IBM Security

Intelligence. Integration. Expertise.

IBM SECURITY ACCESS MANAGER
VERSION 9.0.6

OpenID Connect (OIDC)
Includes use of Access Policies, Advanced Configuration, and Mapping Rules

Lab Guide

Jon Harry

Version 1.3
February 2019

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Table of Contents

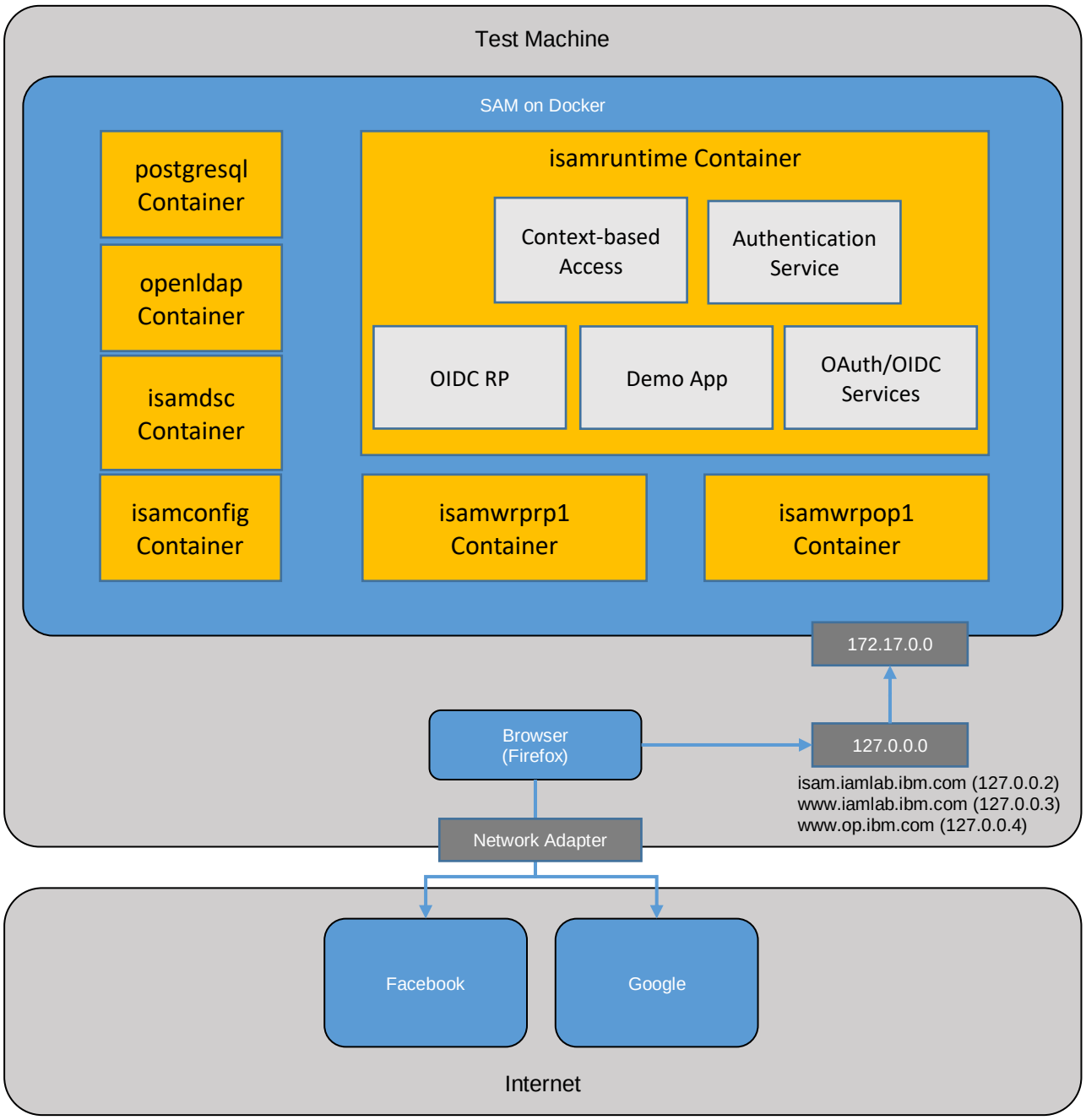
1 Introduction

This lab guide details configuration of an OpenID Connect Provider (OP) and a Relying Party (RP). It shows how a Relying Party can be configured against public OIDC providers, such as Google, and non-standard Identity Providers, such as Facebook, which use proprietary APIs for identity exchange. This guide also covers customization of the OP and RP using Mapping Rules and Advanced Configuration capabilities.

The exercises described in this guide are designed to run on a self-contained test machine which has the required software and helper scripts installed. A configuration archive is needed to set up the initial Access Manager environment.

1.1 High Level Architecture and Networking

The high-level architecture for the environment described in this document may be summarized as follows:



1.1.1 Test Machine

The test machine is a physical or virtual machine which has the following components installed:

- **Docker** – This provides the container services used to explore native Docker installation of Access Manager. It includes a command line tool (docker) for management.
- **Docker Compose** – This tool provides automation for native Docker which can be used to create and manage multi-container environments more efficiently that using native Docker commands. It includes a command line tool (docker-compose).
- **Browser** – A browser is required for accessing the Access Manager admin console and Reverse Proxy. This cookbook was written using Firefox but any up-to-date browser should work.

The Test Machine requires internet connectivity for connection to Google and Facebook. An internet connection is also required for environment setup.

The Test Machine needs to have at least 3 local IP addresses available for the Access Manager components to bind to. The provided scripts assume use of loopback addresses (127.0.0.2, 127.0.0.3, and 127.0.0.4). If external connectivity to the Access Manager components is required, you will need to use externally addressable IP addresses instead.

1.1.2 Google Account

To set up Google as an OIDC Provider, you will need a Google account.

1.1.3 Facebook Account

To set up Facebook as a Social Sign-On provider, you will need a Facebook account. This account will need to have a *verified* phone number associated with it (or you will have to verify a phone number during the lab). If you prefer not to share a phone number with Facebook you will not be able to complete this part of the lab.

1.1.4 TOTP Client

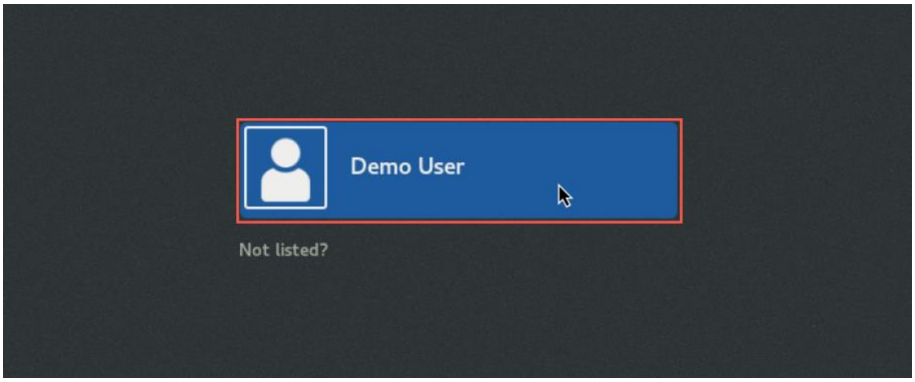
The Advanced Configuration and Access Policy exercise uses TOTP for 2 Factor Authentication. You will need a TOTP client. If you have the IBM Verify app on iOS or Android, this can be used. If you are using another OS, you can also use the Google Authenticator app.

2 Getting Started

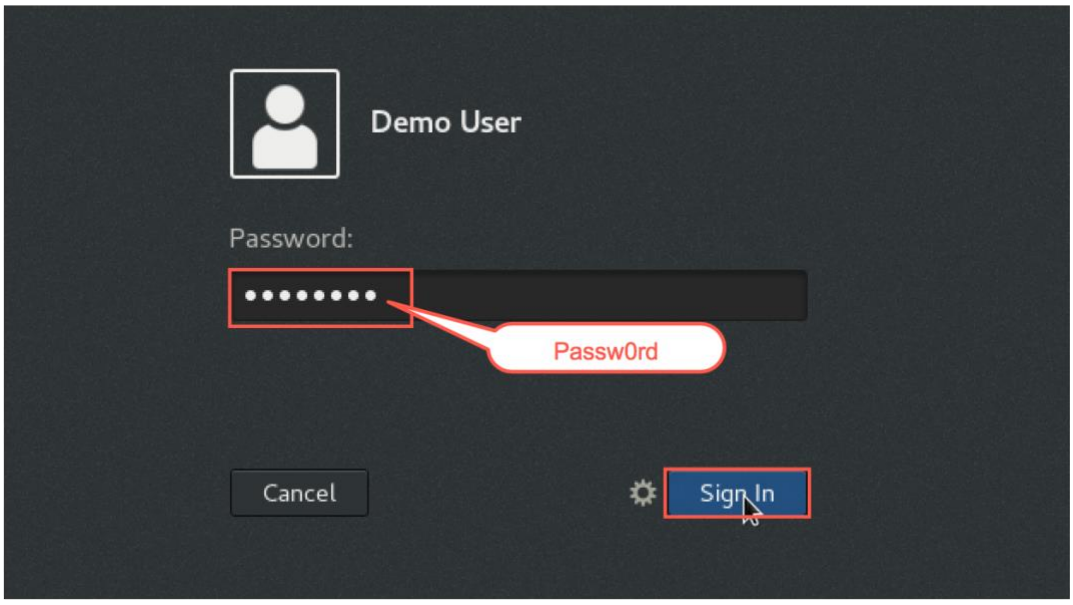
2.1 Start Centos 7 Virtual Machine and sign in.

This lab uses a Centos 7 Virtual Machine which has the required software and some helper scripts pre-installed. Start this Virtual Machine now.

Once the Virtual Machine is booted, you will be presented with a login page:



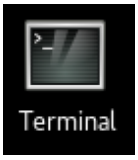
Click on **Demo User**.



Enter **Passw0rd** as the password and click **Sign In**.
The Desktop is displayed:

2.2 Restore the Access Manager Docker environment

Open a terminal window using this icon on the desktop:



To initialize the Access Manager Docker environment for the lab, enter the following command:

```
[demouser@centos ~]$ studentfiles/prepare-lab.sh config-archives/sam906-oidclab-start.tar
Unpacking archive...
WARNING: This command will delete all docker assets
         (containers, volumes, and networks) described in:
         compose project /home/demouser/studentfiles/iamlab.
Press ENTER to continue (or ctrl-c to abort).
No iamlab containers to clean up.
Restoring keys to /home/demouser/dockerkeys
Creating key share at /home/demouser/dockershare/composekeys
Done.
Restoring compose project to /home/demouser/studentfiles/iamlab
Updating IPs in .env...
Starting configuration container...
…
Done.
Check /home/demouser/studentfiles/iamlab/.env for environment information.
Run docker-compose logs -f in /home/demouser/studentfiles/iamlab to monitor environment.
```

Wait for a minute to give the environment time to start.

2.3 Connect to Configuration Container LMI

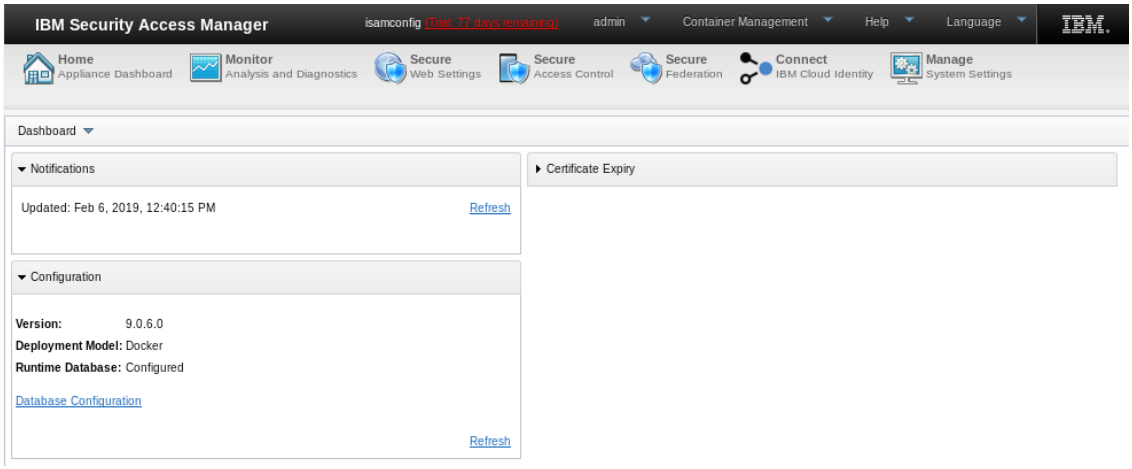
Open the Firefox browser using the following icon on the Centos VM Desktop:

Navigate to the following URL: **https://isam.iamlab.ibm.com**.

If you see a "Connection not secure" warning, click **Advanced→Add Exception…->Confirm Security Exception** to add an exception.

Enter **admin** as the username and **Passw0rd** as the password. Then click **Login**.

The LMI Dashboard is displayed:



<u>You are now ready to start the labs.</u>

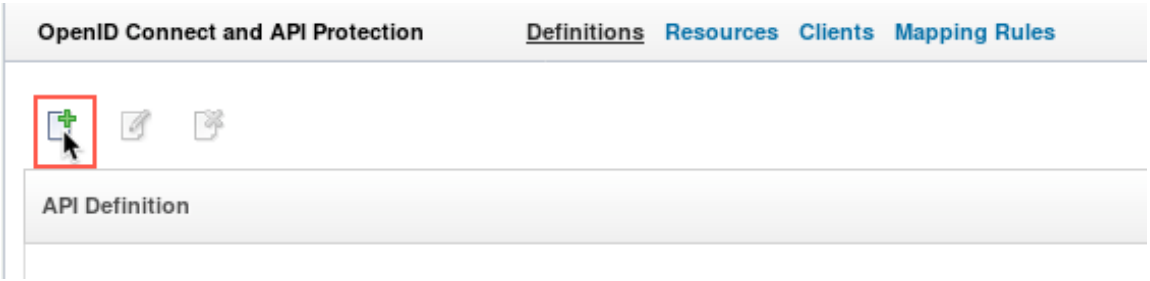
3 Configure OpenID Connect Provider (OP)

In this section we will set up an OIDC Provider in our Access Manager system, configure the *op1* Reverse Proxy instance as the point of contact, and create a Relying Party definition.

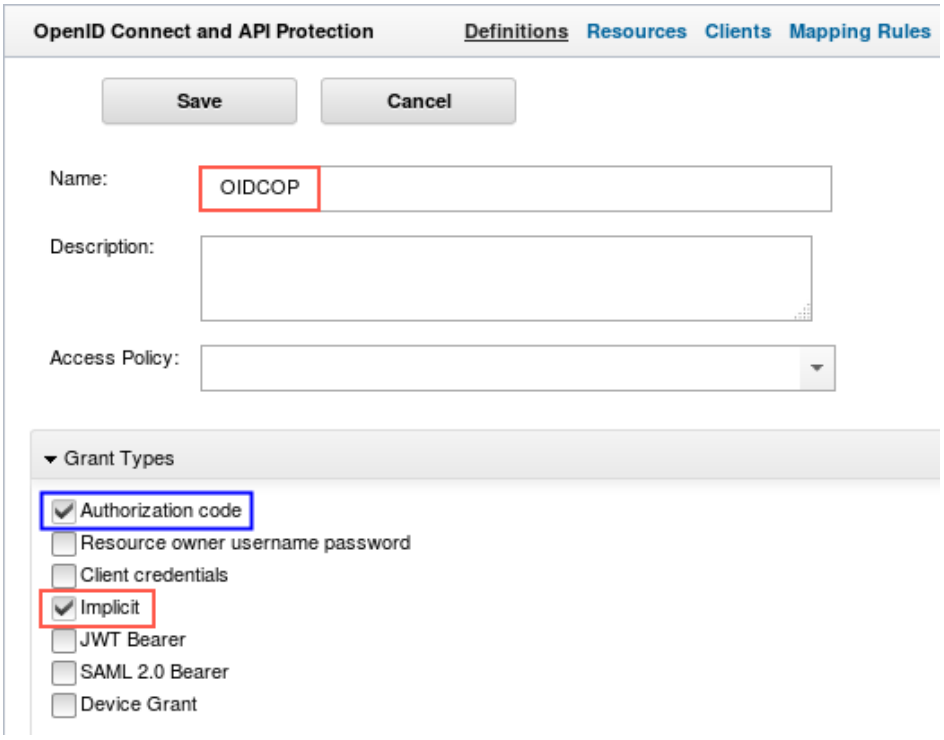
3.1 Create OP Definition

Open ID Connect is built on top of the OAuth 2.0 protocol and, in Access Manager 9.0.4.0 and above, the OAuth 2.0 and OIDC provider services have been consolidated. The OAuth 2.0 and OIDC Provider functions are available in both the AAC and Federation add-on modules.

Navigate to **Secure Federation→Manage: OpenID Connect and API Protection** in the mega-menu.



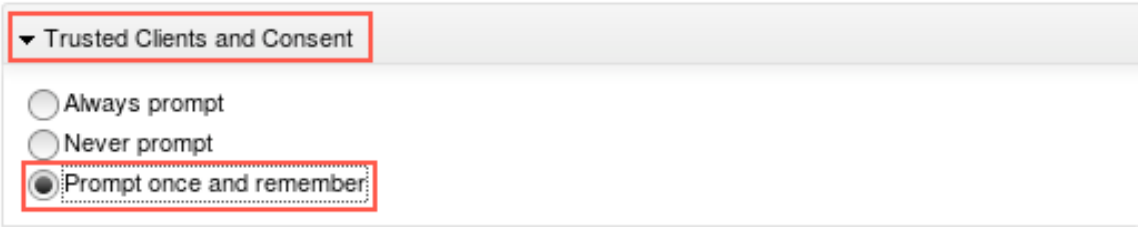
Click **Create** button to add a new definition.



Enter **OIDCOP** as the *Name* of the definition.

This name is used as part of the Metadata URL so good to keep it short and simple.

We want this provider to support the *Implicit* flow (where all communication is via the browser) in addition to the *Authorization Code* flow so check the **Implicit** checkbox.



Expand the **Trusted Clients and Consent** section and select the **Prompt once and remember** radio-button. This will cause the OP to ask for user consent before providing identity data to the Relying Party.

Check the **OpenID Connect** checkbox to enable this definition for OpenID Connect in addition to OAuth 2.0.

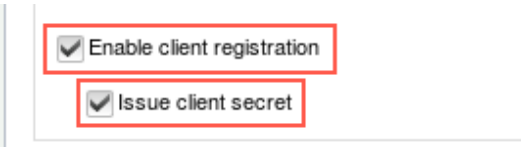
Enter **https://www.op.ibm.com** as the *Issuer Identifier*. This value can be any unique URL. Setting it to the URL of the Point of Contact is sensible.

Enter **https://www.op.ibm.com/mga** as the *Point of Contact prefix*. This prefix is used to generate all the URLs advertised by this provider. It needs to be set to the URL that clients will use to access the provider and must include the junction name that connects to the Runtime (if applicable). We'll have to specify this same junction name when setting up the Point of Contact.

When you leave the Point of Contact text entry box, the *Metadata URI* is automatically completed. Make a note of this, you will need it when configuring the Relying Party.

> You can't easily copy this field. For convenience it is also available in file: *~/studentfiles/oidc/metadata-url*.

Select **server** from the *Certificate Label for Signing* drop-down box.



Check the box to **Enable client registration** and the box to **Issue client secret**. This enables OpenID Connect dynamic client registration (which is a new capability in SAM 9.0.5.0).

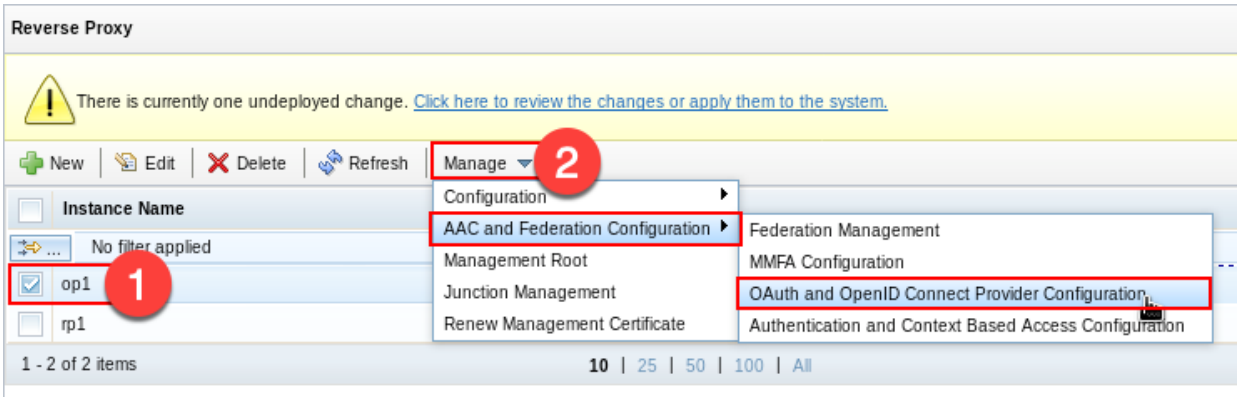
Scroll to the top of the page and click **Save**. The OIDC definition is saved.

3.2 Configure Reverse Proxy as Point of Contact

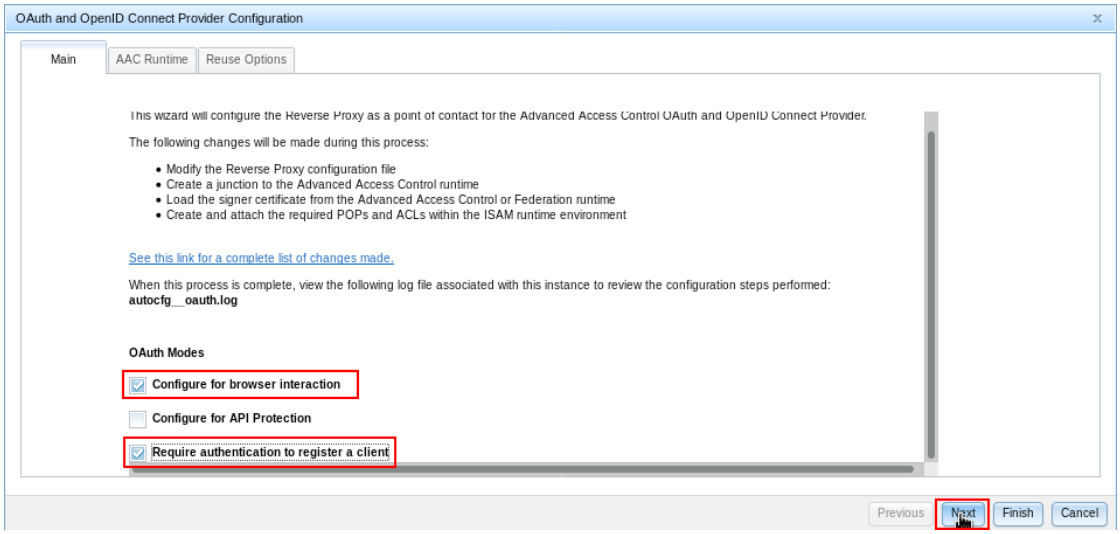
In an Access Manager environment, clients access the OIDC services (which run in the Runtime) via a Reverse Proxy. This Reverse Proxy needs to be configured with a junction to the runtime and have the appropriate Access Controls set up for the endpoints (e.g. metadata URL needs unauthenticated access).

A wizard is available to perform the required configuration. We will now use it.

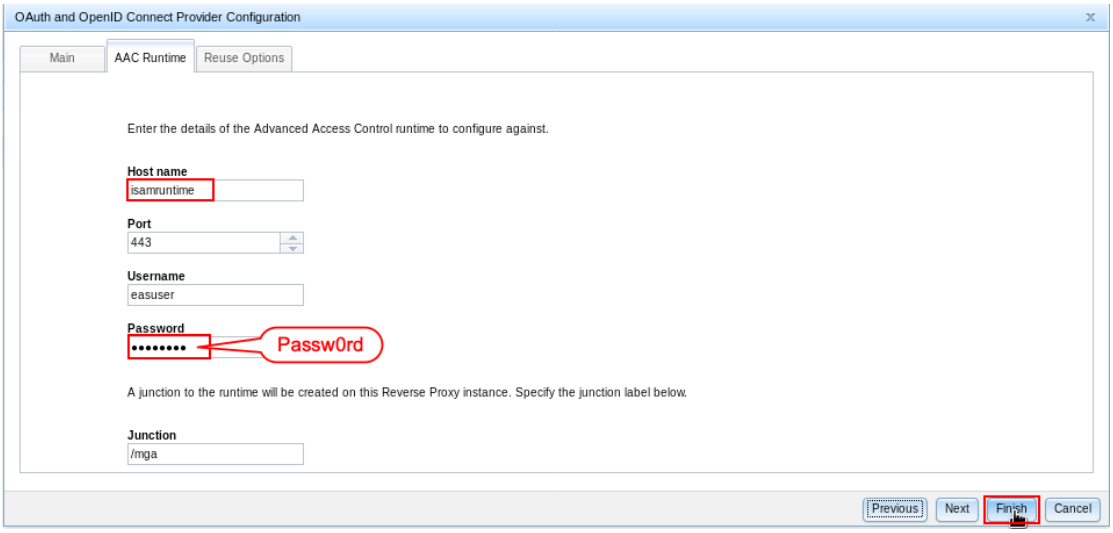
Navigate to **Secure Web Settings→Manage: Reverse Proxy** in the mega-menu.



Select the check-box for the **op1** instance. Click **Manage** and then select **AAC and Federation Configuration→OAuth and OpenID Connect Provider Configuration** from the pop-up menu.



On the *Main* tab of the wizard, select checkboxes for **Configuration for browser interaction** and **Require authentication to register a client**. Then click **Next**.



Enter **isamruntime** as the *Hostname*. This is the hostname that the Reverse Proxy should use to reach the AAC Runtime. Port 443 is already correct.

Enter **Passw0rd** as the *Password* of the *easuser* User. The password for *easuser* has already been set to this value (under *Secure Federation→User Registry)*.

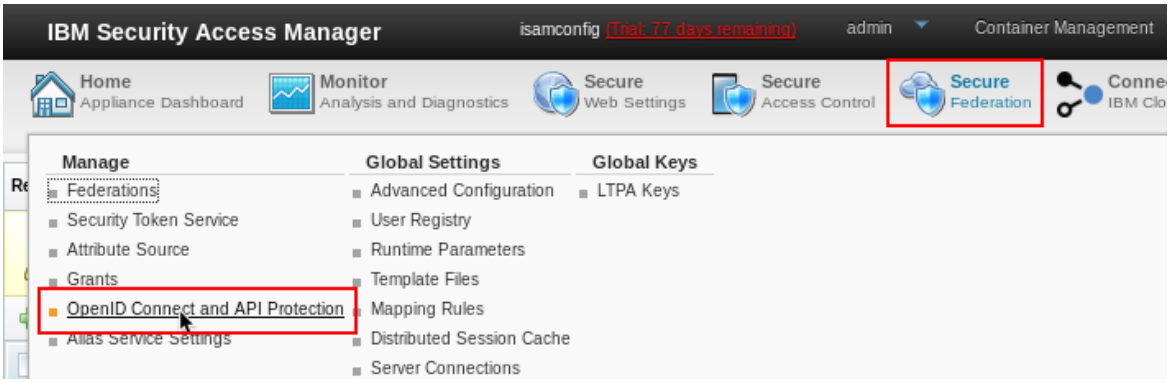
Note the *Junction* name of **/mga**. This needs to match the junction that we specified in the Point of Contact prefix when creating the provider definition.

Click **Finish**.

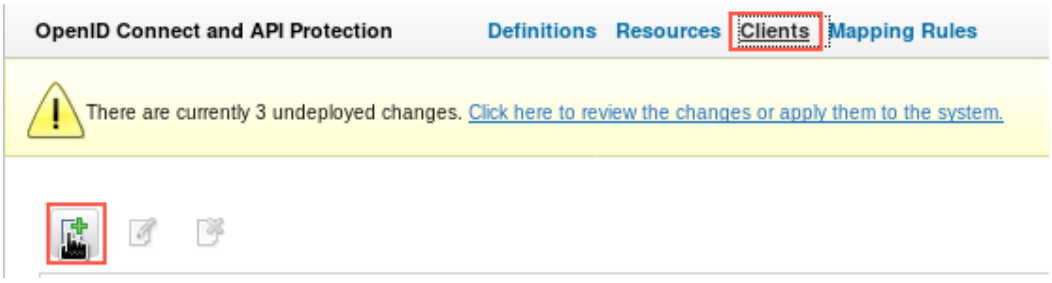
At this point the required changes are made to the Reverse Proxy and the Security Policy. It may take a few seconds to complete.

3.3 Register a Client

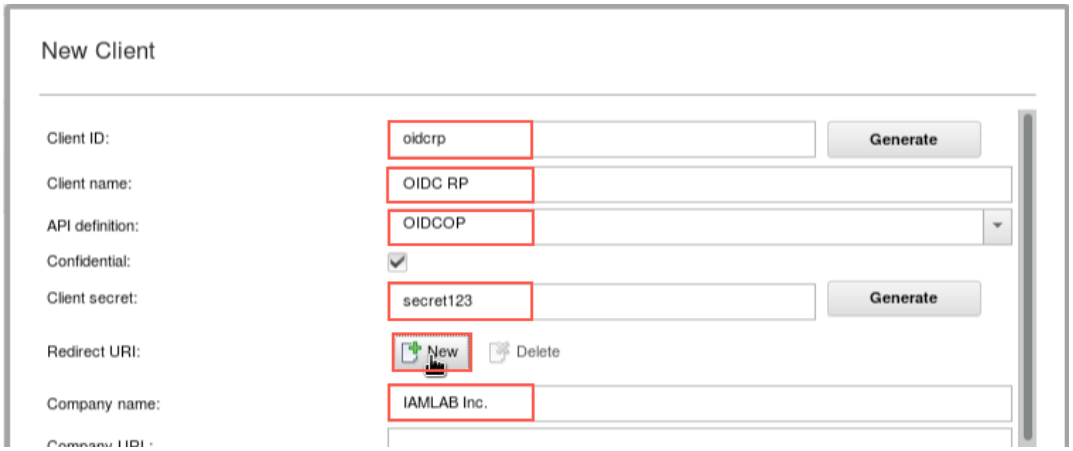
In order for a client to use our OIDC Provider, it must be registered in much the same way as an OAuth 2.0 client must be registered. We will now pre-register a client (which we will create in the next section).



Navigate to **Secure Federation→Manage: OpenID Connect and API Protection** in the mega-menu.



Select the **Clients** tab and then click the **Create** button to add a new client.



Enter **oidcrp** as the *Client ID*. This ID is required when configuring the Relying Party.

Enter a name for the client (e.g. **OIDC RP**). This name will be seen by users if you have consent enabled.

There is only one *API Definition* in our system and it is already selected.

Enter **secret123** as the *Client secret*. Usually you would use something more secure but this one is easy to remember for the lab.

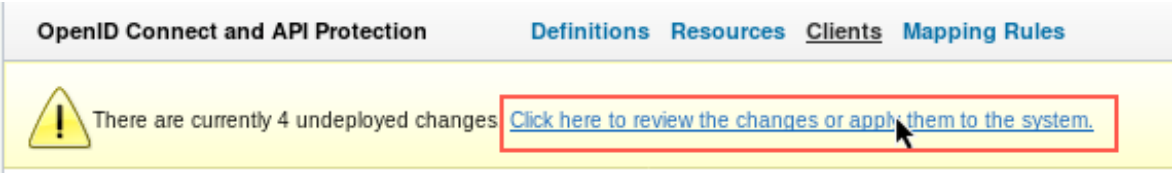
A Company Name is required. Enter something sensible.

For OIDC, we need to define the Redirect URIs that are valid for this client. Click **New** button to add one.

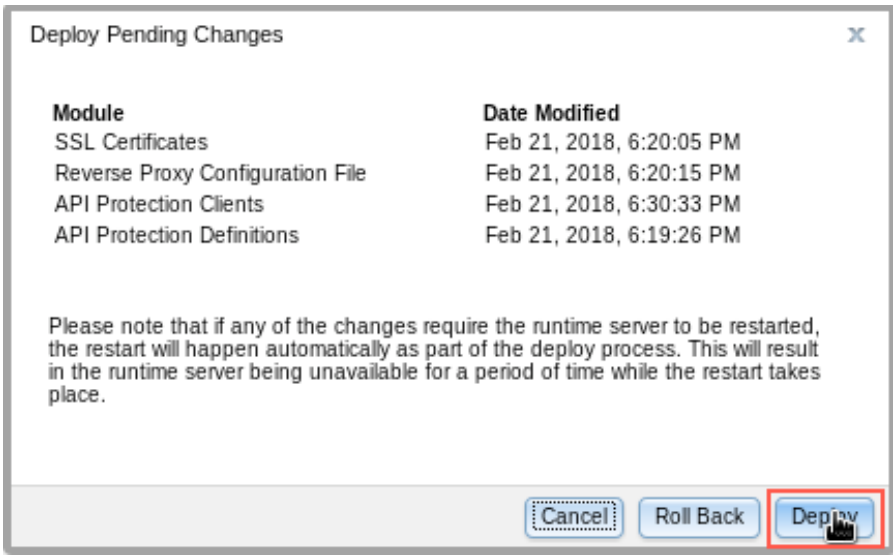
Enter **https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/redirect/ISAMOP** in the Redirect URI box.

> At this point, you would usually have to ask the RP for their redirect URI. The value used here is what the redirect URI of our RP will be when it is created in the next section. We enter it now to save having to come back and change this later.

Click **OK** to save the new client.



Click the link in the yellow warning message to deploy the changes in the configuration container.



Click **Deploy** to confirm.

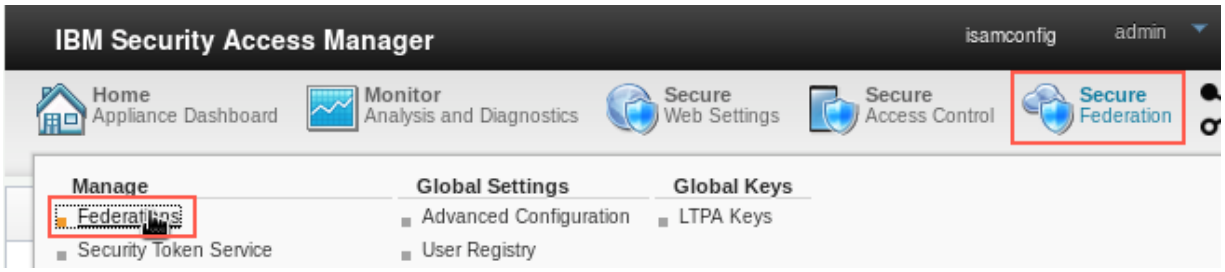
> In a Docker environment, deploying changes does not make them active in the environment. You need to publish a new configuration snapshot and restart/reload the affected components. We will do this later.

4 Configure OpenID Relying Party

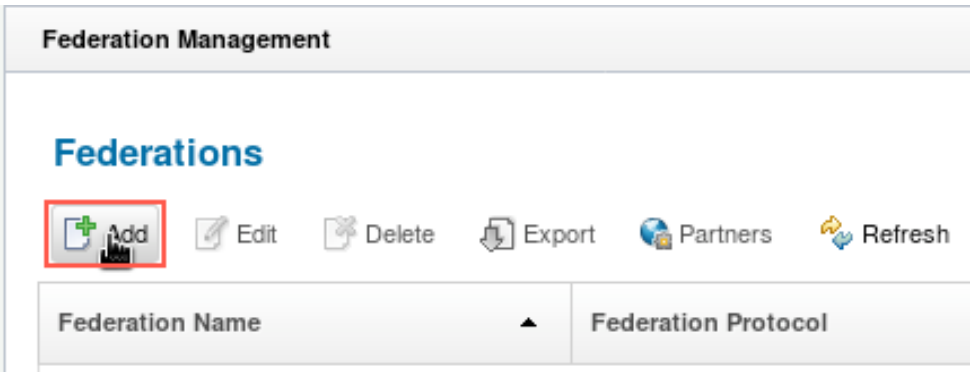
In this section we will set up an OIDC Relying Party in our Access Manager system, configure the *rp1* Reverse Proxy instance as the point of contact, and link it to the OIDC Provider we created earlier.

4.1 Create an OIDC Relying Party Federation

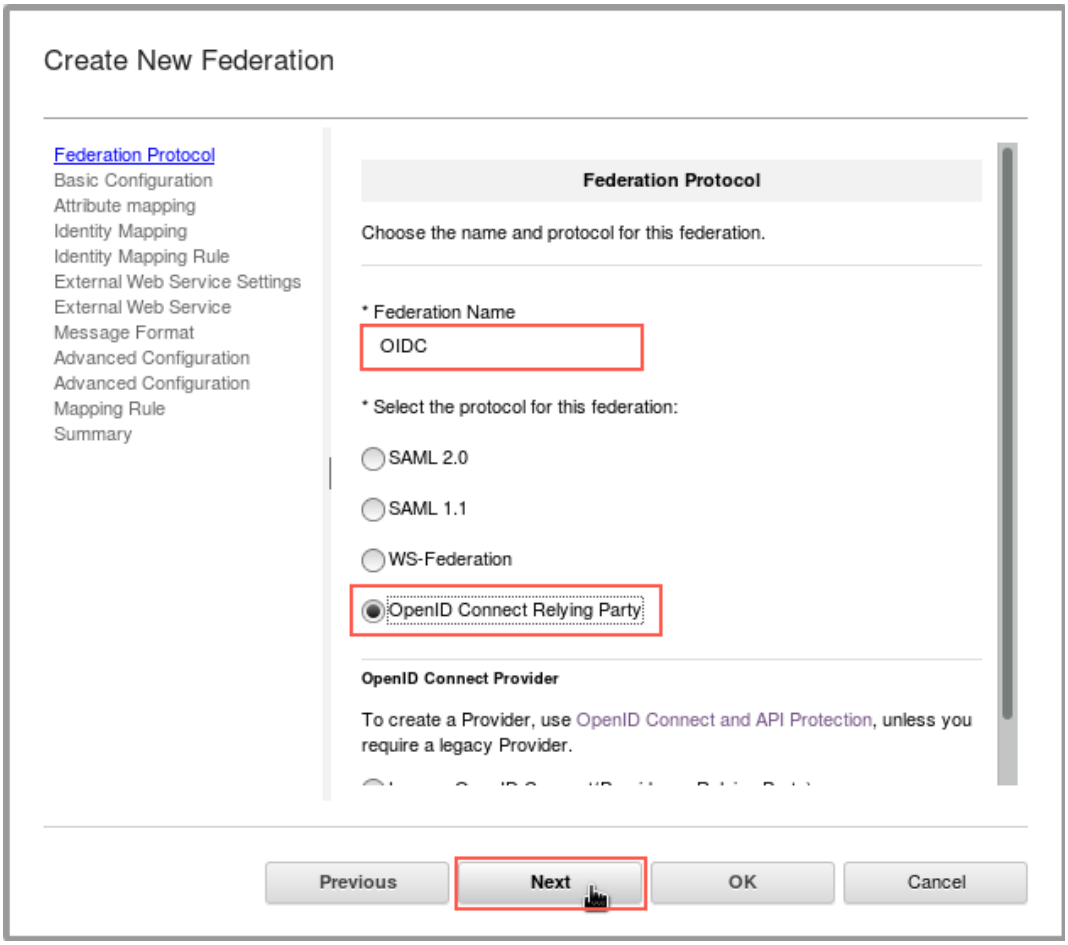
An OIDC Relying Party is configured under Federations. It is NOT available in the Advanced Access Control add-on.



Navigate to **Secure Federation→Manage: Federations** in the mega-menu.

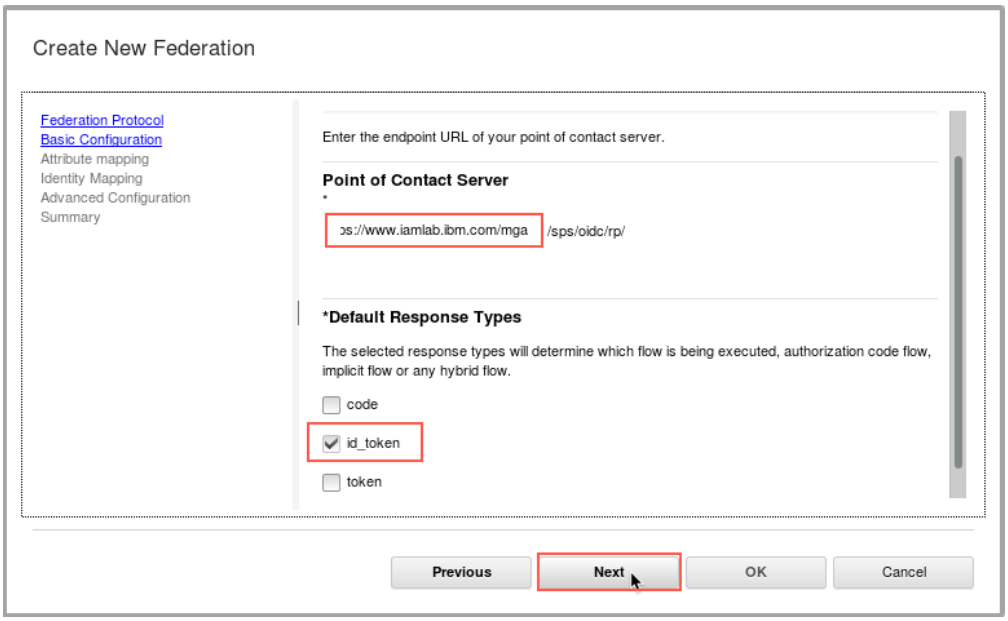


Click **Add**.



Enter **OIDC** as the federation name and select **OpenID Connect Relying Party** as the protocol. Then click **Next**.

The Name given here will appear as part of trigger and redirect URLs so you need to use this exact value.



Enter **https://www.iamlab.ibm.com/mga** as the Point of Contact Server. This URL must be the one that clients will use to connect to the Relying Party Reverse Proxy and must include the junction name that will be

used to connect to the SAM Runtime. The junction name given here is used to create a junction (if it doesn't already exist) when the Point of Contact configuration is performed.

In this environment we will use the Implicit flow. In the implicit flow, the ID Token is returned directly from the OIDC Provider via the browser. There is no requirement for direct communication from the RP to the OP.

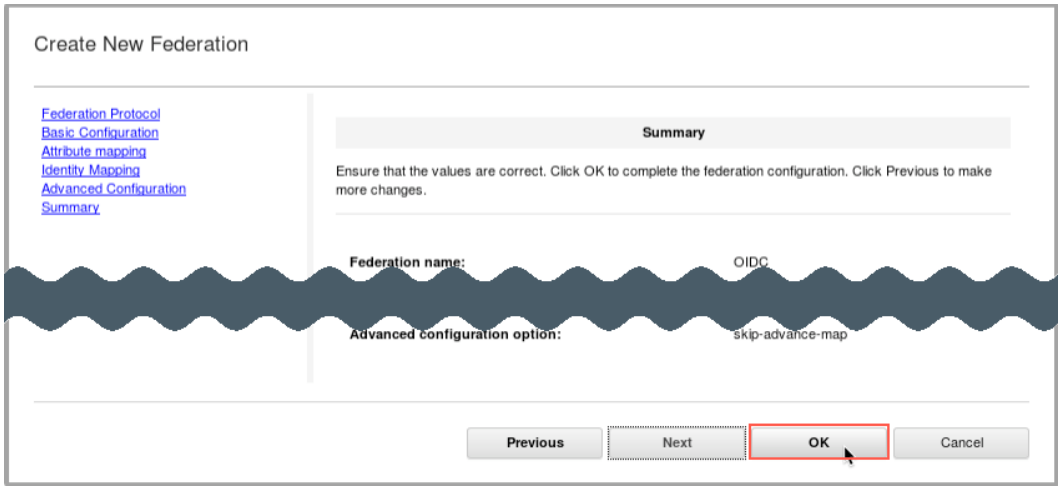
Select the checkbox for **id_token** and then click **Next**.

> To also return Access Token in Implicit mode, select **Token**.
> To use the *Authorization Code* flow, response type **Code** should be selected on its own.

We won't set up Attribute Mapping at the federation level. Click **Next** again.

We won't set up Identity Mapping at the federation level. Click **Next** again.

We won't set up Advanced Configuration at the federation level. Click **Next** again**.**



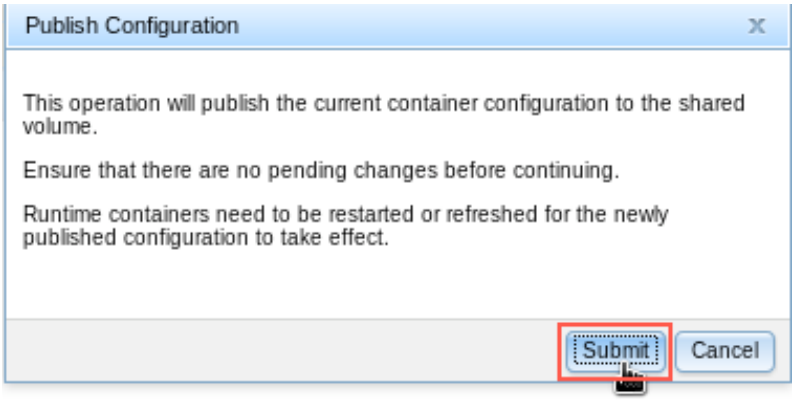
Click **OK** on the summary page to create the federation.

Deploy the outstanding changes using the link in the yellow warning message.

4.2 Extra steps for Docker environment

The next step (running the federation configuration wizard) requires that the federation is active in the SAM Runtime. In a Docker environment that means we need to publish the configuration and wait for the Runtime container to detect the new configuration and reload.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.



Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime container will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue this command to restart the runtime container:
> **`docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime`**

You can monitor the restart of the Runtime container by using the following commands:

```
[demouser@centos ~]$ cd studentfiles/iamlab
[demouser@centos iamlab]$ docker-compose logs -f isamruntime
```

The *-f* flag creates a tail operation. Press **ctrl-c** to terminate (although it can be quite useful to leave this running in the background).

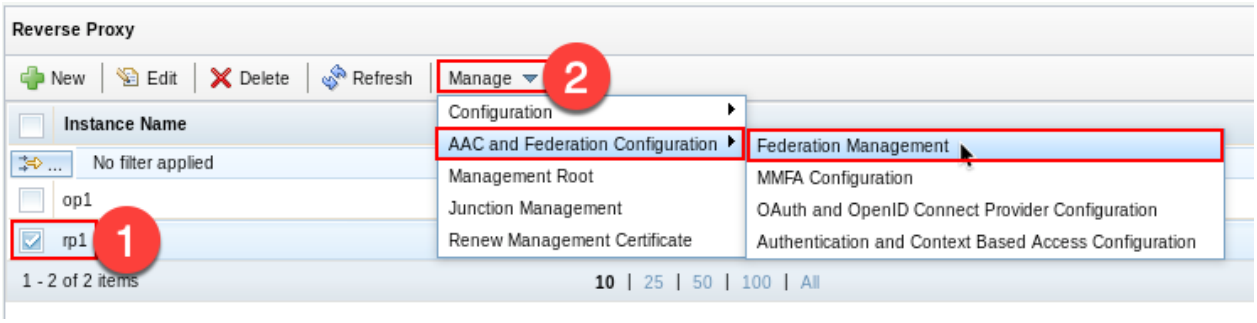
> If you leave off the *isamruntime* parameter, the logs from all containers will be shown.

4.3 Configure Reverse Proxy as Point of Contact

In an Access Manager environment, the OIDC Relying Party logic runs in the Runtime and is accessed via a Reverse Proxy. This Reverse Proxy needs to be configured with a junction to the runtime, have the appropriate Access Controls set up for the endpoints (e.g. unauthenticated access), and needs to be configured to accept the verified identity to create an authenticated session.

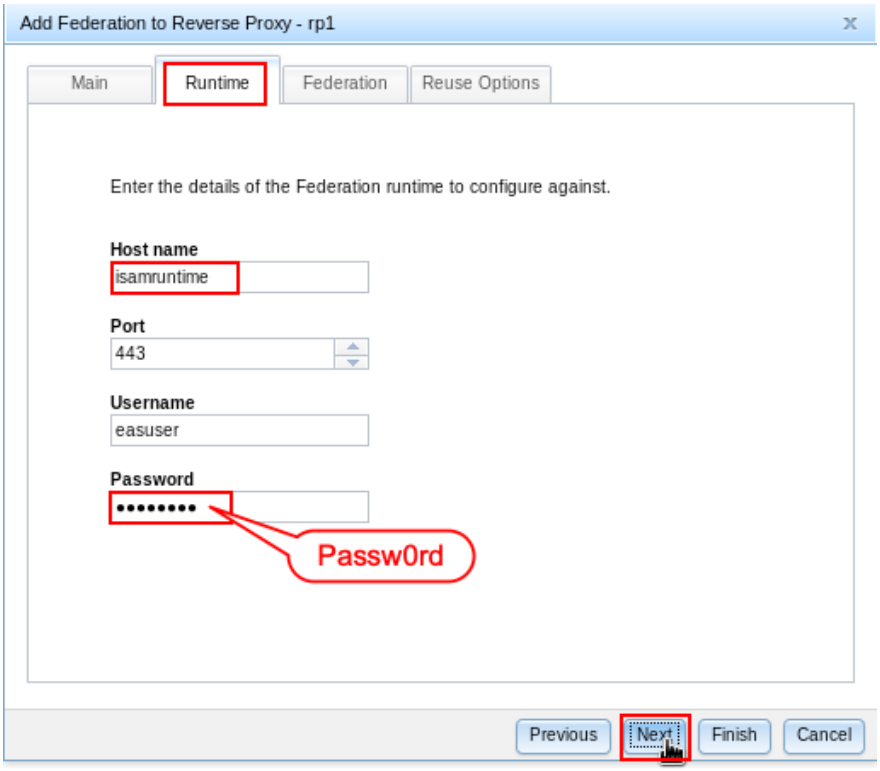
A wizard is available to perform the required configuration. We will now use it now.

Navigate to **Secure Web Settings→Manage: Reverse Proxy** in the mega-menu.



Select the check-box for the **rp1** instance. Click **Manage** and then select **AAC and Federation Configuration→Federation Management** from the pop-up menu.

Click **Add**.



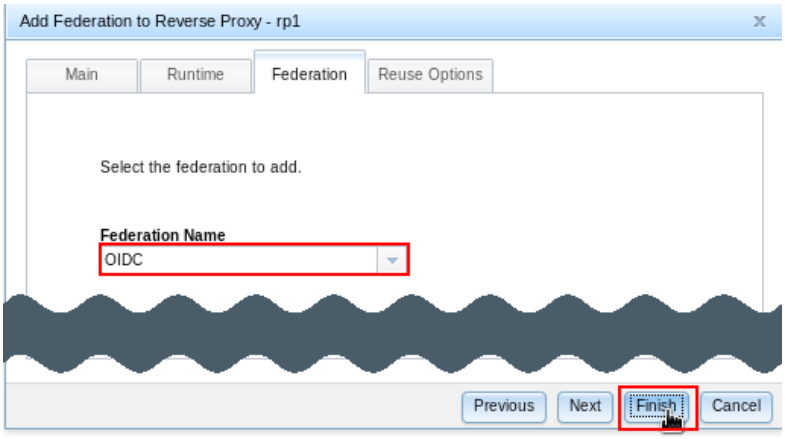
Select the **Runtime** tab in the wizard.

Enter **isamruntime** as the *Host name*. This hostname and port will be used by the Config Container to contact the SAM Runtime so we're using the internal Docker hostname and port here.

Enter **Passw0rd** as the *Password*. The password for the *easuser* user has already been set to this value (under Secure Federation→User Registry).

Click **Next** to move to the *Federation* tab.

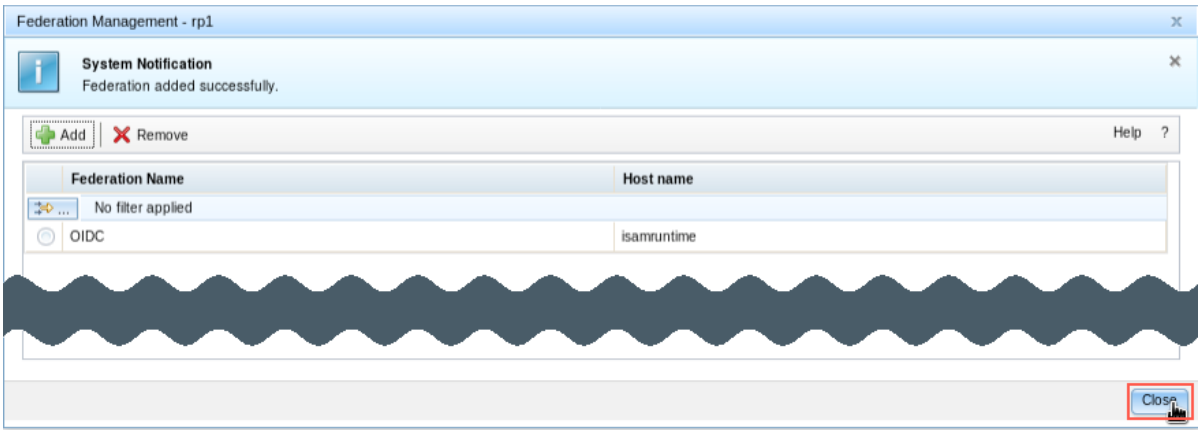
> If you see an error at this point it is likely because the federation runtime could not be accessed. Check the connection details on the *Runtime* tab and try again. Also, make sure you published the configuration (and restarted AAC runtime) after you configured the federation.



In the Federation tab, select **OIDC** from the *Federation Name* drop-down list.

Click **Finish**.

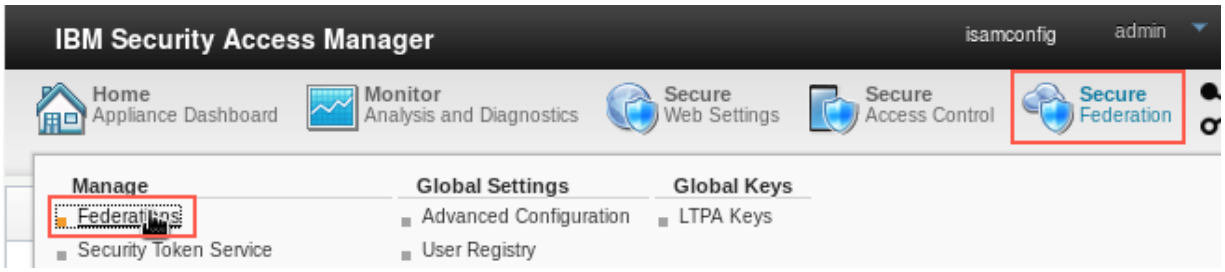
At this point the required changes are made to the Reverse Proxy and the Security Policy. It may take a few seconds to complete.



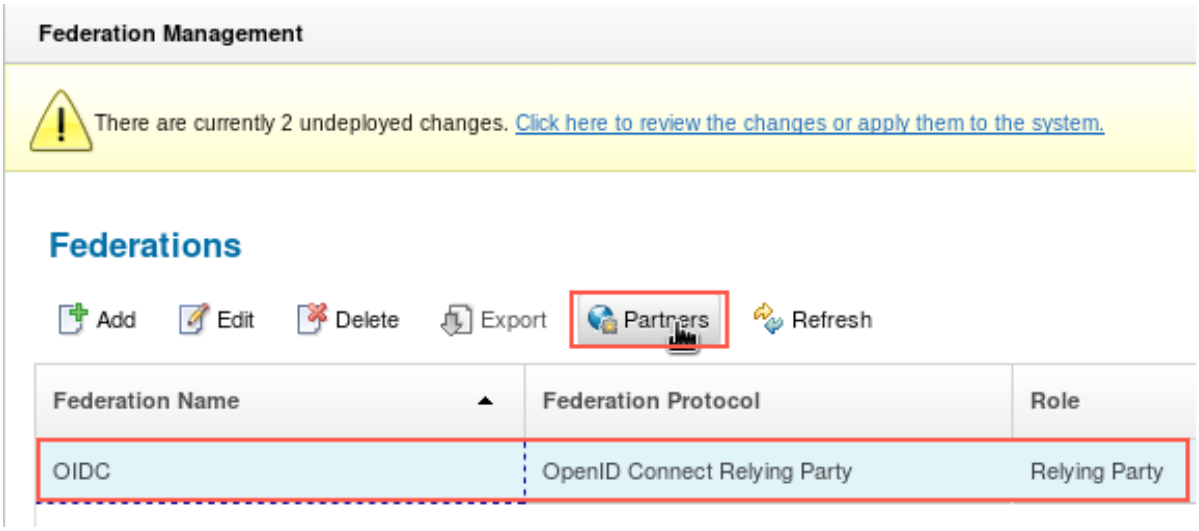
Click **Close** to close the Federation Management overlay.

4.4 Add OIDC OP as a Federation Partner

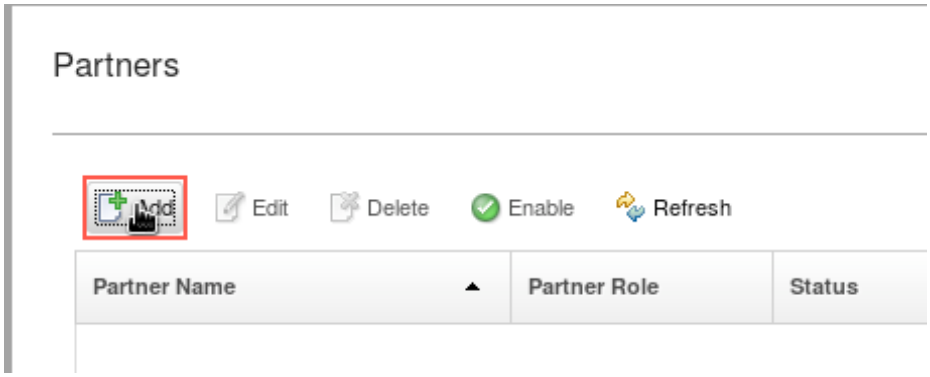
The OIDC OP must now be added to the RP Federation as a partner.



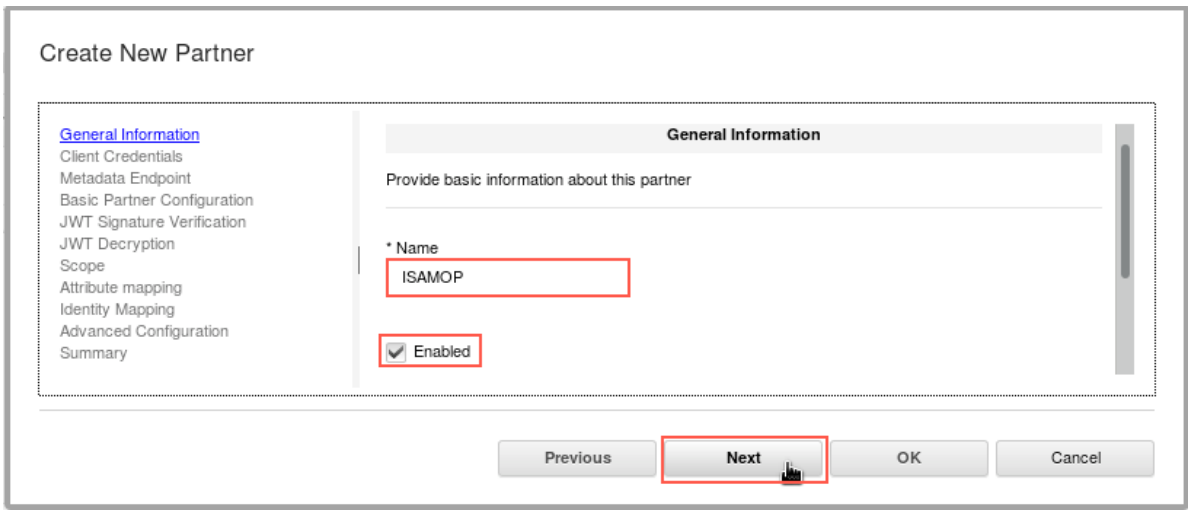
Navigate to **Secure Federation→Manage: Federations** in the mega-menu.



Select the **OIDC** federation and click **Partners**.

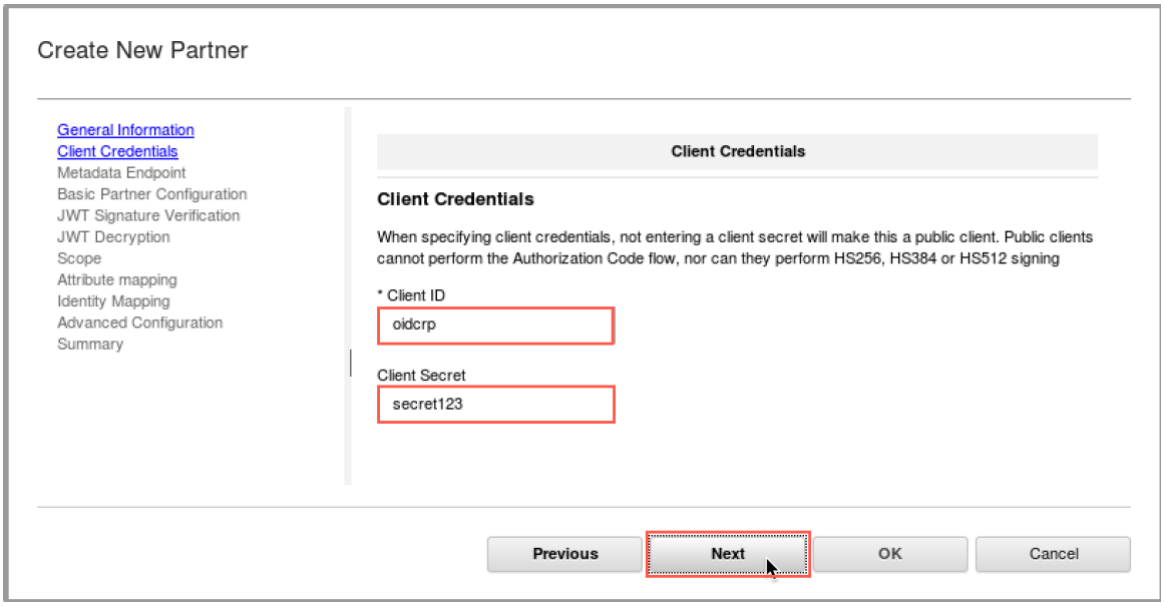


Click **Add**.



Enter **ISAMOP** as the *Name* and select the **Enabled** flag. Then click **Next**.

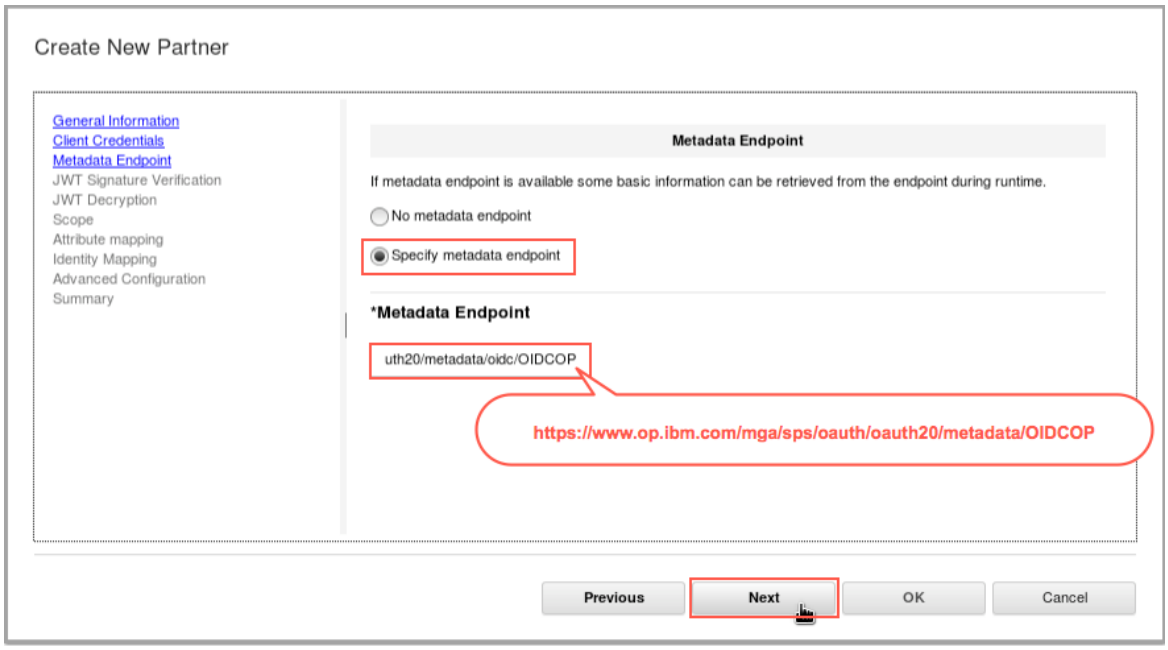
The Name given here will appear as part of trigger and redirect URLs so you need to use this exact value.



Enter **oidcrp** as the *Client ID* and **secret123** as the *Client Secret*.

These values must match the Client ID and Client Secret registered at the OIDC Provider.

Click **Next**.

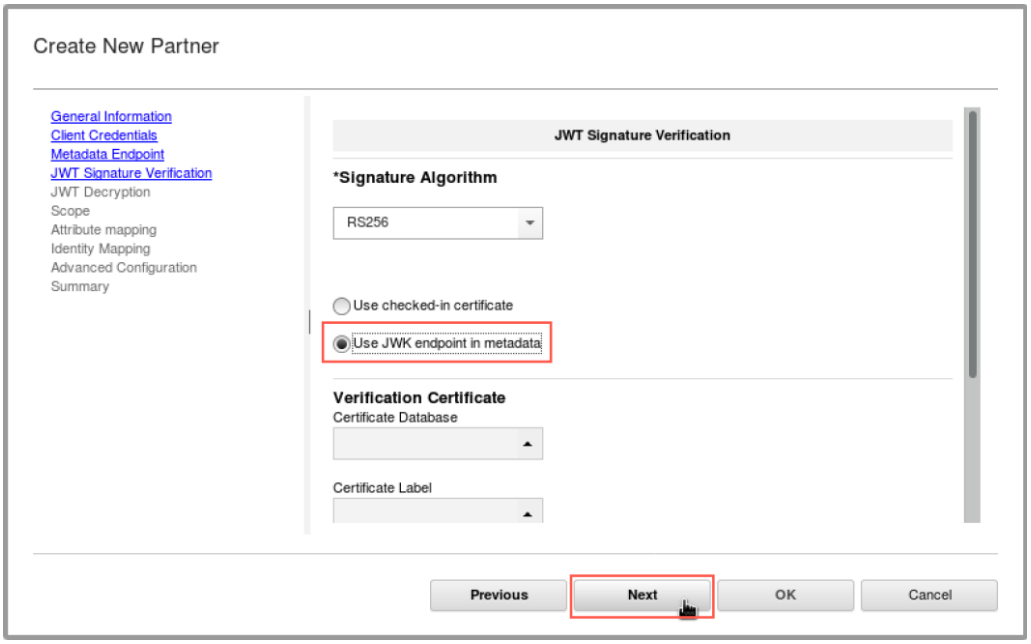


Since our OIDC OP has a metadata endpoint, we can use this to provide dynamic configuration.

Select the radio button for **Specify metadata endpoint**.

You should have noted down the metadata URL from the OP definition. It should be:
https://www.op.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCOP

Click **Next**.



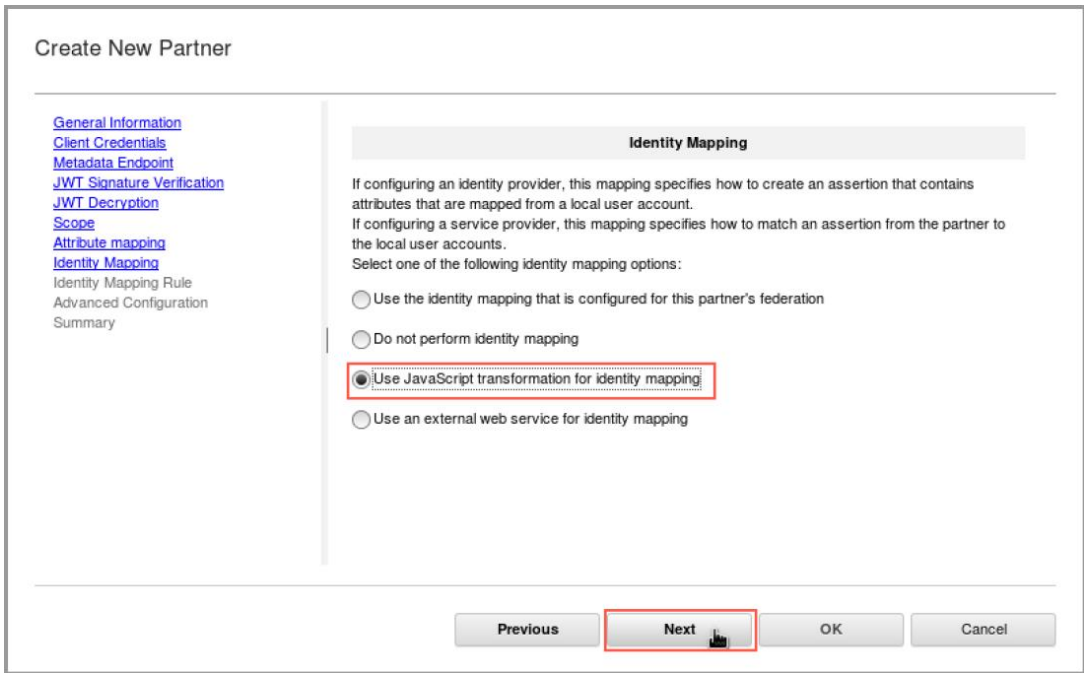
We will use the (default) RS256 signature algorithm. This uses a certificate for signature validation. Since we are using metadata, we can tell the RP to dynamically get the signing certificate of the OP from the JWK endpoint defined in the metadata (rather than manually importing and selecting a certificate).

Select the **Use JWK endpoint in metadata** and click **Next**.

We're not going to encrypt the token contents so just click **Next** again.

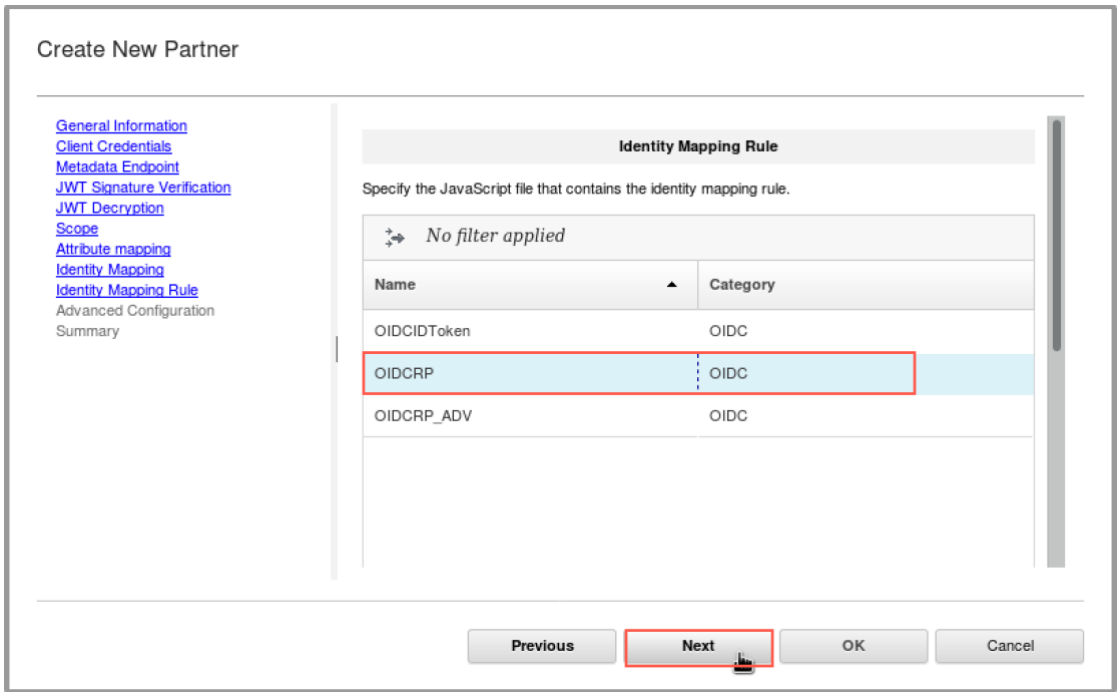
We're not going to change the scopes so just click **Next** again.

We're not going to add any attribute mapping so click **Next** again.

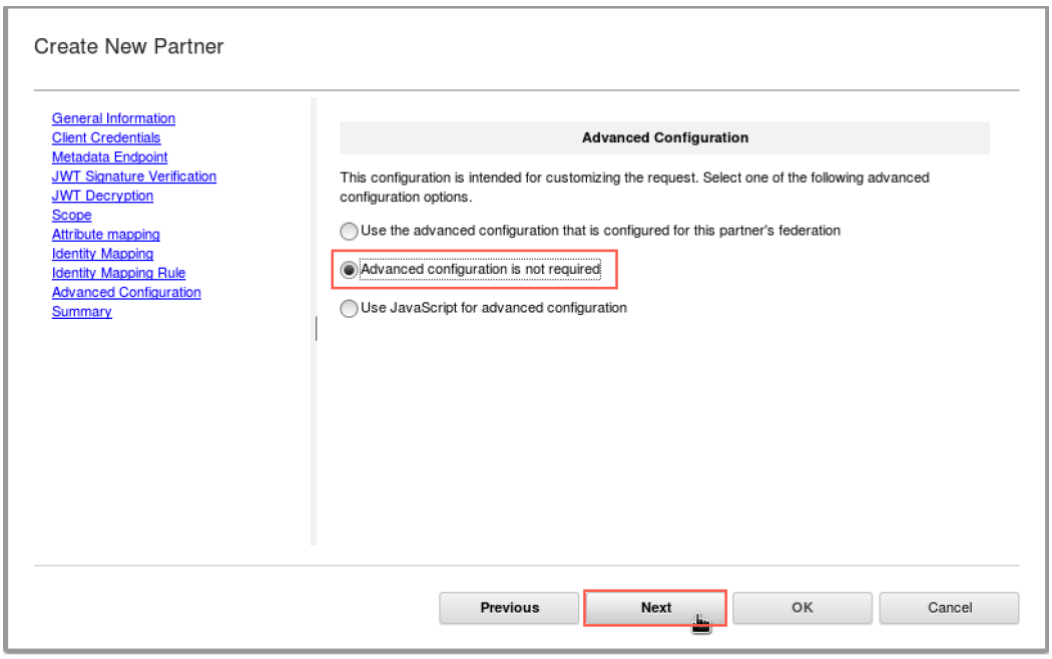


We need to specify an Identity Mapping (we didn't specify one at the federation level). We will use a built-in Javascript transformation.

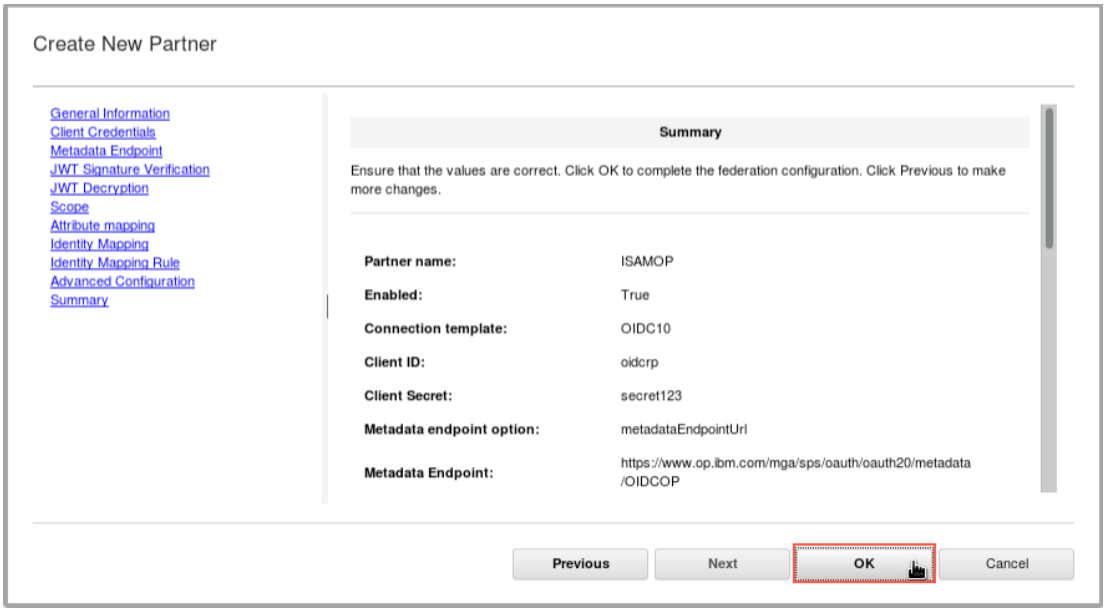
Select the radio-button for **Use JavaScript transformation for identity mapping** and click **Next**.



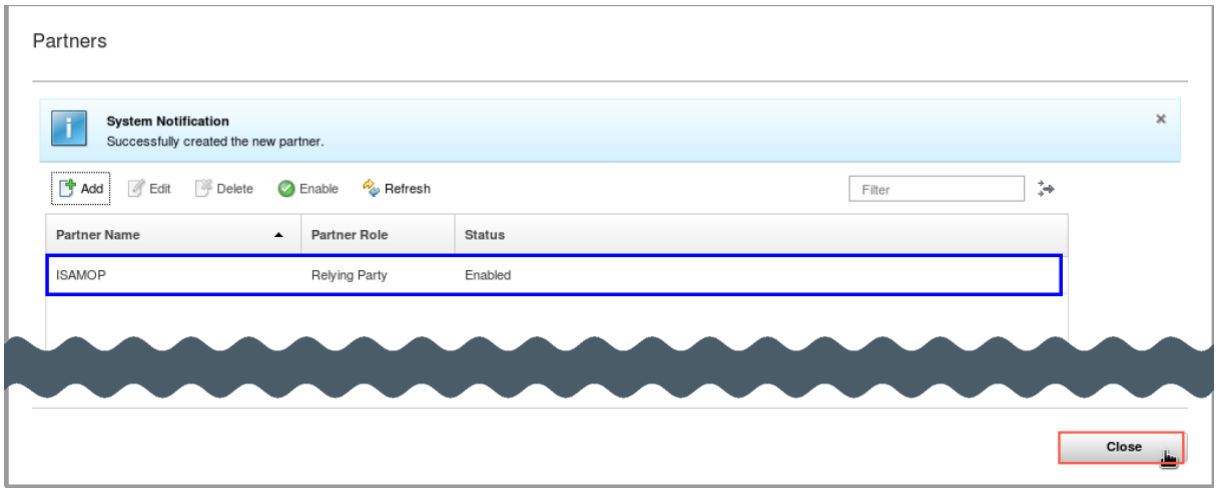
Select the **OIDCRP** mapping rule and click **Next**.



We don't want to use and advanced configuration in this first configuration. Select the radio-button for **Advanced configuration is not required** and click **Next**.



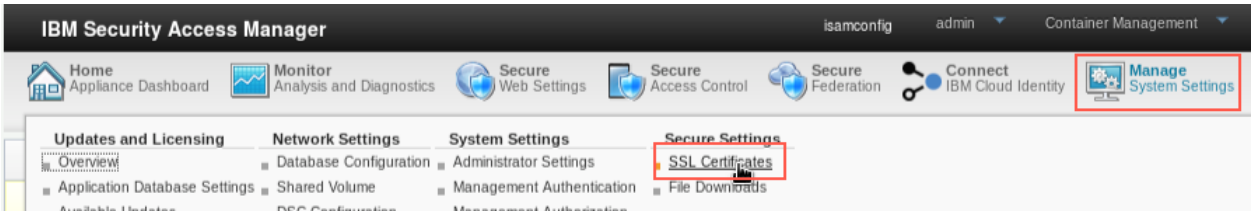
Click **OK** on the summary screen to create the partner definition.



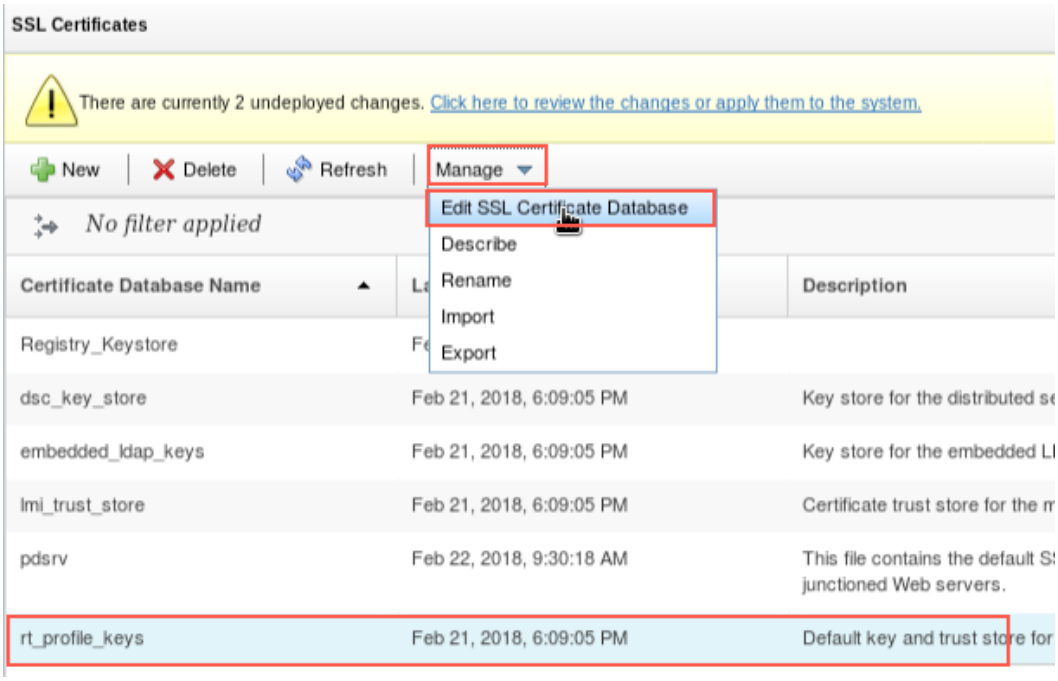
The new partner is created. Click **Close** to close the partner overlay.

4.5 Load OP Server Certificate

In order to allow direct communication from the RP (Runtime container) to the OP (via the OP Reverse Proxy), the Server Certificate of the OP Reverse Proxy must be loaded into the key store of the RP Runtime.

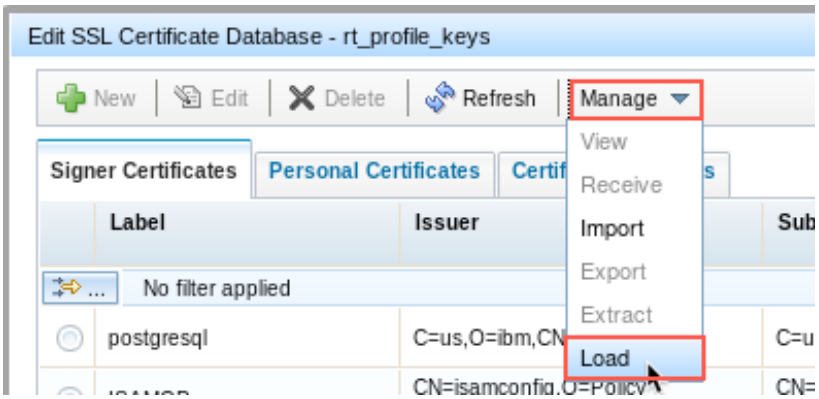


Navigate to **Manage System Settings→Secure Settings: SSL Certificates** in the mega-menu.

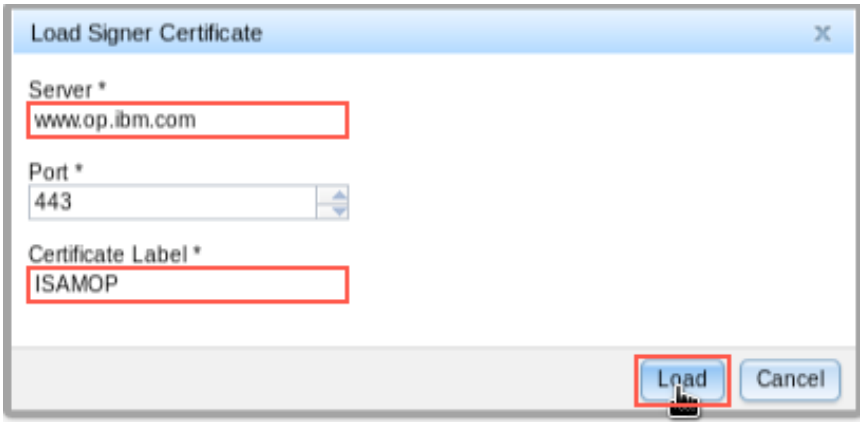


Select the **rt_profile_keys** key store. This is the one that is used by the SAM Runtime.

Click **Manage** and select **Edit SSL Certificate Database** from the pop-up menu.

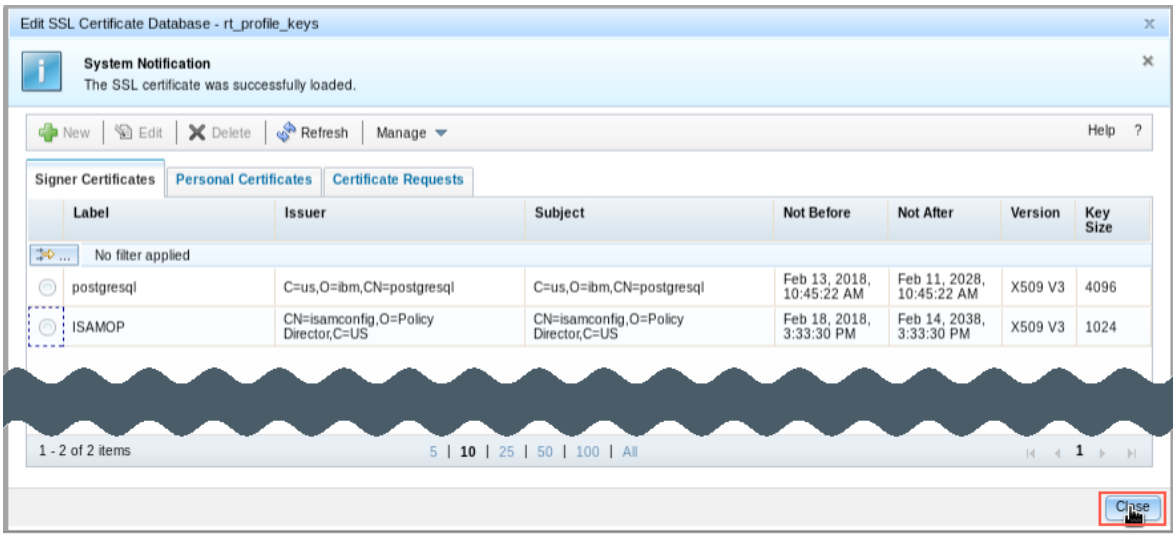


Click **Manage** and then click **Load**.



Enter **www.op.ibm.com** as the *Server* and give a certificate label. Click **Load**.

At this point, the configuration container makes a call out to the server and port given and retrieves the server certificate presented by the server. In this case, the configuration container is able to resolve the *www.op.ibm.com* address because it is listed as an alias for the OP Reverse Proxy container in Docker.



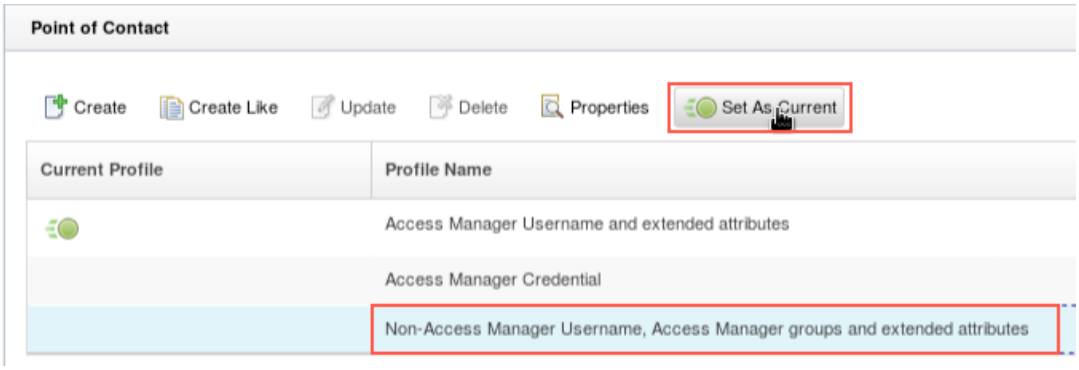
Click **Close** to close the Certificate Database overlay.

4.6 Modify the Point of Contact Profile

By default, the SAM Runtime returns users to the Reverse Proxy in a way which requires these users to exist in the local registry. When working with federated access, this is often not the case. To change the way that users are returned, the point of contact profile must be changed.

It's worth noting that this is a global setting that affects both users authenticated by the AAC Authentication Service and by the Federation Runtime. Sometimes this can be an issue.

Navigate the **Secure Federation→Global Settings: Point of Contact**.



Select the row for **Non-Access Manager Username, Access Manager groups and extended attributes** and then click **Set as Current**.

> This option, also known as *External Users* option, allows the SAM Runtime to specify a username, a set of group memberships and a set of extended attributes. The Reverse Proxy will create a credential with the username specified, with the group memberships specified (which can be used for ACL access control), and with the extended attributes added.

Deploy the changes using the link in the yellow warning message.

4.7 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Reverse Proxy and Runtime containers to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime and reverse proxy containers will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue these commands to restart the runtime and reverse proxy containers:
> ```
> docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime
> docker exec -ti -- iamlab_isamwrprp1_1 isamcli -c reload all
> docker exec -ti -- iamlab_isamwrpop1_1 isamcli -c reload all
> ```

<u>OIDC Federation configuration is now complete.</u>

5 Test SAM→SAM OIDC Federation

We can now test the OIDC Federation between our OP and RP Reverse Proxy instances.

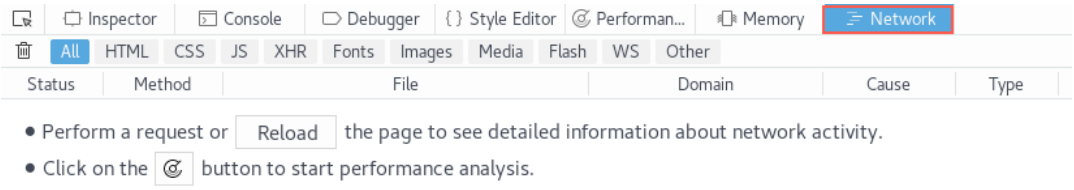
5.1 Run OIDC Flow

In the Firefox browser in the Centos VM, navigate to a protected page on the Relying Party site:
https://www.iamlab.ibm.com/app/mobile-demo/diag

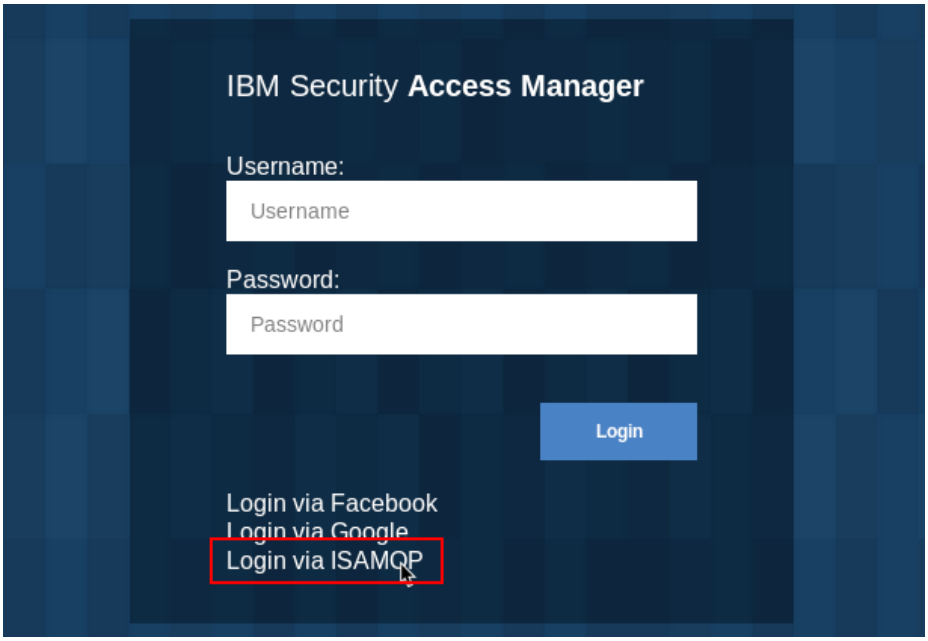
This is a protected page and so the login page of the Relying Party is displayed.

Before continuing, turn on the network trace in the Firefox browser so you can follow the OIDC flow.

Press **Ctrl-Shift-K** to open the network trace tool.



Look at the login page, it has been customized to include additional links to trigger Single Sign On:

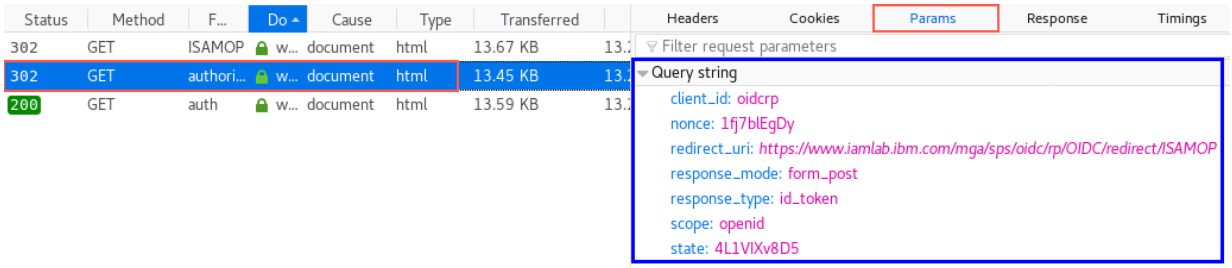


Hover over the link for **Login via ISAMOP**. This allows you to see the trigger URL for starting the OIDC exchange. The URL is: **https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/kickoff/ISAMOP**.

> In general, the format for the OIDC trigger is:
> **/**<*Runtime Junction*>/**sps/oidc/rp/**<*RP Federation Name*>**/kickoff/**<*Partner name*>

Click the **Login with ISAMOP** link.

In the network trace, you can see that the OIDC trigger link at the RP has created a redirect to the OP *authorize* endpoint:

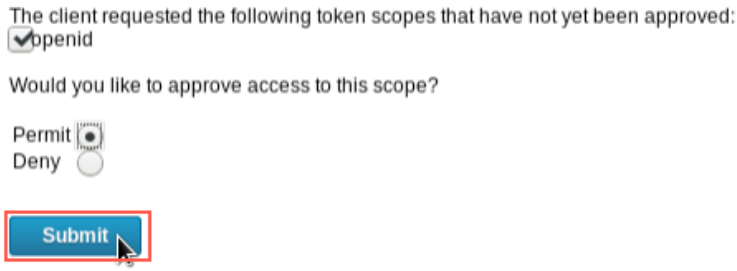


Select the **authorize?...** request and click **Params** tab. You can now see the parameters sent in the query string.

The main browser window is showing the OP login page:

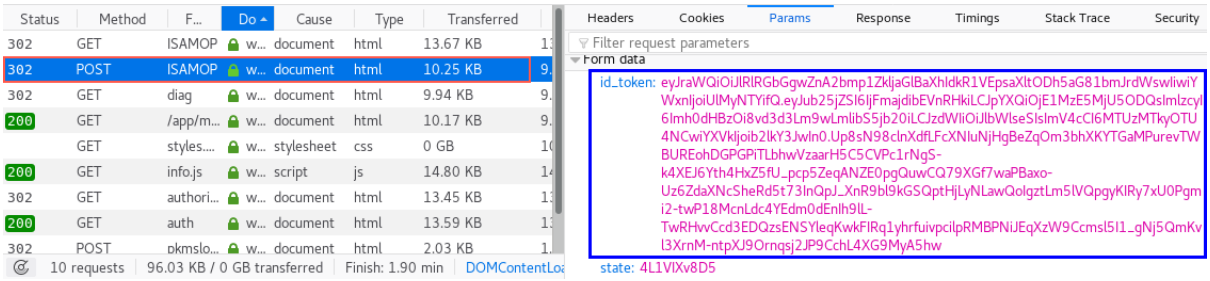
Login with **emily** and **Passw0rd**.

Emily is now shown a *Consent to Authorize* page. This is shown because of the client trust settings of the OIDC Definition (Prompt once and remember). Emily must consent to her identity information being passed to the Relying Party. This page is a template which can be customized.



Click **Submit** to approve the RP for the requested OIDC scope (which allows it to receive an identity token from the OP).

At this point the OIDC flow completes and you are shown the target page on the RP. Look at the trace again:



Select the POST to ISAMOP. This is the response data from the OP being POSTed back to the RP (using a scripted POST set up in the previous response). This is an *Implicit* OIDC flow and so the *id_token* and *access_token* are returned via the browser.

In the browser window, you are on a diagnostics page:

The header of the page shows that the logged in user is **https://www.op.ibm.com/emily**. This username was created in the *OIDCRP* mapping rule that we specified in the RP partner definition for the OP.

You can close the developer tools using the cross in the top-right corner of the developer tools section.

Further down the page, you can see the SAM Credential created at the RP. Review this if you like. When you're done, click the **Logout** link at the top of the diagnostics page to log Emily out from the RP.

5.2 Run the OIDC flow a second time

We will now run the OIDC flow a second time to show that Emily's authorization for the RP has been remembered.

Once again, navigate to: **https://www.iamlab.ibm.com/app/mobile-demo/diag**

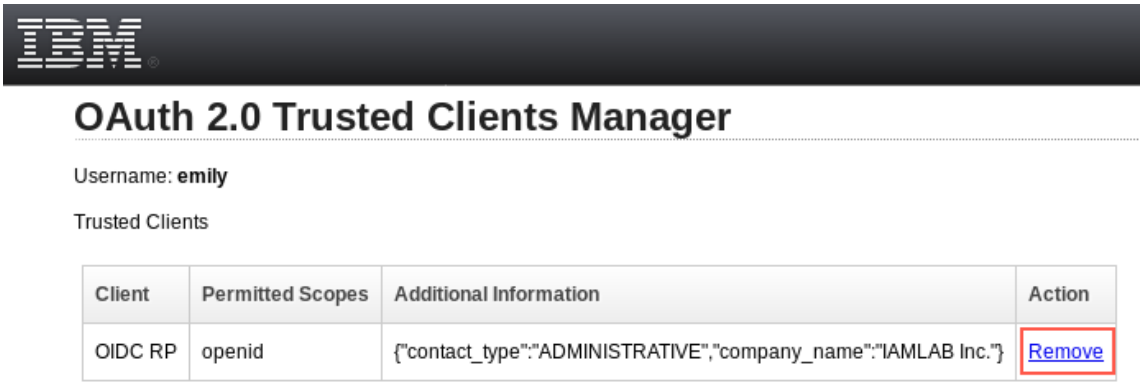
Click the **Login via ISAMOP** link.

Emily is redirected to the OP. She is still authenticated there. Since her approval of the RP to use the OIDC scope is remembered, no prompt is required. The OP returns a token immediately and Emily is logged into the RP. The diagnostics screen is shown again.

Click the logout link to log Emily out of the RP again.

5.3 Review approvals in OP Client Manager

Emily can review the clients that she has authorized in the Client Manager.
Navigate to the following URL: **https://www.op.ibm.com/mga/sps/oauth/oauth20/clients**



The clients that Emily has authorized are displayed together with the permitted scopes. This is a template page which can be customized. Notice the *Additional Information* displayed here. This is information from the RP definition at the OP which could be used to populate this page.

Click **Remove** to remove authorization for the RP.

Repeat the OIDC flow to show that Emily is once again prompted for authorization when the RP attempts to use the OP for single sign-on.

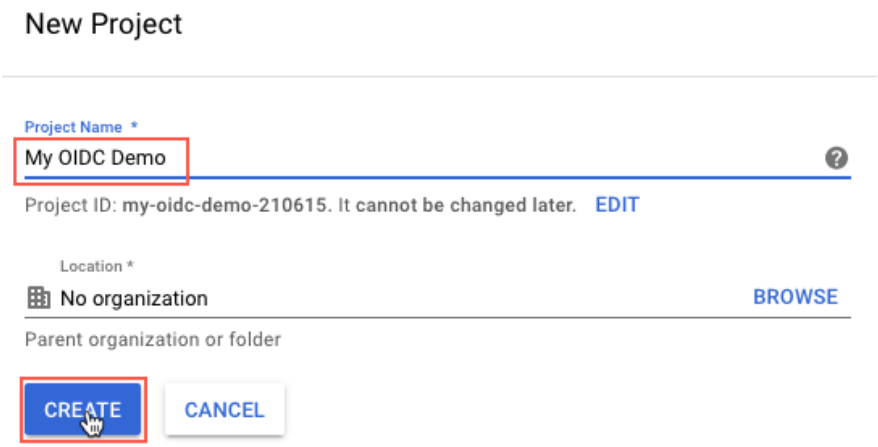
You have successfully shown an end-to-end OpenID Connect flow

6 Integration with Google

In this section we will integrate our Access Manager OIDC Relying Party with Google as an OIDC Provider. Google is fully standards compliant with OpenID Connect and has a metadata URI so the integration is pretty simple.

6.1 Create the Google Client Credentials

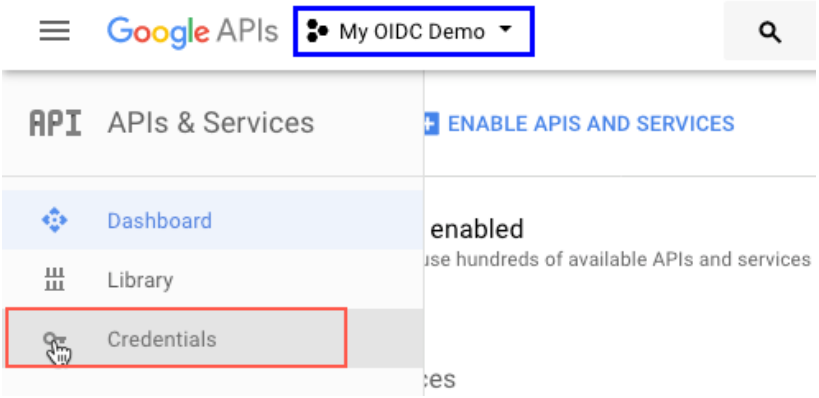
As part of provisioning identity providers, a set of credentials to use with Google APIs will be created.
Note: The screenshots taken may not match what is currently available on the Google developer portal. The images were current in July 2018 but may have changed since then. These steps may need to be modified to reflect different ways to access pages/functionality on the Google developer portal.



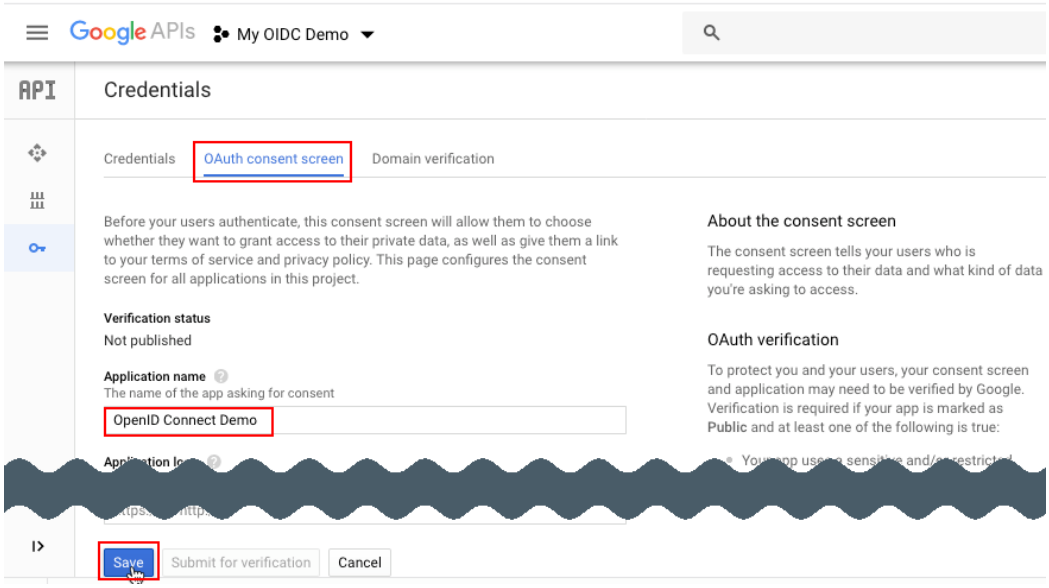
Navigate to https://console.developers.google.com/. If this is the first project for the Google account used, "New project" will be shown automatically. Otherwise, from the project menu dropdown in the top bar, select "New project…"

Create a new project. In this case the project is called **My OIDC Demo**.

Once the project has been created, get to the project dashboard.

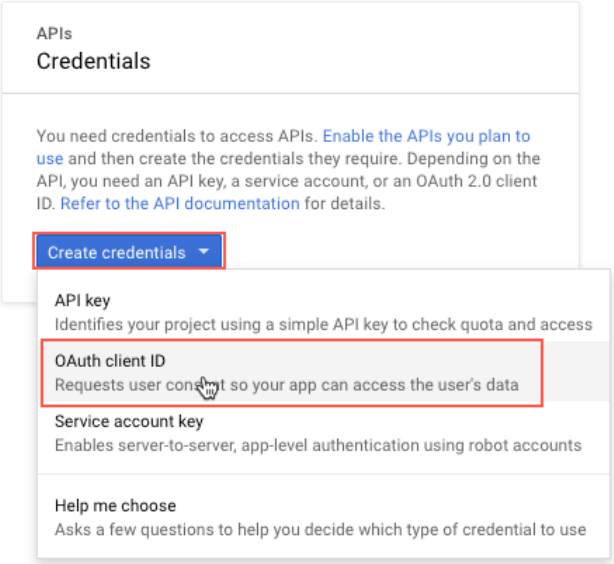


Select **Credentials** from the navigation bar.

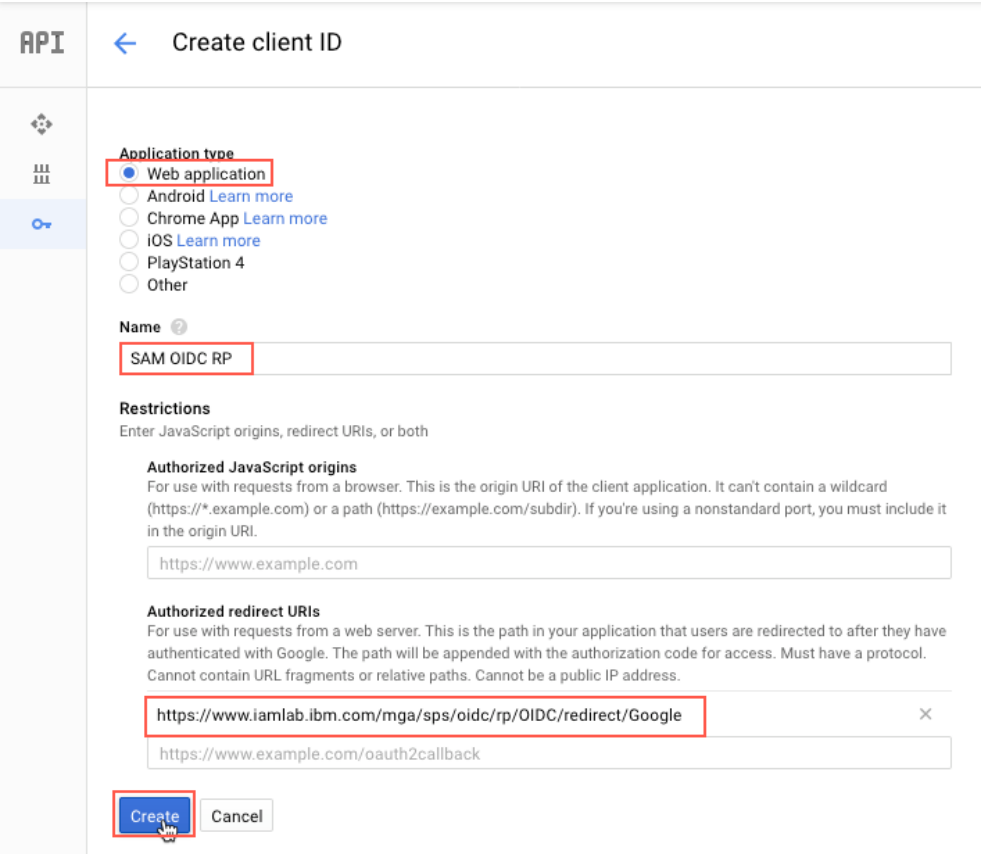


Select the **OAuth consent screen** tab.

Add an **Application name** and click **Save**.



Click **Create credentials** and select **OAuth client ID** from the pop-up menu.



Select **Web application** as the *Application type*.

Add a *Name* client, for example **SAM OIDC RP**.

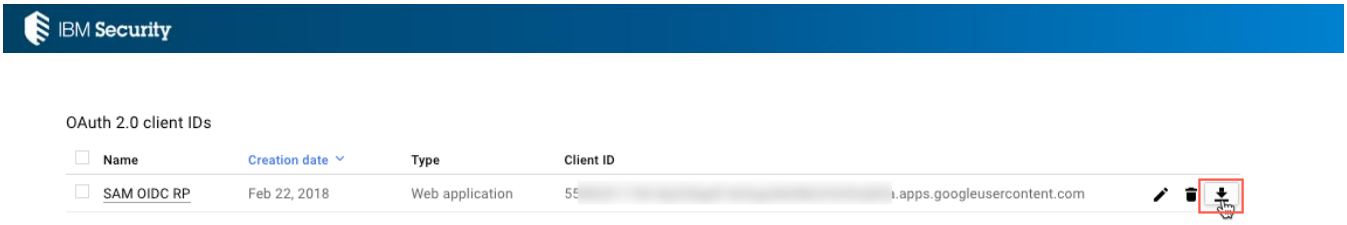
Add an *Authorized redirect URI* with the value:
https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/redirect/Google
Note: You must press enter to add the Redirect URI

> This redirect URI is what our Access Manger RP redirect URI will be assuming we create a new partner named *Google* in our federation named *OIDC*.

Click **Create**.

After creating a client, the Client ID and Secret are shown:

You could note these values here, but we'll download a file containing them in the next step so just click **OK**.

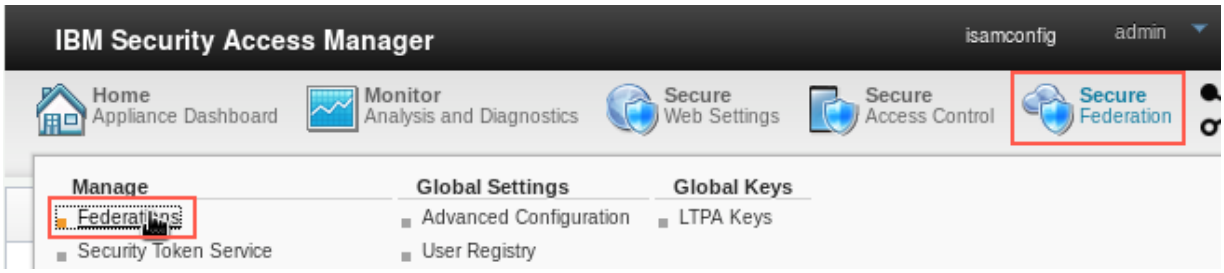


Click the **Download** icon for the new client.

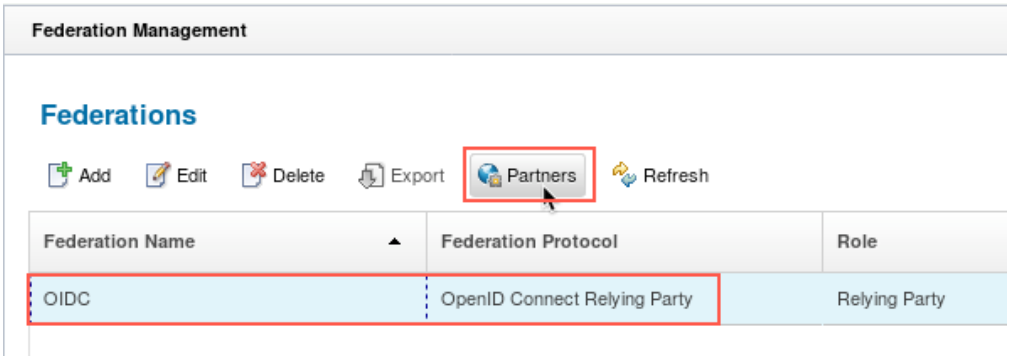
Save the JSON file to the Desktop (or somewhere else you can easily find it). If you open this file, you'll find it contains the Client ID and Client Secret and other URLs needed for RP configuration.

6.2 Create Relying Party Partner in Access Manager

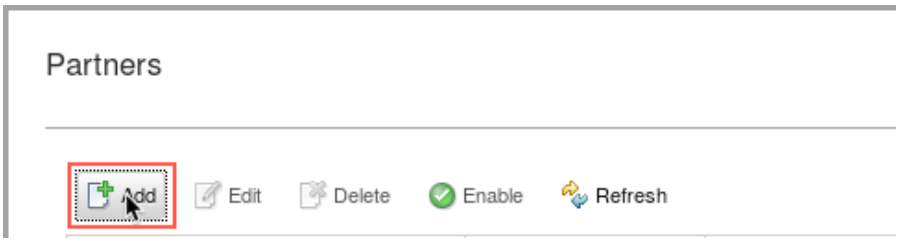
We will now create a partner for our OIDC RP federation in Access Manager for the Google OP.



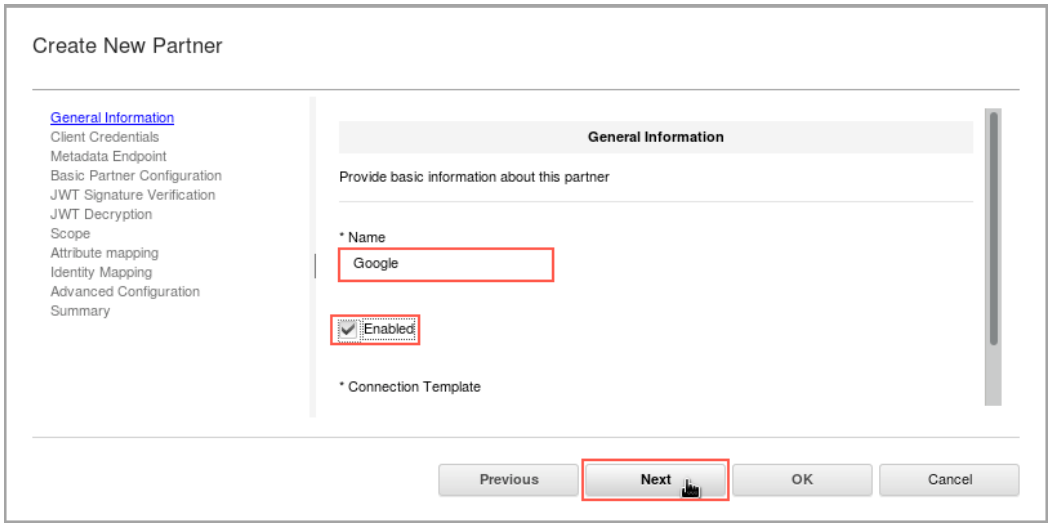
Navigate to **Secure Federation→Manage: Federations** in the mega-menu.



Select the **OIDC** federation and click **Partners**.

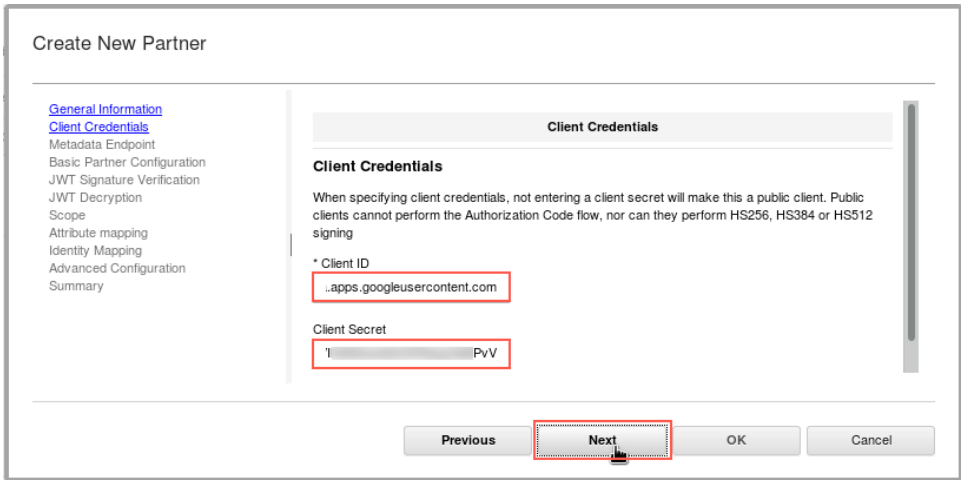


Click **Add**.



Enter **Google** as the *Name* and select the **Enabled** flag. Then click **Next**.

The Name given here will appear as part of trigger and redirect URLs so you need to use this exact value.

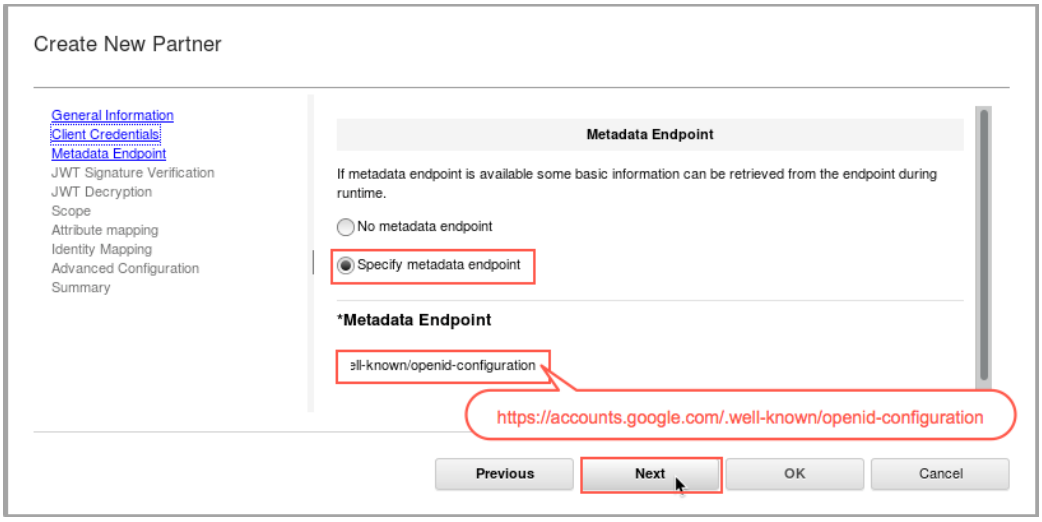


Open the JSON document you downloaded from Google (you can double-click it to open in gedit):


```
{"web":{"client_id":"5xxxxxxxxx6-
9xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxa.apps.googleusercontent.com","project_id":"my-oidc-
demo","auth_uri":"https://accounts.google.com/o/oauth2/auth","token_uri":"https://accounts
.google.com/o/oauth2/token","auth_provider_x509_cert_url":"https://www.googleapis.com/oaut
h2/v1/certs","client_secret":"VIXxXxXxXxXxXxXxXxXxXxXxvV","redirect_uris":["https://www.iaml
ab.ibm.com/mga/sps/oidc/rp/OIDC/redirect/Google"]}}
```


Identify the *client_id* and *client_secret*. Cut and paste them into the fields in the browser.

Click **Next**.

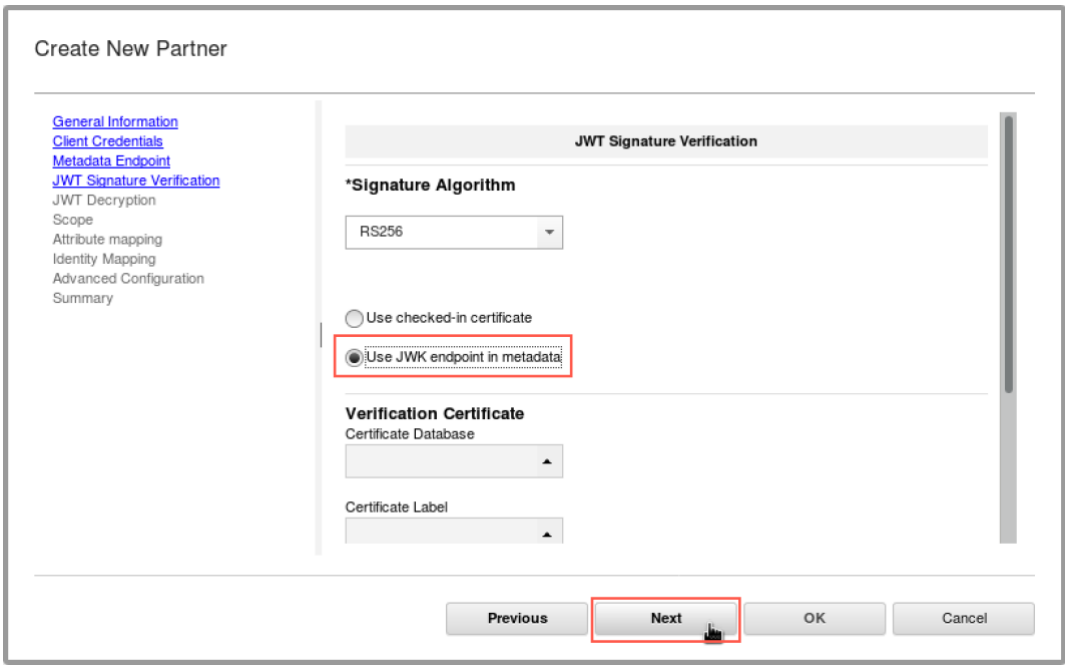


Google has a metadata endpoint. We can use this for dynamic configuration.

Select the radio button for **Specify metadata endpoint**.

You could Google for the OpenID metadata URL for Google but, to save time, it is:
https://accounts.google.com/.well-known/openid-configuration

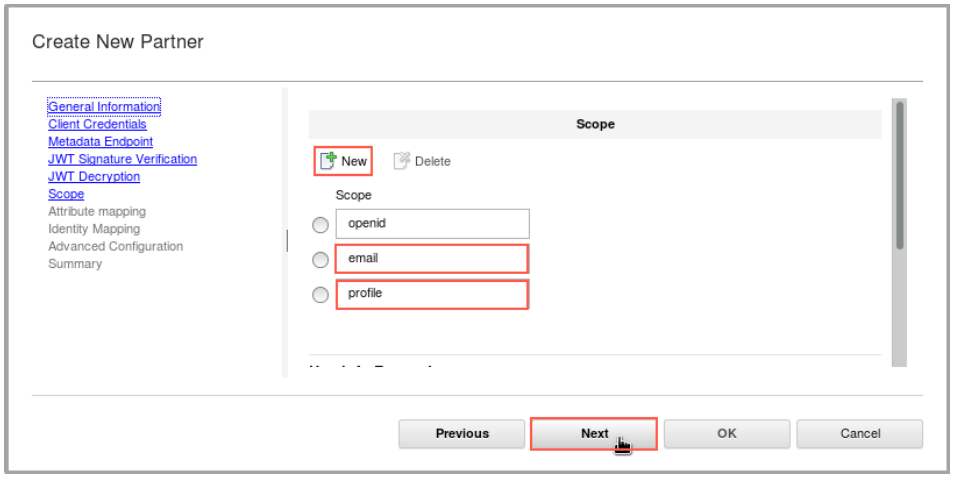
Click **Next**.



Since we are using metadata, we can tell the RP to dynamically get the signing certificate of the OP from the JWK endpoint defined in the metadata (rather than manually importing and selecting a certificate).

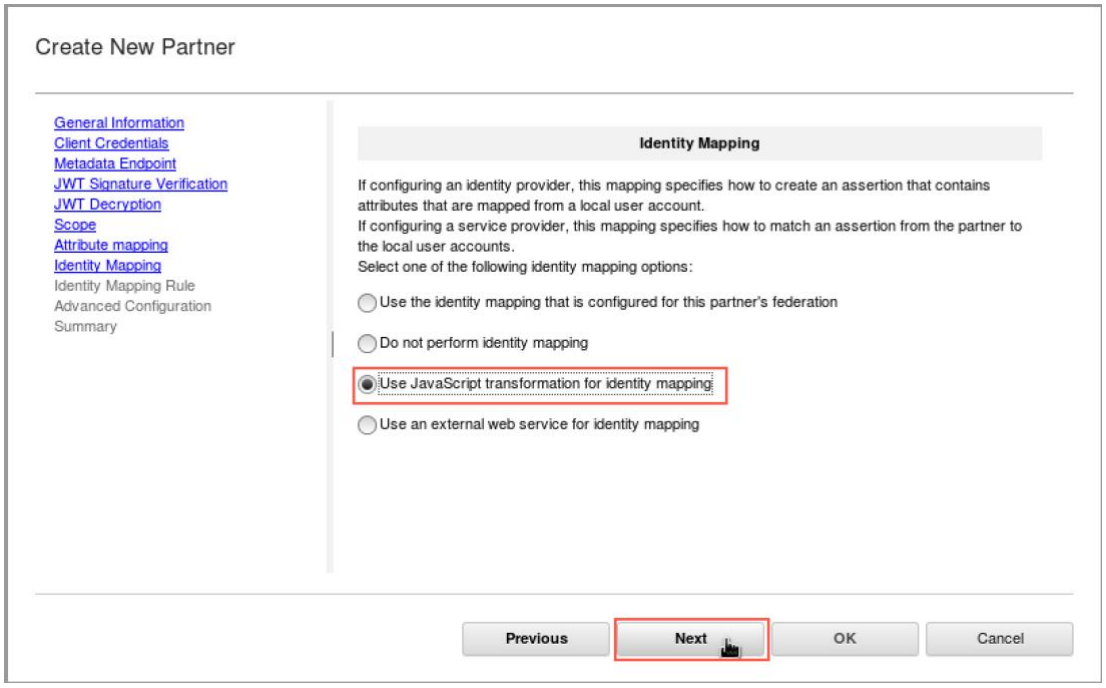
Select the **Use JWK endpoint in metadata** and click **Next**.

We're not going to encrypt the token contents so just click **Next** again.



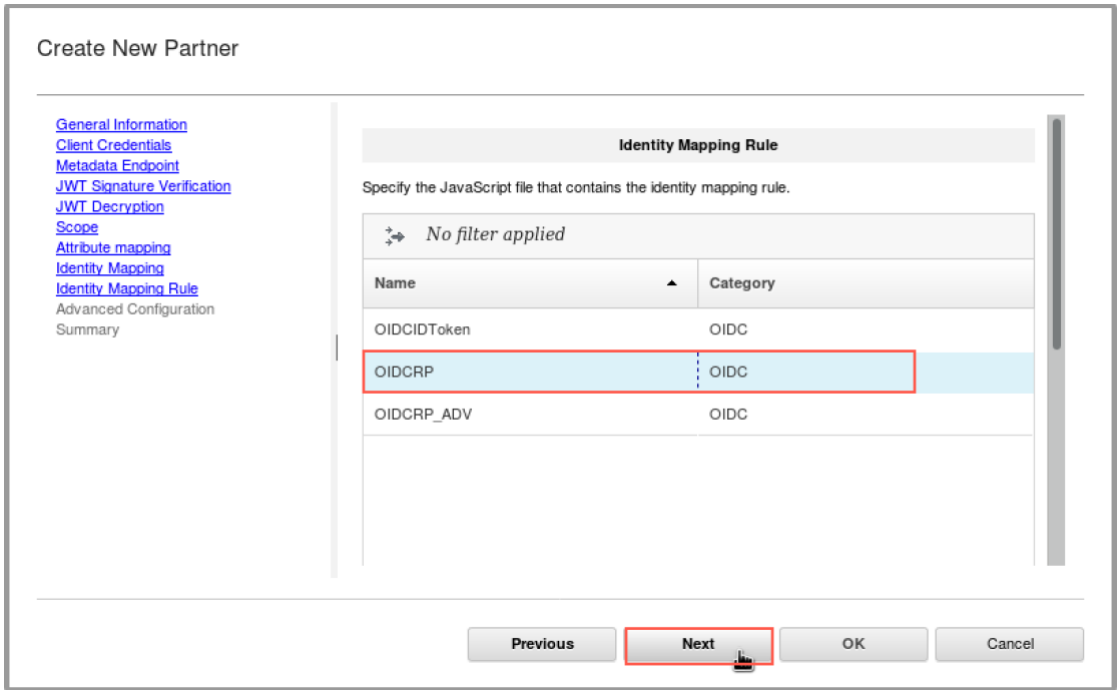
Google supports the standard OIDC scopes which are *oidc*, *email*, and *profile*. Use the **New** button to add **email** and **profile** scopes. Then click **Next**.

We're not going to add any attribute mapping so click **Next** again.

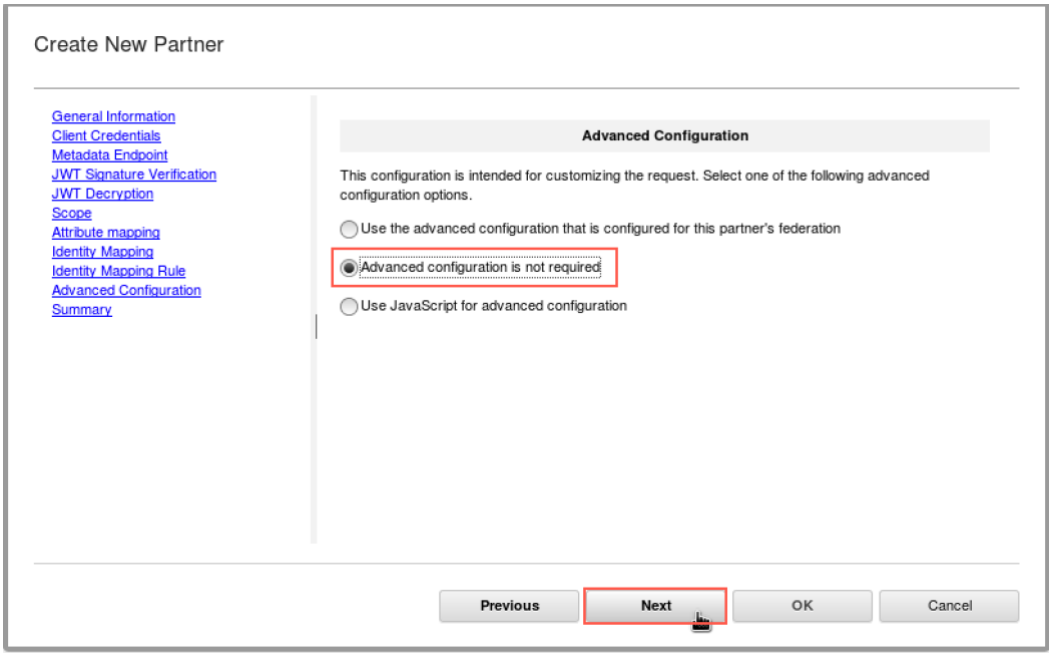


We need to specify an Identity Mapping (we didn't specify one at the federation level). We will use a built-in Javascript transformation.

Select the radio-button for **Use JavaScript transformation for identity mapping** and click **Next**.



Select the **OIDCRP** mapping rule and click **Next**.



We don't want to use and advanced configuration in this first configuration. Select the radio-button for **Advanced configuration is not required** and click **Next**.

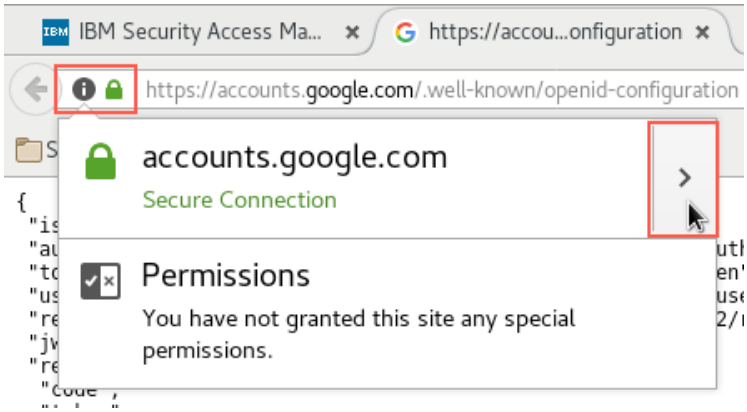
Click **OK** on the summary screen to create the partner definition.

The new partner is created. Click **Close** to close the partner overlay.

6.3 Get Google Root CA Certificates

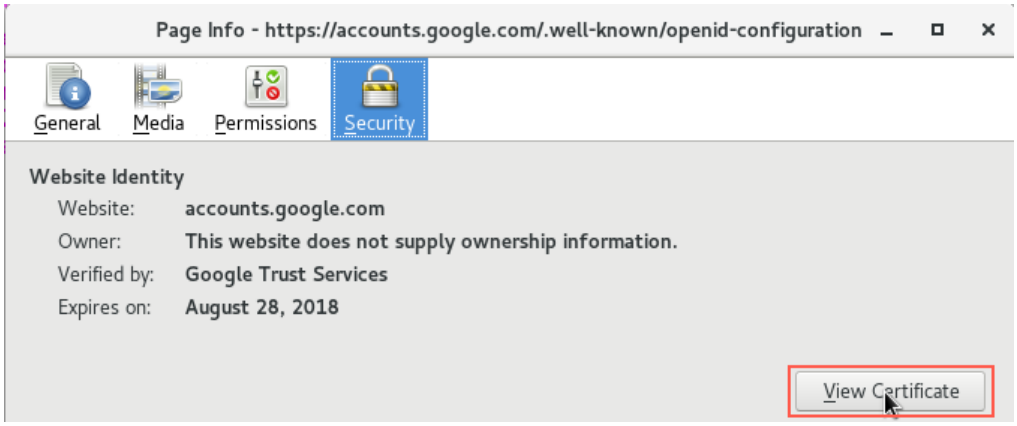
In order to allow direct communication from the RP (Runtime container) to Google servers (metadata and JWKS endpoints), the Root CA certificate used by the Google endpoints must be loaded into the key store of the RP Runtime. We will download this certificate and then import to Access Manager.

Open a browser and navigate to **https://accounts.google.com/.well-known/openid-configuration**.

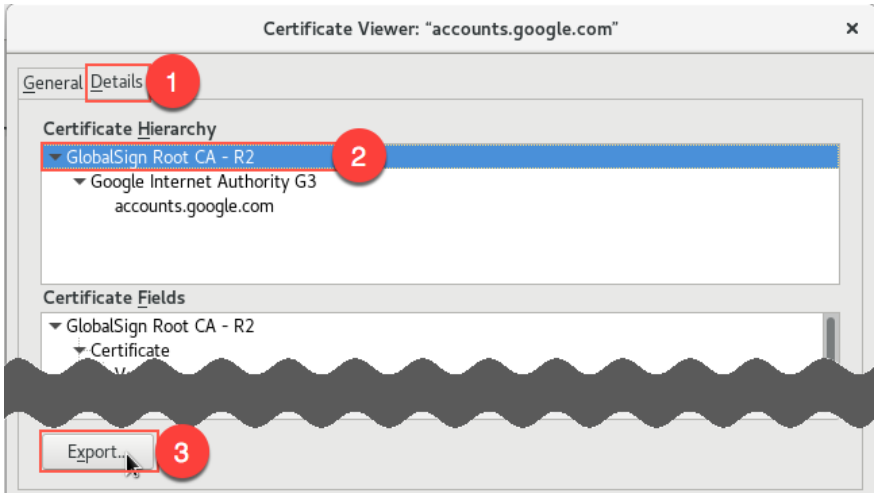


Click on the padlock icon next to the URL and then click the arrow in the pop-up window.

Click **More Information**.



Click **View Certificate**.



Select the **Details** tab.

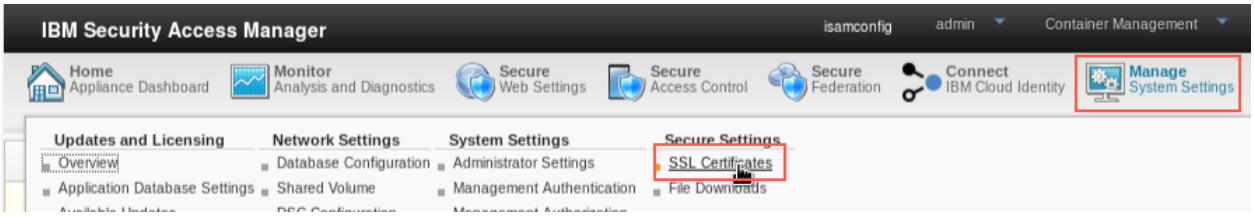
Select the root CA (*GlobalSign Root CA - R2* in this case) and click **Export…** button.

Save the certificate file onto the Desktop (or somewhere else you can easily find it).

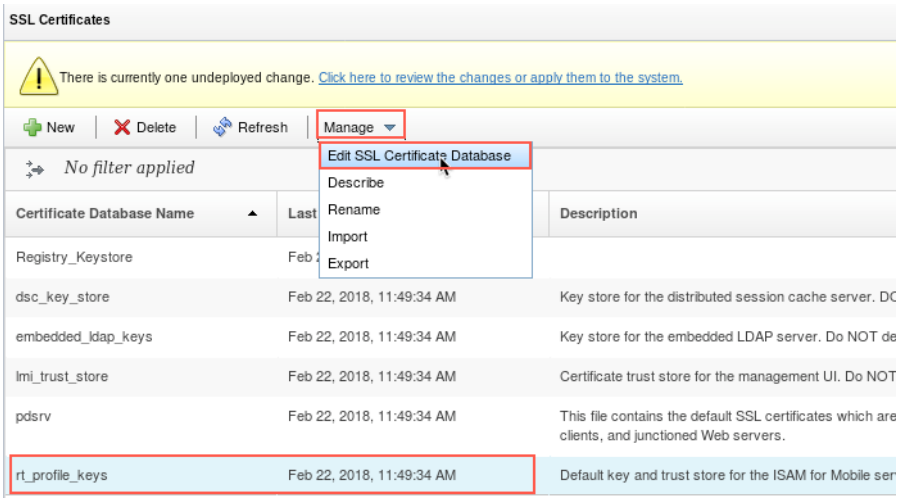
> The other endpoint we need to trust is the JWKS endpoint (https://www.googleapis.com/oauth2/v3/certs). You could check but, at the time of writing, the same CA signs this endpoint so you don't need to download it again.

6.4 Import Google CA Certificate to Runtime key store

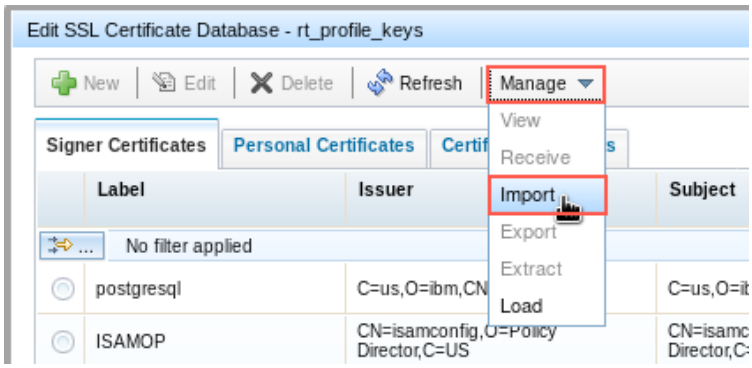
Return to the Access Manager LMI.



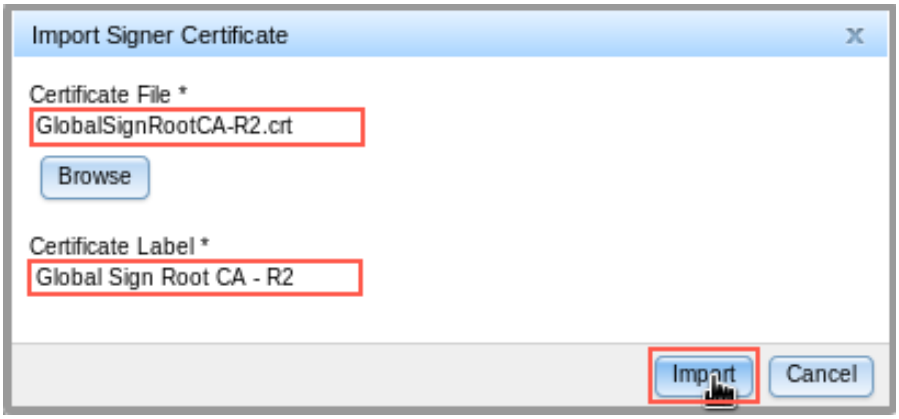
Navigate to **Manage System Settings→Secure Settings: SSL Certificates** in the mega-menu.



Select the **rt_profile_keys** key store. Click **Manage** and select **Edit SSL Certificate Database** from the pop-up menu.



Click **Manage** and select **Import** from the pop-up menu.



Click **Browse** and select the GlobalSign CA certificate you just exported. Enter a Label and click **Import**.

Click **Close** to close the Certificate Database overlay.

Deploy the changes using the link in the yellow warning message.

6.5 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Runtime container to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime container will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue this command to restart the runtime container:
> **`docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime`**

<u>Configuration of Google as an OIDC Provider is now complete.</u>

6.6 Test Google OIDC Flow

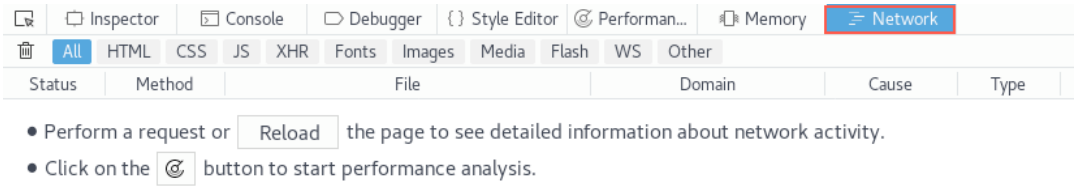
We can now test the OIDC Federation between Google and our OIDC RP.

In the Firefox browser in the Centos VM, navigate to a protected page on the Relying Party site:
https://www.iamlab.ibm.com/app/mobile-demo/diag

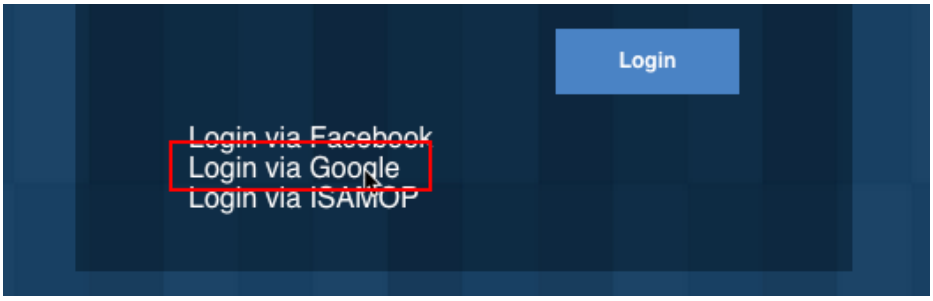
This is a protected page and so the login page of the Relying Party is displayed.

Before continuing, turn on the network trace in the Firefox browser so you can follow the OIDC flow.

Press **Ctrl-Shift-K** to open the network trace tool:

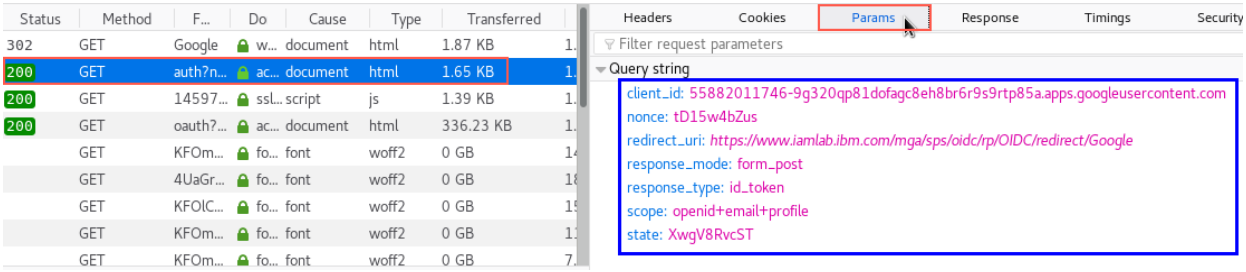


Look at the login page, it has been customized to include additional links to trigger Single Sign On:



Click the **Login with Google** link.

In the network trace, you can see that the OIDC trigger link at the RP has created a redirect to the OP *authorize* endpoint:

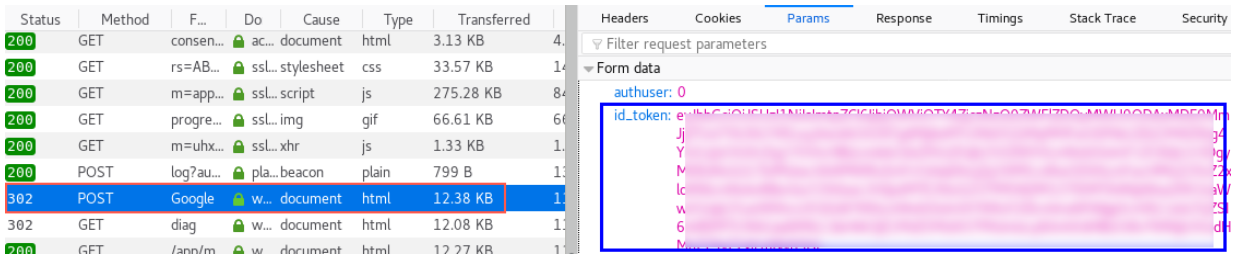


Select the **auth?...** request and click **Params** tab. You can now see the parameters sent in the query string. Notice the 3 scopes being requested (openid, email, and profile).

The main browser window is showing the Google login page.

Login with your Google account. Note that Google does not have a consent page.

At this point the OIDC flow completes and you are shown the target page on the RP. Look at the trace again:



Select the POST to Google. This is the response data from Google being POSTed back to the RP (using a scripted POST set up in the previous response). This is an *Implicit* OIDC flow and so the *id_token* is returned via the browser.

In the browser window, you are on a diagnostics page:

The header of the page shows that the logged in user is **https://accounts.google.com/xxxxxxxx**. This username was created in the *OIDCRP* mapping rule that we specified in the RP partner definition for the OP.

You can close the developer tools using the cross in the top-right corner of the developer tools section.

Further down the page, you can see the SAM Credential created at the RP. Review this if you like. You will see your Name and e-mail address have been populated from Google.

When you're done, click the **Logout** link at the top of the diagnostics page to log out from the RP.

<u>You have successfully configured Google Integration using OpenID Connect</u>

7 Facebook Integration

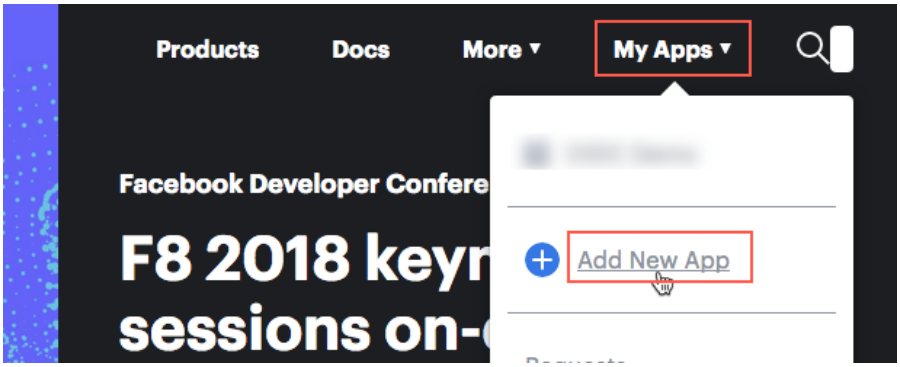
In this section we will integrate our Access Manger OIDC Relying Party with Facebook.

Facebook does not currently support OpenID Connect but we can integrate our Relying Party with some customization. Effectively we use the OAuth 2.0 support to get an Access Token from Facebook and then call the Facebook Graph API to get the user's identity in a custom mapping rule.

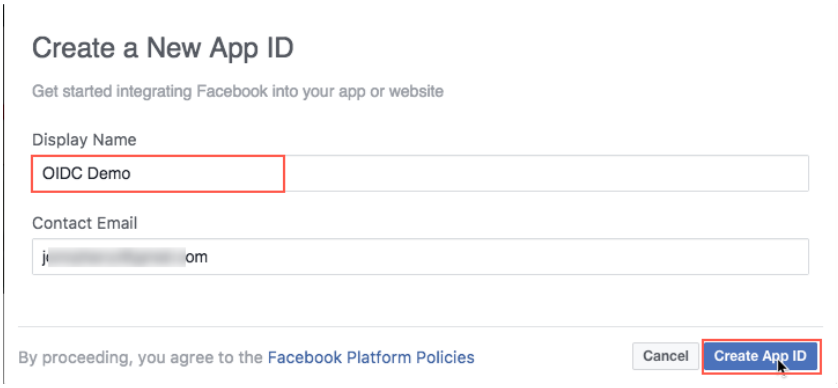
7.1 Set up a Client Application on Facebook

Navigate to **https://developers.facebook.com**

Click **Log In** and login with your Facebook credentials.

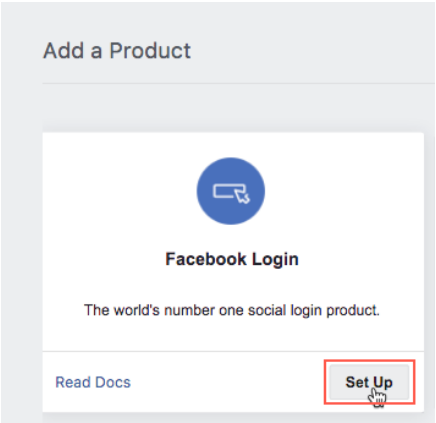


Click **My Apps** and select **Add a New App**.



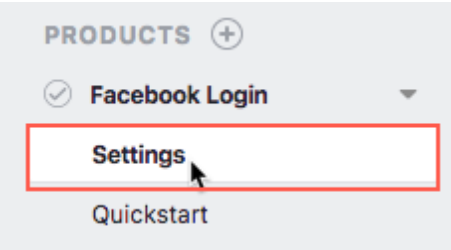
Enter a *Display Name* for your application (e.g. **OIDC Demo**) and click **Create App ID**.

At this point you may be required to complete a security check (e.g. reCAPTCHA).

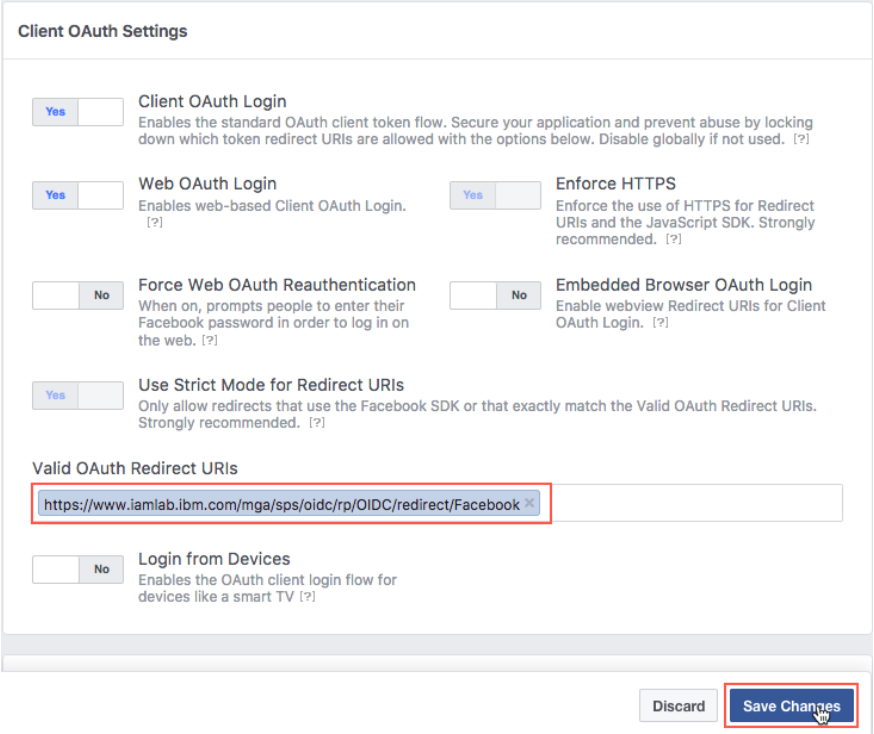


On the dashboard, find the *Facebook Login* product and click **Set Up**.

A Quick Setup wizard is opened but you don't really want to use it.

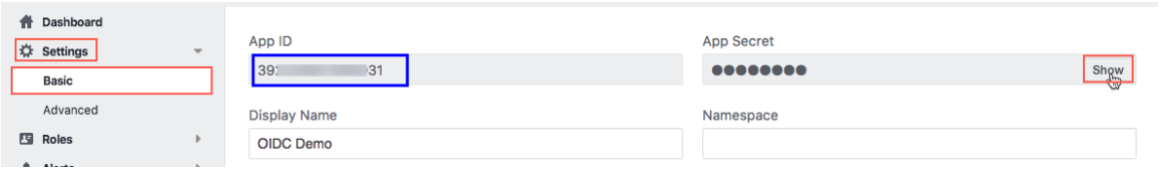


Expand **Facebook Login** product on the navigation bad and click **Settings**.



Add **https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/redirect/Facebook** to the list of *Valid OAuth redirect URIs*.

Click **Save Changes**.



Expand **Settings** on the navigation bar and select **Basic**.

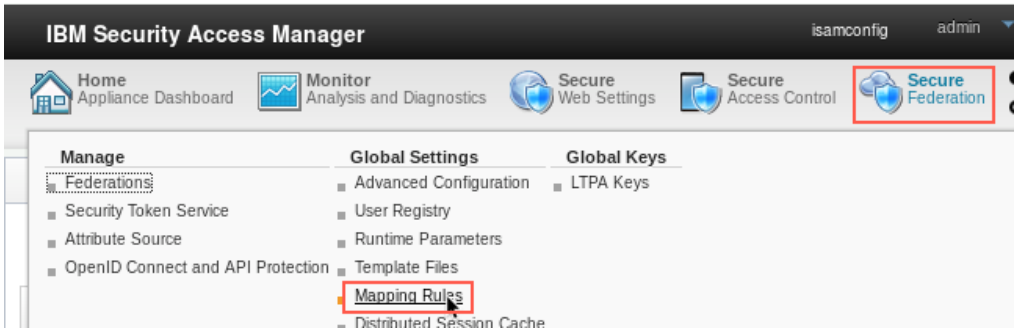
Click **Show** to show the *App Secret*. You may have to enter your Facebook password again.

Make a note of the *App ID* and *App Secret*. You will need these when you configure the Relying Party.

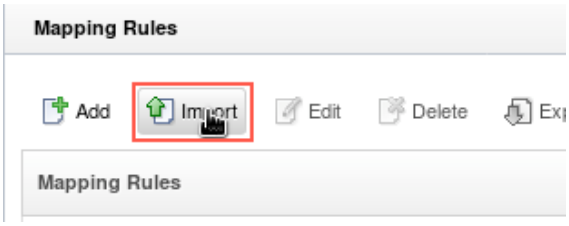
Facebook configuration is complete. You can close this window.

7.2 Import Custom Mapping Rule

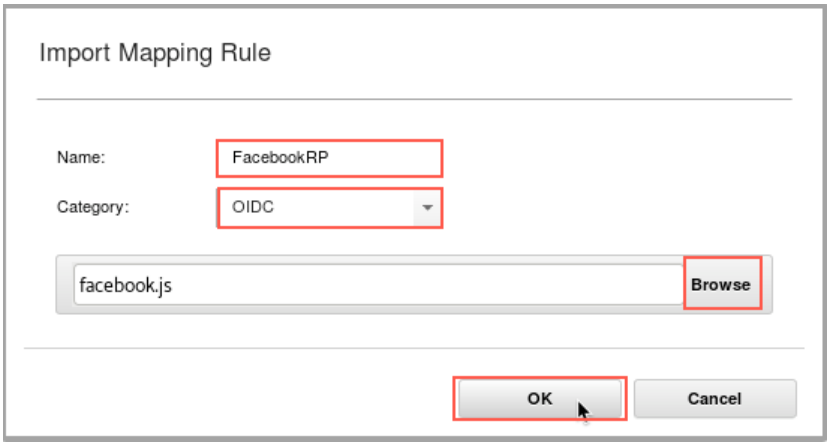
In order to support Facebook with our OIDC Relying Party, we need to add a custom mapping rule. This mapping rule will take the Access Token acquired from the Facebook OAuth Authorize endpoint and use it to call the Facebook Graph API to get the identity of the user.



Navigate to **Secure Federation→Global Settings: Mapping Rules**



Click **Import**.



Enter a name for the rule (without spaces). E.g. **FacebookRP**

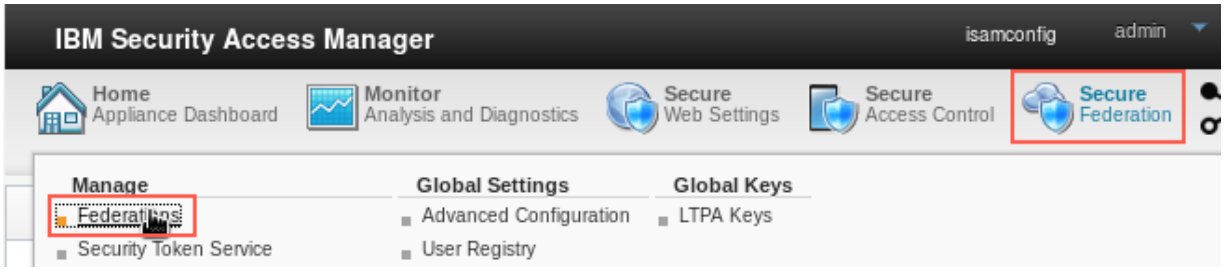
Select **OIDC** as the *Category* from the drop-down list.

Click **Browse** and select file **/home/demouser/studentfiles/oidc/facebook.js**.

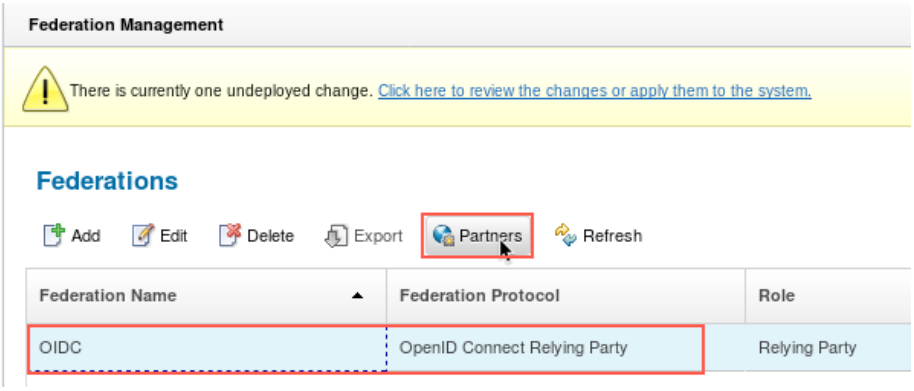
Click **OK**.

7.3 Create Relying Party Partner in Access Manager

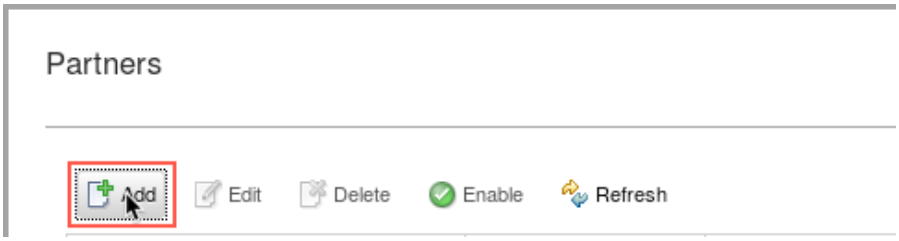
We will now create a partner for our OIDC RP federation in Access Manager for Facebook.



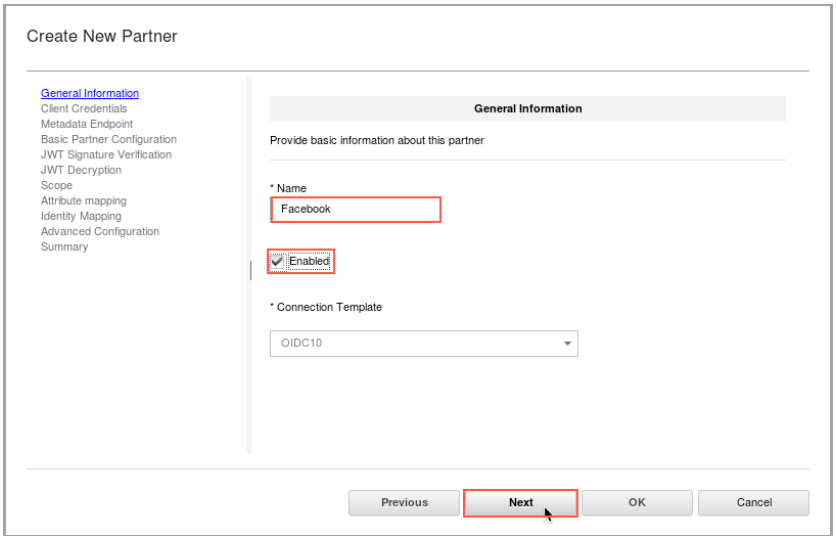
Navigate to **Secure Federation→Manage: Federations** in the mega-menu.



Select the **OIDC** federation and click **Partners**.

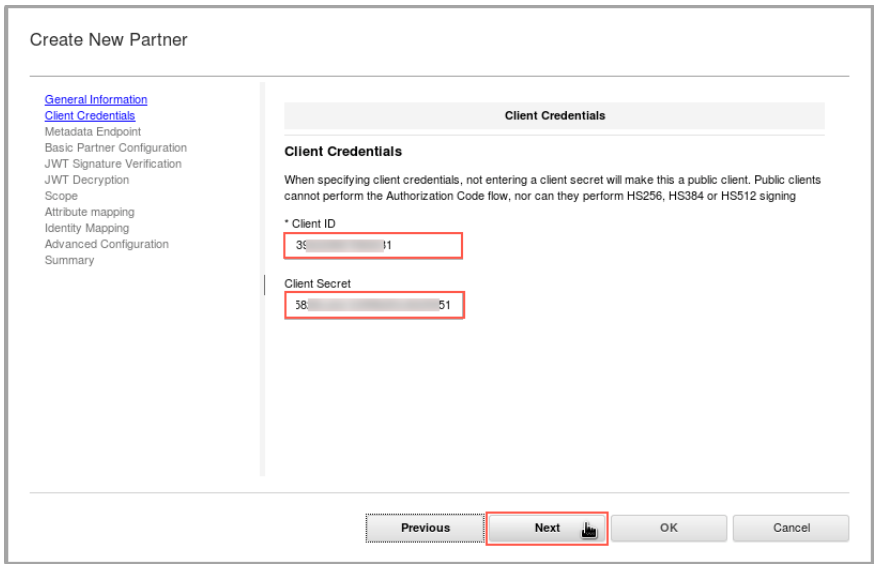


Click **Add**.



Enter **Facebook** as the *Name* and select the **Enabled** flag. Then click **Next**.

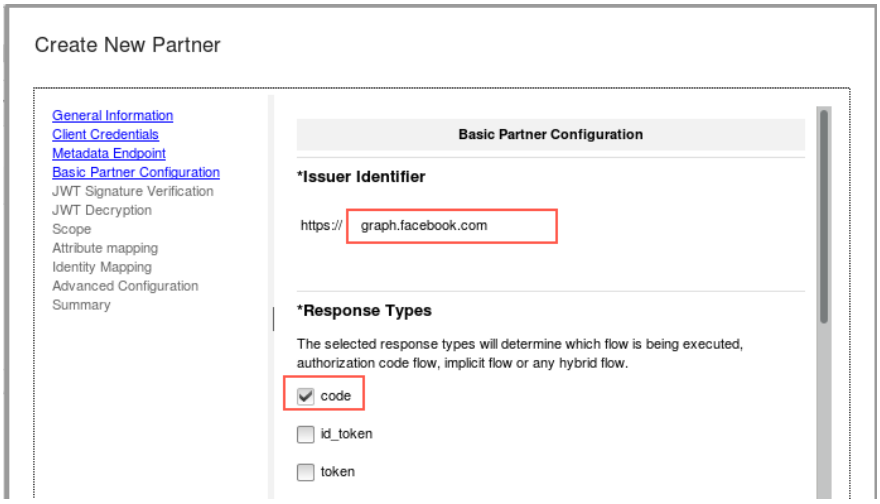
The Name given here will appear as part of trigger and redirect URLs so you need to use this exact value.



Complete the *client_id* and *client_secret* fields using the App ID and Secret you noted down for your registered app.

Click **Next**.

Facebook does not have a metadata endpoint (since it is not a true OIDC Provider) so just click **Next**.

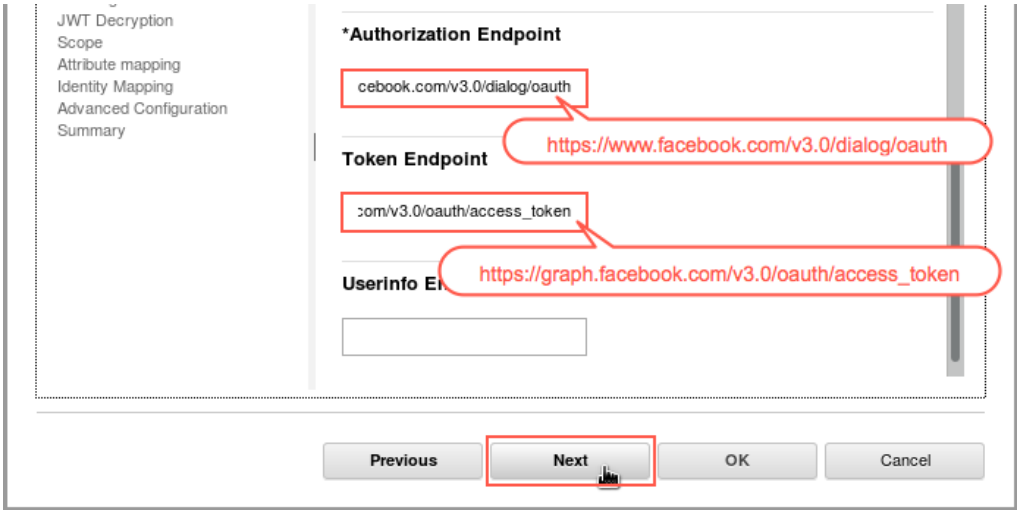


Without the metadata endpoint, we must enter the partner configuration manually.

Enter **graph.facebook.com** as the *Issuer Identifier*.

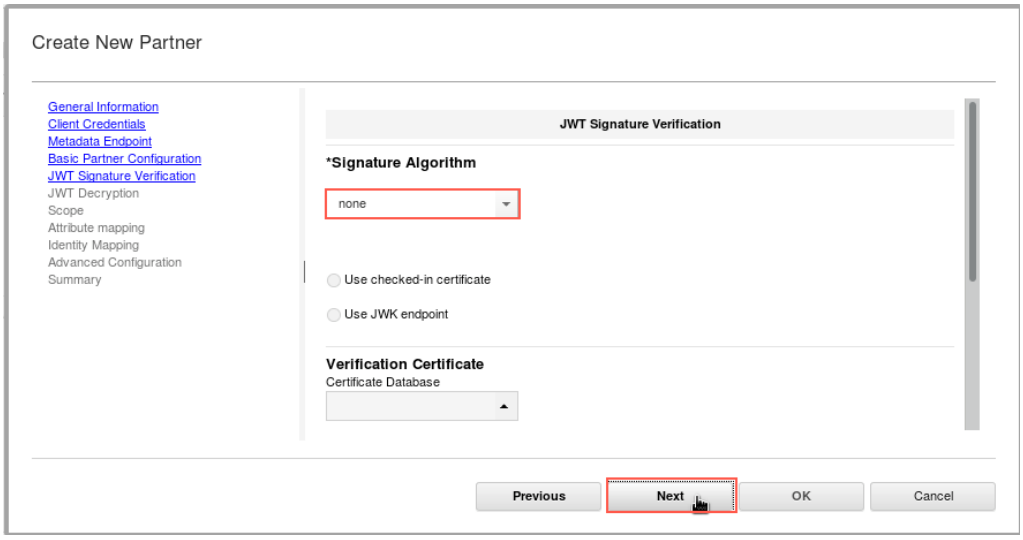
Select **code** under *Response Types*. This means we're going to do an OAuth 2.0 authorization code flow.

If you select *token* here instead, this will cause the OAuth 2.0 implicit flow to be used.



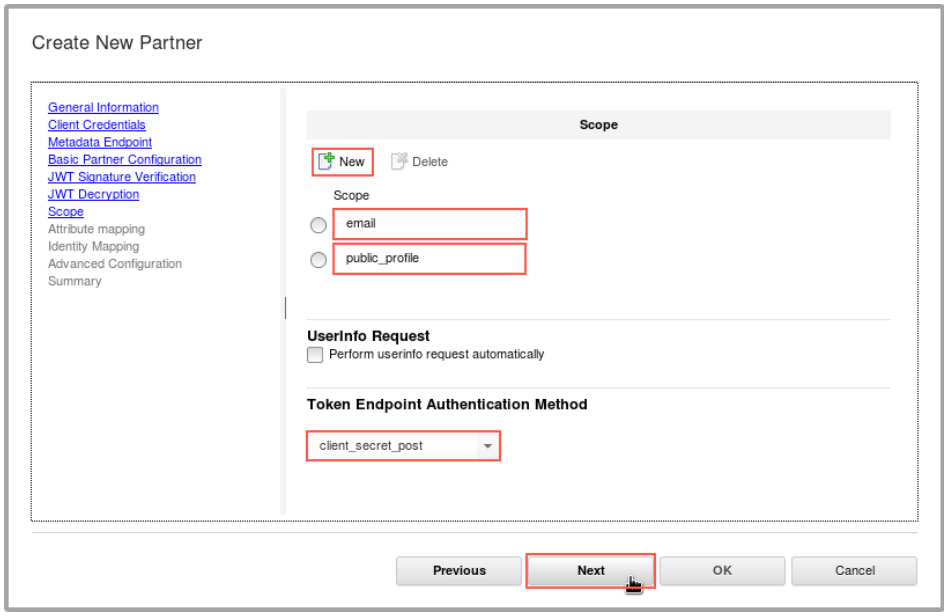
Enter **https://www.facebook.com/v3.0/dialog/oauth** as the *Authorization Endpoint.*
Enter **https://graph.facebook.com/v3.0/oauth/access_token** as the *Token Endpoint.*

Click **Next**.



We are not going to receive any identity token, so the signature algorithm is irrelevant.
Scroll to the top of the page, set *Signature Algorithm* to **none** using the drop-down list, and click **Next**.

We're not going to encrypt the token contents so just click **Next** again.

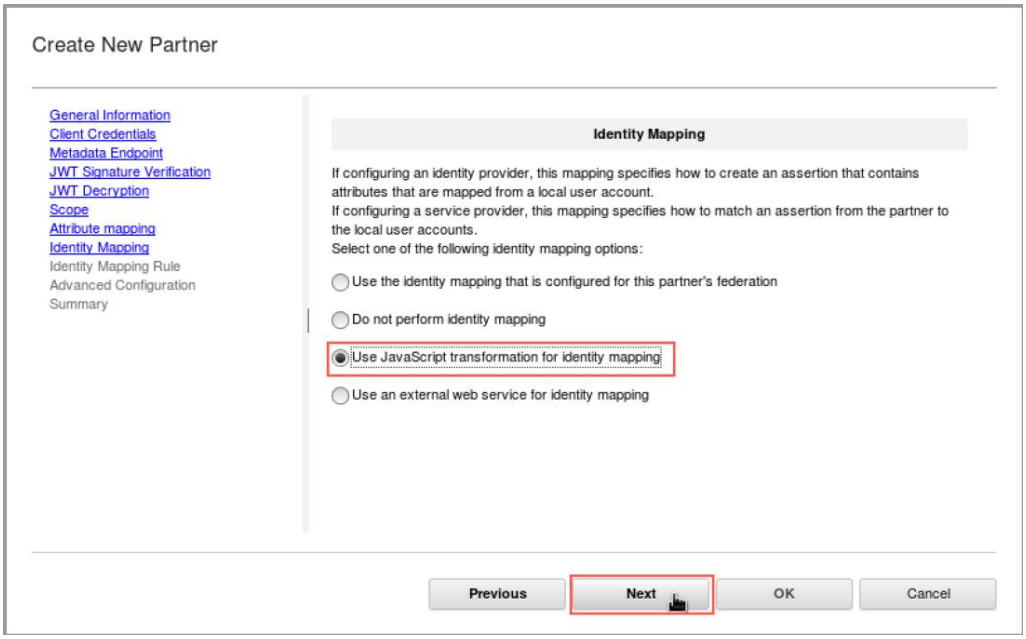


Facebook supports a lot of scopes but the ones we need are *email* and *public_profile*. Since Facebook is not an OIDC Provider, the *oidc* scope needs to be removed.

Replace *oidc* with **email**. Use the **New** button to add **public_profile**.

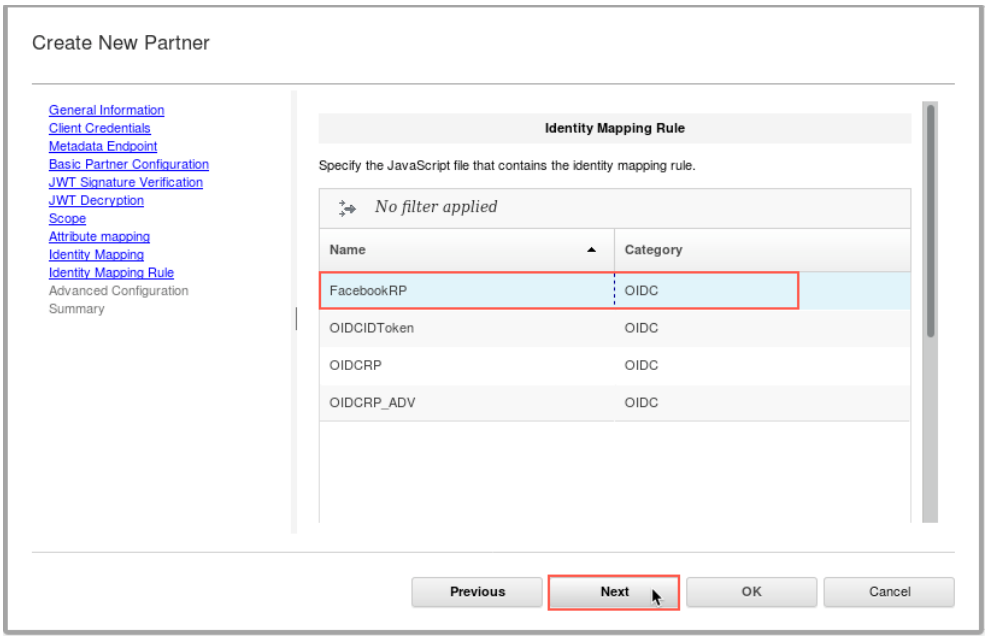
Facebook only supports the *secret post* method for authenticating to the Token Endpoint. Select **client_secret_post** from the drop-down list. Then click **Next**.

We're not going to add any attribute mapping so click **Next** again.

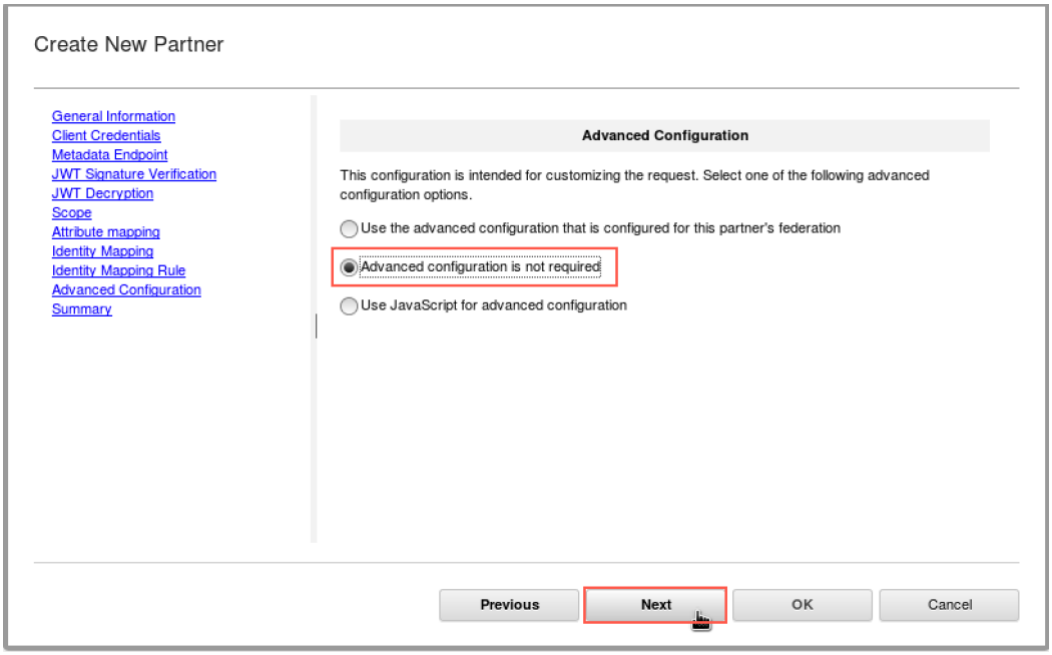


We need to specify an Identity Mapping (we didn't specify one at the federation level). We will use a built-in Javascript transformation.

Select the radio-button for **Use JavaScript transformation for identity mapping** and click **Next**.



Select the **FacebookRP** mapping rule and click **Next**.



We don't want to use and advanced configuration. Select the radio-button for **Advanced configuration is not required** and click **Next**.

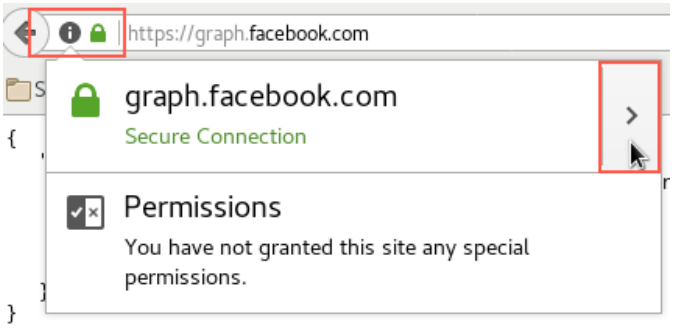
Click **OK** on the summary screen to create the partner definition.

The new partner is created. Click **Close** to close the partner overlay.

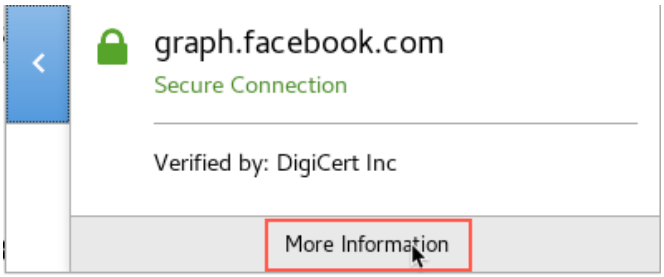
7.4 Get Facebook Root CA Certificate

In order to allow direct communication from the RP (Runtime container) to Facebook servers, the Root CA certificate used by the Facebook Web and API endpoints must be loaded into the key store of the RP Runtime. At the time of writing these both use the same root ceritificate. We will download this certificate and then import to Access Manager.

Open a browser and navigate to **https://graph.facebook.com**.

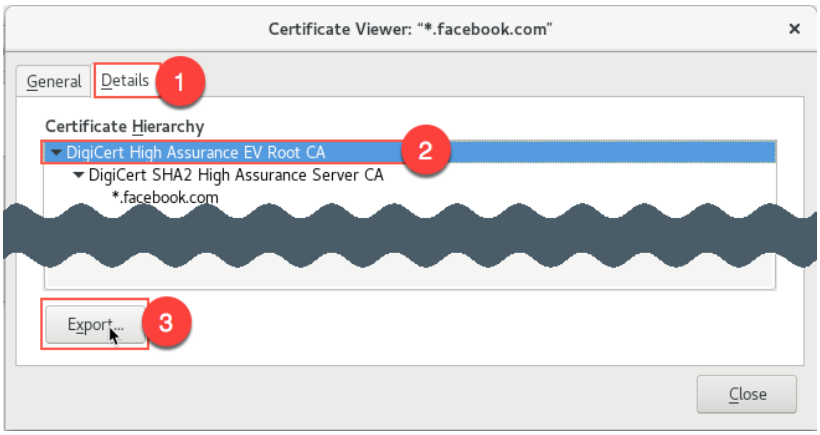


Click on the padlock icon next to the URL and then click the arrow in the pop-up window.



Click **More Information**.

Click **View Certificate**.



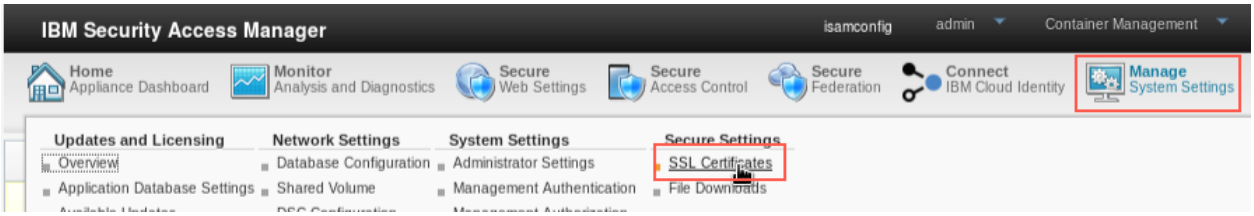
Select the **Details** tab.

Select the root CA (*DigiCert High Assurance EV Root* CA in this case) and click **Export…** button.

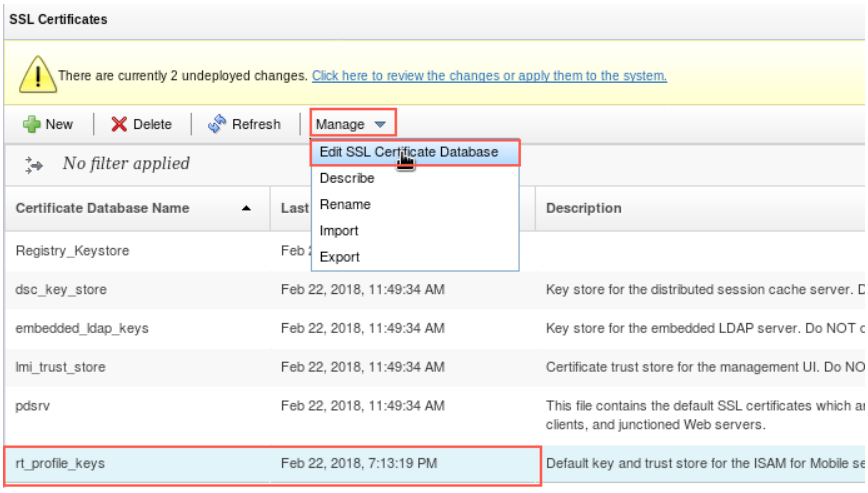
Save the certificate file onto the Desktop (or somewhere else you can easily find it).

7.5 Import Facebook CA Certificate to Runtime key store

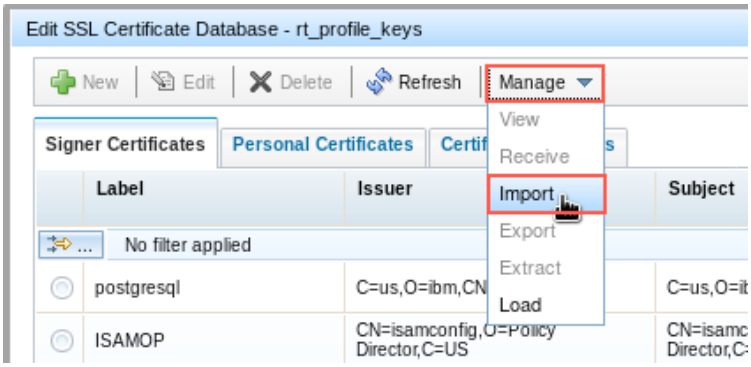
Return to the Access Manager LMI.



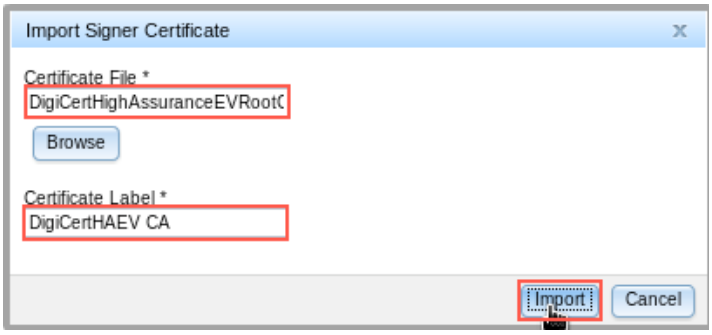
Navigate to **Manage System Settings→Secure Settings: SSL Certificates** in the mega-menu.



Select the **rt_profile_keys** key store. Click **Manage** and select **Edit SSL Certificate Database** from the pop-up menu.



Click **Manage** and select **Import** from the pop-up menu.



Click **Browse** and select the CA certificate you just exported. Enter a Label and click **Import**.

Click **Close** to close the Certificate Database overlay.

Deploy the changes using the link in the yellow warning message.

7.6 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Runtime container to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime container will detect this new snapshot (this is new capability in SAM 9.0.5.0) and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue this command to restart the runtime container:
> **`docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime`**

Configuration of Facebook as an Identity Provider is now complete.

7.7 Test Facebook Social Sign-On Flow

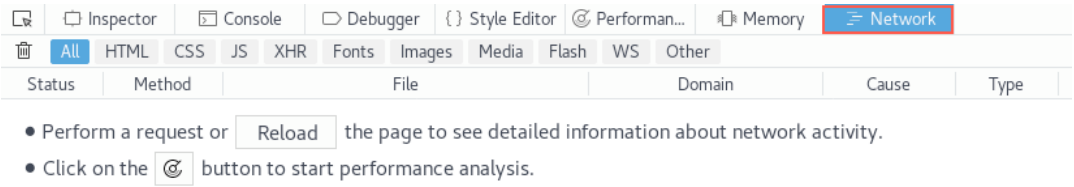
We can now test the Custom RP integration with Facebook.

In the Firefox browser in the Centos VM, navigate to a protected page on the Relying Party site: **https://www.iamlab.ibm.com/app/mobile-demo/diag**

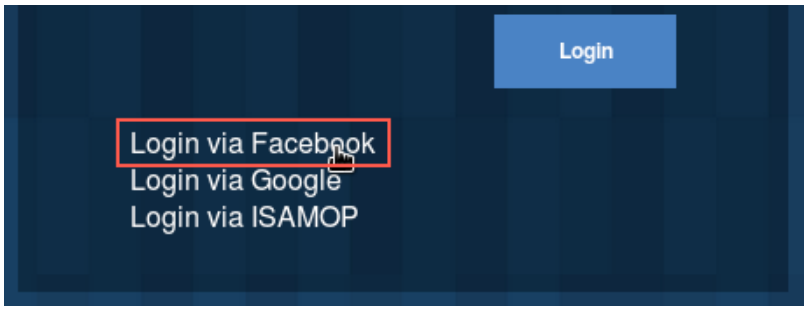
This is a protected page and so the login page of the Relying Party is displayed.

Before continuing, turn on the network trace in the Firefox browser so you can follow the OIDC flow.

Press **Ctrl-Shift-K** to open the network trace tool.

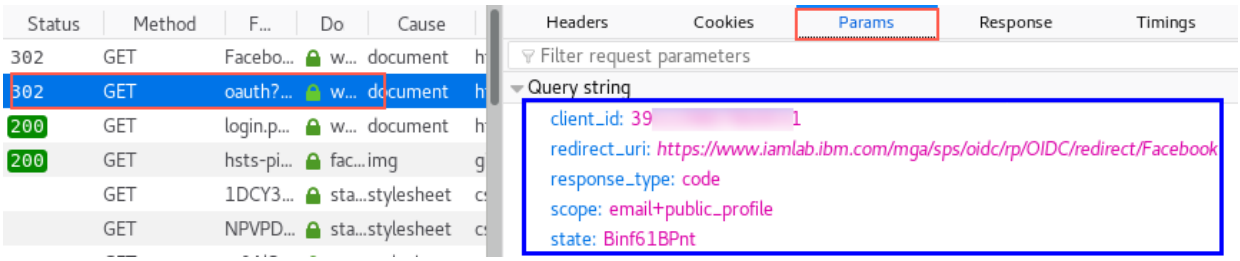


Look at the login page, it has been customized to include additional links to trigger Single Sign On:



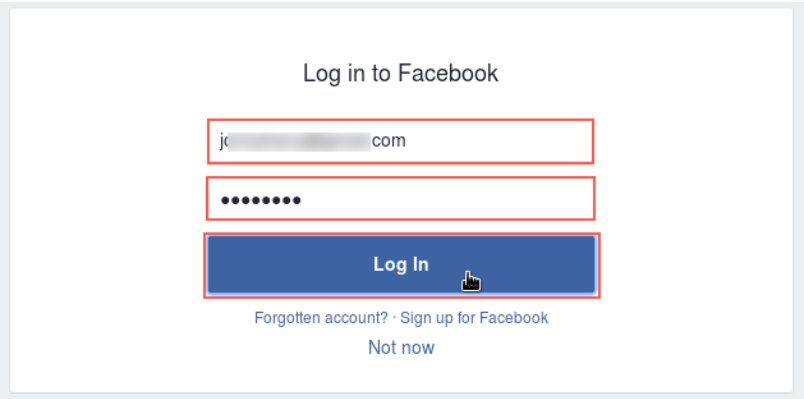
Click the **Login with Facebook** link.

In the network trace, you can see that the OIDC trigger link at the RP has created a redirect to the OP *authorize* endpoint:



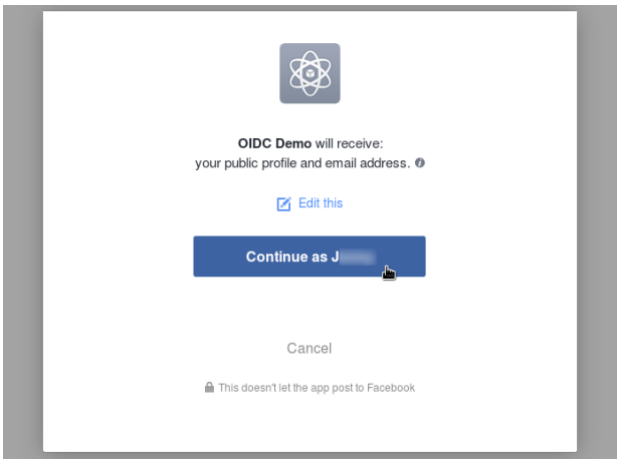
Select the **auth?...** request and click **Params** tab. You can now see the parameters sent in the query string. Notice the 2 scopes being requested (email and public_profile).

The main browser window is showing the Facebook login page:

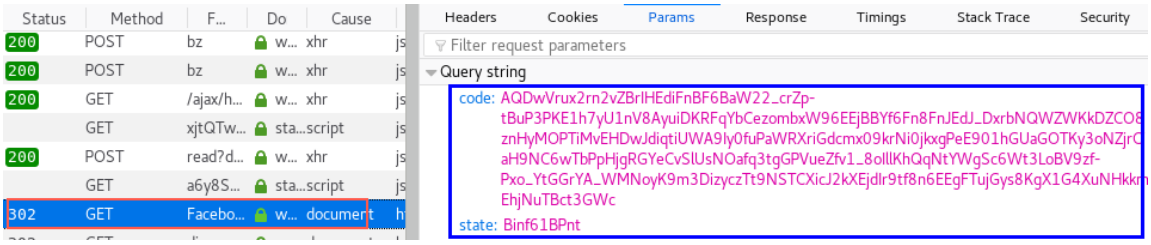


Login with your Facebook account; this might include a 2FA check and offer to remember your browser.

After login, the Facebook Consent page is shown. You can see that the client is requesting access to public profile and e-mail address:



Click **Contine as…** to consent. At this point the OIDC flow completes and you are shown the target page on the RP. Look at the trace again:



Select the GET to Facebook. This is the redirect from Facebook back to the RP redirect URL. This is an *Authorization Code* flow and so a *code* is returned via the browser. Access Manager will use this code to retrieve an Access Token (from the Facebook token endpoint) and then our custom RP Mapping Rule will use the Access Token to retrieve user information using the Facebook Graph API. These steps are performed with direct connections and so are not seen in this browser trace.

In the browser window, you are on a diagnostics page:

The header of the page shows the logged in user. The username is the e-mail address associated with the Facebook account. This username was specified in the *FacebookRP* mapping rule.

You can close the developer tools using the cross in the top-right corner of the developer tools section.

Further down the page, you can see the SAM Credential created at the RP. Review this if you like. You will see first_name, last_name, email, and FacebookID have been populated.

When you're done, click the **Logout** link at the top of the diagnostics page to log out from the RP.

8 Advanced Configuration and Access Policies

In this section we will explore the customizations available for both OIDC Provider (OP) and Relying Party (RP) by setting up a scenario that supports *Authentication-context Class Reference (ACR)*. This allows an RP to request a certain level of authentication from an OP. We will also set up our system so that the RP requests the *authenticationTypes* attribute from the OP so that it knows which authentication types the user has completed at the OP (and can use this information as part of access decisions).

This configuration relies on agreement of ACR values and attribute names to be exchanged by the participants. We'll set up the configuration between our own OP and RP, so we have full control of both sides.

We'll use TOTP as the higher level of authentication that the RP can request. The OP Reverse Proxy in the demo environment is already configured for AAC so is ready to support TOTP.

Once we have the RP configured with the ability to request TOTP authentication from the OP, we'll set up an AAC Authorization Policy and Obligation to require this for specific RP resources.

8.1 Use RP Advanced Configuration Script to modify OIDC Requests

When configuring an OIDC RP, you can specify an advanced configuration mapping to be run when building requests. This allows customization of requests to include additional parameters and claims. We will use this to add two things to requests to the OP */authorize* endpoint:

- An *acr_values* parameter specifying the authentication type required; and
- A *claims parameter* that requests that the OP return the authentication types the user has completed.

8.1.1 Examine Advanced Configuration Script

Let's take a look at the Advanced Configuration script that we'll use with this ACR use case.

It is better to use a text editor to view scripts (rather than the LMI) because it highlights JavaScript syntax making it easier to read. Open a terminal on the Centos VM and enter the following command:

```
[demouser@centos ~]$ gedit studentfiles/oidc/ACR-OIDC-RP-AdvConfig.js &
```

The file is opened in a new text editor window.

The first thing this script does is to retrieve the type of operation that is being processed. This is necessary because the same advanced configuration script is called during all RP operations. For this ACR use-case, we're interested in modifying the *authorize* request and we check for this as follows:

```
// Get the operation
var operation =
stsuu.getContextAttributes().getAttributeValueByNameAndType("operation","urn:ibm:SAM:oidc:
rp:operation");

if(operation == "authorize") {
```

The authentication level required will be passed in the OIDC trigger using the *level* query-string parameter:

```
//Get level parameter from query-string of the kickoff URL
  var myLevels = stsuu.getContextAttributes().getAttributeValuesByName("level");
```

If *level=2* the this means we will request TOTP authentication at the OP. Otherwise we will request password authentication. In this demo, authentication levels are specified using these URNs:

```
//Send ACR of password...
    var myAcr = "urn:demo:password";

    //...Unless level is 2, in which case send ACR of totp
    if (myLevel == "2") {
      myAcr = "urn:demo:totp";
```

The ACR is sent as a request attribute called *acr_values*:

```
stsuu.getContextAttributes().setAttribute(new Attribute("acr_values",
"urn:ibm:SAM:oidc:rp:authorize:req:param", myAcr));
```

In addition to requesting a particular authentication level with an ACR, the request will also include a *claims* parameter which will request that the *authenticationTypes* attribute at the OP be returned in the identity token it generates:

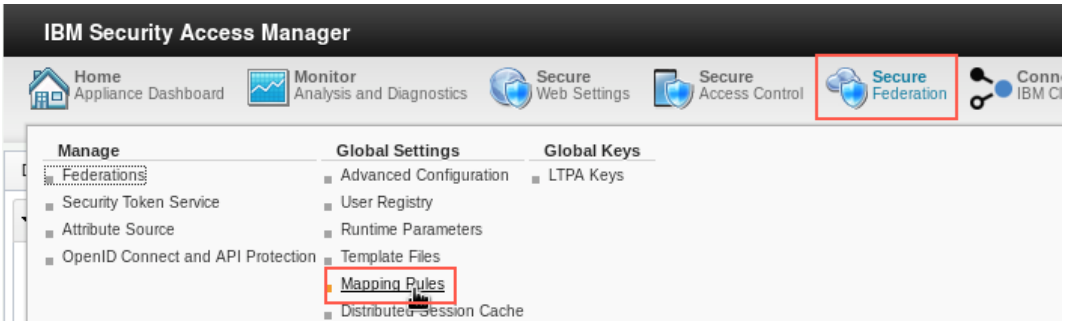
```
// Request that authenticationTypes be returned by the OP (so we will know what
authentication user has done)
var claims = { "id_token": {
    "authenticationTypes": {"essential": false}
  }
};
stsuu.addContextAttribute(new Attribute("claims",
"urn:ibm:SAM:oidc:rp:authorize:req:param", JSON.stringify(claims)));
```

This is sent using JSON format that is specified in the OIDC spec. The *JSON.stringify()* function to convert the JSON to a string for inclusion in the message.

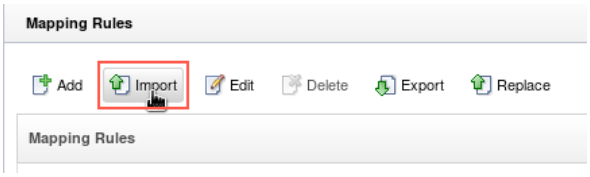
8.1.2 Import Advanced Configuration Script

Open the Firefox browser in the Centos VM and navigate to the Access Manger LMI: **https://isam.iamlab.ibm.com**

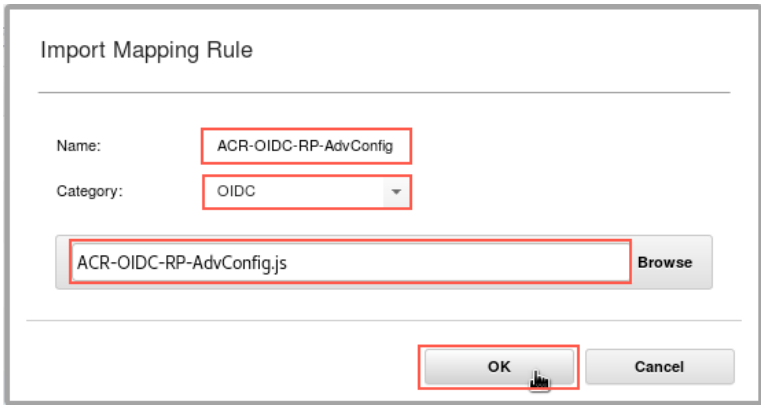
Login with **admin** and **Passw0rd**.



Navigate to **Secure Federation→Global Settings: Mapping Rules** in the mega-menu.



Click **Import**.



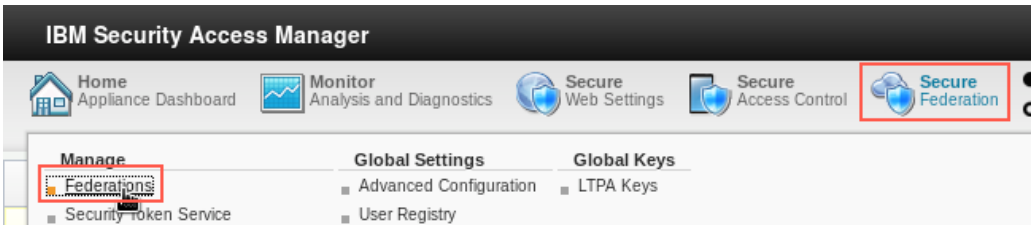
Enter **ACR-OIDC-RP-AdvConfig** as the *Name* and **OIDC** as the *Category*.

Click **Browse** and select file **/home/demouser/studentfiles/oidc/ACR-OIDC-RP-AdvConfig.js**.

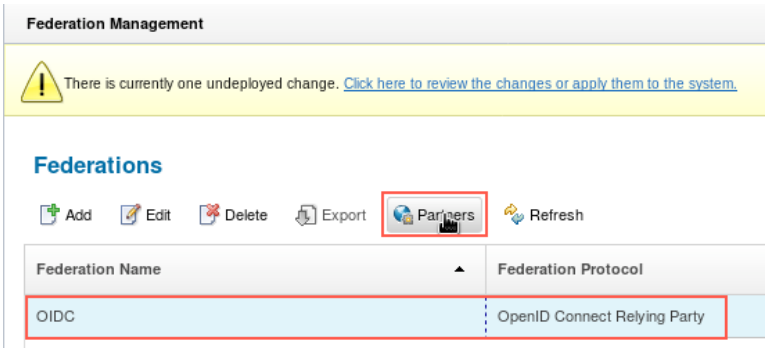
Click **OK**.

8.1.3 Update Relying Party Configuration

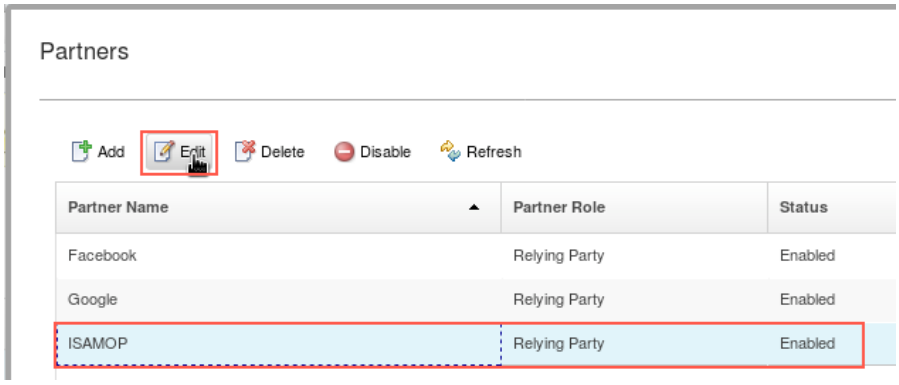
We will now update the existing Relying Party configuration. We will make the changes in the partner definition for the Access Manger OP so that it doesn't affect the other configured connections.



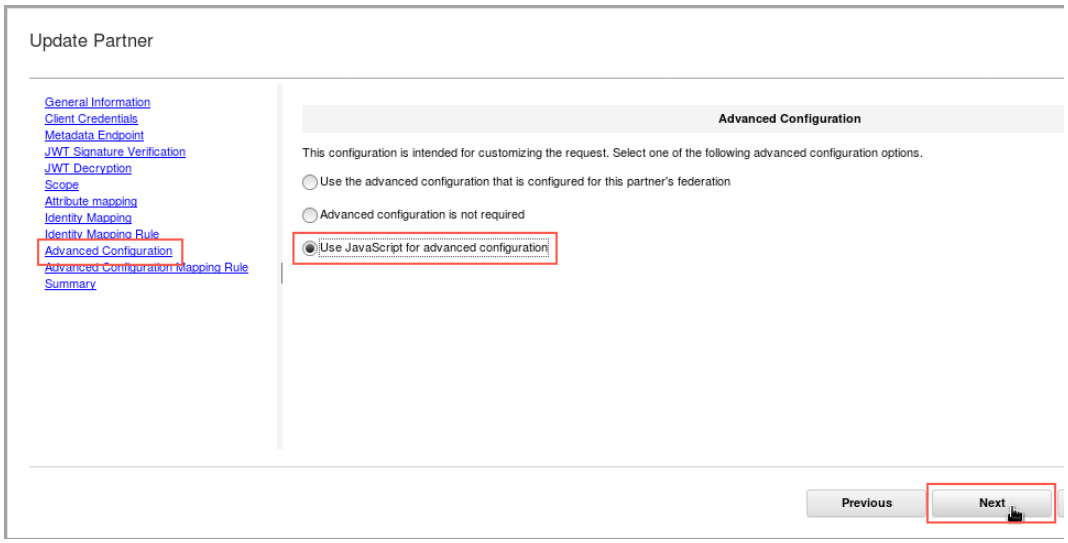
Navigate to **Secure Federation→Manage: Federations** in the mega-menu.



Select the **OIDC** federation and click **Partners**.



Select the **ISAMOP** partner and click **Edit**. The configuration wizard is opened.



Click on the **Advanced Configuration** link in the wizard.
Select radio-button for **Use JavaScript for advanced configuration** and click **Next**.

Select the **ACR-OIDC-RP-AdvConfig** file and then click **Next**.
Click **OK** on the summary page to compete the changes and click **Close** to close the *Partners* overlay.

8.2 Use OP Access Policy to select authentication based on acr_values

The Advanced Configuration at the RP mean that the OP will receive an *acr_values* attribute in OIDC requests to */authorize*. We will now use an Access Policy to read this attribute and trigger TOTP authentication if it has been requested by the RP.

8.2.1 Examine Access Policy Script

Enter the following command in a terminal window:

```
[demouser@centos ~]$ gedit studentfiles/oidc/ACR-OIDC-OP-AccessPolicy.js &
```

The file is opened in a new text editor window.

The first thing this script does is read the content of the *Authentication Class Reference (ACR)* attribute received in the OIDC request. This requires accessing several objects:

```
var protocolContext = context.getProtocolContext();
var authContext = protocolContext.getAuthenticationRequest().getAuthenticationContext();
var acrList = authContext.getAuthenticationClassReference();
```

If the ACR exists, the script checks to see if the list of requested authentication classes contains the value *urn:demo:totp*:

```
if ((acrList != null) && (acrList.contains("urn:demo:totp"))) {
```

If it does, the script then checks to see if the user has already completed a TOTP authentication. To find out, the script reads the *authenticationTypes* attribute from the user object:

```
  var user = context.getUser();
  var authenticationTypesAttribute = user.getAttribute("authenticationTypes");
```

If the attribute exists, the script looks for the authentication policy ID of the built-in TOTP authentication policy. It does two checks because OP attribute format changes based on multi-value attribute behavior:

```
    // Check for TOTP when OP using single comma-separated attribute
    if (authenticationTypes.get(0).contains("urn:ibm:security:authentication:asf:totp")) {
        totpDone = true;
    }

    // Check for TOTP when OP using multi-valued attributes
    if (authenticationTypes.contains("urn:ibm:security:authentication:asf:totp")) {
        totpDone = true;
    }
```

If TOTP already done, the access policy can return an *Allow* decision.

```
    if (totpDone) {
       IDMappingExtUtils.traceString("TOTP Already done");
       context.setDecision(Decision.allow());
```

If not, (or if *authenticationTypes* attribute didn't exist), the script returns a *Challenge* decision to trigger TOTP. The challenge is a redirect to the AAC Authentication Service. Note the use of the *@ACTION@* macro in the target query-string to re-enter the OIDC flow after authentication is complete.

```
        var handler = new RedirectChallengeDecisionHandler();
        IDMappingExtUtils.traceString("CHALLENGE WITH TOTP");
handler.setRedirectUri("/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:totp&Tar
get=https://www.op.ibm.com/mga@ACTION@");
        context.setDecision(Decision.challenge(handler));
```

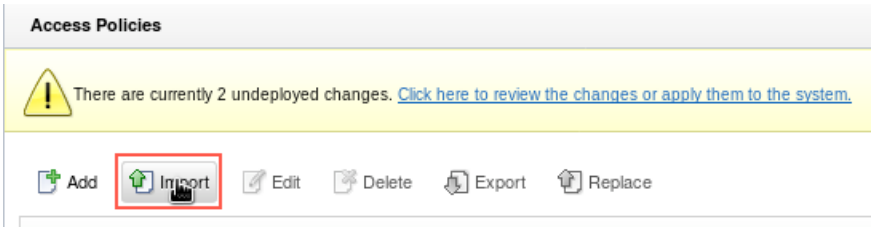
If the ACR wasn't received (or if it doesn't include the TOTP request) the Access Policy simply returns the Allow decision:

```
       IDMappingExtUtils.traceString("ACR doesn't request TOTP");
       context.setDecision(Decision.allow());
```

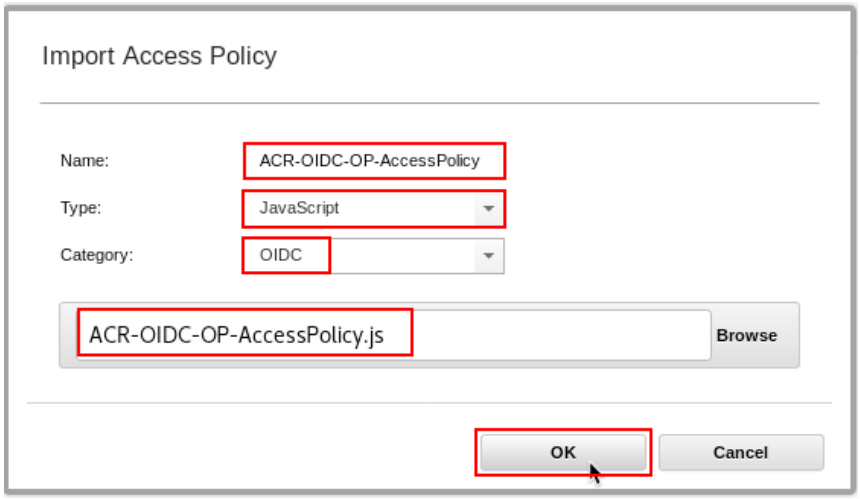
8.2.2 Import Access Policy

Return to the Access Manager LMI.

Navigate to **Secure Federation→Global Settings: Access Policies** in the mega-menu.



Click **Import**.



Enter **ACR-OIDC-OP-AccessPolicy** as the *Name* and **JavaScript** as the *Type*.

Provide a *Category* (you have to type it). **OIDC** seems sensible.

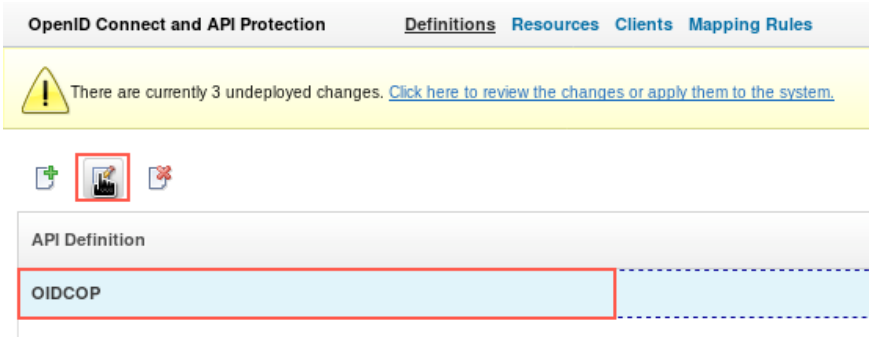
Click **Browse** and select file **/home/demouser/studentfiles/oidc/ACR-OIDC-OP-AccessPolicy.js**.

Click **OK**.

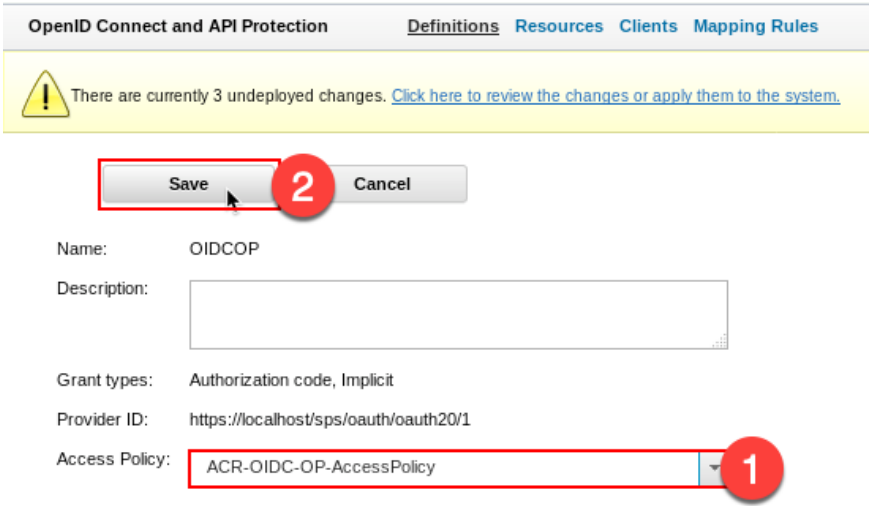
8.2.3 Update OIDC Provider Definition

An Access Policy can be specified on an OIDC/OAuth 2.0 Provider definition. The Policy will be invoked whenever the */authorize* endpoint is accessed. We will now add the policy we just imported to our OP definition.

Navigate to **Secure Federation→Manage: OpenID Connect and API Protection** in the mega-menu.



Select the **OIDCOP** definition and click the **Edit** icon.



Select the **ACR-OIDC-OP-AccessPolicy** from the *Access Policy* drop-down list.

Click **Save**.

8.3 Deploy Changes and Test

Deploy changes using the link in the yellow warning message.

8.3.1 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Runtime container to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime container will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue this command to restart the runtime container:
> **`docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime`**

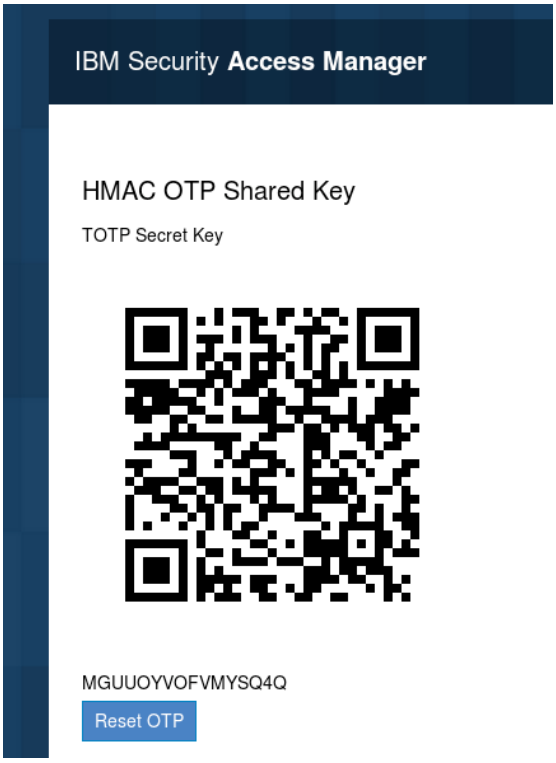
8.3.2 Register TOTP client at OP

In order to allow our test user, Emily, to complete a TOTP challenge, she must register a TOTP client against her Access Manager account at the OP.

> You will need a TOTP client. If you have the IBM Verify app on iOS or Android, this can be used. If you are using another OS, you can also use the Google Authenticator app (or any other app that supports the TOTP standard.

In the Firefox Browser on the Centos VM, navigate the OTP registration URL at the OP:
https://www.op.ibm.com/mga/sps/mga/user/mgmt/html/otp/otp.html

If necessary, login with **emily** and **Passw0rd**. You will see the registration screen:



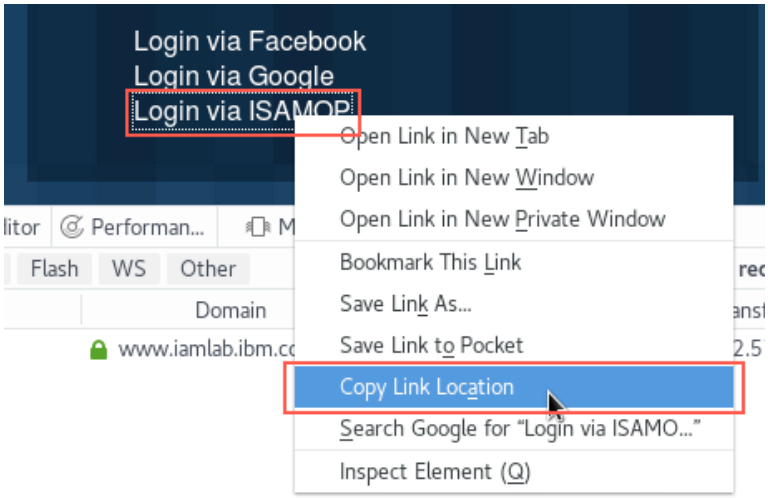
Tell your TOTP application that you want to add an account and then use the device camera to scan the TOTP Secret Key QRCode. If you can't use the camera, you can register with the code instead.

8.3.3 Manually Trigger OIDC with request for TOTP

We will now manually trigger an OIDC flow at the RP. We will include the *level=2* query-string in the trigger URL which will tell the RP to include *"urn:demo:totp"* in the *acr_values* sent to the OP.

In the browser, navigate to: **https://www.iamlab.ibm.com/app/mobile-demo/diag**
You are presented with the RP login page.

We will trace the flow. Press **Ctrl-Shift-K** in the Firefox browser to open the network trace.

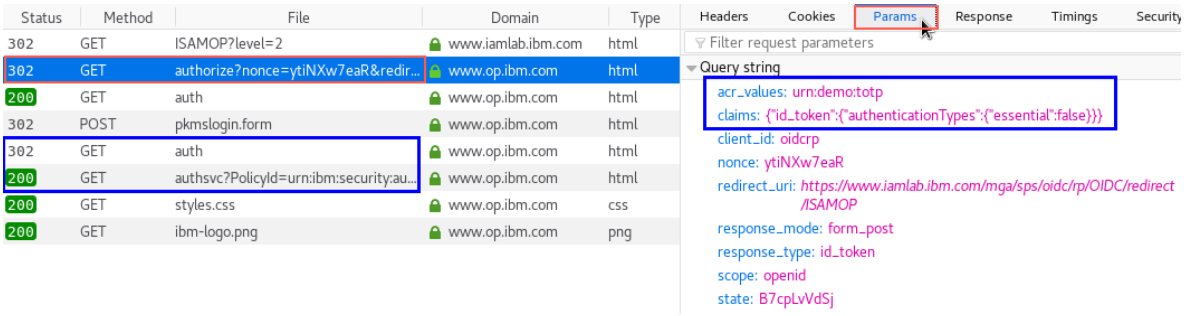


Right-Click on the **Login via ISAMOP** link and click **Copy Link Location**.

Paste this copied URL to the location bar and then edit to add **?level=2** to the end. The full URL should be as follows: **https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/kickoff/ISAMOP?level=2**

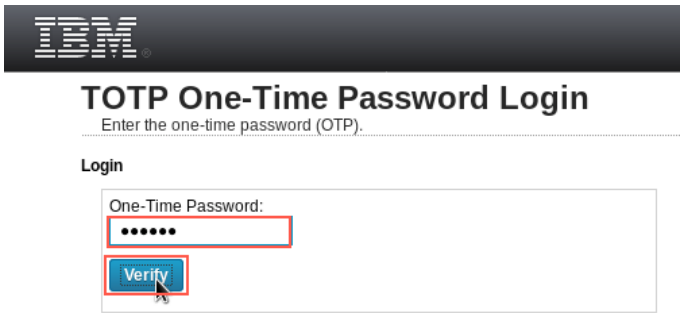
You are redirected to the OP. If you are using a clean browser session you will be on a login page. If so, login with **Emily** and **Passw0rd**. You should now be at a TOTP challenge page.

Before continuing, let's look at the OIDC request that the RP sent:



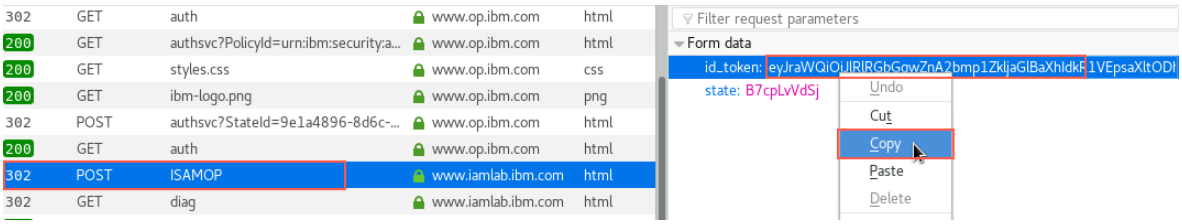
In the network trace, select the request to **authorize?nonce=....** and then select **Params** tab.
You can see the **claims** and **acr_values** parameters are being sent to the OP.

Later in the trace, you'll see a request to *auth* which redirects to *authsvc*. This is where the Access Policy ran and triggered the redirect to the authentication service to perform TOTP authentication.



Use your registered TOTP client to get the current TOTP code. Enter it and click **Verify**.

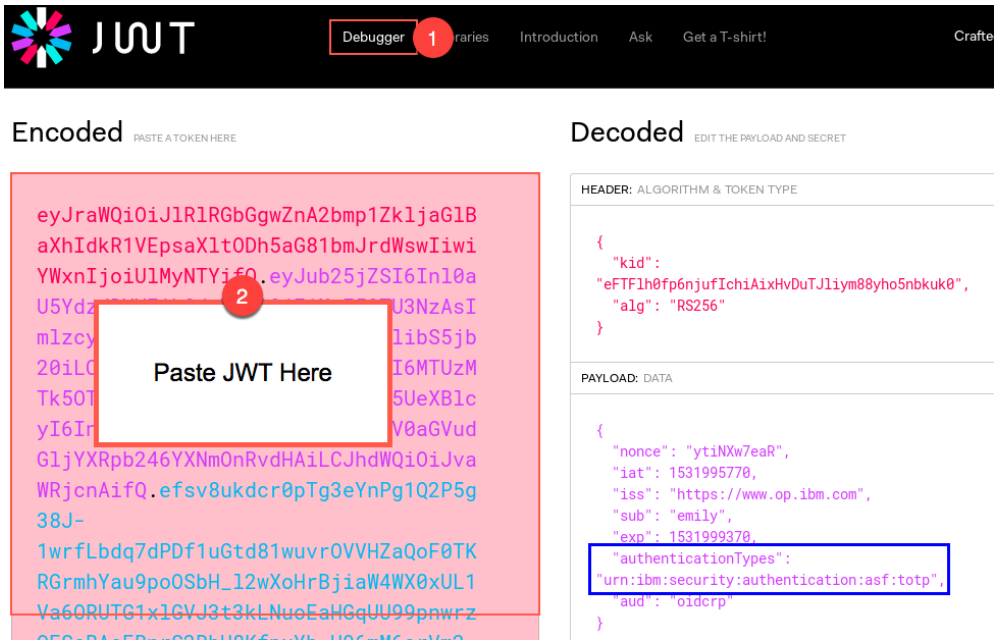
At this point, the OIDC flow completes and you are taken to the diagnostics page at the RP.
If you want to see the ID Token sent by the OP, you can extract it from the Implicit response and decode it.
Let's do that now.



Select the POST to **ISAMOP** in the network trace.

Click on the **id_token** parameter. The screen changes to a single line view of the form data.
Right-click the id_token data and select **Select All** from the pop-up menu.
Right-click the id_token data again and select **Copy** from the pop-up menu.

Open a new browser tab and navigate to: **https://jwt.io**. This site has a good JSON Web Token decoder which we can use.



Select **Debugger** from the title bar and then paste the copied ID Token into the left-hand panel.

The decoded token is shown on the right. You can see the *authenticationTypes* attribute coming from the OP.

The first part of the scenario is complete; we can successfully trigger TOTP at the OP and get back the *authenticationTypes* in the id_token. We now need to configure the RP to map it into the local user credential.

> In SAM 9.0.4.0 the *authenticationTypes* attribute will not be seen in the response. Returning customized id_tokens must be enabled by editing the preTokenMappingRule and setting:
> **var customize_id_token = true;**

8.4 RP Mapping Rule to receive authenticationTypes from OP

We will now configure the RP so that it will populate the *authenticationTypes* attribute returned by the OP into the credential of the local user. This will allow RP policies have visibility of the authentication performed at the OP.

Mapping attributes from the received identity token into the local credential (and setting the local user name) is performed in the RP mapping rule. A new rule is provided which is based on the out-of-the-box *OIDCRP* rule. Only a few small changes are required. One so that it will process the *authenticationTypes* attribute and one to change the format of the local username.

8.4.1 Examine Updated Mapping Rule

Enter the following command in a terminal window:

```
[demouser@centos ~]$ gedit studentfiles/oidc/ACR-OIDC-RP-MappingRule.js &
```

The file is opened in a new text editor window.

The mapping rule reads standard *iss (issuer)* and *sub (subject)* attributes from the incoming token:

```
var iss = stsuu.getAttributeContainer().getAttributeValueByName("iss");
var sub = stsuu.getAttributeContainer().getAttributeValueByName("sub");
```

It then builds the local user name (*principalName*) using just the *sub* attribute (the out-of-the-box rule uses both the *iss* and *sub*). The username will have the format **OIDC/*<OP User>***

```
stsuu.setPrincipalName("OIDC/" + sub);
```

The rest of the mapping rule effectively filters the attributes in the incoming token so that only a specified set are written into the local user credential. This list is specified in the *attrNames* array which has been modified to include the *authenticationTypes* attribute:

```
var attrNames = [
       // authenticationTypes added to the received attribute to be added to credential
       "given_name",
       "family_name",
       "name",
       "email",
       "access_token",
       "authenticationTypes"
];
```

The code reads specified attributes from the incoming *stsuu* (*STS Universal User*) object and stores them in a temporary array called *finalAttrs*:

```
var finalAttrs = [];

for (var i = 0; i < attrNames.length; i++) {
       var attr = stsuu.getAttributeContainer().getAttributeByName(attrNames[i]);
       if (attr != null) {
              finalAttrs.push(attr);
```

It then clears out the attributes from the *stsuu*:

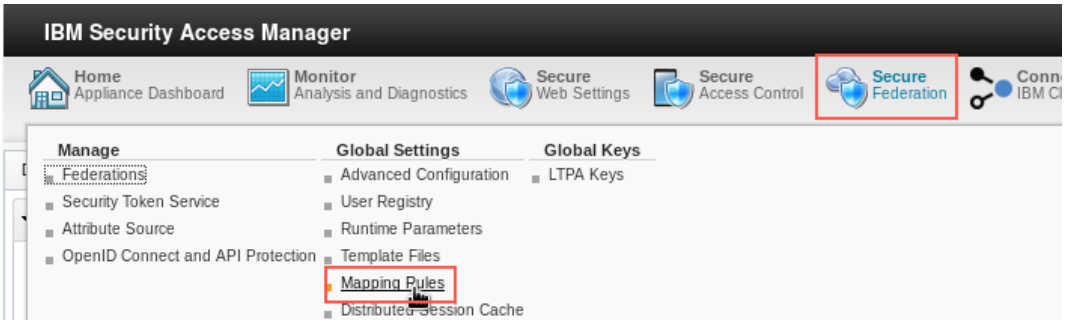
```
stsuu.clearAttributeList();
```

and then re-populates it from *finalAttrs*.

```
for (var i = 0; i < finalAttrs.length; i++) {
       stsuu.addAttribute(finalAttrs[i]);
}
```

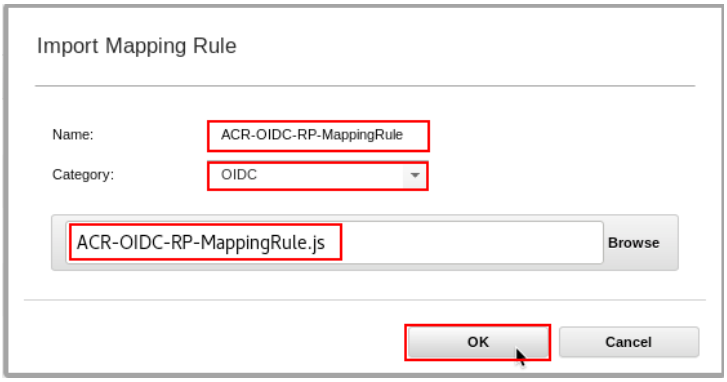
8.4.2 Import Update Mapping Rule

Return to the Access Manager LMI.



Navigate to **Secure Federations→Global Settings: Mapping Rules** in the mega-menu.

Click **Import**.



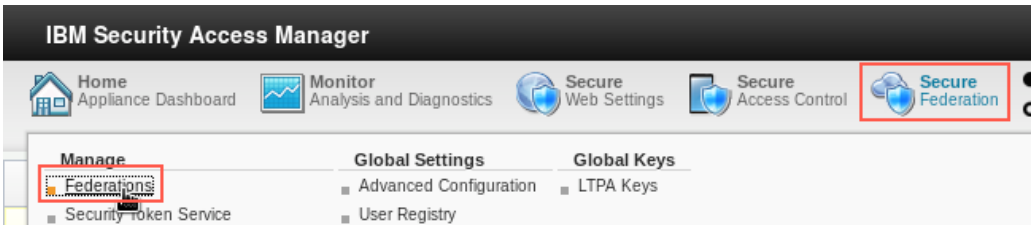
Enter **ACR-OIDC-RP-MappingRule** as the *Name* and **OIDC** as the *Category*.

Click **Browse** and select file **/home/demouser/studentfiles/oidc/ACR-OIDC-RP-MappingRule.js**.

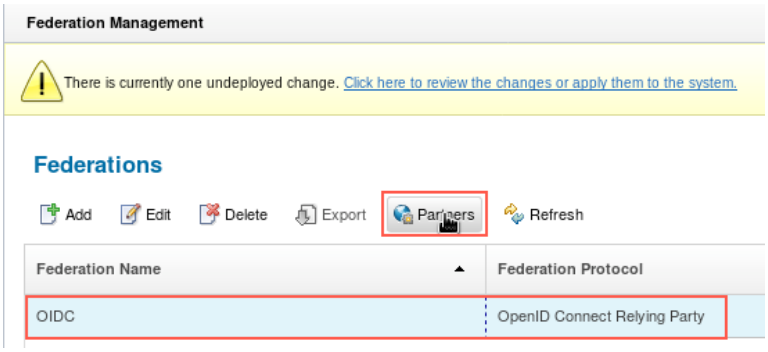
Click **OK**.

8.5 Update Relying Party Configuration

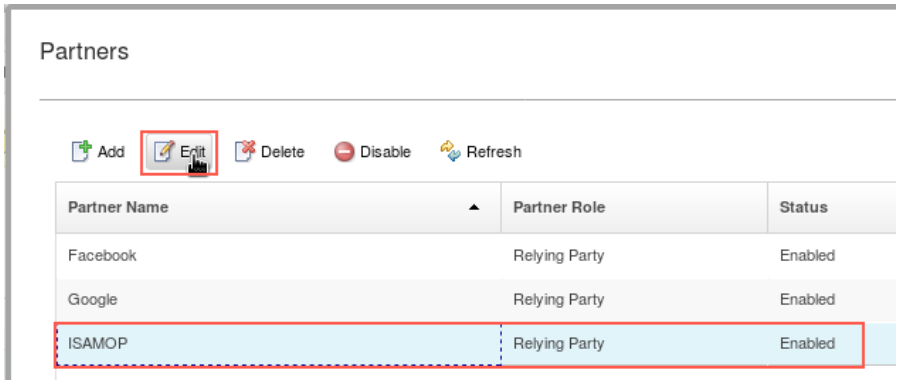
We will now update the existing Relying Party configuration. We will make the changes in the partner definition for the Access Manger OP so that it doesn't affect the other configured connections.



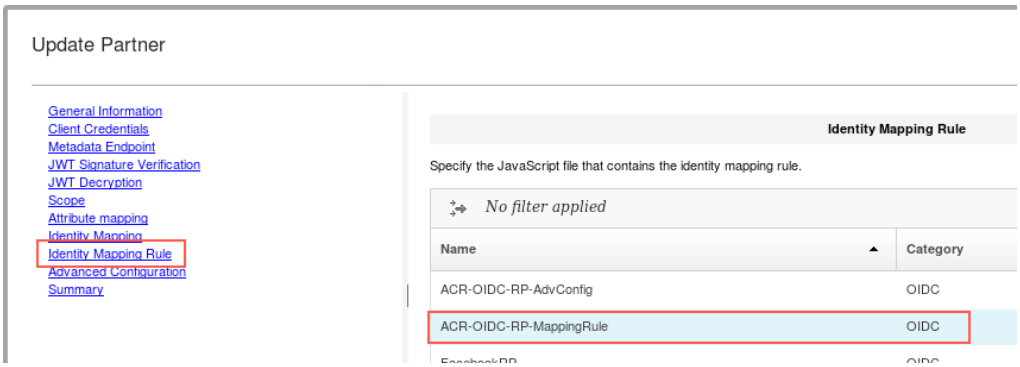
Navigate to **Secure Federation→Manage: Federations** in the mega-menu.



Select the **OIDC** federation and click **Partners**.



Select the **ISAMOP** partner and click **Edit**. The configuration wizard is opened.



Click on the **Identity Mapping Rule** link and then select **ACR-OIDC-RP-MappingRule** from the list.

Click the **Summary** link and then click **OK** to save the updated partner configuration.

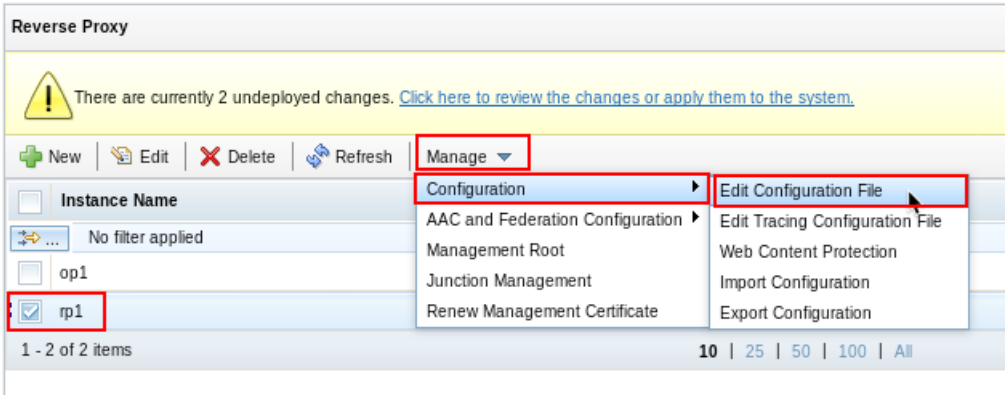
Click **Close** to close the *Partners* overlay.

8.6 Configure multi-value attribute handling at RP

By default, when an Access Manager Reverse Proxy receives multi-valued attributes using the Local User or External User Point-of-Contact profiles, it populates these into a single comma-separated attribute value in the credential. This isn't easy to work with when writing AAC Advanced Authorization policies, because the *contains* function doesn't support this.

We will now change the configuration of the RP Reverse Proxy so it will populate comma-separated attribute values as multi-valued attributes in the user credential.

Navigate to **Secure Web Settings→Manage: Reverse Proxy** in the mega-menu.



Select the check-box for the **rp1** instance. Click **Manage** and then select **Configuration→Edit Configuration File** from the pop-up menus.

Press **Ctrl-f** to open the browser search bar. Enter **create-multi** in the search to locate the configuration entry that needs to be changed.

Update the configuration as follows:

```
# The following configuration entry is used to determine whether multiple
# extended attribute headers of the same name are added to the credential as
# a multi-valued attribute, or a single comma-delimited attribute.
eai-create-multi-valued-attributes = yes
```

Click **Save** to save the updated configuration.

8.7 Deploy Changes and Test

Deploy changes using the link in the yellow warning message.

8.7.1 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Runtime and Reverse Proxy containers to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime and reverse proxy containers will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

> If you're using your own environment without the auto-reload feature enabled, issue these commands to restart the necessary runtime and reverse proxy containers:
> **`docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime`**
> **`docker exec -ti -- iamlab_isamwrprp1_1 isamcli -c reload all`**

8.7.2 Manually Trigger OIDC with request for TOTP

We will now manually trigger an OIDC flow at the RP. We will include the *level=2* query-string in the trigger URL which will tell the RP to include *"urn:demo:totp"* in the *acr_values* sent to the OP.

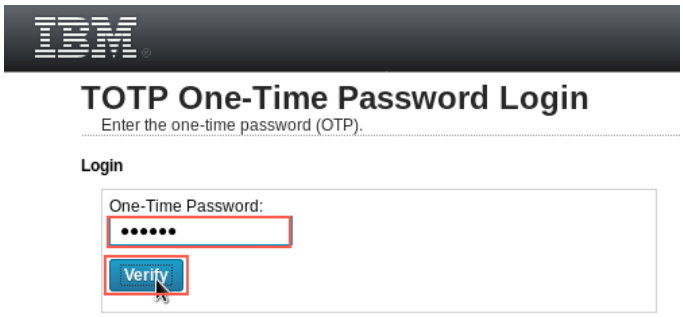
In the browser, navigate to: **https://www.iamlab.ibm.com/app/mobile-demo/diag**
You are presented with the RP login page.

We will trace the flow. Press **Ctrl-Shift-K** in the Firefox browser to open the network trace.

Right-Click on the **Login via ISAMOP** link and click **Copy Link Location**.

Paste this copied URL to the location bar and then edit to add **?level=2** to the end. The full URL should be as follows: **https://www.iamlab.ibm.com/mga/sps/oidc/rp/OIDC/kickoff/ISAMOP?level=2**

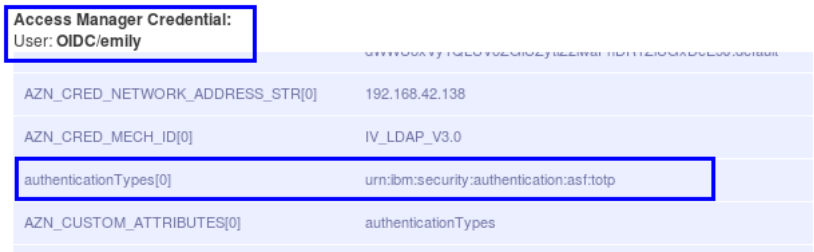
You are redirected to the OP. If you are using a clean browser session you will be on a login page. If so, login with **Emily** and **Passw0rd**. You should now be at a TOTP challenge page.



Use your registered TOTP client to get the current TOTP code. Enter it and click **Verify**.

At this point, the OIDC flow completes and you are taken to the diagnostics page at the RP.

Scroll down the page until you find the Access Manager credential information:



You can see that User is populated as **OIDC/emily**. This was set by the updated RP mapping rule. You can also see that the *authenticationTypes* attribute has been set based on the attribute sent by the OP.

8.8 Set up an Authorization Policy to Trigger TOTP at OP

At this point, our Relying Party has the ability to request TOTP from the OIDC Provider, and has visibility of the authentication methods performed at the OP. We will now create an Advanced Authorization Policy at the Relying Party to tie everything together.

8.8.1 Define an Obligation

In order for an AAC Authorization Policy to trigger an authentication process, we need to define an Obligation. This can then be used in Policies to indicate the need to perform TOTP at the OP.

Open the Access Manager LMI.

Navigate to **Secure Access Control→Policy: Obligations** in the mega-menu.

Click the **Add** button and select **Enforcement Point** from the drop-down menu.

Enter **Trigger TOTP at SAM OP** as the *Name*. This name will appear in the policy editor.

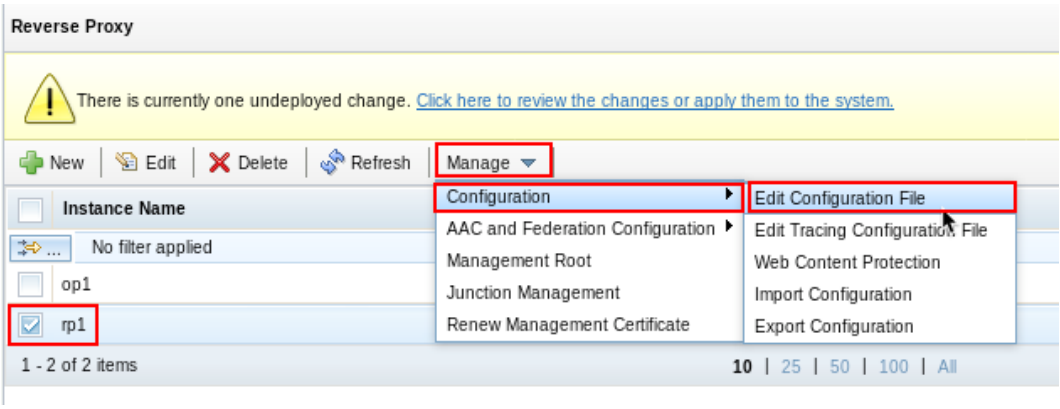
Enter **urn:demo:oidc:totp** as the *Identifier*. We will need to define this in the RP Reverse Proxy.

Click **Save**.

8.9 Define Obligation redirect in Reverse Proxy

We will now add the obligation to the Reverse Proxy configuration. This tells the Reverse Proxy where to redirect to when the obligation ID is received in an authorization decision.

Navigate to **Secure Web Settings→Manage: Reverse Proxy** in the mega-menu.



Select the check-box for the **rp1** instance. Click **Manage** and then select **Configuration→Edit Configuration File** from the pop-up menus.

Press **Ctrl-f** to open the browser search bar. Enter **obligation1** in the search to locate the configuration entry that needs to be changed.

Add an obligation URL mapping as shown. This will trigger the OIDC flow:

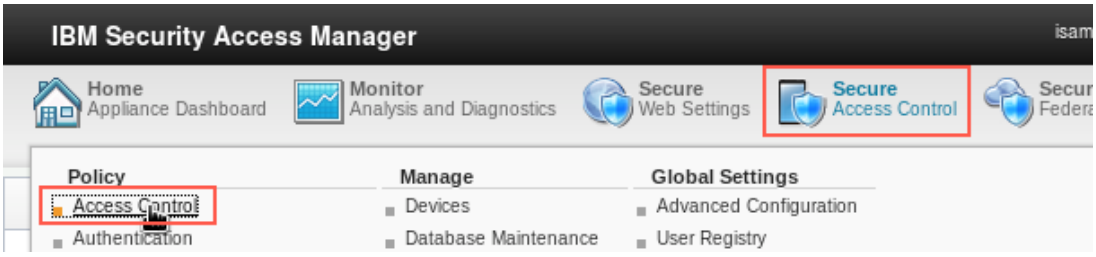
```
[obligations-urls-mapping]
…
urn:ibm:security:authentication:asf:email = /mga/sps/authsvc
# obligation1 = https://example.com/FIM/sps/xauth?AuthenticationLevel=1
urn:demo:oidc:totp = /mga/sps/oidc/rp/OIDC/kickoff/ISAMOP?level=2
```

Click **Save** to save the updated configuration.

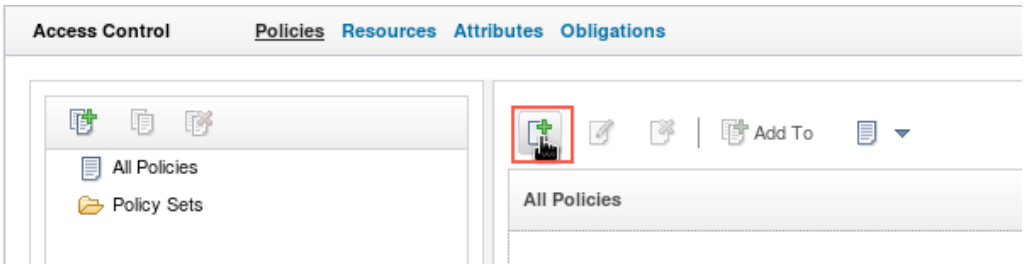
Deploy changes using the link in the yellow warning message. We need to do this in order to have the obligation registered in the configuration container so we can use it in Policies. No need to publish or reload at this point.

8.9.1 Create an Authorization Policy

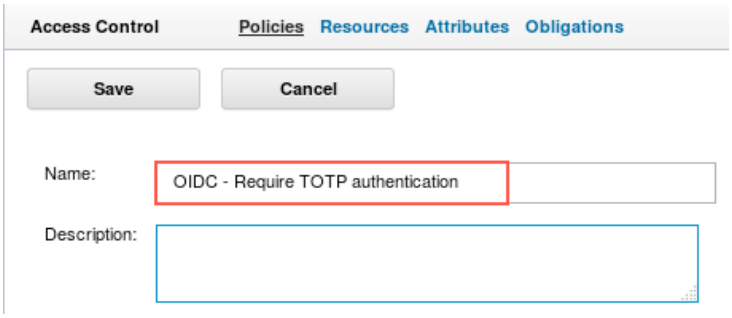
We will now create our Authorization Policy. This will be a simple policy that checks if the user has performed TOTP and, if not, returns an obligation to trigger TOTP at the OP.



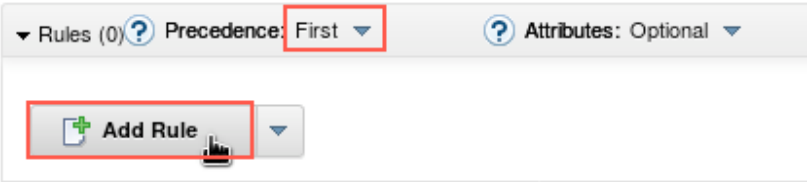
Navigate to **Secure Access Control→Policy: Access Control** in the mega-menu.



Click **Create** button to add a new Policy.

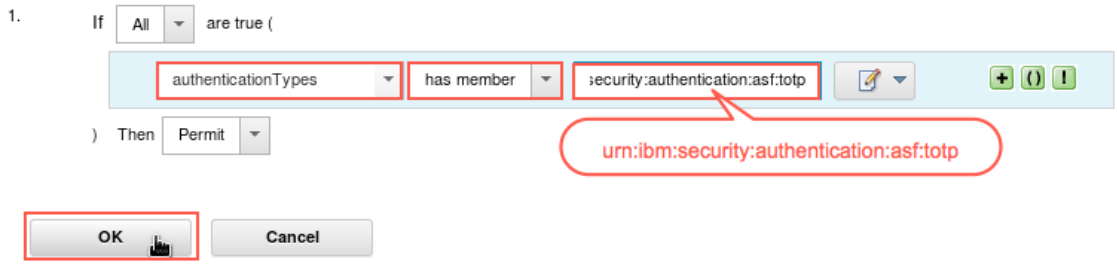


Set the Name to **OIDC - Require TOTP authentication**.



Set the *Precedence* to **First**. This means the first rule that returns a decision will be used.

Click **Add Rule** to add the first rule.

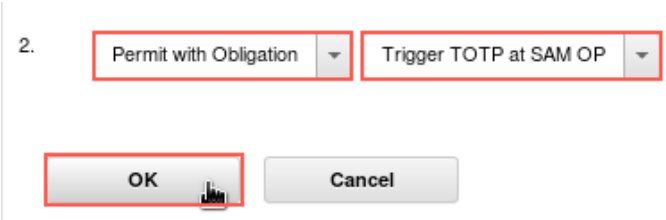


Select **authenticationTypes** from the first drop-down list.

Select **has member** as the comparison operator in the second drop-down list.

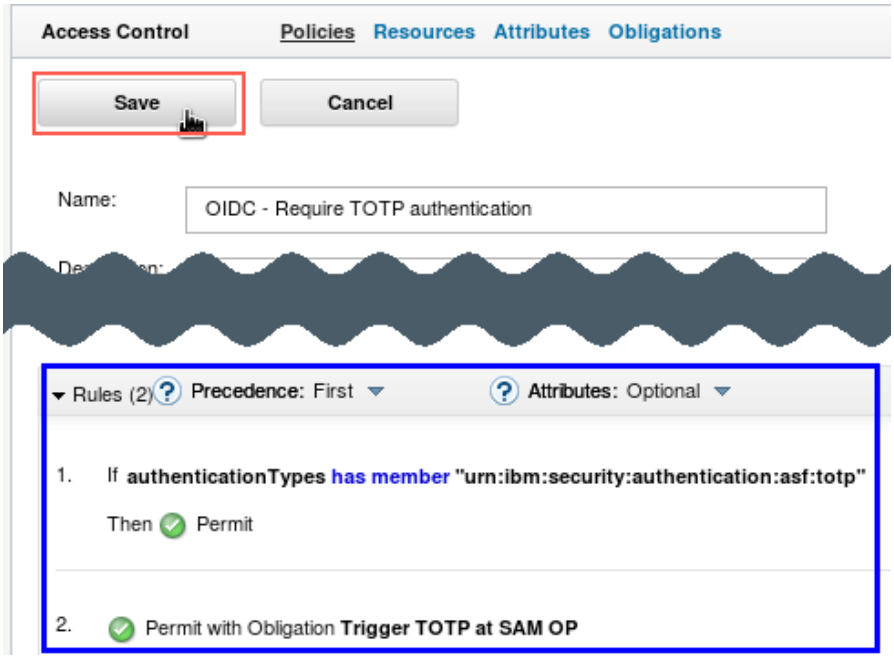
Enter **urn:ibm:security:authentication:asf:totp** in the text box. Click **OK** to save the rule.

Select the **expand** icon on the *Add Rule* button and select **Unconditional rule** from the drop-down menu.



Select **Permit with Obligation** from the decision drop-down list.

Select **Trigger TOTP at SAM OP** as the obligation. Then click **OK** to save the rule.



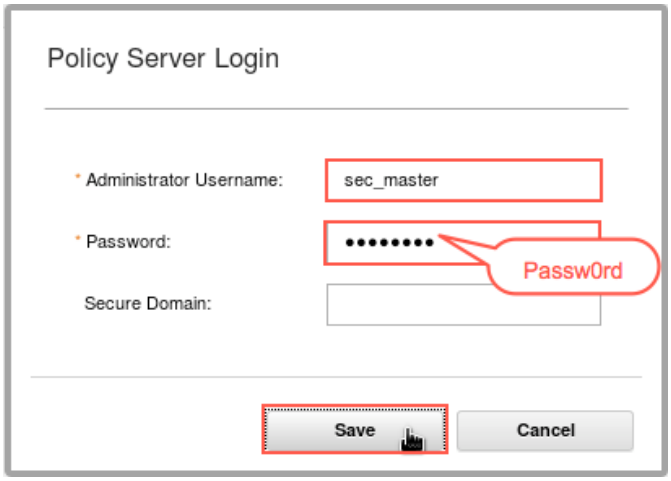
Check the rule is correct and then click **Save** at the top of the page.

8.9.2 Attach Policy to Resource

We will now attach the new policy to a resource. There is a resource in the demo application which is usually used for testing Risk Based Access but we can use it to trigger this policy instead.

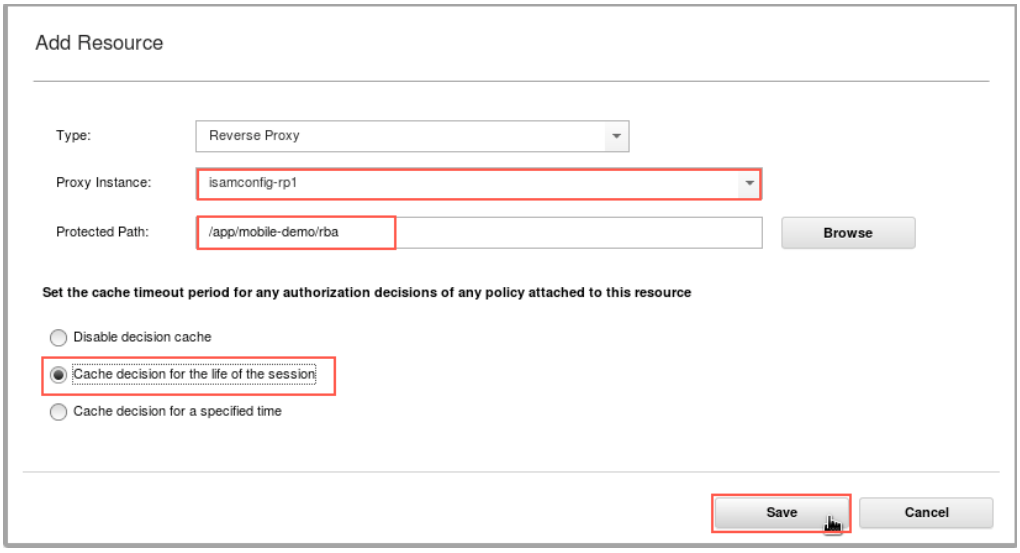
Select the **Resources** tab and then click the **Add** button.

The first time a resource is added, you must provide access to the SAM Policy Server (so it can retrieve available secure objects):



Enter **sec_master** as the *Administrator Username* and **Passw0rd** as the *Password*. Then click **Save**.

Click **Add** button again.



Select **isamconfig-rp1** as the *Proxy instance*.

Enter **/app/mobile-demo/rba** as the *Protected Path*. This is the URL we will protect with our TOTP Policy.

For a simple policy which, once satisfied, doesn't need to be checked again, we can get a significant performance improvement by caching the decision result in the Reverse Proxy. This means that the AAC Authorization engine doesn't have to be invoked for every request after access has been granted.

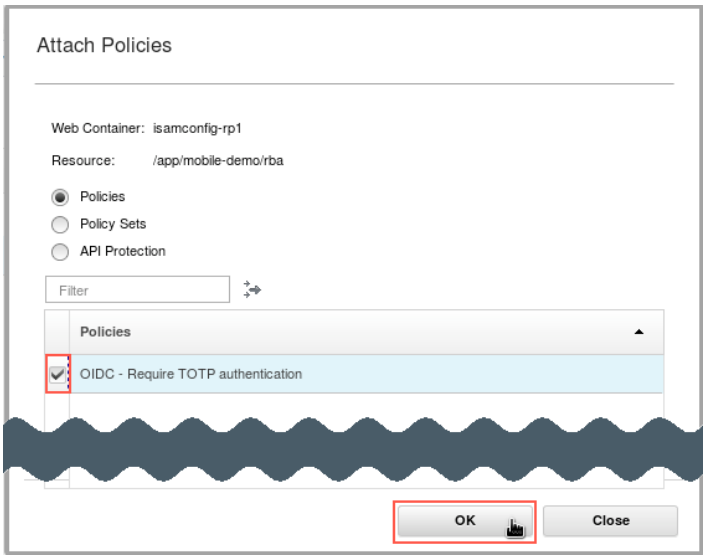
Select radio-button for **Cache decision for the life of the session**.

Click **Save** to save the resource definition.

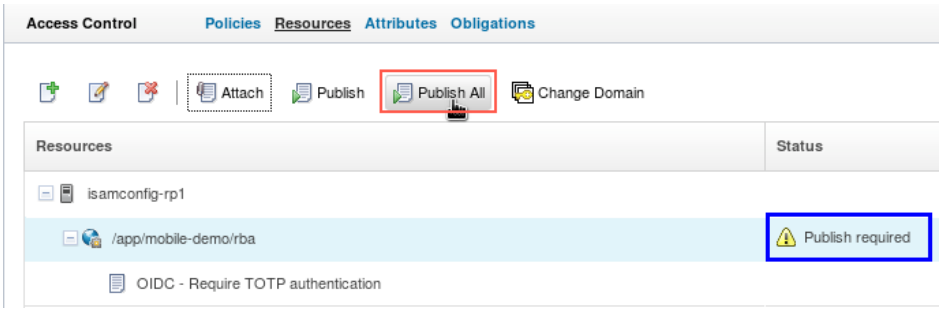
8.9.3 Attach Policy to Resource

We will now attach the OIDC TOTP policy to the RBA resource.

Select the **/app/mobile-demo/rba** resource and click the **Attach** button.



Check the check-box for the **OIDC - Require TOTP authentication** policy and click **OK**.



You can see that the new policy needs to be published. Click the **Publish All** button.

Click **Publish** to confirm the publish operation.

> Note that in a Docker environment, this *Publish* operation is only publishing the policy within the configuration container. A Container Management→Publish is still required to get the policy to the Runtime container.

8.10 Extra steps for Docker environment

In a Docker environment we need to publish the configuration and wait for the Runtime and Reverse Proxy containers to detect the new configuration and reload to activate it.

Click **Container Management** in the title bar and select **Publish Configuration** from the pop-up menu.

Click **Submit** to confirm the publish.

A new configuration snapshot is created. The runtime and reverse proxy containers will detect this new snapshot and automatically reload to pick up the changes. You may need to wait a minute for the reload to complete.

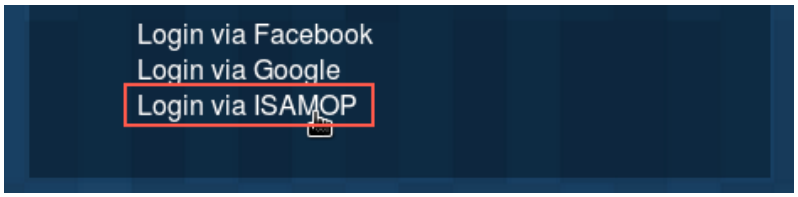
> If you're using your own environment without the auto-reload feature enabled, issue these commands to restart the necessary runtime and reverse proxy containers:
> ```
> docker exec -ti -- iamlab_isamruntime_1 isamcli -c reload runtime
> docker exec -ti -- iamlab_isamwrprp1_1 isamcli -c reload all
> ```

8.11 Test ACR Use-Case

8.11.1 Request RP Demo App homepage

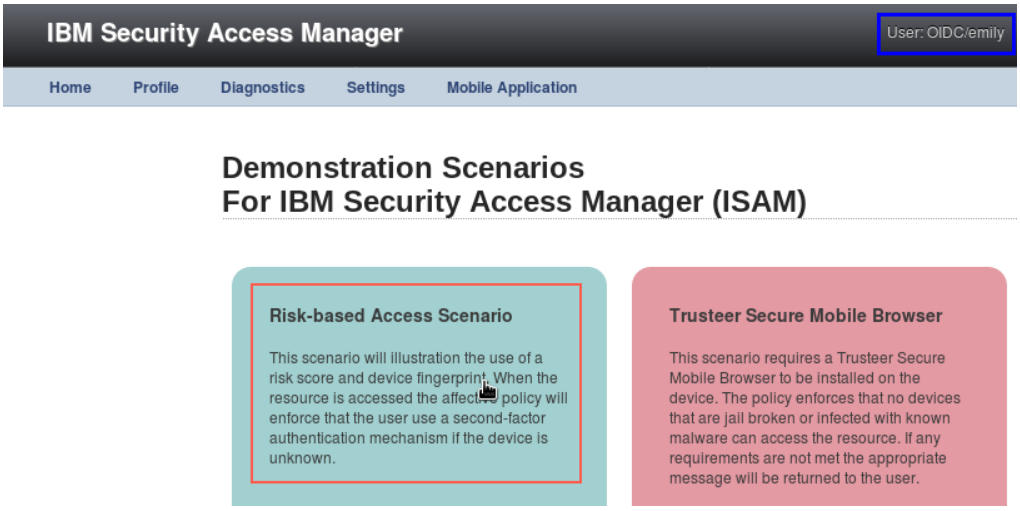
Open a new browser window and navigate to the following URL:
https://www.iamlab.ibm.com/app/mobile-demo/

This is a protected resource and so the login page is displayed:



Click the **Login via ISAMOP** to trigger login at the SAM OIDC Provider. This is a standard OIDC request - no TOTP needed at this point. You are redirected to the OP to authenticate.

Login at the OP using **emily** and **Passw0rd**. You are redirected back to the Relying Party and logged in. The demo app homepage is displayed:

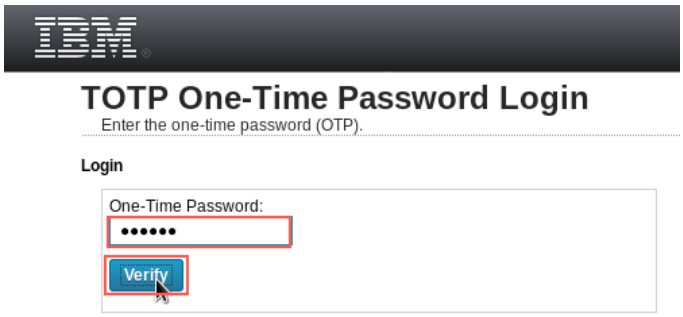


You are logged in to the RP site as **OIDC/emily**.

8.11.2 Access RP resource which requires TOTP

Now we will trigger our authorization policy. Click the **Risk-based Access Scenario** tile.
This makes a request for **/app/mobile-demo/rba** which is where the TOTP policy is attached.

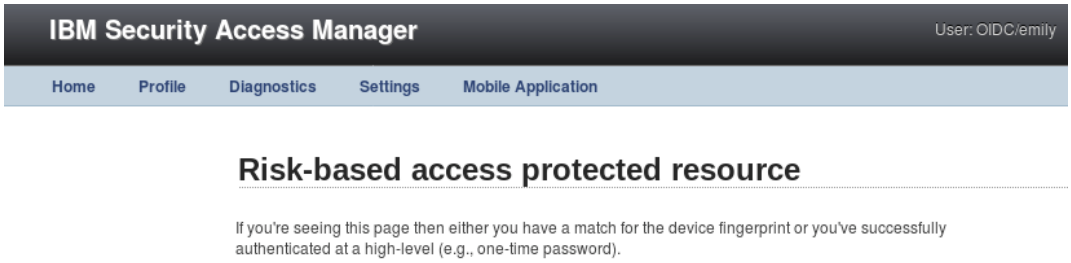
The Policy is executed and returns the OIDC TOTP obligation to the Reverse Proxy. It triggers OIDC with the *level=2* query-string which means that TOTP is requested at the OP. The OP Access Policy triggers TOTP authentication which presents the TOTP challenge page:



Use your registered TOTP client to get the current TOTP code for Emily. Enter it and click **Verify**.

At this point, the OIDC flow completes and you are returned to the RP. The Authorization Policy at the RP executes again but this time TOTP authentication is found in the *authenticationTypes* attribute and so it returns a Permit to the Reverse Proxy (which is cached).

The Reverse Proxy permits the request and the RBA page is returned to the user:



<u>Nice work! You have successfully completed the exercise and this lab guide.</u>