User Account and Authentication Service

Contents

Copyright GE Digital

UAA Service Overview

About the User Account and Authentication Security Service

User Account and Authentication is a security service available in the Predix marketplace.

The User Account and Authentication (UAA) service is the primary authentication service on the Predix platform. It enables developers to add user authentication and authorization capabilities to their application. Application developers can obtain a UAA instance from the Predix marketplace and configure the instance to authenticate trusted users and clients for their application. UAA service offers a virtual OAuth2 server to customers to issue and validate tokens for client applications.

UAA Application or Predix UAA allows Predix customers and developers to create virtual UAA instances with the available controls and configurations. Each UAA instance acts as a separate OAuth server. Predix UAA also offers a UAA Dashboard to manage the UAA instance administrative activities, RESTful endpoints, and a command line (UAAC) tool.

Note: All Predix platform services require a UAA instance to ensure secure access to the service.

UAA includes the following features:

- Identity management through SCIM APIs.
- A complete OAuth 2.0 authorization server.
- Login and logout services for UAA authentication.
- SAML federation capabilities to meet third-party SAML identity provider requirements.

The UAA Server supports the APIs described in the open source UAA documentation .

Additional Information

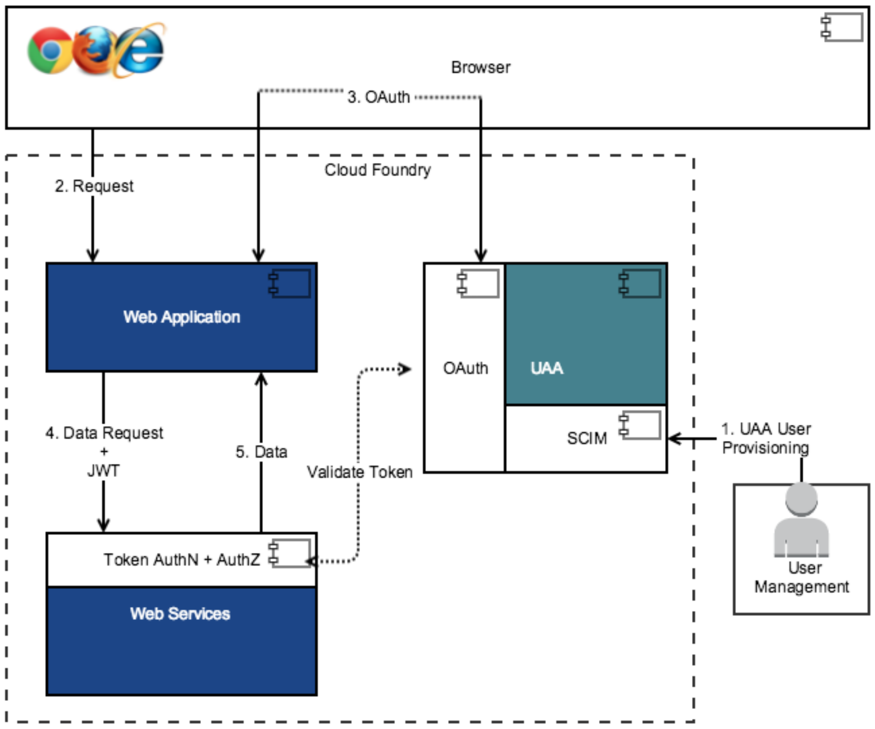
Exploring Security Services Guides

UAA Deployment Architecture

The following topologies are supported for using the UAA service:

- Local identity management in the User Account and Authentication (UAA) server.
- Federated identity management.
- Federated identity management with additional identity management using UAA. In this topology the users are white-listed based on setup in federated identity store and UAA.

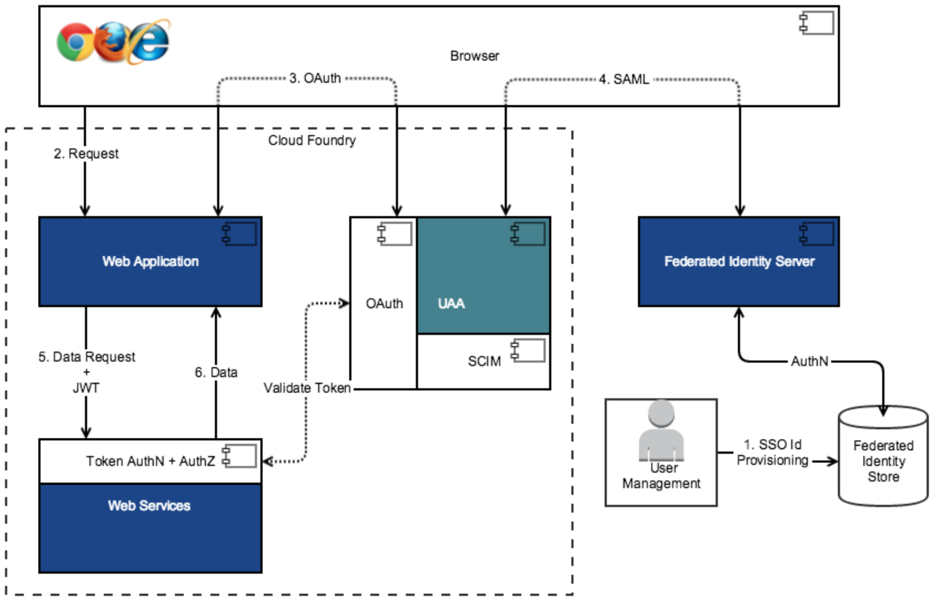
The following diagram shows how authentication in local identity management in the User Account and Authentication (UAA) server.



In this flow:

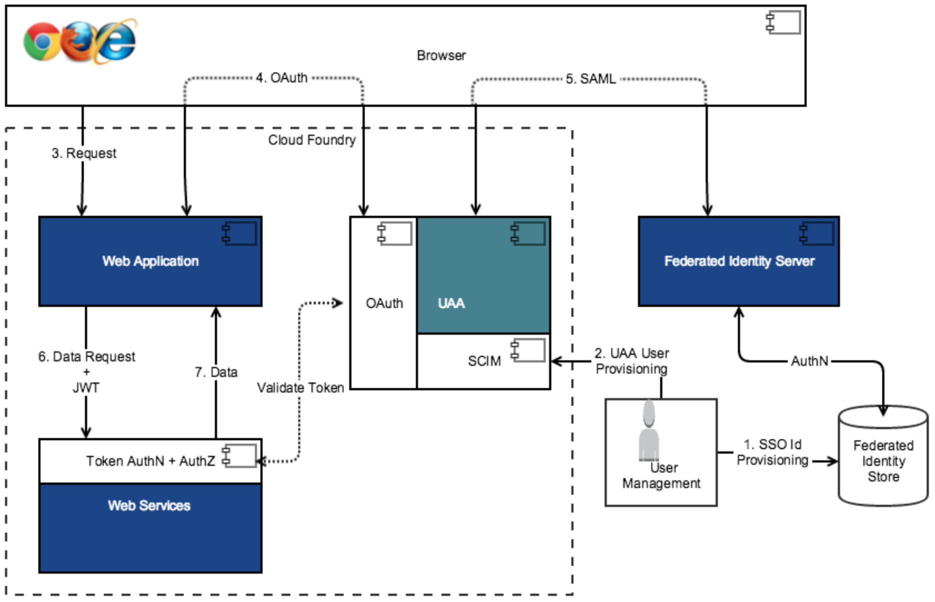
1. An administrator provisions the users through SCIM APIs in UAA.
2. An application user requests data using a browser.
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required.
4. The web application passes the data request and JWT token to the web services.
5. Once the user is authorized, the data is returned to the web application.

The following diagram shows how UAA integrated with federated identity management.



In this flow, instead of performing local authentication using UAA, the users are authenticated using the federated identity store.

The following diagram shows UAA integrated with federated identity management in conjunction with local identity management.



This flow demonstrates authentication using both the UAA service and the federated identity service. The UAA uses the SAML request to communicate with the federated identity services. When an authentication request is generated for UAA, UAA checks the users against the UAA and the federated identity store. This setup is useful for whitelisting the users that can be authenticated. For example, if an organization has a large set of users set up in their federated identity store, but they want only a subset of users to be able to access the services, then they can use this setup to identify the user subset.

Understanding UAA and OAuth2 Access Token Flows

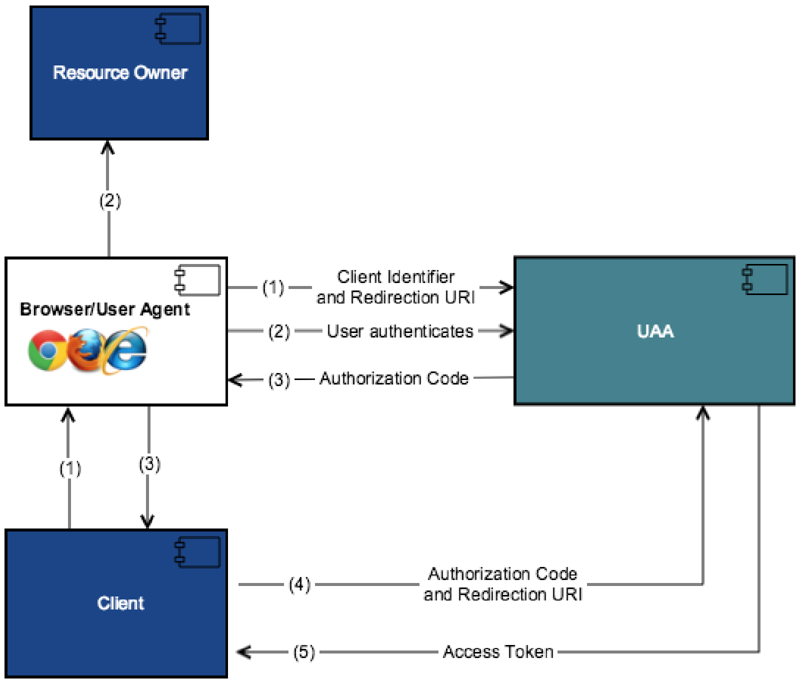
Obtaining Tokens Using Authorization Code Grant

When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code. The OAuth2 endpoint in UAA accepts authorization code to provide an Access Token.

When you use the Authorization Code grant type, the token is stored and managed on the server's side. Therefore any technical information present in the token (such as scope names), is not leaked to the user and the token is less prone to be misused. Additionally, with Authorization Code grant type, it is easier to terminate a user's session without waiting for the token to expire its lifetime. Authorization Code security relies on a robust session management scheme that includes best practices such as HTTP protocol hardening using HTTP security headers, HTTP Cookie security flags, adequate session lifetime, and secure session termination (sign-out mechanism).

For more details on the Authorization Grant type, see RFC 6749.

The following diagram shows the high-level flow for obtaining access tokens using authorization grant.



1. The client requests access to a resource. The request includes client identifier, requested scope, local state, and a redirection URI.
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request.

3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step.
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step.
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.

For detailed information on UAA endpoints used to generate authorization code and Access Tokens, see open-source UAA documentation.

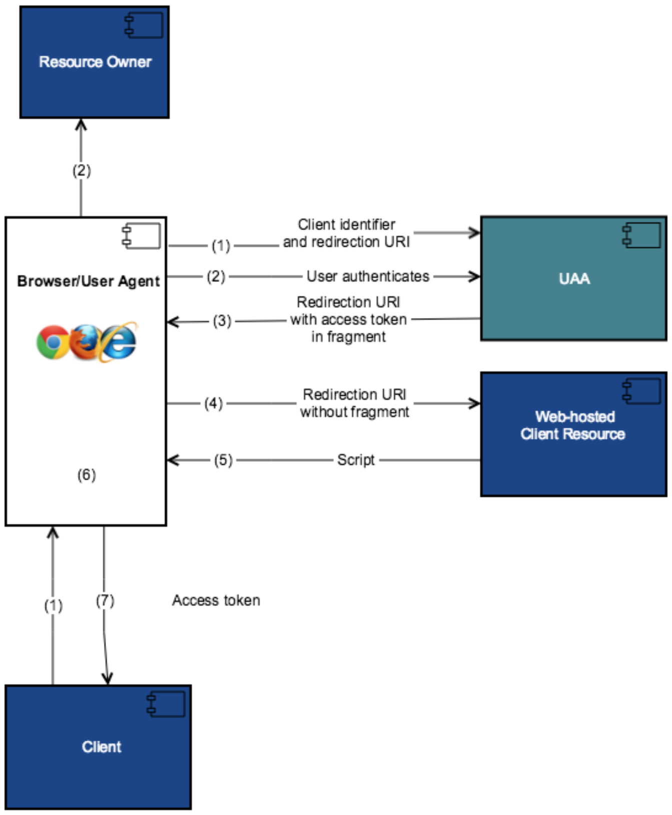
Obtaining Tokens Using Implicit Grant

When you use the implicit grant type, UAA directly issues an Access Token to the client without authenticating the client. This reduces the number of round trips required to obtain an access token. However with this grant type, the access token is transmitted in the URI fragment, which can expose it to unauthorized parties. Additionally if the public clients use implicit flows, UAA cannot determine what client an access token was issued to.

The benefit of using implicit grants is that it improve the responsiveness and efficiency of some clients (for example, a client implemented as an in-browser application). However, you must weigh this benefit against the security implications of using implicit grants.

For more details on the Implicit Grant type, see RFC 6749.

The following diagram shows the high-level flow for obtaining access tokens using implicit grant type.


Resource Owner
(2)
Browser/User Agent
(1) Client identifier and redirection URI
(2) User authenticates
(3) Redirection URI with access token in fragment
UAA
(4) Redirection URI without fragment
(5) Script
Web-hosted Client Resource
(6)
(1)
(7)
Access token
Client

1. The client sends a request to the authorization endpoint in UAA. The request includes client identifier, requested scope, local state, and a redirection URI. Redirection URI is required for UAA to send the Access Token.
2. UAA authenticates the resource owner and establishes whether the resource owner grants or denies the client's access request.
3. If the resource owner grants access, UAA redirects the user-agent back to the client using the redirection URI specified in step 1. The redirection URI includes the access token in the URI fragment.
4. The user-agent makes a request to the web-hosted client resource. The user-agent does not pass the URI fragment information to the client resource.
5. The web-hosted client resource returns a web page that can access the full redirection URI including the URI fragment retained by the user-agent, and can extract the access token contained in the URI fragment.
6. The user-agent extracts the access token. Often extraction includes executing the script provided by the web-hosted client resource locally.
7. The user-agent passes the access token to the client.

For detailed information on UAA endpoints used to generate Access Tokens with implicit grant type, see open-source UAA documentation.

Obtaining Tokens Using Resource Owner Password Credentials Grant

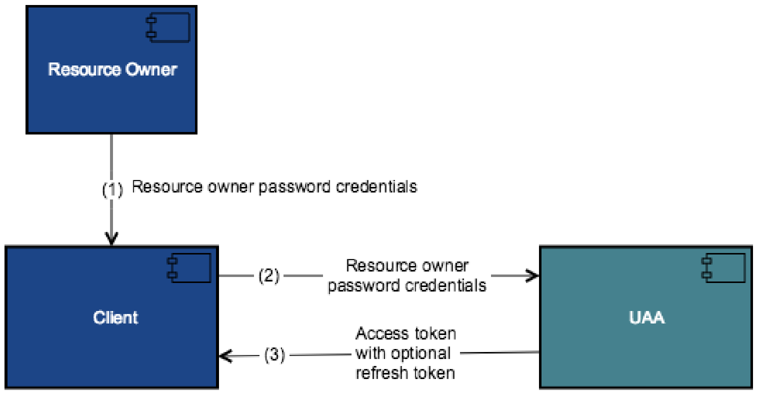
When you use the resource owner password credentials grant type, the OAuth2 endpoint in UAA accepts the username and password and provides Access Tokens.

You must use this grant type only when there is a high degree of trust between the resource owner and the client (for example, the client is part of the device operating system or a highly privileged application), and when other authorization grant types such as an authorization code are not available.

When you use this grant type, the client does not need to store the resource owner credentials for future use. The client simply exchanges the credentials with a long-lived access token or refresh token.

For more details on resource owner password credentials grant type, see RFC 6749.

The following diagram shows the high-level flow for obtaining access tokens using resource owner password credentials grant.


Resource Owner
(1) Resource owner password credentials
Client
(2) Resource owner password credentials
UAA
(3) Access token with optional refresh token

1. The resource owner provides its username and password to the client.
2. The client includes the credentials in the requests for an access token from UAA token endpoint.
3. UAA authenticates the client and validates the resource owner credentials, and if valid, issues an access token.

For detailed information on UAA endpoints used to generate Access Tokens, see open-source UAA documentation.

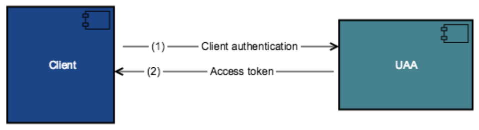
Obtaining Tokens Using Client Credentials Grant

When you use the client credentials grant type, the OAuth2 endpoint in UAA accepts the client id and client secret and provides Access Tokens.

The client credentials can be used when the authorization scope is limited to the protected resources under the control of the client, or to protected resources previously authorized with UAA. Client credentials authorization grant is typically used when the client is acting on its own behalf (the client is also the resource owner) or is requesting access to protected resources based on previous authorization with UAA.

For more details on the Authorization Grant type, see RFC 6749.

The following diagram shows the high-level flow for obtaining access tokens using client credentials grant.



1. The client requests an access token from the token endpoint.
2. UAA authenticates the client, and if valid, issues an access token.

For detailed information on UAA endpoints used to generate Access Tokens with client credentials grant, see open-source UAA documentation.

Understanding UAA and ID Tokens

Using UAA to Obtain ID Tokens

The ID Token contains claims about the authentication of an end-user by the authorization server, and can optionally contain other requested claims. For more information on ID tokens, see the OpenID Connect Core 1.0 specification.

You can obtain ID tokens for the following authorization grant types:

- Authorization Code (`response_type=code`)
- Implicit (`response_type=id_token token` or `response_type=id_token`)
- Hybrid (using other response type values defined in OAuth 2.0 Multiple Response Type Encoding Practices [`OAuth.Responses`])

Get Started With the UAA Service

UAA Security Service Setup

To begin using any secure Predix platform service, you must set up a UAA service instance as the trusted issuer.

Task Roadmap

#	Task	Description
1	(Optional) Configure your proxy settings if necessary.	Depending on your location and network configuration, you may need to configure your proxy settings to access remote resources. See Defining Proxy Connections to Remote Resources.
2	(Optional) Deploy a Predix Hello World Web application.	Creating and Deploying a Simple Web App to Cloud Foundry.
3	Create the UAA service instance.	See Creating a UAA Service Instance on page 10.
4	Create OAuth2 clients to setup access to your service authenticated using UAA.	When you create a UAA instance, an admin client is automatically created for you to access UAA for additional configuration. You can create a new client for your service instance with specific scopes. If an Oauth2 client already exists, you can update the client to add your service instance. See Creating an OAuth2 Client on page 17.
5	Create Groups to set up the required permissions for the users.	A group represents the privileges of a user. See Creating Groups in a UAA Instance on page 27.
6	Create Users in the UAA instance.	For UAA to authenticate the users, the users must first be created in UAA. You can either create users locally within UAA or federate to an external identity provider. For creating users locally, see Creating Users in a UAA Instance on page 24. To set up federated identity with UAA, see Managing Identity Providers.
7	Bind your application to the service instance.	See Binding an Application to the UAA Instance on page 13.

Creating a UAA Service Instance

You can create multiple instances of the UAA service in your space.

About This Task

As a best practice, first delete any older unused instances before creating a new one.

Procedure

1. Sign into your Predix account at https://www.predix.io.
2. Navigate to **Catalog** > **Services**, then click the **User Account and Authentication** tile.

3. Click **Subscribe** on the required plan.
4. Complete the fields on the **New Service Instance** page.

Field	Description
Org	Select your organization.
Space	Select the space for your application.
Service instance name	Enter a unique name for this UAA service instance.
Service plan	Select a plan.
Admin client secret	Enter a client secret (this is the admin password for this UAA instance). The client secret can be any alphanumeric string. **Note:** Record the client secret in a secure place for later use.
Subdomain	(Optional) Enter a subdomain you might need to use in addition to the domain created for UAA. You must not add special characters in the name of the subdomain. The value of sub-domain is case-insensitive.

5. Click **Create Service**.

Results

Your UAA instance is created with the following specifications:

- A client identifier (`admin`).

  Note: An `admin` client is required for bootstrap purposes. You can create additional clients to use with your application.
- A client secret (that you specified while creating the service).

To retrieve additional details of your instance, you can bind an application to your instance.

Using the Command Line to Create a UAA Service Instance

Optional procedure for using the command line instead of the graphical user interface to create a UAA service instance.

About This Task

You can create up to 10 instances of UAA service in your space. If you need additional instances, you must delete an older unused instance and create a new one.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Note: If you are a GE employee, you must use the `cf login --sso` command to log into Cloud Foundry. After you enter your SSO, you will receive a one-time passcode URL. Copy this URL and paste it in a browser to retrieve your one-time passcode. Use this code with the `cf` command to complete the CF login process.

   Depending on your Predix.io registration, the value of `<API_Endpoint>` is one of the following:

   - Predix US-West
     `https://api.system.aws-usw02-pr.ice.predix.io`
   - Predix US-East

  `https://api.system.asv-pr.ice.predix.io`
- Predix Europe
  `https://api.system.aws-eu-central-1-pr.ice.predix.io`

  For example,

  ```
  cf login -a https://api.system.aws-usw02-pr.ice.predix.io
  ```

2. List the services in the Cloud Foundry marketplace by entering the following command.

   ```
   cf marketplace
   ```

   The UAA service, `predix-uaa`, is listed as one of the available services.
3. Create a UAA instance by entering the following command.

   ```
   cf create-service predix-uaa <plan> <my_uaa_instance> -c
   '{"adminClientSecret":"<my_secret>","subdomain":"<my_subdomain>"}'
   ```

   where:

   - `cf` stands for the CLI command, `cloud foundry`
   - `cs` stands for the CLI command `create-service`
   - `<plan>` is the plan associated with a service. For example, you can use the `tiered` plan for the `predix-uaa` service.
   - `-c` option is used to specify following additional parameters.
     - `adminClientSecret` specifies the client secret.
     - `subdomain` specifies a sub-domain you might need to use in addition to the domain created for UAA. This is an optional parameter. You must not add special characters in the name of the sub-domain. The value of sub-domain is case insensitive.

   Note: Cloud Foundry CLI syntax can differ between Windows and Linux operating systems. See the Cloud Foundry help for the appropriate syntax for your operating system. For example, to see help for the `create service` command, run `cf cs`.

Results

Your UAA instance is created with the following specification:

- A client identifier (`admin`).

  Note: An `admin` client is created for bootstrap purposes. You can create additional clients to use with your application.
- A client secret (that you specified while creating the service).

To retrieve additional details of your instance, you can bind an application to your instance.

Example

Create a predix-uaa service instance with client secret as admin and sub-domain as ge-digital:

```
cf cs predix-uaa tiered test-1 -c
'{"adminClientSecret":"admin","subdomain":"ge-digital"}'
```

This is how it appears in VCAP SERVICES when using the `cf env <app_name>` command:

```
"VCAP_SERVICES": {
"predix-uaa": [
   {
    "credentials": {
     "dashboardUrl": "https://uaa-dashboard.run.asv-
pr.ice.predix.io/#/login/04187eb1-
e0cf-4874-8218-9fb77a8b4ed9",
     "issuerId": "https://04187eb1-
e0cf-4874-8218-9fb77a8b4ed9.predix-uaa.run.asv-
pr.ice.predix.io/oauth/token",
     "subdomain": "04187eb1-e0cf-4874-8218-9fb77a8b4ed9",
     "uri": "https://04187eb1-
e0cf-4874-8218-9fb77a8b4ed9.predix-uaa.run.asv-
pr.ice.predix.io",
     "zone": {
      "http-header-name": "X-Identity-Zone-Id",
      "http-header-value": "04187eb1-
e0cf-4874-8218-9fb77a8b4ed9"
     }
    },
    "label": "predix-uaa",
    "name": "testuaa",
    "plan": "Tiered",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [],
    "volume_mounts": []
   }
  ],
```

Binding an Application to the UAA Instance

About This Task

You must bind your application to your UAA instance to provision its connection details in the VCAP_SERVICES environment variable. The Cloud Foundry runtime uses the VCAP_SERVICES environment variable to communicate with a deployed application about its environment.

You can retrieve the following UAA instance details from the VCAP_SERVICES environment variable:

- A `dashboard_url` for your instance. You can use this URL to access the dashboard for managing this instance of UAA.
- A `subdomain` that specifies a sub-domain you can use in addition to the domain created for UAA.
- A `uaa_instance_uri` for your instance.
- A `uaa_instance_issuerId` for your instance. The `issuerID` is required when you create an instance of another service that uses your UAA instance for authentication.
- A `uaa_instance_GUID` is the `zoneID` for your instance.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Depending on your Predix.io registration, the value of `<API_Endpoint>` is one of the following:

   - Predix US-West
     `https://api.system.aws-usw02-pr.ice.predix.io`
   - Predix US-East
     `https://api.system.asv-pr.ice.predix.io`
   - Predix Europe
     `https://api.system.aws-eu-central-1-pr.ice.predix.io`

   For example,

   ```
   cf login -a https://api.system.aws-usw02-pr.ice.predix.io
   ```

2. Bind the application to the service instance by entering the following command:

   ```
   cf bind-service <your_app_name> <uaa_instance_name>
   ```

   The `<uaa_instance_name>` instance is bound to your application, and the following message is returned:

   ```
   Binding service <uaa_instance_name> to app <your_app_name> in org
   predix-platform / space predix as userx@ge.com...
   OK
   TIP: Use 'cf restage' to ensure your env variable changes take effect
   ```

3. Verify the binding by entering the following command:

   ```
   cf env <your_app_name>
   ```

   Messages that are similar to the following messages are returned:

   ```
   Getting env variables for app myApp in org predix-platform / space
   security as userx@ge.com...
   OK
   ...
     ],
     "predix-uaa": [
           {
           "credentials":
           {
            "dashboardUrl": "https://uaa-dashboard.run.asv-
   pr.ice.predix.io/#/login/ff27c315-d027-4d1d-a30c-64f49b369ed9",
            "issuerId":
             "https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-
   uaa.run.aws-usw02-pr.ice.predix.io/oauth/token",
            "subdomain": "ff27c315-d027-4d1d-a30c-64f49b369ed9",
            "uri":
              "https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-
   uaa.run.aws-usw02-pr.ice.predix.io",
             "zone": {
                       "http-header-name": "X-Identity-Zone-Id",
                       "http-header-value": "ff27c315-d027-4d1d-
   a30c-64f49b369ed9"
   ```

```
                    }
        },
        "label":
        "predix-uaa",
        "name":
        "my_uaa_instance",
        "plan":
        "free",
        "tags":
        []
        }
        ],
```

In this sample, the following values are displayed:

- `dashboard_url` = `https://uaa-dashboard.run.asv-pr.ice.predix.io/#/login/ff27c315-d027-4d1d-a30c-64f49b369ed9`
- `uaa_instance_issuerId` = `https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/token`
- `subdomain` = 04187eb1-e0cf-4874-8218-9fb77a8b4ed9
- `uaa_instance_uri` = `https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-uaa.run.aws-usw02-pr.ice.predix.io`
- `uaa_instance_GUID` = `ff27c315-d027-4d1d-a30c-64f49b369ed9`

Unbinding the UAA Instance From Your Application

Procedure

Unbind a service instance:

```
cf unbind-service <your_app_name> <uaa_instance_name>
```

The service instance is unbound from the application, and the following message is returned:

```
Unbinding app predix-service from service userX in org predix-
platform / space predix as userx@ge.com...
OK
```

Deleting a UAA Instance

Procedure

Delete a service instance:

```
cf delete-service predix-uaa <uaa_instance_name>
```

Configuring the UAA Service Instance

About UAA Dashboard

The UAA Dashboard is a graphical user interface available on predix.io to configure and manage your UAA security service instance.

Some of the tasks that you can accomplish using UAA Dashboard are:

- View the clients, users and identity providers related to your UAA instance.
- View clients associated with your service instances.
- Create and manage clients.
- Create and manage users and groups.
- Manage SSO configuration with SAML or OpenID Connect (OIDC) identity providers.
- Manage password policies for user passwords.

View, Create, and Manage Clients

You can create additional clients for different applications that authenticate with the same UAA server instance. You can assign different access permissions to each client depending on your application's requirement. Use the **Client Management** tab in UAA Dashboard to view, create and manage your OAuth clients.

View, Create, and Manage Users and Groups

The authentication flow for authorization grant type such as Authorization Code, Implicit, and Resource Owner Password involves users that provide their username and password credentials to an application. You can either create users locally within UAA or federate to an external identity provider. Use the **User Management** tab in the UAA Dashboard to create users locally in UAA and manage them for your application. Additionally you can create groups that represent the privileges a user can potentially have.

Manage SSO configuration with SAML or OpenID Connect (OIDC) identity providers

Use the **Identity Providers** tab in the UAA Dashboard to manage your identity providers and service providers in UAA. For more information on integrating with identity providers, see Managing Identity Providers.

Add and Manage Password Policies

Use the **Password Policy** tab in the UAA Dashboard to configure the user accounts in UAA with password policy such as length, accepted or required character types, expiration times, and reset policy.

Managing Clients

Creating an OAuth2 Client

You can create OAuth2 clients with specific permissions for your application to work with Predix Platform services. Often this is the first step after creating an instance of a service.

About This Task

When you create an instance of UAA, the UAA Dashboard is available for configuring that instance of UAA. You can use the Client Management tab in the UAA Dashboard to create the OAuth2 clients.

If you are prefer using the UAA command-line interface (UAAC) instead of UAA Dashboard to create an OAuth2 client, see Using UAAC to Create an OAuth2 Client on page 20

Procedure

1. In the Predix.io Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Client Management** tab.

   The Client Management tab has two views, **Clients** and **Services** . The **Services** view displays the service instances that you have created for your services.

   Note: The service instances displayed in the Services view were created while using the UAA that you are trying to configure. Service instances that you created using other UAA instances are not displayed on this page.
6. Click **Create Client** to open the **Create Client** form.
7. Complete the **Create Client** form.

Field	Description
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	Choose one or more of the following grant types: • **authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code. • **client_credentials** When you use the client credentials grant type, the OAuth2 endpoint in UAA accepts the client ID and client secret and provides Access Tokens. • **password** When you use the resource owner password credentials grant type, the OAuth2 endpoint in UAA accepts the username and password and provides Access Tokens. • **refresh_token** The refresh tokens are credentials used to obtain access tokens. You can choose this option to obtain refresh token from UAA. You can then use the refresh token to obtain a new access token from UAA when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope. • **implicit** When you use the implicit grant type, UAA directly issues an Access Token to the client without authenticating the client. This reduces the number of round trips required to obtain an access token. For more information on grant types, see RFC 6749.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.

Field	Description
Redirect URI	Specify a redirect URI to redirect the client after login or logout (for example, `http://example-app.com/callback`). Use this URI when you start using UAA as the service provider for your external Identity provider. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints. You must specify a Redirect URI value if you use the Authorization Code or Implicit authorization grant type. When you use the Authorization Code grant type, the Redirect URI is your application's endpoint or callback that expects user authorization code. When you use the Implicit grant type, the Redirect URI is the end point where UAA sends the bearer token. Unique Resource Identifier consists of: • Access Protocol, `http` or `https` • Domain or IP address • Access Port such as 80 or 443 • Path If you have a specific URL for your application callback, you can use that to set the Redirect URI value for the related client. For example, `https://your-app-domain.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. You can specify multiple values for Redirect URI as a list of allowed destinations that UAA server can redirect the users. For example, `https://yourappdomain1.run.aws-usw02-pr.ice.predix.io/path1/path2/callback,https://yourappdomain2.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. If the subdomain of your application is dynamic, you can set the value of Redirect URI using wilcards. For example, `https://*.your-app-domain.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. **Note:** You must only use '*' for a domain that is exclusive to your application (Such as `your-app-domain` in example above). This prevents the redirect to be routed to an application that you do not own. You cannot use `*` in the top domain and sub domain (such as `predix.io` in the example above).
Scopes	Scopes are permissions associated with an OAuth Client to determine user access to a resource through an application. The user permissions are for authorization grant types `authorization_code`, `password` and `implicit`. By default, the admin client is assigned all required scopes. For a new client, an administrator can select the scopes to be added based on client requirements. For a list of available scopes, see Scopes Authorized by the UAA. To use an OAuth2 client for your Predix Platform service instance, you must update your OAuth2 client to add scopes that are specific to each service after adding the client to the service instance.

Field	Description
Authorities	Authorities are permissions associated with the OAuth Client when an application or API is acting on its own behalf to access a resource with its own credentials, without user involvement. The permissions are for the `client_credentials` authorization grant type. By default, the admin client is assigned all required authorities. For a new client, an administrator can select the authorities to be added based on client requirements. The list of authorities matches the list of scopes. For a list of available UAA scopes, see Scopes Authorized by the UAA. To use an OAuth2 client for your Predix Platform service instance, you must update your OAuth2 client to add authorities that are specific to each service after adding the client to the service instance. **Note:** An admin client is not assigned the default authority to change the user password. To change the user password, you must add the `uaa.admin` authority to your admin client.
Auto Approved Scopes	Specify scopes that can be approved automatically for the client without explicit approval from a resource owner.
Allowed Providers	Specifies the names of the external identity providers, if any. This field is required if you are using external identity providers with UAA as a service provider.
Access Token Validity	Specifies the access-token expiration time in ms.
Refresh Token Validity	Specifies the refresh-token expiration time in ms.

Next Steps

Updating the OAuth2 Client for Services on page 21 for your service specific information.

Using UAAC to Create an OAuth2 Client

You can use the UAA command-line interface (UAAC) instead of UAA Dashboard to create an OAuth2 client.

About This Task

You can use the UAAC, to manage your UAA instance. For more information on installing the command-line interface, see https://github.com/cloudfoundry/cf-uaac.

Procedure

1. Specify your UAA instance as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   `<uaa_instance_url>` is the URL to your trusted issuer, for example, `https://11fa0273-9e2a-37e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io`. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.
2. Log in using the administrative client.

   ```
   uaac token client get admin
   ```
3. Specify the administrative client secret at the prompt.

4. Use the following command to create the OAuth2 client:

```
uaac client add [client_name]
    --authorities "uaa.resource"
    --scope "openid"
    --autoapprove "openid"
    --authorized_grant_types [authorization_code|implicit|password|
client_credentials|refresh_token]
    --redirect_uri [redirect_uri_1, redirect_uri_2, ...]
```

   For more information on UAA options such as `scopes` and `authorized_grant_types`, see the UAA documentation at https://github.com/GESoftware-CF/uaa/blob/master/docs/UAA-APIs.rst.

Next Steps

Updating the OAuth2 Client for Services on page 21 for your service specific information.

Updating the OAuth2 Client for Services

To use an OAuth2 client for secure access to your Predix Platform service instance from your application, you must update your OAuth2 client to add additional authorities or scopes that are specific to each service.

About This Task

To enable your application to access a platform service, your JSON Web Token (JWT) must contain the scopes required for a platform service. For example, some of the scope required for Access Control service are `acs.policies.read acs.policies.write`.

The OAuth2 client uses an authorization grant to request an access token. Based on the type of authorization grant that you have used, you must update your OAuth2 client to generate the required JWT. For more information on how the OAuth2 client is created, see Creating OAuth2 client.

If you use the UAA Dashboard to create additional clients, the client is created for the default `client_credentials` grant type. Some required authorities and scopes are automatically added to the client. You must add additional authorities or scopes that are specific to each service.

In addition, the admin client is not assigned the default authority to change the user password. To change the user password, you must add the `uaa.admin` authority to your admin client.

Use the following procedure to update the OAuth2 client.

Procedure

1. In the Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Client Management** tab.

   The Client Management tab has two views, **Clients** and **Services**. The **Services** view displays the service instances that you have created for your services.

   Note: The service instances displayed in the **Services** view are the instances that you created using the UAA that you are trying to configure. The service instances that you created using some other UAA instance are not displayed on this page.
6. Select the **Switch to Services View** option.

7. In the **Services** view, select the service that you need to update.
8. Choose an existing client or choose the **Create a new client** option. If you chose to create a new client, follow the steps in Creating an OAuth2 Client on page 17.
9. Click **Submit**.
10. Click on the **Switch to Clients View** option.
11. In the **Clients** view, click the edit icon corresponding to the client added in the previous step.
12. Complete the **Edit Client** form.

Field	Description
Authorized Grant Types	Choose one or more of the following grant types: • **authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code. • **client_credentials** When you use the client credentials grant type, the OAuth2 endpoint in UAA accepts the client ID and client secret and provides Access Tokens. • **password** When you use the resource owner password credentials grant type, the OAuth2 endpoint in UAA accepts the username and password and provides Access Tokens. • **refresh_token** The refresh tokens are credentials used to obtain access tokens. You can choose this option to obtain refresh token from UAA. You can then use the refresh token to obtain a new access token from UAA when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope. • **implicit** When you use the implicit grant type, UAA directly issues an Access Token to the client without authenticating the client. This reduces the number of round trips required to obtain an access token. For more information on grant types, see RFC 6749.
Redirect URI	Specify a redirect URI to redirect the client after login (for example, `http://example-app.com/welcome`). This URI is used when you start using UAA as service provider for your external Identify provider.
Scopes	By default, the client is assigned a few required scopes. For a new client, an administrator can select the scopes to be added based on the selected grant type. If you select the `authorization_code`, `password` and `implicit` grant type, you must update the scopes with service specific scopes. For a complete list of required scopes, see Authorities or Scopes Required for Platform Services on page 23. For a list of available UAA scopes, see Scopes Authorized by the UAA.
Authorities	By default, the client is assigned a few required authorities. For a new client, an administrator can select the authorities to be added based on the selected grant type. If you select the `client_credentials` grant type, you must update the authorities with service specific authorities. For a complete list of scopes to be added for each service, see Authorities or Scopes Required for Platform Services on page 23. For a list of available UAA authorities, see Scopes Authorized by the UAA.
Auto Approved Scopes	Specify scopes that can be approved automatically for the client without explicit approval from the resource owner.
Allowed Providers	Specify the names of the external identity providers, if any. This field is required if you are using external identity providers with UAA as a service provider.

Field	Description
Access Token Validity	Specifies the access token expiration time in ms.
Refresh Token Validity	Specifies the refresh token expiration time in ms.

Next Steps

You can complete the following additional tasks in UAA Dashboard:

- If you are using authorization grant type as Authorization Code, Implicit, or Resource Owner Password, you can manage users in UAA.
- You can create password policies for user passwords.
- You can set up external identity provider or use UAA as an identity provider. See Managing Identity Providers.

If you have completed your OAuth2 client setup, you can bind your application to your service instance.

Authorities or Scopes Required for Platform Services

When you create a new OAuth2 client, the client is assigned default scopes and authorities. You must add additional authorities or scopes that are specific to each service.

The following table lists the scopes and authorities specific to each platform service that you must add to your OAuth2 client.

Service Name	Authorities/Scopes
Access Control	• acs.policies.read • acs.policies.write • acs.attributes.read • acs.attributes.write • predix-acs.zones.<acs_instance_guid>.user This value is added by default if you use the UAA Dashboard. It is also generated in the VCAP_SERVICES environment variable as `oauth-scope` when you bind your application to your ACS service instance.
Analytics Catalog	analytics.zones.<service_instance_guid>.user (added by default)
Analytics Runtime	analytics.zones.<service_instance_guid>.user (added by default)
Asset	predix-asset.zones.<service_instance_guid>.user (added by default)
Blockchain as a Service	predix-blockchainapi.zones.<service_instance_guid>.user (added by default)
Event Hub	• Publish ◦ predix-event-hub.zones.<Predix-Zone-Id>.user ◦ predix-event-hub.zones.<Predix-Zone-Id>.wss.publish ◦ predix-event-hub.zones.<Predix-Zone-Id>.grpc.publish • Subscribe ◦ predix-event-hub.zones.<Predix-Zone-Id>.user ◦ predix-event-hub.zones.<Predix-Zone-Id>.grpc.subscribe
Tenant Management	• tms.tenant.read • tms.tenant.write • predix-tms.zones.<tms_instance_guid>.user (added by default)

Service Name	Authorities/Scopes
Time Series	• Data ingestion ◦ timeseries.zones.<Predix-Zone-Id>.user (added by default) ◦ timeseries.zones.<Predix-Zone-Id>.ingest • Data queries ◦ timeseries.zones.<Predix-Zone-Id>.user (added by default) ◦ timeseries.zones.<Predix-Zone-Id>.query
View	• views.zones.<view_instanceId>.user (added by default) • views.admin.user • views.power.user

Changing the Administrative Client Secret

Procedure

1. Specify your UAA instance as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   `<uaa_instance_url>` is the URL to your trusted issuer, for example, `https://11fa0273-9e2a-37e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io`. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.
2. Log in using the administrative client.

   ```
   uaac token client get admin
   ```
3. Specify the administrative client secret at the prompt.
4. Use the following command to change the client secret.

   ```
   uaac secret change
   ```
5. Enter the current administrative client secret at the prompt.
6. Enter the new administrative client secret at the prompt.
7. Reenter the new administrative client secret at the prompt.

   Note:

   Do not specify the administrative client secret using the `-s` option. Your terminal history retains a copy of those credentials.

Managing Users

Creating Users in a UAA Instance

You can create users locally in UAA for authentication and assign them to the required groups from the UAA dashboard.

Before You begin

- Log in to Predix.io.

About This Task

When you create a UAA instance, an admin client is automatically created for you so that you can configure your UAA instance. The admin client is assigned all the required authorities and scopes by default.

Note: The admin client is not assigned the authority to be able to change the user password by default. If you need the ability to update or change the user password, you must add the `uaa.admin` authority to your admin client. You can use the UAA command-line interface (UAAC) to add the `uaa.admin`authority to your admin client. For more information on installing the command-line interface, see https://github.com/cloudfoundry/cf-uaac.

If you prefer using the UAAC to create the users, see Using UAAC to Create Users in a UAA Instance on page 26.

Use the following procedure to create users locally through the UAA dashboard.

Procedure

1. In the **Console** view, select the **Space** where your services are located.
2. In the **Services Instances** page, select the UAA instance that you need to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In the UAA Dashboard, select the **User Management** tab.

   The User Management tab has two sections, **Users** and **Groups**. The **Groups** section displays the groups that you created in your UAA instance.
6. Click on the **Create User** button to open the **New User** form.
7. Specify the following values in the **New User** form:

Field	Description
Regular User	Choose this option to set up local users in your UAA. The **Regular User** is not configured through any external Identity Provider (IdP).
Shadow User	Choose this option to create a local user in UAA corresponding to the user defined in your external IdP. The **Shadow User** option is useful if you need to white list users to authenticate only a subset of users setup in your identity provider. To setup individual shadow users, ensure that the option to create shadow users is not selected while configuring a new IdP.
User Name	Specify the user name. If you are setting up a shadow user, this value must match the user name defined in your IdP.
Email	Specify the email address. If you are setting up a shadow user, this value must match the user name defined in your IdP.
Password	Specify the password. An administrator can set password policies to define the permitted structure of the password. For more information, see Creating Password Policies on page 27. This option is not required if you are setting up a shadow user.
Given Name	Specify the first name of the user.
Family Name	Specify the last name of the user.
Origin	Specify the name of the IdP that this user is configured in. The Origin option is available only if you are setting up a Shadow user.
Groups	Select the groups to associate the user with. For more information on groups, see Creating Groups in a UAA Instance on page 27.

Field	Description
Active	Select this option to allow your Regular or Shadow user to login.
Verified	Select this option to indicate that this Regular or Shadow user is a verified user. Verified users are the users who are verified using an autogenerated email invite sent from UAA at the time of account creation.

Using UAAC to Create Users in a UAA Instance

Optional procedure to create users in a UAA instance using the UAAC instead of UAA dashboard. .

About This Task

For applications accessing your UAA instance, you can create additional clients and users with required scopes.

Note: The admin client is not assigned the authority to be able to change the user password by default. If you need the ability to update or change the user password, you must add the `uaa.admin` authority to your admin client. You can use the UAA command-line interface (UAAC) to add the `uaa.admin`authority to your admin client. For more information on installing the command-line interface, see https://github.com/cloudfoundry/cf-uaac.

Procedure

1. Create a new user.

```
uaac user add <my-user> --emails <my_user>-user@ge.com --password
<my_password>
```

2. Create the groups in your UAA instance.

   For more information on available groups, see https://github.com/cloudfoundry/uaa/blob/master/docs/UAA-APIs.rst#scopes-authorized-by-the-uaa.

   For example:

```
uaac group add scim.read
uaac group add scim.write
```

3. Add the new user to the required groups.

   For example:

```
uaac member add zones.<my_uaa_instance>.admin <my-user>
uaac member add scim.read <my-user>
uaac member add scim.write <my-user>
uaac member add clients.write <my-user>
uaac member add clients.read <my-user>
uaac member add clients.admin <my-user>
uaac member add clients.secret <my-user>
```

4. Verify that the user is created with the correct scope.

```
uaac token owner get <my-oauth-client> <my-user>
uaac token decode
```

Creating Groups in a UAA Instance

If you design your application to authorize using specific scopes, you can create groups corresponding to those scopes in UAA and assign users to those groups. When the users log into your web application, the application redirects them to UAA. If a user is in the specified group and you chose to authorize the web application with that scope, the web application gets a signed token that contains that scope.

About This Task

Predix platform services have scopes specific to each service. When you create users for these services, you can create groups corresponding to these scopes to provide permissions specific to a service. After creating groups, you can assign users to the required groups.

For example, if you use the Time Series service, you must create the `timeseries.zones.<instance_id>.user` and `timeseries.zones.<instance_id>.ingest` groups for users with data ingestion permission.

For a list of scopes for all platform services, see Authorities or Scopes Required for Platform Services on page 23.

Use the following procedure to create groups in UAA:

Procedure

1. In the Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance that you need to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **User Management** tab.

   The User Management tab has two sections, **Users** and **Groups**. The **Groups** section displays the groups that you have created in your UAA instance.
6. Click on the **Create Group** option to open the **New Group** form.
7. Specify the following values in the **New Group** form:

Field	Description
Display Name	Specify the name of the group.
Description	Specify the description of the group.

Creating Password Policies

You can configure the password policy for passwords and client secrets in UAA for parameters such as length, accepted or required character types, expiration times, and reset policy.

About This Task

When you create an instance of UAA, an internal Identity Provider of type `uaa` is automatically created with the default password policy. You can create new password policies for clients in your instance of UAA. You can create policies for both user passwords and for client secrets.

You can change the password policies at any time. Change in the password policy affects all users, including any existing users in your UAA instance.

Procedure

1. In the Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance that you need to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Password Policy** tab.
6. Specify the following values in the **Password Policy** form:

Field	Description
Set Password Length	Specifies the minimum to maximum number of characters required for a valid password.
Requirements	Specifies the type of characters required for a valid password.
Expiration	Specifies the number of months after which current password expires.
Lockout Policy	Specifies the amount of time (in seconds) for which the account is locked when the number of failed attempts has exceeded the set limit within the specified time.

Managing Identity Providers

Managing Identity Providers

An Identity Provider (IdP) manages accounts for users who may need secure access to the applications or services. A Service Provider (SP) is the server receiving request from a user for access to a service or application. In a typical SAML flow, when a user requests a service from the SP, the SP first requests and obtains an identity assertion from the IdP. The IdP receives the request from SP and generates an identity assertion based on the user account information. SP then decides whether to perform the service based on assertion provided by IdP. UAA supports SAML protocol for communicating with IdPs or SPs.

You can configure your UAA instance to act as an IdP or use an external IdP. The following scenarios determine the type of configuration you need for your UAA:

- If you administer users accounts locally in UAA using UAA SCIM APIs or UAA dashboard, then UAA is your default identity provider. You do not need any additional configuration for identity provider in UAA.
- If you provision your user accounts remotely on an external IdP such as Company SSO, you can configure UAA as SP that redirects to external IdP. For more information, see Configuring UAA as Service Provider for External Identity Provider.
- If you have applications that provide SP capability (For example, GitHub Enterprise or ServiceNow), you can configure UAA as IdP. For more information, see Configuring UAA as an Identity Provider.
- It is possible to configure UAA as both SP and IdP. However such a configuration is useful only as a test environment. To set up UAA as SP and IdP, you can complete steps for configuring UAA as both SP and IdP.

Configuring UAA as Service Provider for External Identity Provider

If you provision your user accounts remotely on an Identity Provider (IdP) such as Company SSO, you can configure UAA as Service Provider (SP) that redirects to external IdP.

Before You begin

- Obtain your Identity Provider (IdP) metadata from your IdP administrator.
- Log In to Predix.io and go to Console view.

About This Task

Complete the following procedure to configure UAA as SP for external IdP:

Procedure

1. In the Services Instances page on Predix.io, select the UAA instance that you need to configure.
2. Select the **Configure Service Instance** option.
3. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
4. In UAA Dashboard, click on the **Identity Providers** tab.
5. In the **External Identity Provider** section, select **New Identity Provider**.
6. In the **New Identity Provider** form, specify the following information and press **Submit**:

Name of the Field	Description
Name	Specify the name of your IdP.
Description	Specify a short description for your IdP.
Type	Select the type of IdP- SAML or OIDC.
Email Domain	Specify the email domain that UAA can use to identify the identity provider if you have configured more than one identity provider. To support this feature, UAA provides a configuration of IdP discovery. By default, this configuration is set to False. UAA can be configured to set IDP Discovery to true if you have configured more than one identity provider. For more information, see Identifying IdP Using Email Domain on page 34
Active	Select this option to set the IdP as active. If you have multiple IdPs defined for a single UAA instance, UAA interacts with only active IdPs. If you have multiple active IdPs for a single UAA instance, you must ensure that the clients related to the IdPs are updated with the corresponding information. Although it is possible to update a single client to interact with multiple IdPs, as a best practice, you can define a new client for each of your application to interact with UAA.

If you are setting up an OpenID Connect (OIDC) IdP, the following field are available:

Name of the Field	Description
Username Mapping	Specify the username attribute defined in your IdP.
Authorization Endpoint Url	Specify the authorization endpoint for OIDC token.
Token Endpoint Url	Specify the token URL for OIDC authorization code.

Name of the Field	Description
Token Key Endpoint Url	Specify the token key endpoint for token verification.
Token Issuer	Specify the URL of your OIDC provider.
Relying Party Client ID	Specify the OIDC provider client id.
Relying Party Client Secret	Specify the OIDC provider client secret.
Automatically create a shadow user on login	Select this option to create a local user in UAA corresponding to each user defined in the identity provider that you are configuring. The local user is created when the user logs-in for the first time. This feature enables a user to be authenticated by UAA without requiring you to create it in UAA also. By default this option is turned off. You must create each user in UAA even if the user exists in IdP for authentication from UAA. This option is useful if you need to white list users to authenticate only a subset of users setup in your identity provider. To white list users, you can turn this option off while configuring a new IdP, and individually create users in UAA corresponding to users that are defined in your IdP that you need authenticated.

If you are setting up a SAML IdP, the following fields are available:

Name of the Field	Description
Name ID Format	Specify the IdP ID format that UAA SP must use. This is useful if the IdP metadata contains multiple ID formats and you need to specify the format that SP must use. You can copy the ID format string from the metadata that you specify.
Metadata	Specify the IdP metadata and download the UAA SP metadata in this field. You must obtain the IdP metadata from your IdP administrator. You must supply the UAA SP metadata to your IdP administrator.
Email	(Optional) Specify the attribute defined in your IdP that corresponds to the `email` attribute in UAA.
Given Name	(Optional) Specify the attribute defined in your IdP that corresponds to the `Given Name` attribute in UAA.
Family Name	(Optional) Specify the attribute defined in your IdP that corresponds to the `Family Name` attribute in UAA.
Automatically create a shadow user on login	Creates a local user in UAA corresponding to each user defined in the identity provider that you are configuring. The local user is created when the user logs-in for the first time. This feature enables a user to be authenticated by UAA without requiring you to create it in UAA also. If this option is turned off, you must create each user in UAA even if the user exists in IdP for authentication from UAA. This option is useful if you need to white list users to authenticate only a subset of users setup in your identity provider. To white list users, you can turn this option off while configuring a

Name of the Field	Description
	new IdP, and individually create users in UAA corresponding to users that are defined in your IdP that you need authenticated.

The IdP is displayed in the list of Identity Providers.

7. To set up the client for the IdP to access UAA, click on the **View/Edit** option next to the name of your IdP.
8. Specify a client for your IdP.

   You can either choose an existing client or create a new client. As a best practice for development, you can add a new client for each application.

   To create a new client, click on the **Create new client for this IDP** option. By default, the Create Client form is populated for grant type Authorization Code. You can update it to use implicit or password grant types also. For client credentials grant type, you must first create a user.

   For more information on creating a client, see Creating an OAuth2 Client on page 17.

Configuring UAA as a Service Provider Using Scripts

As an optional procedure, you can use UAAC and configuration scripts instead of UAA Dashboard to configure UAA as a service provider.

Before You begin

- Download the following scripts from GitHub.
  - `create-saml-idp.sh`
  - `create-client-for-idp.sh`

About This Task

You can configure UAA as Service Provider (SP) that redirects to external Identity Provider (IdP) such as Company SSO.

You can use the UAA command-line interface (UAAC), to manage your UAA instance. For more information on installing the command-line interface, see https://github.com/cloudfoundry/cf-uaac.

Procedure

1. Specify your UAA instance as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   `<uaa_instance_url>` is the URL to your trusted issuer, for example, `https://11fa0273-9e2a-37e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io`. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.
2. Log in using the administrative client.

   ```
   uaac token client get admin
   ```
3. Specify the administrative client secret at the prompt.
4. Export SAML Service Provider metadata from UAA. Provide this metadata to your IdP administrator so it can be imported into the IdP.

The `uaa-sp-metadata.xml` file is the Service Provider SAML metadata generated by your UAA instance. It is available at the following endpoint:

`<uaa_instance_url>/saml/metadata/alias/<uaa_instance_guid>.cloudfoundry-saml-login`

For example,

`https://13fa0384-9e2a-48e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io/saml/metadata/alias/13fa0384-9e2a-48e2-9d06-2c95a1f4f5ea.cloudfoundry-saml-login`

5. Your IdP administrator will configure their IdP, import the SAML Service Provider metadata, and generate the Identity Provider metadata. Obtain this Identity Provider metadata from your IdP administrator. You need to import this metadata into UAA.
6. Validate the SAML Identity Provider metadata by running through a XML parser. Ensure that it contains a valid XML header such as:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   ```

   If your file does not contain the required header, add it to the top of your metadata file.
7. Create the SAML Identity Provider for the specified UAA instance using the `create-saml-idp.sh` script. The script uses the specified SSO metadata file.

   ```
   ./create-saml-idp.sh -n <my-idp-name> -m <idp-metadata>.xml -a -c
   <mapping-config-file> -g <groups-config-file>
   ```

   - `-n` specifies a name for your identity provider.
   - `-m`specifies the SSO metadata provided by your IdP administrator. For example, `idp-metadata.xml`
   - `-a` specifies that a shadow user should be created in UAA. By default, a shadow user is not created. Using the `-a` option is required when you set up the identity provider for the first time.
   - `-c` is an optional parameter that specifies configuration mapping file. A configuration mapping file contains attribute mappings required to convert SAML assertion attributes to JWT token.
     An example of configuration mapping file is as follows:

     ```
     {
      "email":"mail",
      "given_name":"first name",
      "family_name":"last name"
      }
     ```
   - `-g` is an optional parameter that specifies groups configuration file. A groups configuration file contains any groups that need to be mapped from external IdP.
     An example of groups configuration file is as follows:

     ```
     [
     "group1",
     "group2",
     "group3",
     "group4",
     "group5"
     ]
     ```

For example, to use GE SSO, you can specify:

```
./create-saml-idp.sh -n gesso-staging-idp -m gesso-staging-idp-
metadata.xml -a
```

Note: If required, you can update the IdP configuration after configuring UAA for your IdP. For more information, see Updating IdP Configuration.

8. Validate that the identity provider that you specified is present.

```
uaac curl /identity-providers
```

9. Use the `create-client-for-idp.sh` script to create a client with the specified identity provider (IdP) as the allowed provider for the client. Each OAuth client can specify one or more identity providers responsible for authenticating the user. The default identity provider is `uaa`, which means users are authenticated using the UAA login page. Use this script to specify a different identity provider for user login, such as a SAML identity provider. However, it does not allow you to specify multiple identity providers.

```
./create-client-for-idp.sh -c <client_id> -s <client-secret> -p <my-
sso-idp> -r <redirect_uri>
```

   - `-c` specifies the client Id.
   - `-s` specifies the client secret.
   - `-p` specifies a name for your identity provider.
   - `-r` specifies your applications URI.

   For example,

```
./create-client-for-idp.sh -c apm_app -s changeme -p gesso-staging-
idp -i  -r Predix-HelloWorld-WebApp-myapp.run.aws-usw02-
pr.ice.predix.io
```

10. Validate that your client is present and `allowed providers` attribute is set to your IdP.

```
uaac client get <client_id>
```

Results

To test your setup, use the following URL:
`<uaa_instance_url>/oauth/authorize?client_id=<my_app>&response_type=code&redirect_uri=<redirect_uri>`

Note: Use this URL for testing if you are using the Authorization Code authorization grant.

- `<client_id>` is the name of the client provisioned in the step 6
- `redirect_uri` is the application URL. For example, `https://security-predix-seed.grc-apps.svc.ice.ge.com`.

The request is redirected to your IdP login page where you can enter the credentials for user that you provisioned for your IdP. If login is successful, you are redirected back to `redirect_uri`.

Note: Use browser plugin tools such as `SAML Tracer` for Firefox to validate SAML flow.

Updating IdP Configuration

Before You begin

- Download the following scripts from GitHub.
  - `update-saml-idp.sh`

About This Task

You can update UAA configuration to reflect any changes to your IdP setup.

Procedure

1. Specify your UAA instance as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   `<uaa_instance_url>` is the URL to your trusted issuer, for example, `https://11fa0273-9e2a-37e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io`. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.
2. Log in using the administrative client.

   ```
   uaac token client get admin
   ```
3. Specify the administrative client secret at the prompt.
4. Update the SAML Identity Provider for the specified UAA instance using the `update-saml-idp.sh` script. The script uses the specified SSO metadata file.

   ```
   ./update-saml-idp.sh -n <my-idp-name> -m <idp-metadata>.xml -d <idp-
   id> -a -c <mapping-config-file> -g <groups-config-file>
   ```

   - `-n` specifies the name of identity provider that you specified while creating the IdP and it corresponds to the name attribute for `/identity-providers` payload.
   - `-d` specifies the IdP Id. IdP Id is auto generated id from UAA and corresponds to `id` attribute for `/identity-providers` payload.
   - For description of other parameters, see Configuring UAA as a Service Provider Using Scripts on page 31.

Identifying IdP Using Email Domain

You can configure UAA to use the email domain to identify the identity provider if you have configured more than one external identity providers.

About This Task

When a user logs in using a set email Id, UAA can redirect the call to the right identity provider if multiple identity providers are present. To configure UAA to identify the identity provider, you set the IdP discovery configuration in UAA. By default, this configuration is set to False.

Procedure

1. In UAA Dashboard, select the **Customization** tab.
2. In the **Customization** page, turn on the **Enable IDP Discovery** option.

3. Click **Save**.

Configuring UAA as an Identity Provider

If you have applications that provide Service Provider (SP) capability (For example, GitHub Enterprise or ServiceNow), you can configure UAA as an Identity Provider (IdP).

Before You begin

- Obtain your SP metadata from your administrator.
- Log In to Predix.io and go to Console view.

About This Task

Complete the following procedure to configure UAA as SAML IdP to integrate with other service providers.

Procedure

1. In the Services Instances page on Predix.io, select the UAA instance that you need to configure.
2. Select the **Configure Service Instance** option.
3. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
4. In UAA Dashboard, click on the **Identity Providers** tab.
5. In the **UAA as Identity Provider** section, select **New Service Provider**.
6. In the **New Service Provider** form, specify the following information and press **Submit**:

Name of the Field	Description
Name	Specify the name of your SP.
Entity ID	Specify the Entity ID for your SP. This value must match the value in the SP metadata.
Active	Select this option to set the the SP as active. If you have multiple SPs defined for a single UAA instance, UAA interacts with only active SPs. If you have multiple activeSPs for a single UAA instance, you must ensure that the clients related to the SPs are updated with the corresponding information. Although it is possible to update a single client to interact with multiple SPs, as a best practice, you can define a new client for each of your application to interact with UAA.
Metadata	Specify the SP metadata and download the UAA IdP metadata in this field. You must obtain the SP metadata from your SP administrator. You must supply the UAA IdP metadata to your SP administrator.
UAA Validate SAML Metadata	(Optional) Select this option for UAA to check the SP metadata for valid signature value. UAA performs this check every time a user initiates the SAML communication.

The SP is displayed in the list of Service Providers.

Configuring UAA as an Identity Provider Using Scripts

As an optional procedure, you can use the configuration scripts instead of UAA Dashboard to configure UAA as Identity Provider.

Before You begin

- Download the following script from GitHub.

- `create-saml-sp.sh`

About This Task

You can configure UAA as SAML identity provider to integrate with other service providers.

Procedure

1. Specify your UAA instance as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   `<uaa_instance_url>` is the URL to your trusted issuer, for example, `https://11fa0273-9e2a-37e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io`. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.
2. Log in using the administrative client.

   ```
   uaac token client get admin
   ```
3. Specify the administrative client secret at the prompt.
4. Export Identity Provider (IdP) metadata from UAA. Provide this metadata to your Service Provider (SP) administrator so it can be imported into the SP application.

   The `uaa_idp_metadata.xml` file is the IdP metadata generated by your UAA instance. It is available at the following endpoint:

   `<uaa_instance_url>/saml/idp/metadata`

   For example,

   `https://13fa0384-9e2a-48e2-9d06-2c95a1f4f5ea.predix-uaa.run.aws-usw02-pr.ice.predix.io/saml/idp/metadata`
5. Import the UAA IdP metadata into your SP application.

   This step depends on the type of service provider you use. For more information, refer to the documentation for your service provider.
6. Provision a user for UAA IdP.

   ```
   uaac target <UAA_IDP_INSTANCE_URL>
   uaac token client get admin
   client secret: <admin client secret>
   uaac user add <user-name> --given_name <first-name> --family_name
   <last-name> --emails <email> -p <password>
   uaac group add zones.uaa.admin
   uaac member add zones.uaa.admin <user-name>
   ```
7. Add Service Provider configuration to UAA IdP.

   a) Obtain an access token. The client that you use to obtain access token must have the `sps.write` scope.

      You are redirected to UAA IdP login page. Enter credentials for user provisioned in step 3.

      For more information on obtaining access tokens, see Using UAA to Obtain OAuth2 Access Tokens.

   b) Use the `create-saml-sp.sh` script to add the SP configuration.

      ```
      ./create-saml-sp.sh -n <uaa-sp-name> -m uaa-sp-metadata.xml -i
      ```

8. Check that the SP configuration was successfully added.

```
uaac curl /saml/service-providers
```

OIDC Federated Authentication

OIDC Federated Authentication

OpenID Connect (OIDC) is a simple identity layer built on top of the OAuth 2.0 protocol. It enables clients to verify the identity of an end-user based on the authentication performed by an authorization server or identity provider (IdP) and obtains basic profile information of an end-user in an interoperable REST-like manner.

OpenID Connect allows one or more relying parties (RP) or clients to delegate user authentication to an OpenID Provider (OP) or authentication servers. The OpenID Provider authenticates users and provides claims to relying parties. These claims are user attributes such as first and last name, email address, and department. As a result, relying parties are freed from the need to run a login process, and users have fewer credentials to manage.

OIDC implements authentication as an extension of OAuth 2.0 protocol, it provides information about the end-user in the form of `id_tokens`. The `id_token` is a signed data structure that contains authenticated user attributes that are encoded as a JSON Web Token (JWT). In addition to identifying the end-user, or subject, the `id_token` also identifies the token issuer and client application to which the token was issued.

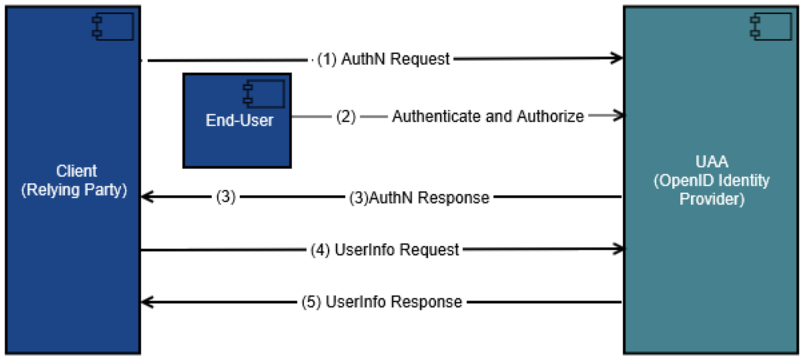
OIDC Key Features:

- Simple to integrate with applications.
- Works harmoniously with a wide variety of applications
- Offers features and security options that meet enterprise requirements.
- Uses JWTs for portability and support for a range of signature and encryption algorithms

OIDC Implementation in UAA

OIDC implementation in UAA facilitates the users by reducing the number of passwords and other credentials that they have to manage during the authentication process. You can configure UAA as an Open ID Identity Provider or Identity Provider to authenticate with another instance of UAA.

The following diagram shows the OIDC flow in UAA.



In this flow:

1. The Client (Relying Party) sends a request to the UAA or any configured External OpenID Identity Provider.

   Note: In the above flow, we have shown UAA as the OpenID Identity Provider, you can also configure other external identity providers such as Google or your Company SSO.
2. The UAA authenticates the end-user and obtains authorization. User logs in using a web browsers and enters login credentials for authentication.
3. UAA returns an ID Token and an Access Token. The AuthN response is a redirect to the client application (if configured).
4. The client sends a request with the Access Token to the user info endpoint in UAA.
5. UAA returns claims (data) about the end-user.

Relying Party (Client Information)

The relying party or the client is usually your web or mobile application. You must secure the following Client information for OIDC configuration.

Fields	Values
Client ID	*`oidc_client_id`* OAuth 2.0 Client Identifier valid at the Authorization Server (OpendID Providers).
Client Secret	*`oidc_client_secret`* OAuth 2.0 Client Identifier secret.
Authorized Grant Types	`authorization_code` When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code.
Redirect URI	The URL to which a user is redirected to after logging in at the OpenID Provider. requires unauthenticated access.`https://`

Fields	Values
	`*.`*`<Identity Provider UAA base url>`*`/login/callback/*`
Scopes	Include the `openid` scope in your client application.
Authorities	Include the `openid` authority in your client application.

Related concepts

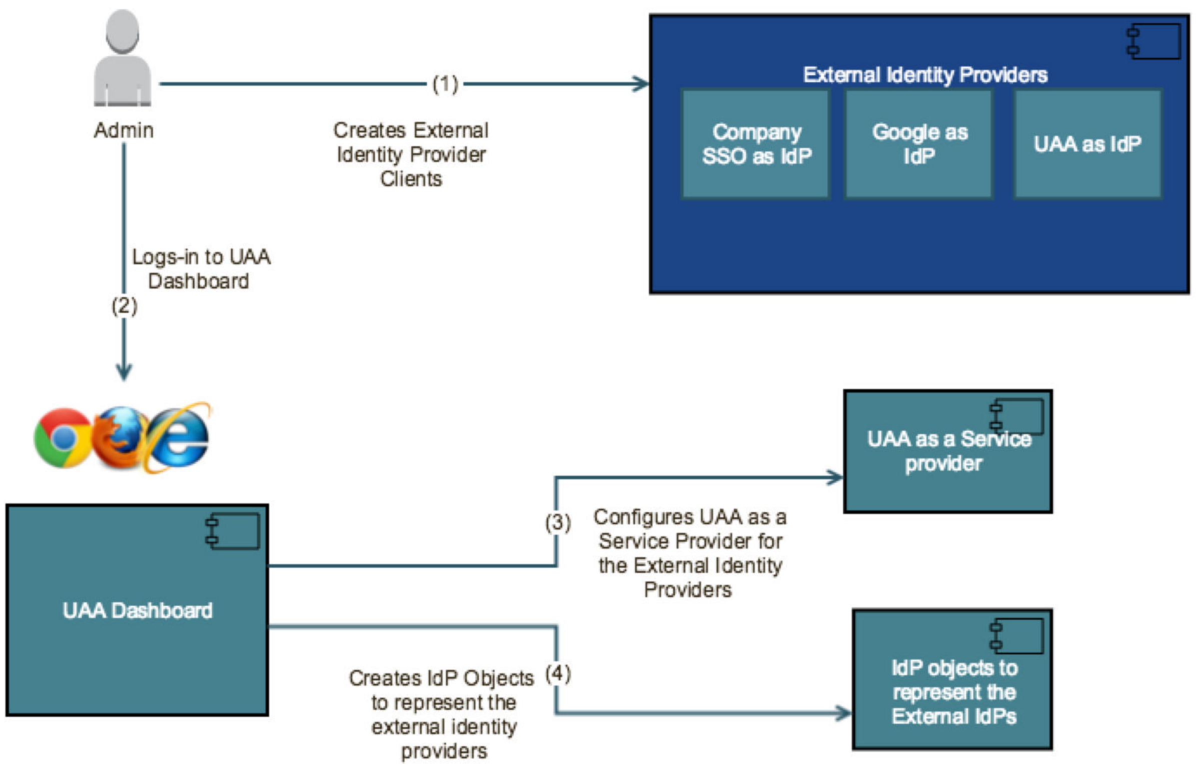
OIDC Federated Authentication Scenarios Supported in UAA on page 39

OIDC Federated Authentication Scenarios Supported in UAA

You can configure OIDC federated authentication using the UAA dashboard for the following scenarios :

- UAA to UAA OIDC Federated Authentication
- UAA to Company SSO Authentication
- UAA to Trusted External Identity Provider Authentication

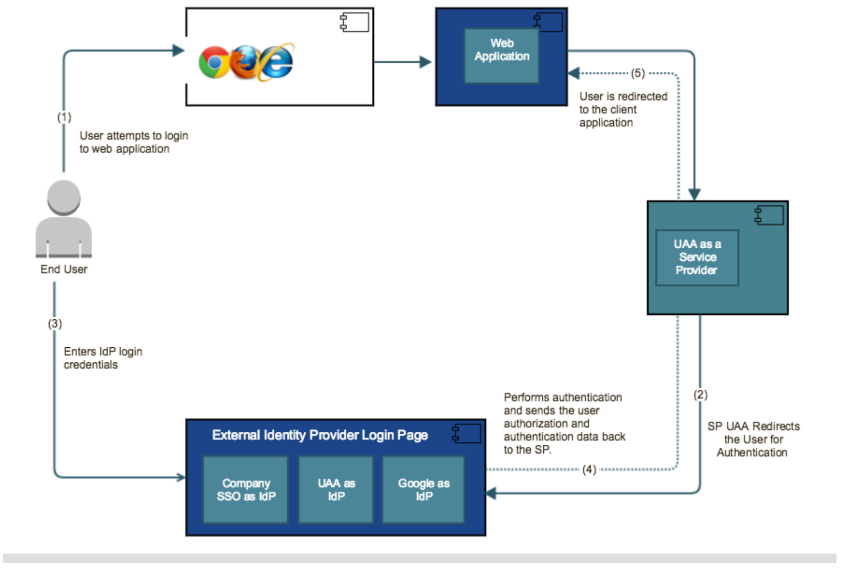
The following diagram shows the UAA SP to IdP flow for the Admin user:


Admin
(1)
Creates External Identity Provider Clients
External Identity Providers
Company SSO as IdP
Google as IdP
UAA as IdP
Logs-in to UAA Dashboard
(2)
UAA as a Service provider
UAA Dashboard
(3)
Configures UAA as a Service Provider for the External Identity Providers
(4)
Creates IdP Objects to represent the external identity providers
IdP objects to represent the External IdPs

In this flow the Admin:

1. Creates external identity provider clients for OIDC federated authentication.
2. Logs-in to the UAA dashboard.
3. Creates UAA service provider for external identity providers.
4. Creates IdP objects to represent the external identity providers.

The following diagram shows the UAA SP to External IdP flow for the End user:


Web Application
(5)
User is redirected to the client application
(1)
User attempts to login to web application
End User
UAA as a Service Provider
(3)
Enters IdP login credentials
Performs authentication and sends the user authorization and authentication data back to the SP.
(2)
SP UAA Redirects the User for Authentication
(4)
External Identity Provider Login Page
Company SSO as IdP
UAA as IdP
Google as IdP

1. User attempts to login to the web application and is directed to UAA service provider.
2. UAA service provider redirects the user to external identity providers for authentication.
3. User logs in to the External identity and enter the IdP login credentials. The external identity providers perform authentication and sends the user authorization and authentication data back to the UAA service provider. UAA generates access and id tokens based on user authentication.
4. User is redirected to the client application after the authentication is completed.

UAA to UAA OIDC Federated Authentication – You can configure OIDC federated authentication between two instances of UAA, and designate one as the identity provider (IdP) and the other as the service provider (SP) to allow the UAA service provider users to redirect to UAA identity provider for authentication. UAA (SP) instance acts as the server that receives the authentication and authorization data while the UAA (IdP) acts as the server that receives the authentication request, performs user authentication, and sends the authentication and authorization data back to the SP. See Configuring UAA as a Service Provider for OIDC Federated Authentication on page 42 for details on configuring UAA as a Service Provider.

UAA to Company SSO Authentication – You can configure your Company's SSO as the OIDC identity provider and UAA as the service provider to allow the users from your company to redirect to Company's SSO identity provider for authentication. When UAA is configured with Company SSO IdP, users can log in using their Company SSO credentials. The UAA (SP) must be set up with `openid`and any other scopes specific to your Company's SSO configuration that when UAA exchanges the authorization code for Company SSO token, the token includes required attributes for user authentication. You must set up your Company SSO as the allowed provider for your IdP client. See Example: Configuring UAA SP for OIDC Authentication Using Company SSO IdP on page 51

Note: Consult with your Company SSO administrator or refer to any Company SSO related documentation to include the required scopes.

UAA to Trusted External Identity Provider – You can also configure an OIDC compliant external identity provider such as Google to perform user authentication with UAA as the service provider (SP) to allow the users to redirect to the external identity provider for authentication. When Google is set up as an external OIDC federated identity provider, users can log in using their Google account credentials. UAA (SP) must be set up with the required `openid` and `email` scopes so that when UAA exchanges the authorization code for Google IdP token, the token includes required attributes for user authentication.

You must set up your Google IdP client before configuring UAA as the service provider for OIDC federated authentication. See Example: Configuring UAA to Google IdP OIDC Federated Authentication on page 58

Configuring UAA as a Service Provider for OIDC Federated Authentication

About This Task

You can configure UAA as a service provider for an external identity provider such as Company SSO or Google. You can also configure UAA as a service provider for another UAA instance that you may want to set up as your identity provider.

Before You begin

- Create your external identity providers and secure the required client information for authentication. If you are configuring OIDC federated authentication between two instances of UAA, see Configuring UAA as an OIDC Identity Provider on page 45.
- Log In to Predix.io and go to the Console view.
- Create or select a UAA instance that you want to configure as your identity provider.

The **Client Management** form represents your Client application, you can use the information in the Client Management form to verify your Redirect URIs or other client specific information.

Procedure

1. Navigate to the **Client Management** tab in the same UAA instance and click on **Create Client** information.

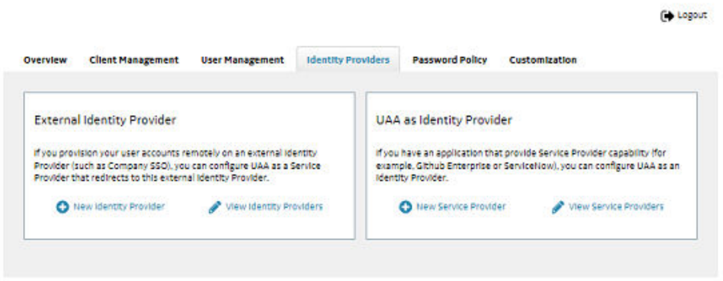
   Ensure the following information is added to the **Create Client** form.

Fields	**Descriptions**
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	**authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.
Redirect URI	Specify a redirect URI to redirect the client after login or logout. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints. For UAA to Company SSO OIDC configuration, the Redirect URI is the Url of your Company SSO.
Scopes	Scopes are permissions associated with a Client to determine user access to a resource through an application. **Note:** • For OIDC configuration, UAA (IdP) clients must have the `openid` scope. • For UAA to Company SSO OIDC configuration, you may want to include the `profile` scope in addition to the `openid` scope.

Fields	Descriptions
	• For UAA to Google IdP OIDC configuration, you must include `email` scope in addition to the `openid` scope
Allowed Providers	Select the OpenID Connect identity provider that you created through the identity provider configuration. **Note:** If an OIDC identity provider is not created prior to the service provider configuration, you may want to return to this task again to select the **Allowed Providers** after you create an identity provider for your service provider.

2. Click **Save**.
3. Click on the **Identity Providers** tab, navigate to **External Identity Provider** section and then click on the **New Identity Provider**.

   Example:



External Identity Provider

If you provision your user accounts remotely on an external identity Provider (such as Company SSO), you can configure UAA as a Service Provider that redirects to this external identity Provider.

UAA as Identity Provider

If you have an application that provide Service Provider capability (for example, Github Enterprise or ServiceNow), you can configure UAA as an Identity Provider.

4. In the **New Identity Provider** form, specify the following information :

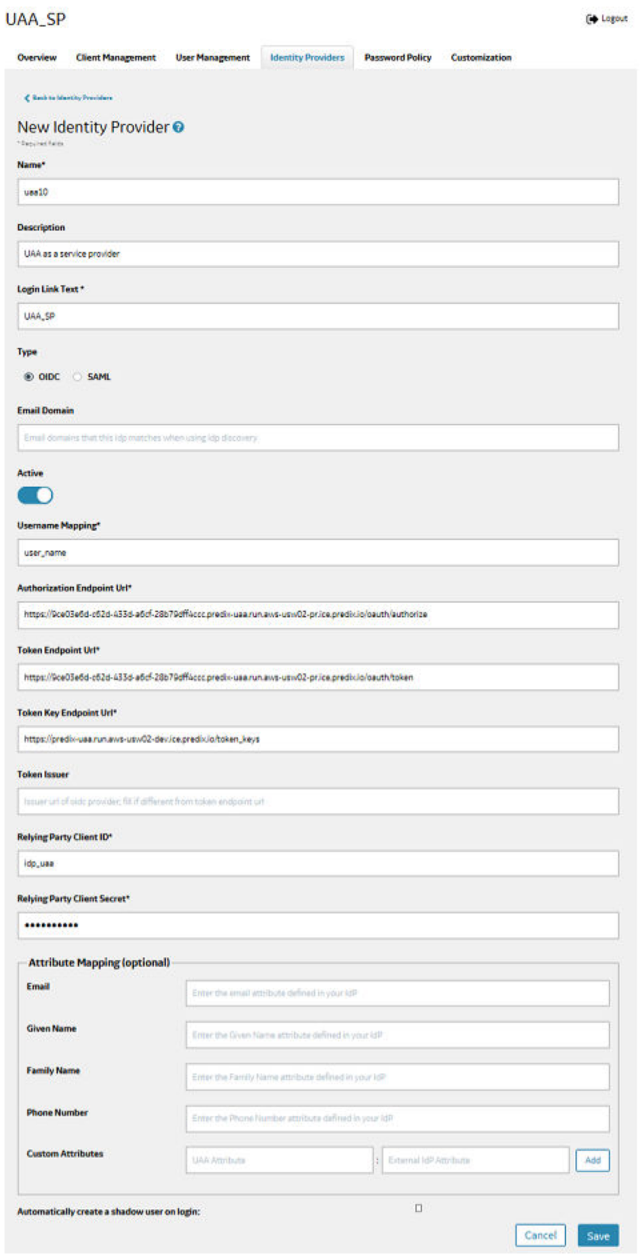
   Tip: OIDC compliant external IdPs have a “well-known” URL that provides information on Identity Provider fields such as, authorization endpoint, token endpoint etc. For example, you can browse the Google OIDC well-known endpoint link, https://accounts.google.com/.well-known/openid-configuration for information on Identity Provider field values.

Field	Description
Name	Specify the name of your IdP.
Description	(Optional) Specify a short description for your IdP.
Login Link Text	Specify the login text that you want to display for your IdP.
Type	Select OIDC as the type.
Email Domain	(Optional) Specify the email domain that UAA can use to identify the identity provider if you have configured more than one identity provider. To support this feature, UAA provides a configuration of IdP discovery. By default, this configuration is set to False. You can configure UAA to enable IDP Discovery if you have configured more than one identity provider. For more information, see Identifying IdP Using Email Domain on page 34
Active	Select this option to set the IdP as active.
Username Mapping	Specify the username attribute defined in your IdP. Username mapping is a required field to map `user_name` attribute (a unique identifier associated with a user) with a specific user from the IdP.

Field	Description
	Note: You must ensure that the OIDC attribute mapped to the `username` is unique for each user. Although OIDC specification guarantees a unique `sub` attribute for each user in the ID token, your OIDC provider may choose to provide a more suitable username attribute in a human-readable format.
Authorization Endpoint Url	Specify the authorization endpoint for OIDC token.
Token Endpoint Url	Specify the token Url for retrieving an OIDC token with an authorization code.
Token Key Endpoint Url	Specify the token key endpoint Url for token verification.
Token Issuer	Specify the Url of your OIDC provider.
Relying Party Client ID	Specify the OIDC provider Client id. Relying Party Client ID and Client Secret are checks to match the client ID and client secret stored in the authorization server before issuing an authorization token.
Relying Party Client Secret	Specify the OIDC provider client secret.
Attribute Mapping	(Optional) Specify the user attributes and any custom attributes specific to the user. Attribute Mapping allows you to propagate user's name attribute along with any other stored user information such as email addresses, first or last names etc., from the identity provider (For example, Company SSO) to the UAA service provider. These attributes along with any other stored user information are shared with applications via the OpenID tokens.
Automatically create a shadow user on login	Select this option only if you want to automatically create a local user in UAA to represent the federated users. You must manually create the local users when this option is set to a **false** value. Setting a **false** value allows an admin to control which users from an IdP can have access. In other words, admin can whitelist certain users to grant access to them. Setting this option to a **true**value grants access to any user from the federated IdP.

5. Click **Save**.

Example



Name*	uaa10
Description	UAA as a service provider
Login Link Text *	UAA_SP
Type	[✓ OIDC] SAML
Email Domain	Email domains that this idp matches when using idp discovery
Active	●
Username Mapping*	user_name
Authorization Endpoint Url*	https://9ce03e6d-c62d-433d-a6cf-28b79dff4ccc.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/authorize
Token Endpoint Url*	https://9ce03e6d-c62d-433d-a6cf-28b79dff4ccc.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/token
Token Key Endpoint Url*	https://predix-uaa.run.aws-usw02-dev.ice.predix.io/token_keys
Token Issuer	Issuer url of oidc provider, fill if different from token endpoint url
Relying Party Client ID*	idp_uaa
Relying Party Client Secret*	••••••••••

Attribute Mapping (optional)

Email	Enter the email attribute defined in your IdP
Given Name	Enter the Given Name attribute defined in your IdP
Family Name	Enter the Family Name attribute defined in your IdP
Phone Number	Enter the Phone Number attribute defined in your IdP
Custom Attributes	UAA Attribute : External IdP Attribute

Automatically create a shadow user on login: ☐

Next Steps

Verifying UAA to UAA OIDC Configuration on page 49

Configuring UAA as an OIDC Identity Provider

About This Task

You can set up your configuration to use one instance of UAA as a service provider (SP) and the other as the identity provider (IdP) from the UAA dashboard. We recommend that you configure your UAA (IdP) instance before configuring the UAA (SP) instance.

Before You begin

- Select or create a UAA instance that you want to set up as your IdP.

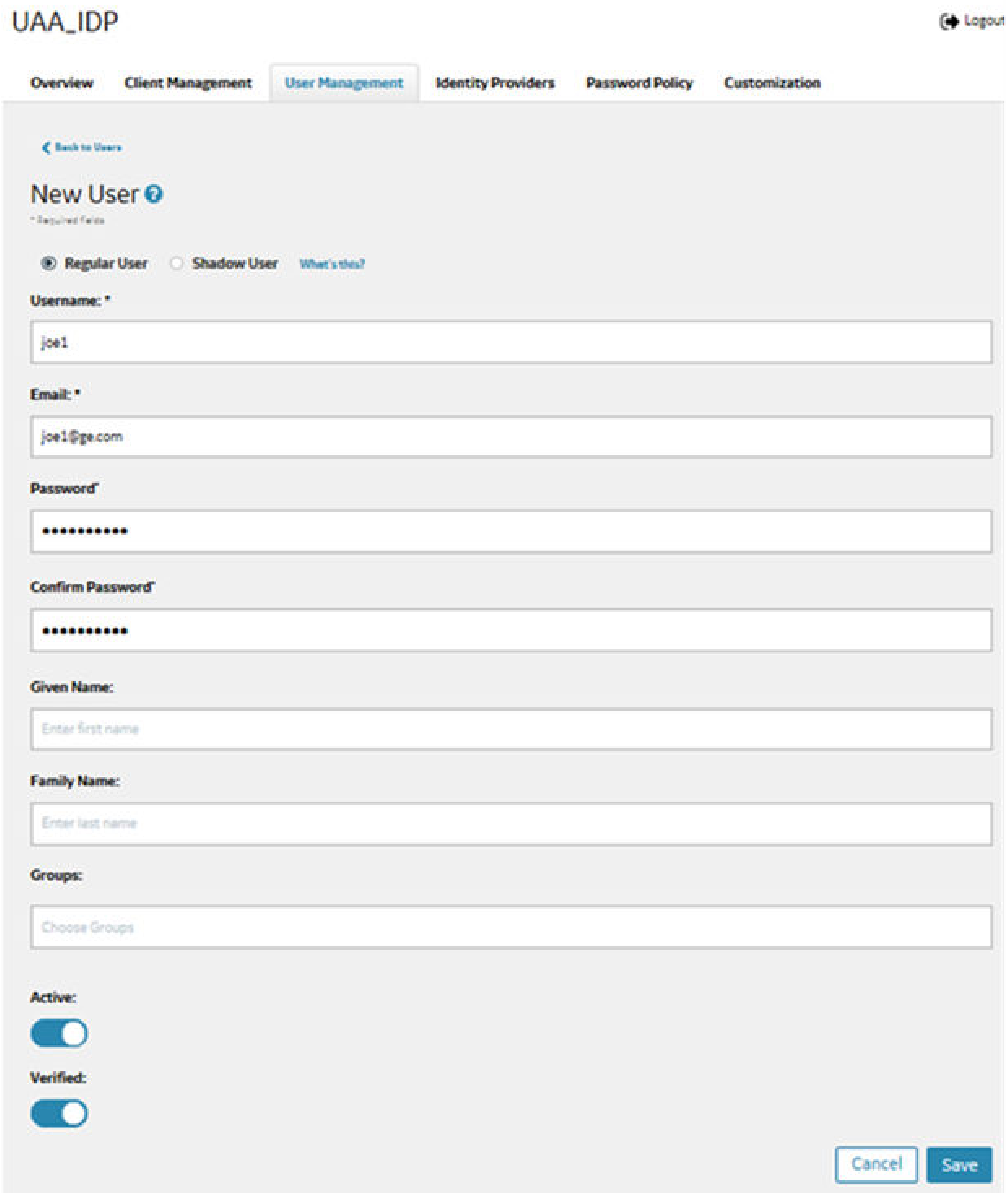
Procedure

1. Open your UAA (IdP) instance and add the **New User** information in the **User Management** tab.

   Ensure that the following information is entered and saved in the **New User** form.

Field	Description
Regular User	Choose this option to set up local users in your UAA. The **Regular User** is not configured through an external Identity Provider (IdP).
Shadow User	Choose this option to create a local user in UAA corresponding to the user defined in your external IdP. The **Shadow User** option is useful if you need to whitelist users to authenticate only a subset of users setup in your identity provider. To set up individual shadow users, ensure that the option to create shadow users is not selected while configuring a new IdP.
User Name	Specify the user name. If you are setting up a shadow user, this value must match the user name defined in your IdP.
Email	Specify the email address. If you are setting up a shadow user, this value must match the user name defined in your IdP.
Password	Specify the password. An administrator can set password policies to define the permitted structure of the password. For more information, see Creating Password Policies on page 27. This option is not required if you are setting up a shadow user.
Active	Select this option to allow your Regular or Shadow user to login.
Verified	Select this option to indicate that this Regular or Shadow user is a verified user.

Example:


UAA_IDP

Username: * joe1

Email: * joe1@ge.com

Password* ••••••••••••

Confirm Password* ••••••••••••

Given Name:

Family Name:

Groups:

Active: ●

Verified: ●

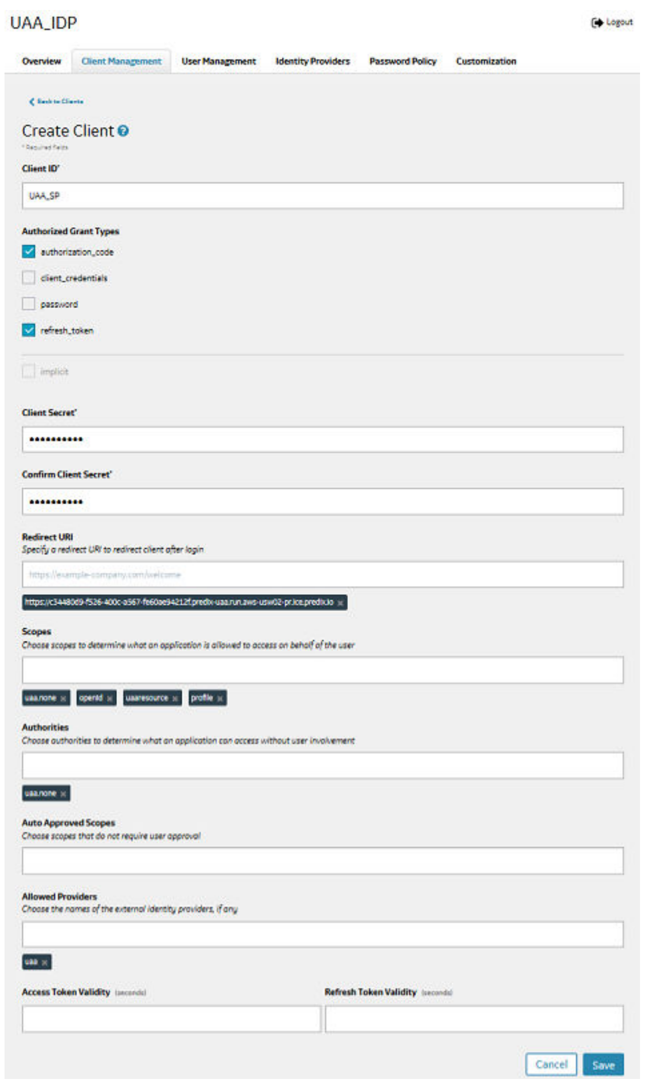
2. Click **Save**.

   The Client Management form represents your Client application, you can use the information in the Client Management form to verify the destination of your Redirect URIs or other client specific information.
3. Navigate to **Client Management** tab and click on **Create Client** information.

   Ensure the following information is added to the Client form

Fields	Descriptions
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	**authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.
Redirect URI	Specify a redirect URI to redirect the client after login or logout. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints.
Scopes	Scopes are permissions associated with a Client to determine user access to a resource through an application. Add the `openid` and `uaa.resource` scopes. **Note:** For OIDC configuration, UAA (IdP) clients must have the `openid` scope.

Example:



Create Client

* Required Fields

Client ID*: UAA_SP

Authorized Grant Types
- [✓] authorization_code
- [ ] client_credentials
- [ ] password
- [✓] refresh_token
- [ ] implicit

Client Secret*: ••••••••••

Confirm Client Secret*: ••••••••••

Redirect URI
Specify a redirect URI to redirect client after login
https://example-company.com/welcome
https://c34480d9-f526-400c-a567-fe60ae94212f.predix-uaa.run.aws-usw02-pr.ice.predix.io

Scopes
Choose scopes to determine what an application is allowed to access on behalf of the user
uaa.none, openid, uaaresource, profile

Authorities
Choose authorities to determine what an application can access without user involvement
uaa.none

Auto Approved Scopes
Choose scopes that do not require user approval

Allowed Providers
Choose the names of the external identity providers, if any
uaa

Access Token Validity (seconds)

Refresh Token Validity (seconds)

4. Click **Save**.

Next Steps

Configuring UAA as Service Provider for External Identity Provider on page 29

Related tasks

Verifying UAA to UAA OIDC Configuration on page 49

Verifying UAA to UAA OIDC Configuration

Before You begin

Ensure you have the following information ready before you perform the verification task:

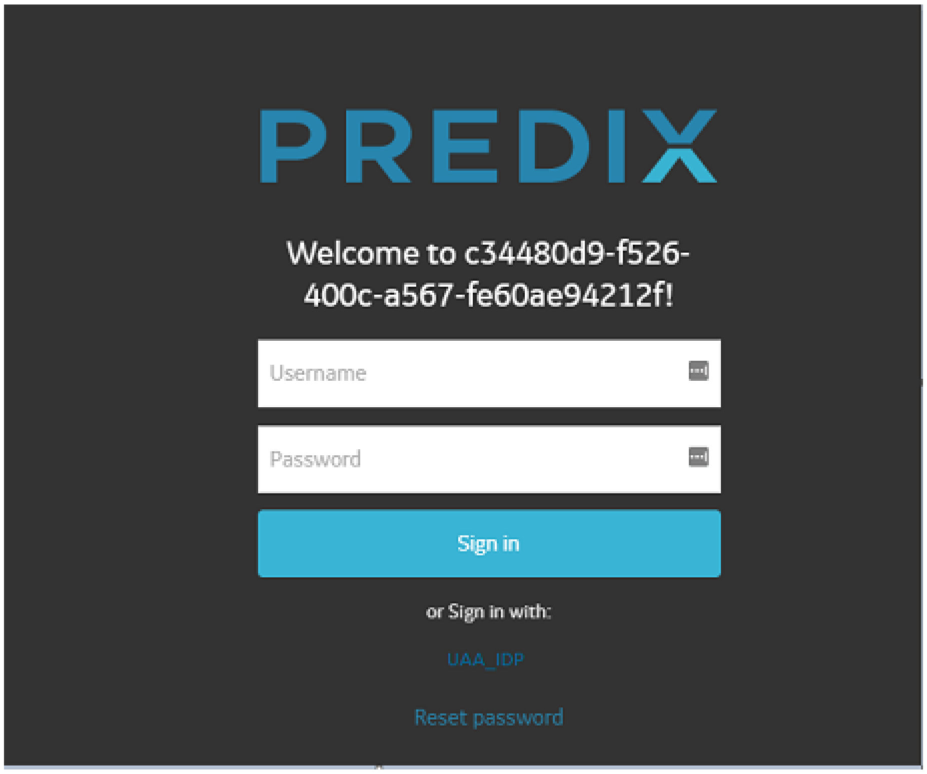
- Redirect URI for the UAA(IdP)
- User credentials (username, email, and password) for the user that you created with **OpenId** scope in UAA(IdP). See Configuring UAA to UAA OIDC Federated Authentication

Procedure

1. Enter the Redirect URI link for the UAA(IdP) in your browser.

   A UAA login screen displays with the **Link Login Text** that you entered for your IdP login.

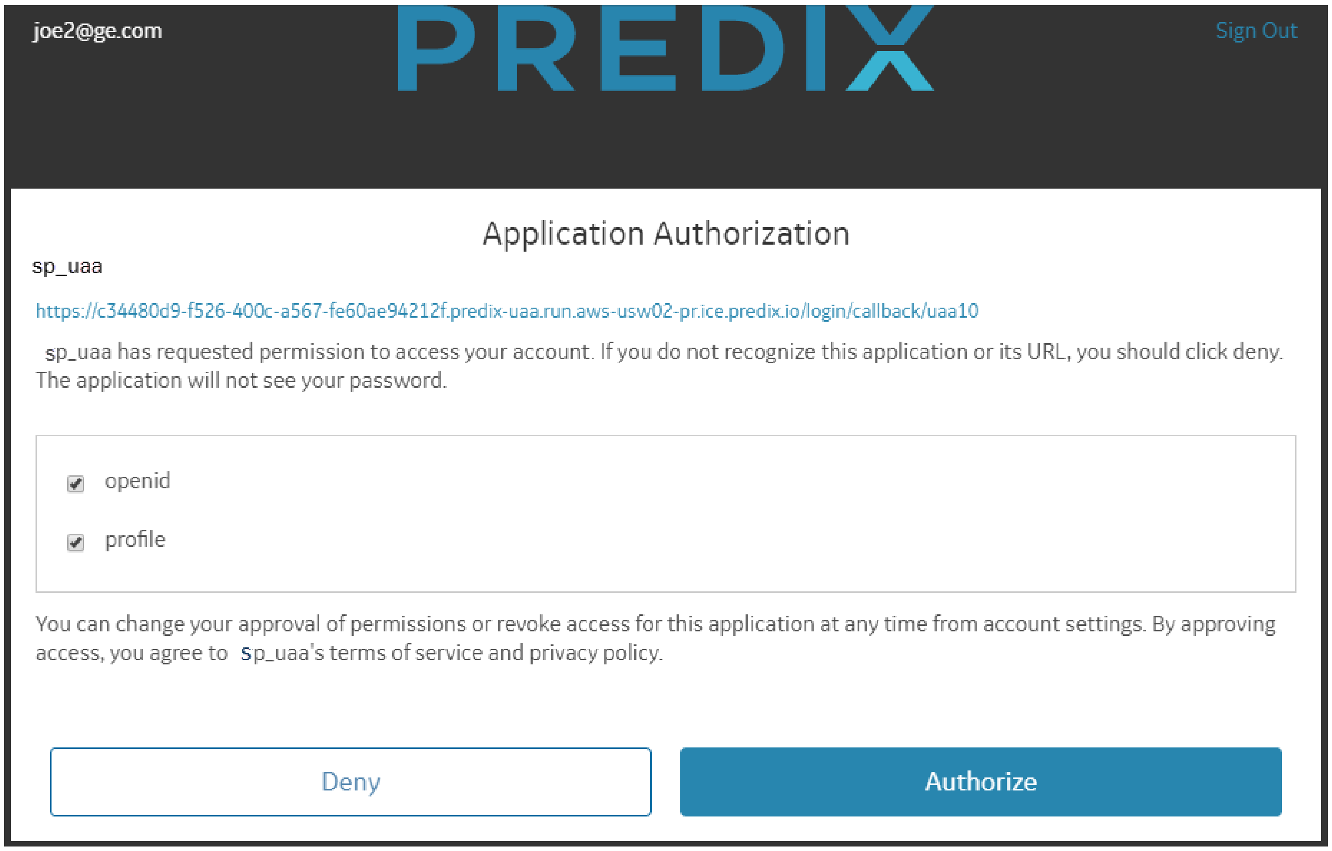
   Example:

   If the **Login Link Text** (UAA_IDP in the above example) matches the text that you entered in the UAA_SP's **Identity Provider Form** for the IdP login, it confirms that the OIDC configuration is completed successfully.
2. Click on the **Login Link Text**, you are redirected to the IdP login page. You can enter your user credentials (email/username and password) that you created in UAA_IdP.

   On successful login, the **Application Authorization** window displays. You can choose to **Authorize** or **Deny** access to your application.

   Example:



Application Authorization

sp_uaa

https://c34480d9-f526-400c-a567-fe60ae94212f.predix-uaa.run.aws-usw02-pr.ice.predix.io/login/callback/uaa10

sp_uaa has requested permission to access your account. If you do not recognize this application or its URL, you should click deny. The application will not see your password.

- ✓ openid
- ✓ profile

You can change your approval of permissions or revoke access for this application at any time from account settings. By approving access, you agree to sp_uaa's terms of service and privacy policy.

Example: Configuring UAA SP for OIDC Authentication Using Company SSO IdP

About This Task

The following example shows the UAA SP configuration from the dashboard for OIDC federated authentication with your Company SSO.

Note: This example does not include instructions to configure your Company SSO Client information.

Before You begin

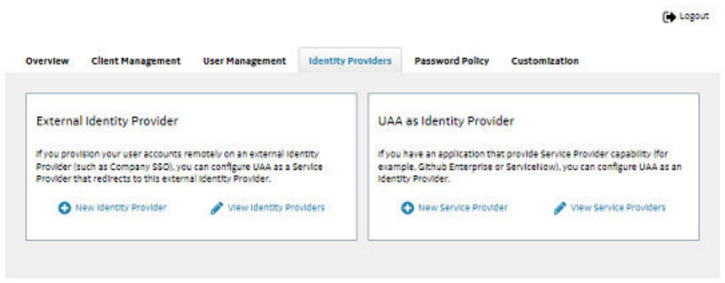
- Create the Company SSO identity provider information.
- Create or select a UAA instance that you want to configure as the SP for your Company SSO IdP.

Complete the following procedure to configure Company SSO as the external identity provider.

Procedure

1. Click on the **Identity Providers** tab, navigate to **External Identity Provider** section and then click on the **New Identity Provider**.

   Example:



External Identity Provider

If you provision your user accounts remotely on an external Identity Provider (such as Company SSO), you can configure UAA as a Service Provider that redirects to this external Identity Provider.

UAA as Identity Provider

If you have an application that provide Service Provider capability (for example, Github Enterprise or ServiceNow), you can configure UAA as an Identity Provider.

2. In the **New Identity Provider** form, specify the following information :

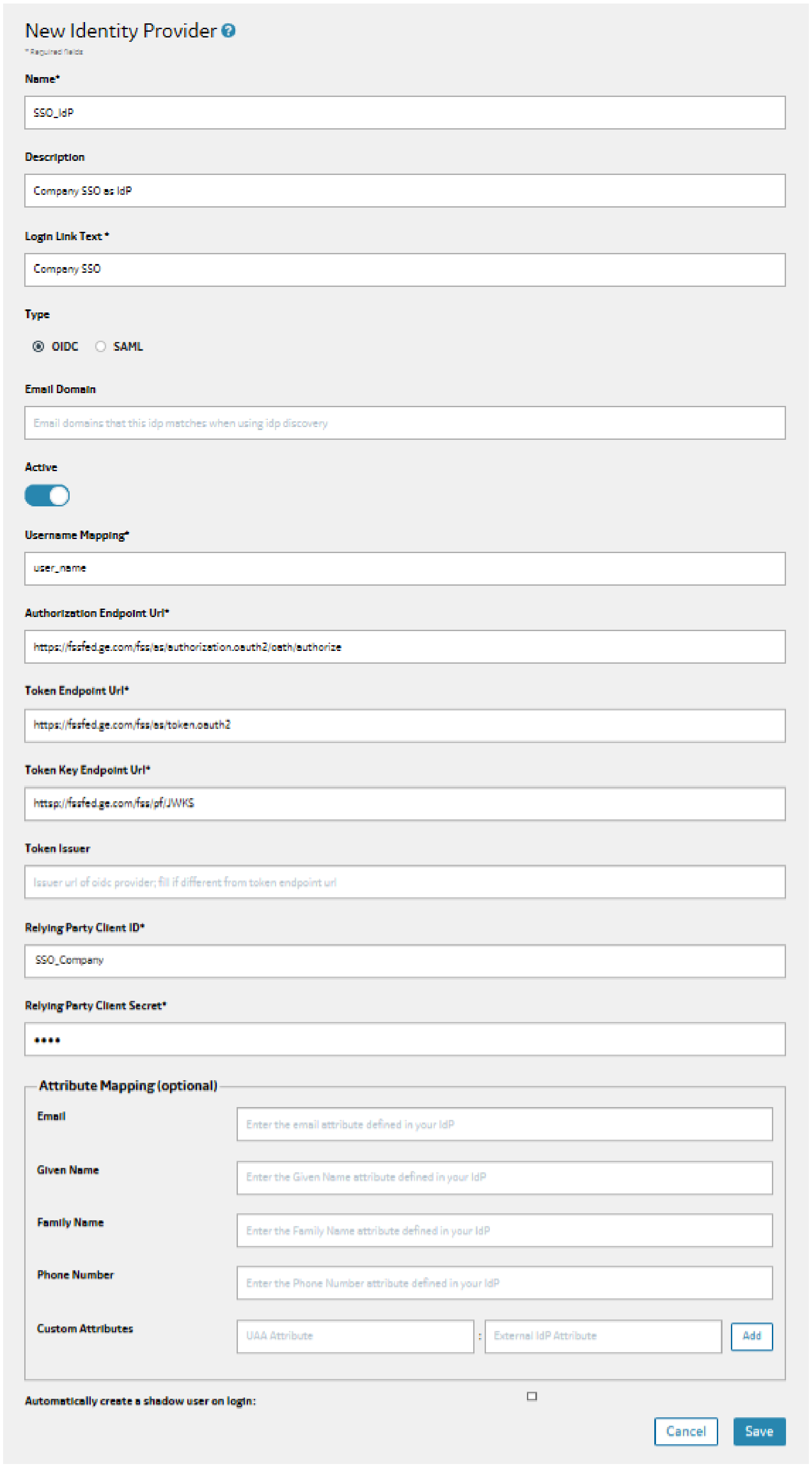
Field	Description
Name	Specify the name of your IdP.
Description	(Optional) Specify a short description for your IdP.
Login Link Text	Specify the login text that you want to display for your IdP.
Type	Select OIDC as the type.
Email Domain	(Optional) Specify the email domain that UAA can use to identify the identity provider if you have configured more than one identity provider.
Active	Select this option to set the IdP as active.
Username Mapping	Specify the username attribute defined in your IdP. Username mapping is a required field to map `user_name` attribute (a unique identifier associated with a user) with a specific user from the IdP. **Note:** You must ensure that the OIDC attribute mapped to `user_name` is unique for each user. Although OIDC specification guarantees a unique `sub` attribute for each user in the ID token, your OIDC provider may choose to provide a more suitable username attribute in a human-readable format.
Authorization Endpoint Url	Specify the authorization endpoint for OIDC token.
Token Endpoint Url	Specify the token Url for OIDC authorization code.
Token Key Endpoint Url	Specify the token Url for retrieving an OIDC token with an authorization code.
Token Issuer	Specify the Url of your OIDC provider.
Relying Party Client ID	Specify the OIDC provider Client id. Relying Party Client ID and Client Secret are checks to match the client ID and client secret stored in the authorization server before issuing an authorization token.
Relying Party Client Secret	Specify the OIDC provider client secret.
Attribute Mapping	(Optional) Specify the user attributes and any custom attributes specific to the user. Attribute Mapping allows you to propagate user's name attribute along with any other stored user information such as email addresses, first or last names etc., from the identity provider (Company SSO) to the service provider (UAA). These attributes along with any other stored user information are shared with applications via the OpenID tokens.

Field	Description
Automatically create a shadow user on login	Select this option only if you want to create a local user in UAA corresponding to each user defined in the identity provider that you are configuring. You must manually create the local users when this option is set to a **false** value. Setting a **false** value allows an admin to control which users from an IdP can have access. In other words, admin can whitelist certain users to grant access to them. Setting this option to a **true**value grants access to any user from the federated IdP.

Example:

Note: In the following example we have used these Urls:

- `https://fssfed.ge.com/fss/as/authorization.oauth2` as the Authorization End-point Url
- `https://fssfed.ge.com/fss/as/token.oauth2` as the Token End-point Url
- `httsp://fssfed.ge.com/fss/pf/JWKS` as the Token Key End point Url.



New Identity Provider

* Required fields

Name*

SSO_IdP

Description

Company SSO as IdP

Login Link Text *

Company SSO

Type

[✓ OIDC] SAML

Email Domain

Email domains that this idp matches when using idp discovery

Active

Username Mapping*

user_name

Authorization Endpoint Url*

https://fssfed.ge.com/fss/as/authorization.oauth2/oath/authorize

Token Endpoint Url*

https://fssfed.ge.com/fss/as/token.oauth2

Token Key Endpoint Url*

httsp://fssfed.ge.com/fss/pf/JWKS

Token Issuer

Issuer url of oidc provider; fill if different from token endpoint url

Relying Party Client ID*

SSO_Company

Relying Party Client Secret*

••••

Attribute Mapping (optional)

Email	Enter the email attribute defined in your IdP
Given Name	Enter the Given Name attribute defined in your IdP
Family Name	Enter the Family Name attribute defined in your IdP
Phone Number	Enter the Phone Number attribute defined in your IdP
Custom Attributes	UAA Attribute : External IdP Attribute Add

Automatically create a shadow user on login: ☐

Cancel Save

3. Click **Save**.

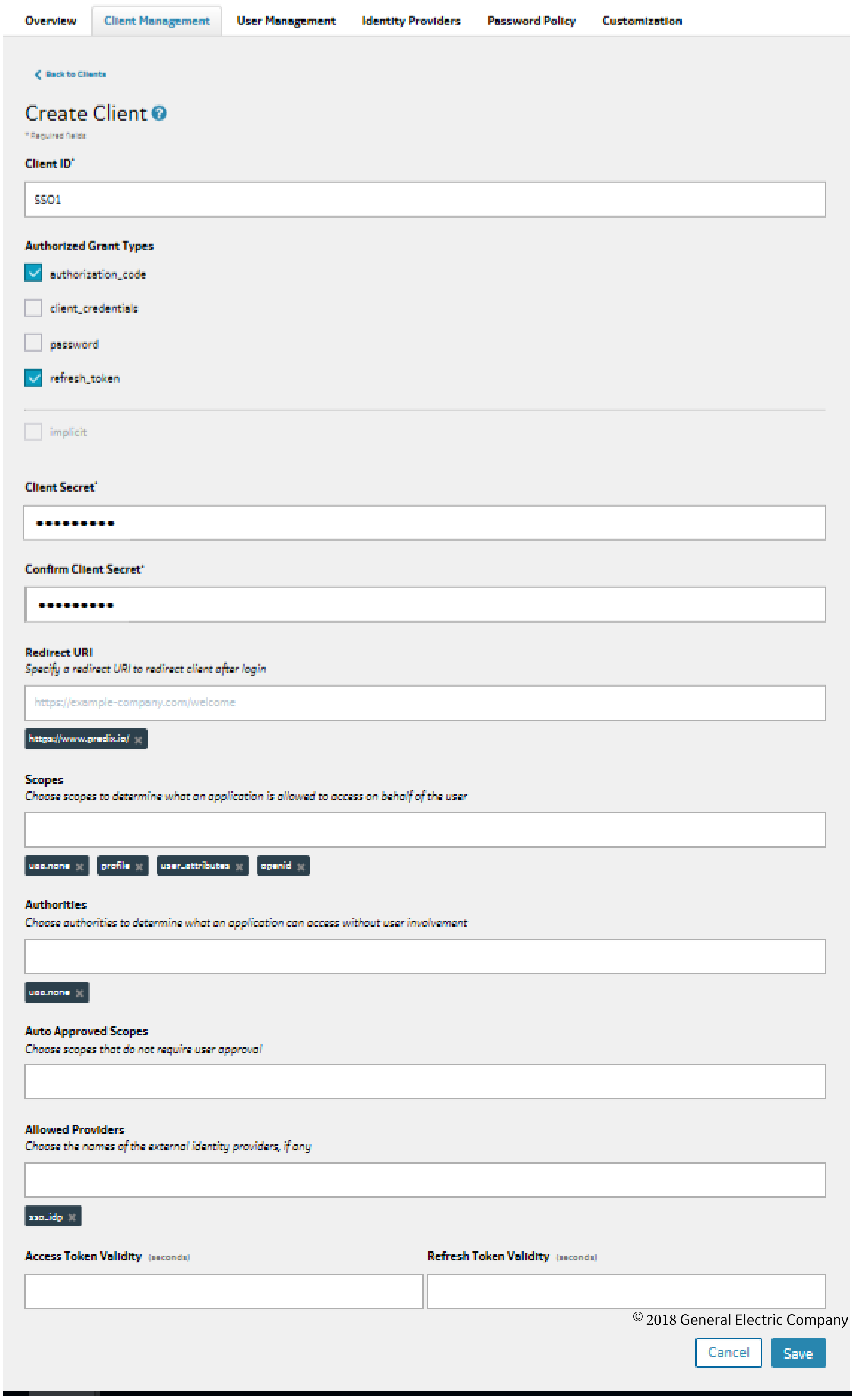
   The **Client Management** form represents your Client application, you can use the information in the Client Management form to verify the destination of your Redirect URIs or other client specific information.
4. Navigate to **Client Management** tab in the same UAA instance and click on **Create Client** information.

   Ensure the following information is added to the **Create Client** form

Fields	Descriptions
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	**authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.
Redirect URI	Specify a redirect URI (Url of your Company SSO) to redirect the client after login or logout. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints. The Redirect URI for this client is the link to the application that the user is trying to use. Once authentication is successful, UAA sends the application user back to the application.
Scopes	Scopes are permissions associated with a Client to determine user access to a resource through an application. Add the `openid`, `user_attributes`, and `profile` scopes. **Note:** Depending upon your Company's SSO setup, the scopes may vary for your configuration. Consult with your Company SSO administrator or any related documentation to include the required scopes.
Allowed Providers	Select the OpenID Connect identity provider that you created. For this example, we created a provider name, “ SSO_IdP” in **Step 2**.

Example:

Note: In the following example, we have used the Predix.io Url (https://www.predix.io/) as our Redirect URI.



Create Client

* Required fields

Client ID*

SSO1

Authorized Grant Types

- ✓ authorization_code
- ☐ client_credentials
- ☐ password
- ✓ refresh_token
- ☐ implicit

Client Secret*

••••••••••

Confirm Client Secret*

••••••••••

Redirect URI

Specify a redirect URI to redirect client after login

https://example-company.com/welcome

https://www.predix.io/ ×

Scopes

Choose scopes to determine what an application is allowed to access on behalf of the user

uaa.none × profile × user_attributes × openid ×

Authorities

Choose authorities to determine what an application can access without user involvement

uaa.none ×

Auto Approved Scopes

Choose scopes that do not require user approval

Allowed Providers

Choose the names of the external identity providers, if any

sso_idp ×

Access Token Validity (seconds)

Refresh Token Validity (seconds)

5. Click **Save**.

Example: Verifying Company SSO as an OIDC IdP Configuration

Before You begin

Ensure that you have your UAA service provider instance link handy. Your UAA instance link must be in this format, `https://<uaa_zone_id>.<uaa_url>/login`.

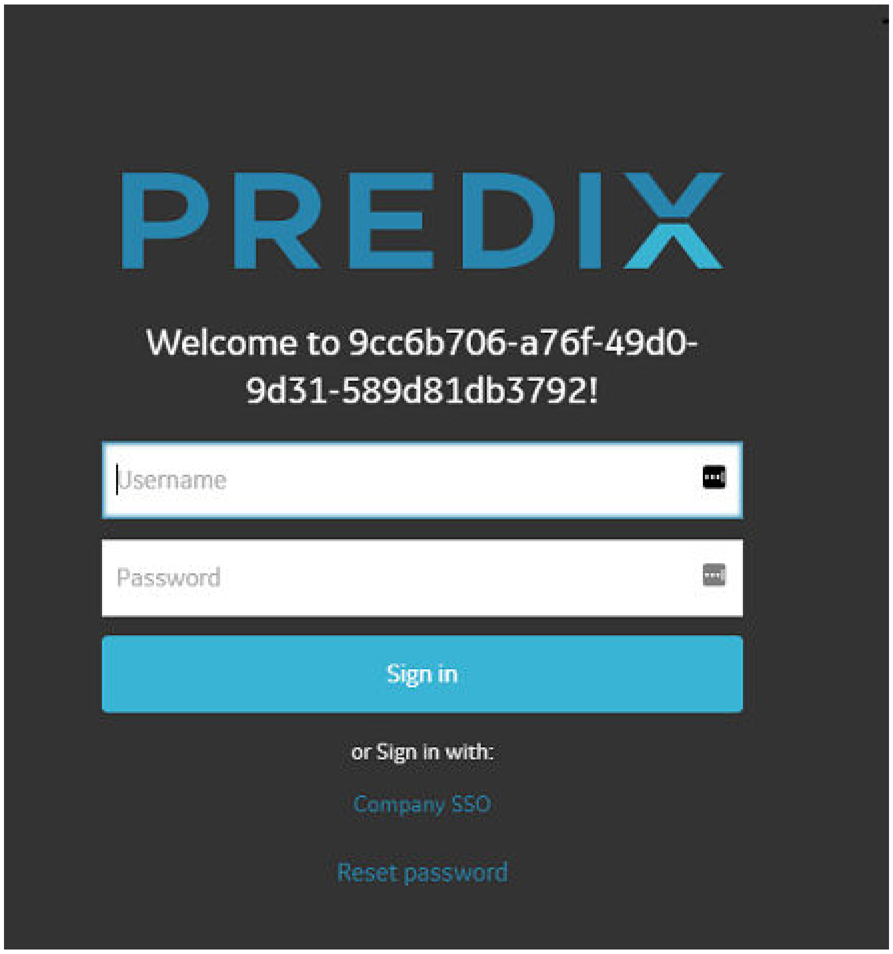
Procedure

1. Open a new browser and enter the UAA instance link that you had configured for Company SSO IdP.

   Example: `https://9cc6b706-a76f-49d0-9d31-589d81db3792.predix-uaa.run.aws-usw02-pr.ice.predix.io/login`

   A UAA dashboard login screen displays with the **Link Login Text** that you entered for your IdP login.

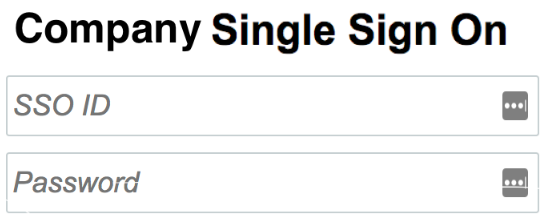
   Example:



Welcome to 9cc6b706-a76f-49d0-9d31-589d81db3792!

2. Click on the **Login Link Text** ("Company SSO" in the above example) link and login.

   You are redirected to your Company SSO login page, where you can enter your SSO credentials and back to UAA before you finally land on the Redirect URI page specified in your UAA (SP) settings.



Example: Configuring UAA to Google IdP OIDC Federated Authentication

About This Task

You must configure the Google IdP Client information before you configure UAA as your Service Provider for OIDC federated authentication with Google IdP.

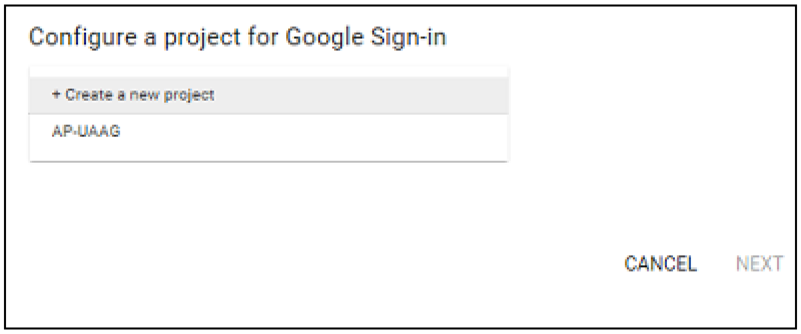
Follow these steps to set up your Google IdP Client:

Setting Up Google IdP Client Information

Procedure

1. Create a new project in Google Sign-in for Websites. If you already have a project you can skip this step and select your project.

   Example

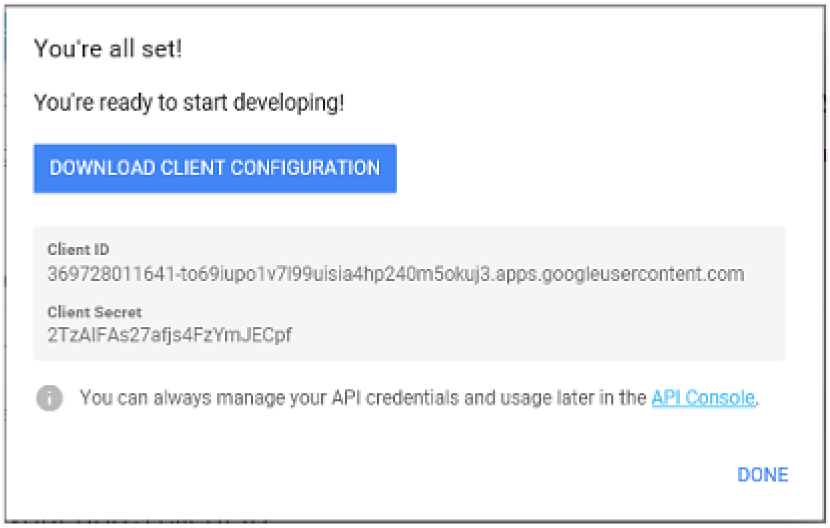
   

2. Click **Next** to add the Authorized Redirect URI. For this configuration, the Authorized Redirect URI is your UAA service provider instance Zone ID and UAA Url. You can enter this information in the following format:

   ```
   https://<uaa_zone_id>.<uaa_url>/login/callback/*
   ```

3. Download and save the Client ID and Client Secret for your Google IdP client. This is your relying party client ID and secret. You will need this information when you configure UAA as a service provider.

   Example



You're all set!

You're ready to start developing!

Client ID
369728011641-to69iupo1v7l99uisia4hp240m5okuj3.apps.googleusercontent.com

Client Secret
2TzAlFAs27afjs4FzYmJECpf

You can always manage your API credentials and usage later in the API Console.

Configuring UAA as an SP for OIDC Authentication with Google IdP

About This Task

The following example shows how to configure UAA as a service provider for Google identity provider.

Note: Any changes to the external user accounts must be performed on the external identity provider directly.

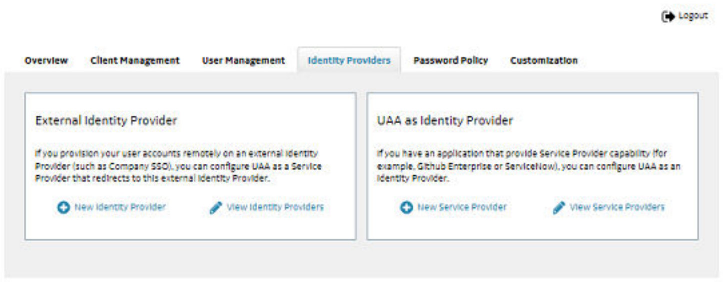
Before You begin

- Configure your Google IdP client information as outlined in the *Setting Up Google IdP Client Information* section.
- Download and save the Client ID and Client Secret for your Google IdP client. You will need this information to configure UAA as a service provider.

Procedure

1. Click on the **Identity Providers** tab, navigate to **External Identity Provider** section and then click on the **New Identity Provider**.

   Example:



External Identity Provider

If you provision your user accounts remotely on an external identity Provider (such as Company SSO), you can configure UAA as a Service Provider that redirects to this external Identity Provider.

UAA as Identity Provider

If you have an application that provide Service Provider capability (for example, Github Enterprise or ServiceNow), you can configure UAA as an Identity Provider.

2. In the **New Identity Provider** form, specify the following information :

Field	Description
Name	Specify the name of your IdP.
Description	(Optional) Specify a short description for your IdP.

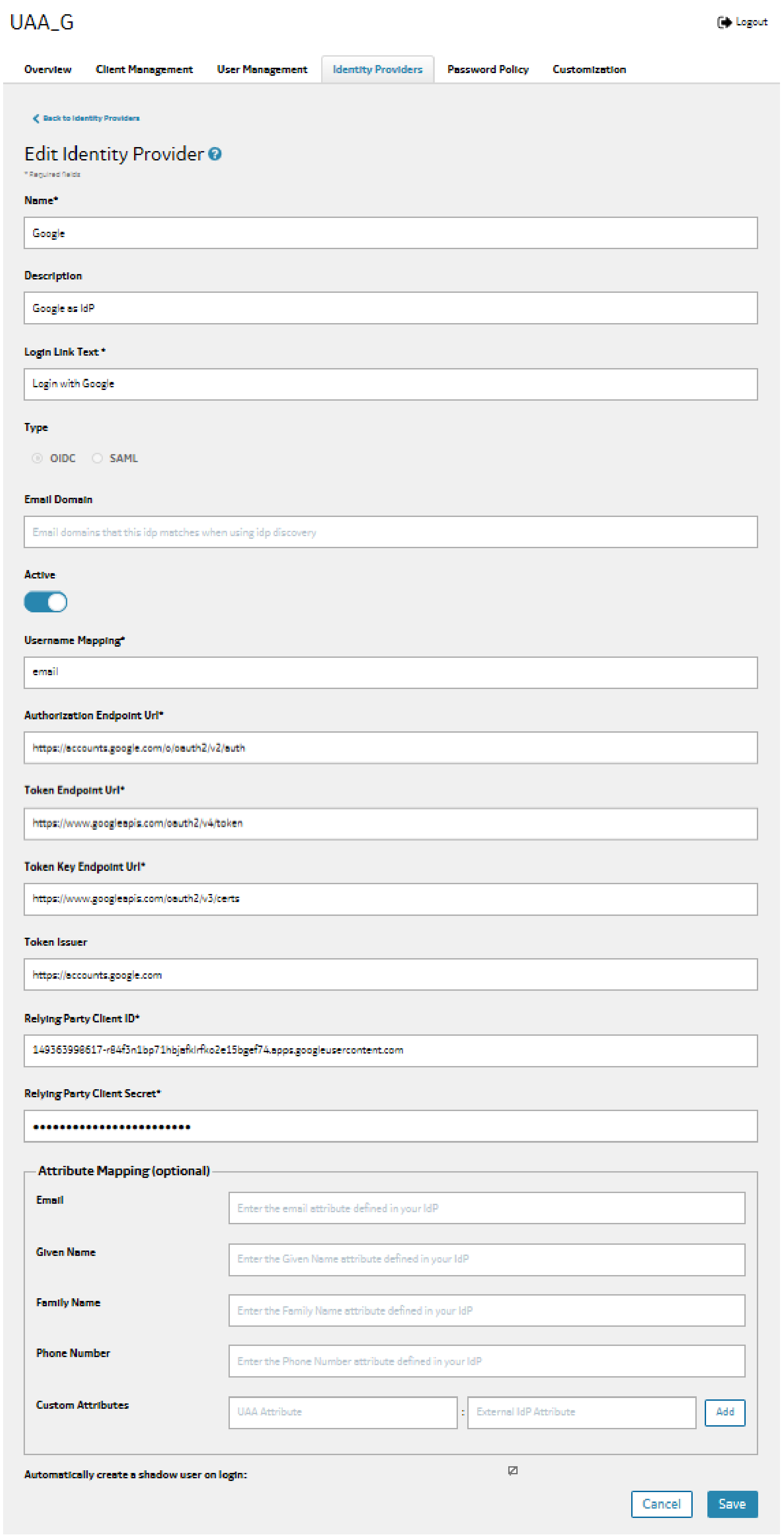
Field	Description
Login Link Text	Specify the login text that you want to display for your IdP.
Type	Select OIDC as the type.
Email Domain	(Optional) Specify the email domain that UAA can use to identify the identity provider if you have configured more than one identity provider.
Active	Select this option to set the IdP as active.
Username Mapping	Specify the username attribute defined in your IdP. Username mapping is a required field to map `user_name` attribute (a unique identifier associated with a user) with a specific user from the IdP. **Note:** You must ensure that the OIDC attribute mapped to `username` is unique for each user. Although OIDC specification guarantees a unique `sub` attribute for each user in the ID token, your OIDC provider may choose to provide a more suitable username attribute in a human-readable format.
Authorization Endpoint Url	Specify the authorization endpoint for OIDC token.
Token Endpoint Url	Specify the token Url for OIDC authorization code.
Token Key Endpoint Url	Specify the token key endpoint Url for token verification.
Token Issuer	Specify the Url of your OIDC provider.
Relying Party Client ID	Specify the OIDC provider Client id. Relying Party Client ID and Client Secret are checks to match the client ID and client secret stored in the authorization server before issuing an authorization token. Use the Client ID that your downloaded and saved in **Step 2** of the *Setting Up Google IdP Client Information* section.
Relying Party Client Secret	Specify the OIDC provider client secret. Use the Client Secret that your downloaded and saved in **Step 2** of the *Setting Up Google IdP Client Information* section.
Attribute Mapping	(Optional) Specify the user attributes and any custom attributes specific to the user. Attribute Mapping allows you to propagate user's name attribute along with any other stored user information such as email addresses, first or last names etc., from the identity provider (Google) to the service provider (UAA). These attributes along with any other stored user information are shared with applications via the OpenID tokens.
Automatically create a shadow user on login	Select this option to create a local user in UAA corresponding to each user defined in the identity provider that you are configuring.

Example:

Note: In the following example we have used these Urls:

- `https://accounts.google.com/o/oauth2/v2/auth` as the Authorization End-point Url
- `https://www.googleapis.com/oauth2/v4/token` as the Token End-point Url
- `https://www.googleapis.com/oauth2/v3/certs` as the Token Key End point Url.

Example:



UAA_G

Logout

Overview | Client Management | User Management | Identity Providers | Password Policy | Customization

Back to Identity Providers

Edit Identity Provider

* Required fields

Name*	Google
Description	Google as IdP
Login Link Text *	Login with Google
Type	OIDC SAML
Email Domain	Email domains that this idp matches when using idp discovery
Active	
Username Mapping*	email
Authorization Endpoint Url*	https://accounts.google.com/o/oauth2/v2/auth
Token Endpoint Url*	https://www.googleapis.com/oauth2/v4/token
Token Key Endpoint Url*	https://www.googleapis.com/oauth2/v3/certs
Token Issuer	https://accounts.google.com
Relying Party Client ID*	149363998617-r84f3n1bp71hbjefklrfko2e15bgef74.apps.googleusercontent.com
Relying Party Client Secret*	••••••••••••••••••••••••

Attribute Mapping (optional)

Email	Enter the email attribute defined in your IdP
Given Name	Enter the Given Name attribute defined in your IdP
Family Name	Enter the Family Name attribute defined in your IdP
Phone Number	Enter the Phone Number attribute defined in your IdP
Custom Attributes	UAA Attribute : External IdP Attribute Add

Automatically create a shadow user on login: ☑

Cancel Save

3. Click **Save**.
4. Login to UAA Dashboard and create or select a UAA instance that you want to configure as your service provider.

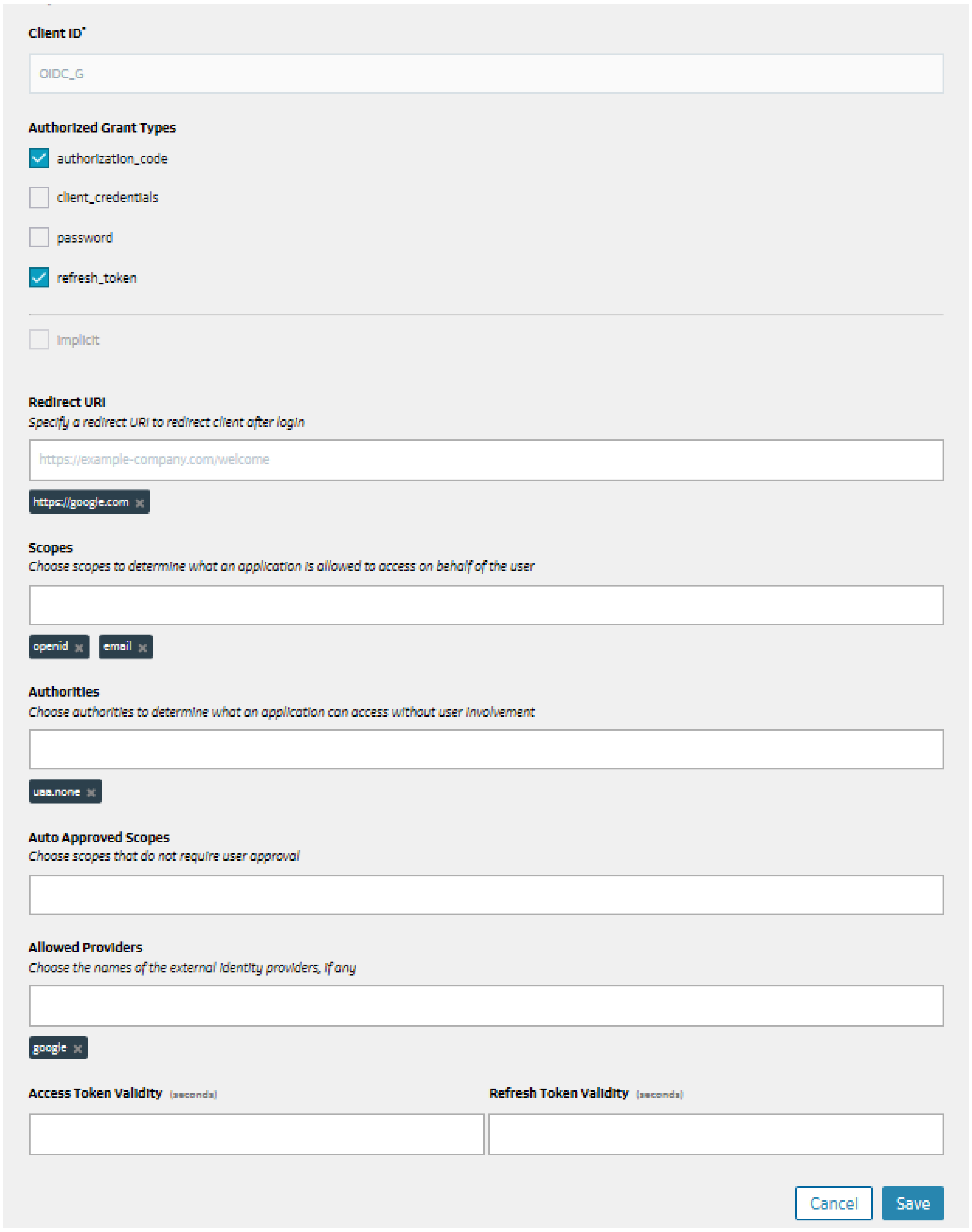
   The **Client Management** form represents your Client application, you can use the information in the Client Management form to verify the Redirect URIs or other client specific information.

   You can use the Client Management form to enter auto-approve scopes for this client. If there are scopes that you want automatically populated (without user consent), you can add these scopes to the Client Management form.
5. Navigate to **Client Management** tab in the same UAA instance and click on **Create Client** information.

   Ensure the following information is added to the **Create Client** form

Fields	**Descriptions**
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	**authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.
Redirect URI	Specify a redirect URI to redirect the client after login or logout. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints. For example: `https://google.com`
Scopes	Scopes are permissions associated with a Client to determine user access to a resource through an application. **Note:** • For UAA to Google IdP OIDC configuration, you must include `email` scope in addition to the `openid` scope
Allowed Providers	Select the OpenID Connect identity provider that you created through the identity provider configuration. For this example, we created a provider named, " Google" in **Step 2**.

Example:



Client ID*

OIDC_G

Authorized Grant Types

- [x] authorization_code
- [ ] client_credentials
- [ ] password
- [x] refresh_token
- [ ] implicit

Redirect URI

Specify a redirect URI to redirect client after login

https://example-company.com/welcome

https://google.com

Scopes

Choose scopes to determine what an application is allowed to access on behalf of the user

openid email

Authorities

Choose authorities to determine what an application can access without user involvement

uaa.none

Auto Approved Scopes

Choose scopes that do not require user approval

Allowed Providers

Choose the names of the external identity providers, if any

google

Access Token Validity (seconds)

Refresh Token Validity (seconds)

6. Click **Save**.

Next Steps

1. Example: Updating Google IdP for Required Scope on page 64
2. Example : Verifying the Google IdP Configuration on page 67

Example: Updating Google IdP for Required Scope

About This Task

You must update the Google IdP to include the required `email` scope. This feature is not available through the UAA dashboard and you must use the `uaac` commands through your Terminal or Command Prompt to perform this task.

Follow these steps to update the IdP using the `uaac` to include the `email` scope

Procedure

1. Specify your UAA SP instance Url as the intended target.

   ```
   uaac target <uaa_instance_url>
   ```

   For example, `uaac target https://uaa-dashboard-uaa.predix-uaa.run.aws-usw02-dev.ice.predix.io`.
2. Login using the administrative client

   `uaac token client get admin`
3. Use the `uaac curl /identity-providers` command to locate your Google IdP. Copy and save the value for your Google identity provider ID.

   When you enter this command, you will see a list of all configured Identity Providers in this zone.

   For example:

   ```
   uaac curl /identity-providers
    {
     "type": "oidc1.0",
      "config": "{\"emailDomain\":null,\"additionalConfiguration\":null,
   \"providerDescription\":\"Google as Idp\",
        \"externalGroupsWhitelist\":[],\"attributeMappings\":
   {\"user_name\":\"user_name\",\"phone_number\":\"925-343-4444\",
   \"given_name\":\"fname\",\"user.attribute.oidcattrkey\":
        \"oidcattrvalue\",\"family_name\":\"lname\",\"email\":\"mail\"},
   \"addShadowUserOnLogin\":false,\"storeCustomAttributes\":true,
       \"authUrl\":\"https://test-oidc.predix-uaa.run.aws-usw02-
   dev.ice.predix.io/login/oauth/authorize\",\"tokenUrl\":\"https://
   test-oidc.predix-uaa.run.aws-usw02-dev.ice.predix.io/login/oauth/
   token\",
        \"tokenKeyUrl\":\"https://test-oidc.predix-uaa.run.aws-usw02-
   dev.ice.predix.io/login/oauth/token_key\",\"tokenKey\":null,
   \"linkText\":\"Google\",\"showLinkText\":true,
       \"clientAuthInBody\":false,\"skipSslValidation\":true,
   \"relyingPartyId\":\"oidcGClient\",\"scopes\":[\"openid\",\"profile
   \"],\"issuer\":\"https://test-oidc.predix-uaa.run.aws-usw02-
   dev.ice.predix.io/login/oauth/token\",
       \"responseType\":\"code\",\"userInfoUrl\":null,\"discoveryUrl
   \":null}", "id": "9832a29d-f79e-4a48-8814-4ab12319ce59",
   "originKey": "abcd", "name": "abcd", "version": 0, "created":
   1517445351932,
    "last_modified": 1517445351932, "active": true, "identityZoneId":
   "bb0cf71b-148b-445b-bea2-b34501e5526b" },
   ```

```
{
   "type": "oidc1.0",
    "config": "{\"emailDomain\":null,\"additionalConfiguration
\":null,
     \"providerDescription\":\"Google as IdP\",
\"externalGroupsWhitelist\":[],\"attributeMappings\":{\"user_name\":
\"user_name\",
      \"user.attribute.oidcattrkey-2\":\"oidcattrvalue-2\",
\"given_name\":\"fname\",\"family_name\":\"lname\",\"email\":\"mail
\"},\"addShadowUserOnLogin\":false,\"storeCustomAttributes\":true,
     \"authUrl\":\"https://uaa.system.aws-usw02-dev.ice.predix.io/
login/oauth/authorize\",\"tokenUrl\":\"https://uaa.system.aws-usw02-
dev.ice.predix.io/login/oauth/token\",
     \"tokenKeyUrl\":\"https://uaa.system.aws-usw02-
dev.ice.predix.io/login/oauth/token_key\",\"tokenKey\":null,
\"linkText\":\"Google\",\"showLinkText\":true,\"clientAuthInBody
\":false,
     \"skipSslValidation\":true,\"relyingPartyId\":\"oidcGClient\",
\"scopes\":[\"openid\",\"profile\"],\"issuer\":\"https://
uaa.system.aws-usw02-dev.ice.predix.io/login/oauth/token\",
    \"responseType\":\"code\",\"userInfoUrl\":null,\"discoveryUrl
\":null}", "id": "e9821af2-fc07-41e2-8025-1c99977d7919",
"originKey": "oidc-google-idp1516910034792", "name": "oidc-google-
idp1516910034792",
    "version": 2, "created": 1516910093721, "last_modified":
1516910143500, "active": false, "identityZoneId":
"bb0cf71b-148b-445b-bea2-b34501e5526b"

}
```

In the example above, we located the Google IdP named `oidc-google-idp1516910034792` and the ID value for this IdP, `e9821af2-fc07-41e2-8025-1c99977d7919`. We must save this ID value for retrieving information specific to the Google identity provider.

4. Retrieve the information specific to your Google IdP configuration using the `uaac curl /identity-providers/{id}` command and add it in JSON form.

   For example:

```
uaac curl /identity-providers/e9821af2-fc07-41e2-8025-1c99977d7919

uaac curl /identity-providers/e9821af2-fc07-41e2-8025-1c99977d7919 -
X PUT -H "Content-Type:application/json" -d
' {
    "type": "oidc1.0",
    "config": "{\"emailDomain\":null,\"additionalConfiguration
\":null,\"providerDescription\":\"Google as IdP\",
              \"externalGroupsWhitelist\":[],\"attributeMappings\":
{\"user_name\":\"user_name\",\"phone_number\":\"925-343-4444\",
               \"user.attribute.oidcattrkey-2\":\"oidcattrvalue-2\",
\"given_name\":\"fname\",\"family_name\":\"lname\",\"email\":\"mail
\"},
               \"addShadowUserOnLogin\":false,\"storeCustomAttributes
\":true,\"authUrl\":\"https://accounts.google.com/o/oauth2/v2/auth",
                \"tokenUrl\":\"https://www.googleapis.com/oauth2/v4/
token",\"tokenKeyUrl\":\"https://www.googleapis.com/oauth2/v3/certs",
                \"tokenKey\":null,\"linkText\":\"Google\",
\"showLinkText\":true,\"clientAuthInBody\":false,\"skipSslValidation
\":true,\"relyingPartyId\":\"oidcGClient\",
                 \"scopes\":[\"openid\",\"profile\",\"email\"],
\"issuer\":\"https://uaa.system.aws-usw02-dev.ice.predix.io/login/
oauth/token\",\"responseType\":\"code\",
                  \"userInfoUrl\":null,\"discoveryUrl\":null}",
    "id": "e9821af2-fc07-41e2-8025-1c99977d7919",
    "originKey": "oidc-Google-idp1516910034792",
    "name": "oidc-Google-idp1516910034792",
    "version": 2,
    "created": 1516910093721,
    "last_modified": 1516910143500,
    "active": false,
    "identityZoneId": "bb0cf71b-148b-445b-bea2-b34501e5526b"
  }'
```

5. Update your IdP configuration to include `email` in scope.

   a. Open a separate text editor window and enter the following:

   ```
   uaac curl /identity-providers/{id} -X PUT -H "Content-
   Type:application/json" -d '<JSON received from last step>'
   ```

   b. Paste the Google IdP configuration information that you copied from Step 4 inside the `'<JSON received from step 4>'` quotes.

   c. Manually, add the `email` scope in the JSON :

   ```
   \"scopes\":[\"openid\",\"profile\",\"email\"]
   ```

6. Copy the code from the text editor and paste into your Terminal and press the **Enter** key.

   For example:

```
uaac curl /identity-providers/ed729b3e-8fc8-4c09-aa03-c43e8957d0cf -
X PUT -H "Content-Type:application/Json" -d '
{
 "type": "oidc1.0",
  "config": "{\"emailDomain\":null,\"additionalConfiguration\":null,
\"providerDescription\":\"Google as IdP\",\"externalGroupsWhitelist
\":[],
  \"attributeMappings\":{\"user_name\":\"user_name\",
\"user.attribute.oidcattrkey-2\":\"oidcattrvalue-2\",\"given_name\":
\"fname\",\"family_name\":\"lname\",\"email\":\"mail\"},
   \"addShadowUserOnLogin\":true,\"storeCustomAttributes\":true,
\"authUrl\":\"https://accounts.google.com/o/oauth2/v2/auth",
    \"tokenUrl\":\"https://www.googleapis.com/oauth2/v4/token",
\"tokenKeyUrl\":\"https://uaa.system.aws-usw02-dev.ice.predix.io/
login/oauth/token_key\",
    \"tokenKey\":null,\"linkText\":\"Google\",\"showLinkText\":true,
\"clientAuthInBody\":false,\"skipSslValidation\":true,
\"relyingPartyId\":\"149363998617-
r84f3n1bp71hbjafklrfko2e15bgef74.apps.googleusercontent.com\",
\"scopes\":[\"openid\",
    \"profile\",\"email\"],\"issuer\":\"https://uaa.system.aws-usw02-
dev.ice.predix.io/login/oauth/token\",\"responseType\":\"code\",
\"userInfoUrl\":null,\"discoveryUrl\":null}",

      "id": "e9821af2-fc07-41e2-8025-1c99977d7919",
      "originKey": "google",
      "name": "Google",
      "version": 2,
      "created": 1516910093721,
      "last_modified": 1516910143500,
      "active": true,
```

```
        "identityZoneId": "bb0cf71b-148b-445b-bea2-b34501e5526b"
}'
```

A **200** response and shows that the `email` scope is added to the Google identity provider configuration.

Next Steps

Example : Verifying the Google IdP Configuration on page 67

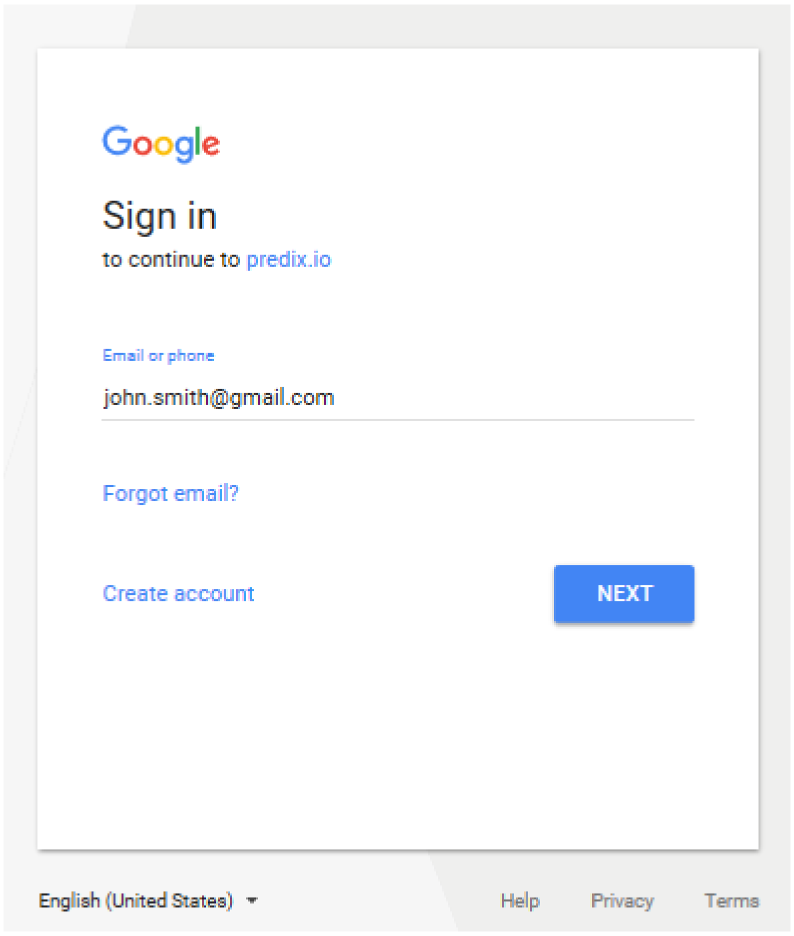
Example : Verifying the Google IdP Configuration

Procedure

1. Open a new browser to make an authorization call using the following format: `http://<UAA_ZONE>.<UAA_URL>/oauth/authorize?response_type=code&client_id=<CLIENT_ID>`

   Example: `https://bfbef8d1-5624-4c2a-857b-4867900c96b6.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/authorize?response_type=code&client_id=OIDC_G`
2. You will be redirected to your Google account page. If you are not logged into your Google account. You will need to enter your google account credentials (email and password).

   Example



Google

Sign in

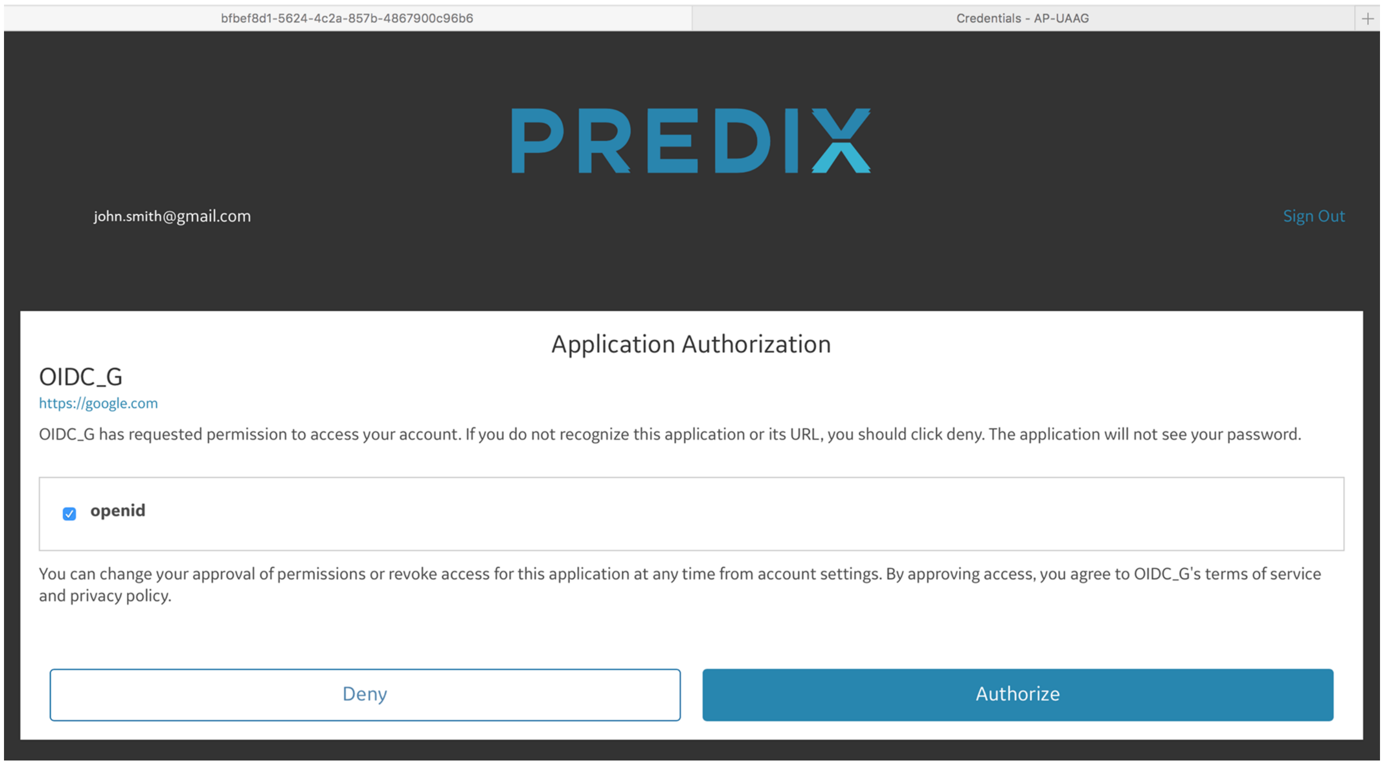
to continue to predix.io

Email or phone

john.smith@gmail.com

3. On successful login, the **Application Authorization** window displays. You can choose to **Authorize** or **Deny** access to your application.

   Example



Application Authorization

OIDC_G

https://google.com

OIDC_G has requested permission to access your account. If you do not recognize this application or its URL, you should click deny. The application will not see your password.

✓ openid

You can change your approval of permissions or revoke access for this application at any time from account settings. By approving access, you agree to OIDC_G's terms of service and privacy policy.

4. Click on **Authorize** to access your application.

Managing Tokens

Using UAA for Token Validation

UAA uses token based authentication, issuing tokens for client applications to use when they act on behalf of Cloud Foundry users. Token based authentication is required to develop secure applications that can only be accessed based on a security token that is generated for the user on authentication.

After generating an Access Token, the client accessing your application presents the bearer token to your application for authentication. Your application validates this token using UAA. A token can be validated in one of the following ways:

- Validate the token remotely using UAA

UAA provides an endpoint (`/check_token`) to validate an access token coming from a resource server. For more details on the endpoint, see UAA Documentation.

- Validate the token locally

  Predix platform provides a Java-based FastTokenService library to validate the token locally. FastTokenService performs the following tasks:

  - Downloads and caches the UAA identity zone public key and uses this key to validate the token signature.
  - Validates that the token is issued by a valid issuer.
  - Validates that the token has not expired.
  - Validates that the token is not issued for a future date.

  For more information on FastTokenService library, see Setting up Fast Token Validation on page 69.

The type of token validation that you use depends on the validation needs of your application. For example, if you need to use token revocation, then you must use remote token validation. However if you only need to validate that the token has not expired, you can validate the token locally and minimize the network request to UAA. While remote validation offers complete validation support, the local validation is much faster and less taxing on the network.

The following table shows the different scenarios where remote and local validation can be used:

Validation Requirement	Remote Validation	Local Validation
Is the token tampered?	Yes	Yes
Is the token expired?	Yes	Yes
Is the token issued for a future date?	Yes	Yes
Was the token revoked?	Yes	No
Is the user still available and active?	Yes	No
Is the token issued by UAA?	Yes	Yes
Is the issuing application still available?	Yes	No

Setting up Fast Token Validation

About This Task

In a typical security scenario, when your application uses UAA, the client accessing your application presents its UAA token issued for authentication. Your application redirects this token to UAA for validation. If your application requires frequent validation of these tokens, it can have an impact on the performance of the application. To mitigate this risk, you can use the Predix `FastTokenService` library to fetch the UAA token signing key at startup and perform the validation. The `FastTokenService` library is available through maven artifactory.

To use the `FastTokenService` library,

Procedure

1. Add the `uaa-token-lib` dependency to your application POM file.

```
<dependency>
    <groupId>com.ge.predix</groupId>
    <artifactId>uaa-token-lib</artifactId>
    <version>3.1.1</version>
    <type>pom</type>
</dependency>
```

2. Update your Spring configuration:

```
<bean id="fasttokenServices"
        class="com.ge.predix.uaa.token.lib.FastTokenServices">
        <property name="storeClaims" value="true" />
        <property name="trustedIssuers">
            <list>
                  <value>${issuerId}</value>
            </list>
        </property>
</bean>
```

   In this example, if `storeClaims` is set to `true`, it includes all claims received from the UAA/check_token endpoint as string request parameters.

3. Update your Spring configuration to include reference to `fastTokenService`.

```
 <oauth:resource-server id="oauth2ServiceFilter"
        token-services-ref="fasttokenServices" />
```

   Note:

   The Spring security configuration refers to `oauth2ServiceFilter` as follows:

```
<http pattern="/" request-matcher="ant"
    xmlns="http://www.springframework.org/schema/security"
    disable-url-rewriting="true" use-expressions="true"
    entry-point-ref="preAuthenticationEntryPoint" create-
session="stateless">

    <intercept-url pattern="/about"
access="isFullyAuthenticated()" />
    <anonymous enabled="false" />
    <custom-filter ref="oauth2ServiceFilter"
position="PREAUTH_FILTER" />
</http>
```

4. Access authenticated `zoneId` in your code.

```
ZoneOAuth2Authentication zoneAuth = (ZoneOAuth2Authentication)
      SecurityContextHolder.getContext().getAuthentication();
      String zoneId = zoneAuth.getZoneId();
```

Configuring UAA to Set the Token Expiration Time

You can configure your UAA client to set the expiration time for tokens to regulate the process of token expiry.

About This Task

Depending on the authorization grant type, the tokens are assigned a default value. If needed, you can update the token expiry within a set range. To set the expiration time, you can use the Configuration tab in the UAA Dashboard.

Procedure

1. In the Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance that you need to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Customization** tab.
6. Update the following values in the **Customization** page:

Field	Value
Access Token Validity	Specify the time for which the access token is valid.
Refresh Token Validity	Specify the time for which the refresh token is valid.

7. Click **Save**.

Viewing Token Details

You can use the UAA Dashboard to view the details of a token assigned to a client.

About This Task

Procedure

1. In the Predix.io Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Client Management** tab.
6. Select the client for which you need to view the token details.

   The client details are displayed in the right pane. For information on how to create a new client, see Creating an OAuth2 Client on page 17.
7. In the clients details pane, select the **Generate Token** option.
8. Specify the **Client Secret** and click **Generate**.

   The token details are displayed.

Multifactor Authentication

UAA Multifactor Authentication

UAA multifactor authentication (MFA) is a mechanism to add an additional factor to user authentication and verification process. The MFA works in conjuction with the primary method for user authentication i.e., using the password. When MFA is implemented, users attempt to login with their password and are redirected to MFA registration or code verification page. Users are signed in only after the second authentication factor is satisfied.

UAA supports Time-based One-Time Password (TOTP) protocol based secondary authenticators like the Google Authenticator.

Note: Currently, Google Authenticator is the only type of authenticator that UAA recognizes. Ability to onboard other secondary authenticators such as, YubiKey, RSA SecureID, Duo Security etc., will be added in future releases.

Creating an MFA Provider

About This Task

Multifactor authentication (MFA) providers are scoped to an IdentityZone in UAA. As an Admin, you can create an MFA provider on a per zone basis. Once you set up MFA for a UAA zone, you can enable or disable the MFA provider for users in that zone.

Procedure

Create a Google Authenticator type of authenticator for your zone using the create an MFA provider API. This is a one time operation.

Note: You cannot update same providers or add different types of authenticators on a per zone basis. UAA will add this capability in future releases.

Next Steps

After you create an MFA provider for a UAA zone, you must enable MFA for users in that zone, see Enabling an MFA Provider for a UAA Zone on page 72 for more information.

Enabling an MFA Provider for a UAA Zone

About This Task

Once you create an MFA provider for a UAA zone, you can enable MFA for users in that zone.

When MFA provider is enabled, all users in a given zone are required need to enter their password and go through the Google Authenticator code verification process to successfully authenticate with UAA.

Procedure

1. Set the `mfaConfig` property in the IndentityZone configuration to enable MFA for that zone.
2. Update the IndentityZone configuration using the update the IdentityZone API. The `providerName` is the unique name of the provider used at the time of creation.

Related tasks

Disabling an MFA Provider for a UAA Zone on page 72

Disabling an MFA Provider for a UAA Zone

About This Task

You can disable an MFA provider for a UAA zone. When you disable the MFA configuration for a user in a specific zone, you do not need to modify or delete the MFA provider created earlier for this zone.

Procedure

Set `mfaConfig` property to false to disable the MFA for a given zone.

Understanding MFA User Flow

If you are a first time application user, you must register your phone application with your UAA account before you start using the Google Authenticator.

Registering First Time Application Users

About This Task

You must enter your username and password to login to UAA. After logging in, UAA detects if MFA is enabled for your zone and redirects you to complete the first time registration process.

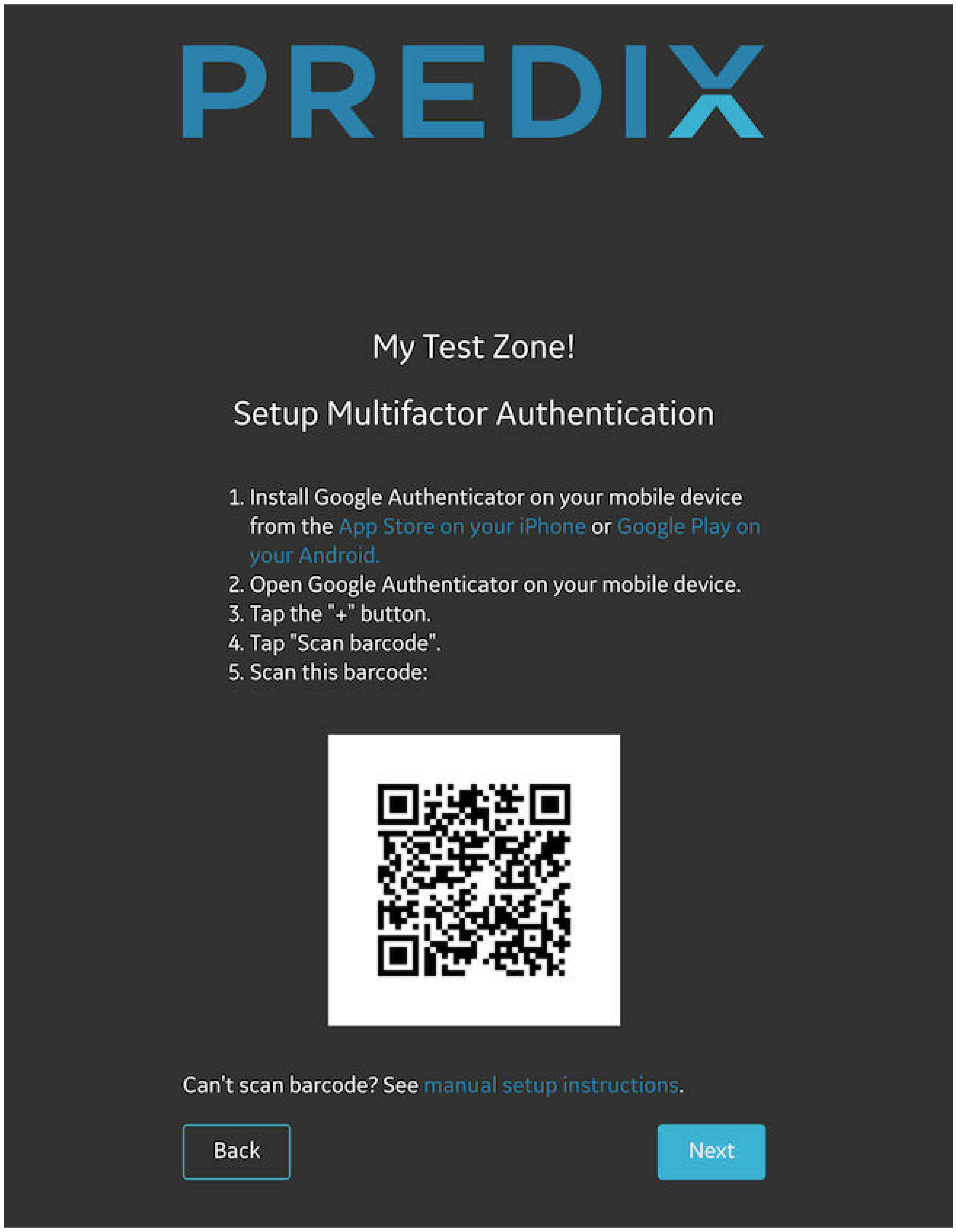
You can start using the Gooogle Authenticator after you register your phone application with your UAA account.

Procedure

Scan the QR code appearing on your screen to follow the registration process.

Note: This is a required step for all first time users. If you have already completed this step once, you can skip this step and proceed to enter your 6-digit verification code.

The following screenshot shows a set of instructions to install Google Authenticator on your mobile device and the QR code that you can scan as part of the registration process.



My Test Zone!

Setup Multifactor Authentication

1. Install Google Authenticator on your mobile device from the App Store on your iPhone or Google Play on your Android.
2. Open Google Authenticator on your mobile device.
3. Tap the "+" button.
4. Tap "Scan barcode".
5. Scan this barcode:

Can't scan barcode? See manual setup instructions.

Next Steps

Entering One-Time Authentication Code on page 74

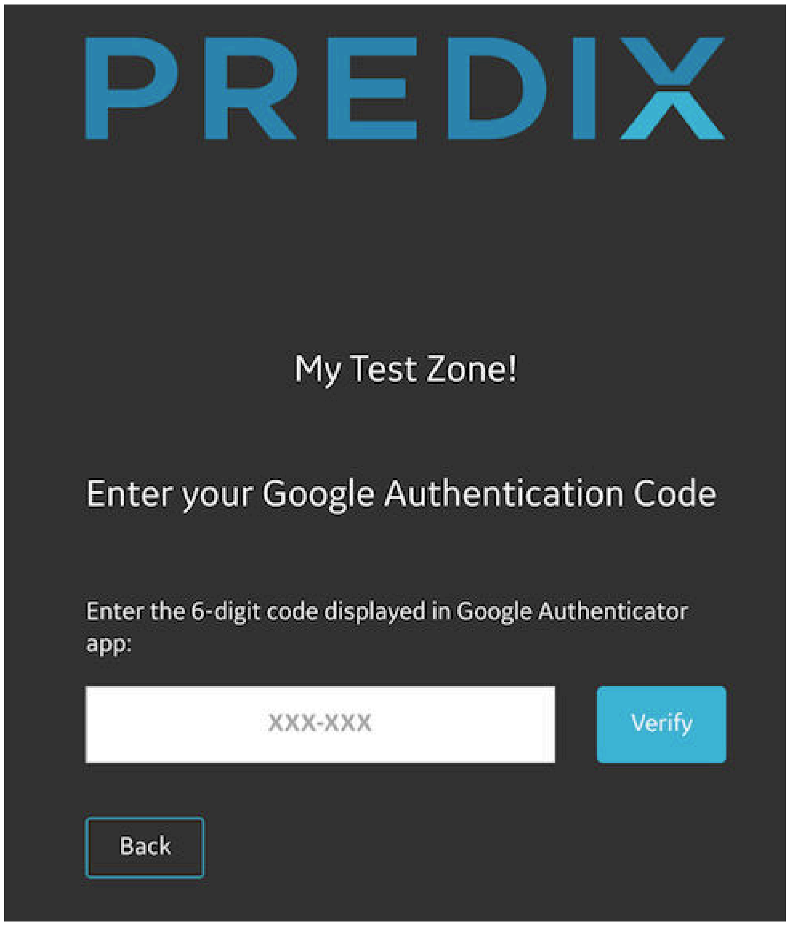
Entering One-Time Authentication Code

After you register your applicationw ith UAA account, you can use the generated 6-digit Google Authentication code to complete the second factor of the authentication process.

Procedure

1. Click **Next** to use the generated code and complete the authentication process.

   The following screenshot shows the field to enter your 6-digit Google Authenticator code.


My Test Zone!

Enter your Google Authentication Code

Enter the 6-digit code displayed in Google Authenticator app:

2. Enter your 6-digit code and click **Verify** to complete the registration process.

Deleting MFA Registration

You can delete the MFA registration for a specific user. The MFA registration for a user is tied with the Google Authenticator application used during MFA configuration. If the user uninstalls the application or switches to a new phone, the UAA account must registered again with a new authenticator application.

1. Admins can delete the MFA credentials for a user using delete MFA registration API.

   Note: Deleting an MFA provider will delete the MFA registration for all the users for that provider.
2. Any users with deleted MFA registration must go through MFA Registration process as described in the Registering First Time Application Users on page 73

.

Using UAA for Application Login or Logout Window

Using UAA to Set Up Application Login or Logout

When you bind your application to UAA and set up your application users in UAA, UAA provides a login screen for the users to access your application.

You can customize the UAA login window as per your application's requirements. Your application users can use the login screen to reset their passwords if needed.

Customizing the UAA Login Window for Your Application

When you bind your UAA instance with your application to provide your application users with a login window, the default login window displays standard UAA window. The default window displays the predix logo, header, and footer information. You can customize this window for each UAA instance to suit your application's requirements.

About This Task

You can customize the UAA Login window to display the information specific to your application. You can add information such as company name, product name, and product logo. You can also update the header and footer information displayed on the login page.

To customize your login page:

Procedure

1. In the Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance that you need to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Customization** tab.
6. Update the following values in the **Customization** page:

Field	Value
Welcome to <GUID>	Replace the GUID value in the welcome message.
Name	Specifies the name of your company, application or product.
Upload Product Logo	Click to choose a Product logo image file in PNG format to upload.
Upload Favicon Logo	Click to upload a shortcut icon or URL icon in PNG format for your application.
Footer	Specify the text to appear as footer in your login screen.
Footer Links	Specify values for URLs that you need to show in your login screen.
Logout Redirect URL	Specify a default URL value to redirect users after logout.
Allow Logout Redirect URL Overwrite	Turn this option off if you need to prevent users from specifying a redirect URI as a logout parameter. The users are then redirected to the default logout redirect URI.
Logout Redirect Parameter Whitelist	Specify a list of redirect URLs that the users can use as logout parameter.

For more information on other UAA configuration options, see https://docs.cloudfoundry.org/api/uaa/#retrieving-an-identity-zone.

Note: The configuration tab in the UAA Dashboard also provides fields for setting duration for token validity and enabling automatic discovery of identity providers. For more information, see Configuring UAA to Set the Token Expiration Time on page 70 or Configuring UAA as Service Provider for External Identity Provider on page 29.

7. Click **Save**.
8. View your login window to validate the changes.

Troubleshooting UAA Service

Troubleshooting UAA Service

The following are general troubleshooting tips and issues you may experience when using Security services.

- Troubleshooting UAA Command Line Interface (UAAC) Installation Issues
- Troubleshooting Oauth2 Issues

Troubleshooting UAA Command Line Interface (UAAC) Installation Issues

There are a number of ways you can install UAAC. The procedure lists recommended installation steps for Windows and Linux users.

Windows

1. Install Ruby from http://rubyinstaller.org/downloads.
   Use the stable version of Ruby (version 2.1.x).

   Note: The 64-bit versions of Ruby are relatively new and not all the packages have been updated to be compatible with it. To use the 64-bit version you will require compiler knowledge, and you will need to be prepared to solve dependency issues.
2. Install the Ruby development kit.
   Follow the installation steps at https://github.com/oneclick/rubyinstaller/wiki/Development-Kit.
3. Download Gem for UAAC, `cf-uaac*.gem`, from https://rubygems.org/gems/cf-uaac.
4. Install the `cf-uaac*.gem` gem.

   ```
   gem install cf-uaac*.gem
   ```
5. Test if the installation was successful.

   ```
   uaac help
   ```

Linux

Installation of UAAC on Linux or Mac is easier if you use the `bundler` gem to install all the required dependencies for `cf-uaac` gem install.

1. Verify that the Ruby version of Ruby is a stable version (version 2.1.x).

   ```
   ruby --version
   ```
2. Download Gem for Bundler, `bundler.gem`, from https://rubygems.org/gems/bundler.
3. Install the `bundler` gem.

   ```
   gem install bundler
   ```
4. Generate a Gemfile with the default rubygems.org source.

   ```
   bundle init
   ```

   The Gemfile is created in the same location as the bundler gem.

5. Edit the new Gemfile to include the gems you want to include in the bundle.

```
# A sample Gemfile
source "https://rubygems.org"

# gem "rails"
gem 'nokogiri'
gem 'rack', '~>1.1'
gem 'rspec', :require => 'spec'
gem 'cf-uaac'
```

6. Install all of the required gems from your specified sources.

```
bundle install
```

7. Install the `cf-uaac` gem.

```
gem install cf-uaac
```

8. Test to see if the installation was successful.

```
uaac help
```

Troubleshooting Oauth2 Issues

Predix platform services use OAuth2 for authentication. Errors related to OAuth2 include the `HTTP 401 Authorization` error. To troubleshoot an OAuth2 issue, check the claims in a user or application token. For example, if you receive an HTTP 401 authorization error, you need to make sure that the token was issued from the correct UAA (`iss` claim) with the required scopes (`scope`).

CAUTION:

Using online tools to decode tokens comes with several risks. If a token is stolen, it can be used by anyone to access protected resources without any other credentials. Even after a token expires, its claims information exposes internal details about Predix applications and systems, which can be used to help in other attacks.

Use the following tools to decode tokens on your local machine:

- **UAAC**

  Use the following command to decode a token:

  ```
  uaac token decode
  ```

  Example:

  ```
  uaac token decode
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovLzEyMzQxMzQz
  MTI
  0MTMubXktZG9tYWluLmNvbS9vYXV0aC90b2tlbiIsInNjb3BlIjoidGVzdCIsInN1YiI6
  Imp
  vZUBleGFtcGxlLmNvbSJ9.KGzJ12fOqBF2sxi9F3oFG3FDWBmNES9TU2j7yc_XyP0

  Note: no key given to validate token signature

  iss: https://1234134312413.my-domain.com/oauth/token
  scope: test
  sub: joe@example.com
  ```

Note:

As indicated in the UAAC output message, you can provide a public key to validate the issuer of the token.

- **Base64**

  The JWT claims are Base64-encoded JSON. If you do not need to validate the JWT issuer, use a tool that can decode the token claims section.

  A JWT token uses the following format:

  ```
  base64encoded_header.base64encoded_claims.signature
  ```

  Use the following command to list the claims:

  ```
  echo | cut -f 2 -d . | base64 --decode
  ```

  Example:

  ```
  ~/dev$
  echo
  'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovLzEyMzQxM
  zQzMTI0MTMubXktZG9tYWluLmNvbS9vYXV0aC90b2tlbiIsInNjb3BlIjoidGVzdCIsIn
  N1Y
  iI6ImpvZUBleGFtcGxlLmNvbSJ9.KGzJ12fOqBF2sxi9F3oFG3FDWBmNES9TU2j7yc_Xy
  P0'
  | cut -f 2 -d . | base64 --decode
  {"iss":"https://1234134312413.my-domain.com/oauth/
  token","scope":"test","sub":"joe@example.com"}

  ~/dev$
  ```

SSLException while targeting a UAA instance using UAAC

When targeting multiple UAA instances using UAAC, the client is able to authenticate some UAA instances successfully, but for others it generates an SSLException.

Cause

One probable cause of this issue is your proxy settings. Additionally the versions of UAAC, Ruby, and OpenSSL that you are using do not support a new GE certificate and you need to be able to accept TLS1.2 connections.

Solution

As a workaround, you can use the `--skip-ssl-validation` flag.

Use the following command to check the SSL certificate:

```
openssl ciphers -v | awk '{print $2}' | sort | uniq
```

Use the latest versions of UAAC, Ruby and OpenSSL.

UAA Release Notes

UAA Release Notes

Q1 2017 Release

New Enhancements

The following new enhancements have been added.

Added Support for Configuring OpenID Connect (OIDC) Identity Provider in UAA Dashboard

You can now configure your UAA instance to support an external OIDC identity provider using the UAA dashboard. For more information, see Configuring UAA as Service Provider for External Identity Provider on page 29.

Added Capability to Setup Shadow Users in UAA using UAA Dashboard

You can now use UAA dashboard to create shadow users. A shadow user is a local user in UAA corresponding to each user defined in the external identity provider that you are configuring with your UAA instance. The local user is created when the user logs-in for the first time. This feature enables a user to be authenticated by UAA without requiring you to create it in UAA also.

You can either choose to create shadow users for all users in your identity provider or create shadow users one at a time.

To create shadow users for all users in your identity provider, use the **Automatically create a shadow user on login** option when you configure UAA for a new identity provider. For more information, see Configuring UAA as Service Provider for External Identity Provider on page 29.

To create selected shadow users, select the Shadow User option when you create a new user. This option is useful if you need to white list users to authenticate only a subset of users setup in your identity provider. For more information, see Creating Users in a UAA Instance on page 24.

Q4 2016 Release

New Features

The following new features have been added.

OpenID Connect Endpoints Published

OpenID Connect flow was updated for endpoint information. For more information, see https://docs.cloudfoundry.org/api/uaa/#openid-connect-flow.

Added Client Credentials Lockout Policy

You can now configure your UAA instance client credentials lockout policy to lock the client credentials for specified period of time after specified number of unsuccessful attempts to login.

For more information, see Creating Password Policies on page 27.

New Enhancements

Enhanced UAA Dashboard User Interface

UAA Dashboard user interface was enhanced managing identity providers and managing password policies.

For more information, see Managing Identity Providers and Creating Password Policies on page 27.

Enhanced MetaData Generated by UAA as IdP

- UAA now generates `NameIDPolicy` in the identity provider (IdP) metadata. The `NameIDPolicy` specifies the SAML name identifier format that is used by the IdP. SAML 2.0 requires the IdP to exchange the name identifier format with the service provider (SP) for SSO service setup.
  For more information, see https://github.com/cloudfoundry/uaa/pull/439.
- UAA now generates the `inResponseTo` attribute in the IdP metadata. The attribute was added to `saml2p:Response` to correspond with the `inResponseTo` attribute for `saml2:SubjectConfirmationData`.
  For more information, see https://github.com/cloudfoundry/uaa/pull/441.

Q3 2016 Release

New Features

The following new features have been added.

UAA Dashboard

Predix.io now provides a graphical user interface to configure and manage your UAA instance. When you select the UAA instances from the Services page on Predix.io, you can click on the Configure your instance button to access the UAA Dashboard. Some of the tasks that you can accomplish using UAA Dashboard are:

- View the clients, users and identity providers related to your UAA instance.
- View client and user details.
- View clients associated with your service instances.
- Create and update clients.
- Create and update users.
- Add identity providers for SAML federation flow where UAA is the service provider.

For more information, see Using the UAA Service.

Support for OpenID Connect Relying Party

UAA now supports federating to an OpenID Connect 1.0 compliant Identity Provider in addition to SAML 2.0 providers.

For more information, see Cloud Foundry UAA Documentation.

New Enhancements

The following new enhancements have been added.

Identity Provider Discovery

UAA now supports Identity Provider discovery when multiple SAML or OpenID Connect Identity Providers are enabled for any given Identity Zone.

For more information, see the description of the Identity Zone API in the Cloud Foundry UAA Documentation.

Opaque Tokens

A new token type of opaque was added in addition to JWT tokens.

For more information, see Cloud Foundry UAA Documentation.

Revocable Tokens

UAA now supports revocable tokens. Both JWT and Opaque tokens are revocable. You can configure a JWT token for revocation per Identity Zone. The configuration is turned off by default.

For more information, see Cloud Foundry UAA Documentation.

JWT Key Rotation

UAA now supports specifying multiple signing and verification keys as part of the Identity Zone configuration.

For more information, see Cloud Foundry UAA Documentation.

Q2 2016 Release

New Features

The following new features have been added.

UAA as SAML Identity Provider

You can now configure UAA as SAML identity provider to integrate with other service providers. For more information, see Configuring UAA as an Identity Provider on page 35.

Ability to configure UAA properties per UAA zone

You can now use the UAA APIs to configure the UAA zone that you created for the following values:

- Enable or disable self service links UI. By default, it is set to False (disable).
- Enable or disable internal authorization. By default, it is set to False (disable).
- Enable or disable internal user store management. By default, it is set to False (disable).
- Set the home page redirect.

Allow use of `/check_token` to perform authorization

In addition to validating the token itself, UAA also allows the requester to include a list of scopes to perform authorization against the token. When the requester sends a scope parameter with one or more scopes, UAA validates that the token is authorized for the specified scopes.

For more information, see UAA Documentation.

Support for User Account Verification

The Create a User API now supports an additional attribute called `verified`. By default `verified` is set to `True`. That means that when you create a user, it is authenticated by UAA and additional verification is not required for the user to obtain a token from UAA. When the property is set to `Null`, it defaults to `True`. If your application requires user verification, you can set this attribute to `False`. When the attribute is set to `False`, your application must authenticate the user and then use the Verify User Links API to update the user verification status in UAA. To verify the user, the application must define its own authentication process.

Note: Any users that were created prior to this release are authenticated by default even if the `verified` attribute was set to `False`.