdsso",
            "passwordSecretId": "agent.secret.id"
          },
          "secretsProvider": "SystemAndEnvSecretStore-1",
          "sessionCache": {
            "enabled": false
          }
        }
      }
    ],
    "handler": {
      "type": "Chain",
      "config": {
        "filters": [
          {
            "name": "CrossDomainSingleSignOnFilter-1",
            "type": "CrossDomainSingleSignOnFilter",
            "config": {
              "redirectEndpoint": "/home/cdsso/redirect",
              "authCookie": {
                "path": "/home",
                "name": "ig-token-cookie"
              },
              "amService": "AmService-1",
              "verificationSecretId": "verify",
              "secretsProvider": {
                "type": "JwkSetSecretStore",
                "config": {
                  "jwkUrl": "&
{amInstanceUrl}/oauth2/realms/alpha/connect/jwk_uri"
                }
              }
            }
          }
        ],
        "handler": "ReverseProxyHandler"
      }
    }
}
```

Notice the following features of the route compared to `cdsso.json` in gateway-guide:sso.adoc#proc-cdsso, where Access Management is running locally:

- The AmService `URL` points to Access Management in the Identity Cloud.
- The AmService `realm` points to the realm where you configure your IG agent.

3. Test the setup:

   a. Go to https://openig.ext.com:8443/home/cdsso. The Identity Cloud login page is displayed.

      If you see warnings that the site is not secure, respond to the warnings to access the site.

   b. Log in to Identity Cloud as user `demo`, password `Ch4ng3!t`.

      Access Management calls `/home/cdsso/redirect`, and includes the CDSSO token. The CrossDomainSingleSignOnFilter passes the request to sample app.