IBM Security

IBM SECURITY ACCESS MANAGER

Federation Cookbook

9.0.6.0

Installation
SAML 2.0
OpenID Connect
Secure Token Service

Jon Harry
Pranam Codur
Sumana Narasipur
Steve Nguyen
Ben Harmon
Shane Weeden
Ben Stevens

Version 9.0
December 2018

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.

Document Control

Release Date	Version	Authors	Comments
28 July 2015	1.0	Jon Harry & Pranam Codur	Version 1.0 : Cookbook for ISAM 9.0 SAML 2.0 support
18 Aug 2015	1.1	Pranam Codur	Version 1.1 : Added information about Signing and SLO
18 Sept 2015	3.0	Whole team	Completion of milestone 3 scenarios
12 Oct 2015	4.0	Whole team	Completion of milestone 4 scenarios
17 Nov 2015	5.0	Jon Harry	Update for GA release
18 Nov 2015	5.1	Jon Harry	Activation codes externalized for distribution
15 Jan 2016	5.2	Jon Harry	Added Notices for external distribution
18 Feb 2016	6.0	Ben Stevens	Added OpenID Connect chapters
03 May 2016	7.0	Sumana S. Narasipur, Ram Narayan, Adrian Sasmita, Jon Harry	Update with Point of Contact Management UI, FedCfg UI for SAM 9.0.1.0. Move older version config to Appendix
25th May 2017	8.0	Sumana S. Narasipur	Updates for SAM 9.0.3.0
12th December 2018	9.0	Sumana S. Narasipur	Updates for SAM 9.0.6 which includes OpenID Connect Single Sign On flows, Access Policies, Device Flow.

Table of Contents

1 Introduction 8
1.1 High Level Architecture and Networking 8
1.2 Required Components 8
1.2.1 Access Manager Virtual Appliance ISO Image 8
1.2.2 Access Manager 9.0 Activation Codes 8
1.2.3 Host machine running VMWare 9
1.2.4 VMWare Networking 9
1.2.5 Hosts file 9
1.2.6 Required Files 9
1.3 Manual vs. Programmatic configuration 10
1.3.1 End-to-end SAML 2.0 use case configuration 10
2 Virtual Machine creation and Appliance Install 11
2.1 Create a new virtual machine 11
2.2 Loading the Firmware Image onto the Virtual Appliance 11
3 Appliance Host and Networking Configuration 13
3.1 Manual vs Silent Configuration 13
3.2 OPTION 1: Silent Configuration 13
3.2.1 Use Configuration ISO to configure IP connectivity 13
3.2.2 Complete "First-Steps" process 14
3.3 OPTION 2: Manual Configuration 14
3.4 Check internet connectivity 24
4 Basic Appliance Configuration 25
4.1 Login to the Local Management Interface (LMI) of the Appliance 25
4.2 Enable NTP 27
4.3 Product Activation 29
4.4 Configure Runtime Interfaces 32
4.5 Update Hosts File on the Appliance 36
4.6 Configure ISAM Runtime Component on the Appliance 38
5 Create Reverse Proxy instance 42
5.1 Create new Reverse Proxy 42
5.2 Modify Reverse Proxy Instance Settings 45
5.3 Deploy the Changes and Restart the Reverse Proxy Instance 46
6 Create SAML 2.0 Identity Provider federation 48
6.1 Upload keystore files 48
6.2 Upload mapping rules 50
6.3 Create federation 53
6.4 Export meta-data 63
7 Create SAML 2.0 Service Provider federation 64
7.1 Uploading keystore files 64
7.2 Upload mapping rules 66
7.3 Create federation 69
7.4 Export meta-data 79
8 Configure Reverse Proxy for Federation 80
8.1 ISAM Configuration for the IdP 80
8.1.1 Configure Reverse Proxy for IdP 80
8.1.2 Environment-specific configuration 83
8.2 ISAM Configuration for SP 84
8.2.1 Configure Reverse Proxy for SP 84
8.2.2 Environment-specific configuration 87
8.2.3 Add anonymous user 87
9 Configure Partners 89
9.1 Configuring Partner for the IdP 89
9.2 Configuring Partner for the SP 93
10 Configure test application and test user 97

10.1 Configure test application 97
10.2 Authorize Access to Mobile Demo Application 99
10.2.1 Test Access to Demo Application....100
10.3 Configure test user....101
11 Test Federation....103
12 Customizing IDP Login Screens with Authentication Macros 107
12.1 Replace login.html on the identity provider....107
12.2 Configure authentication macros in the federation runtime 109
12.3 Testing the authentication macros....112
13 OpenID Connect....114
13.1 Open ID Connect Provider....114
13.1.1 Configuring Attribute Sources 114
13.1.2 Create API Definition with OIDC Enabled 116
13.1.3 Configuring Clients 119
13.1.4 Updating easuser password....122
13.1.5 Configuring Reverse Proxy for OpenID Connect Provider....122
13.2 OpenID Connect Relying Party....126
13.2.1 Updating easuser password....126
13.2.2 Uploading mapping rules....127
13.2.3 Configuring OpenID Connect Relying Party Federation for hybrid flow....129
13.2.4 Configuring OpenID Connect Relying Party Federation for implicit flow 134
13.2.5 Configuring OpenID Connect Relying Party Federation for authorization_code flow 138
13.2.6 Configuring OpenID Connect Relying Party Partner for hybrid federation 142
13.2.7 Configuring OpenID Connect Relying Party Partner for implicit federation 148
13.2.8 Configuring OpenID Connect Relying Party Partner for authorization_code federation....153
13.2.9 Configuring Point of Contact profile 158
13.2.10 Adding Signer Certificate....160
13.2.11 Configuring Reverse Proxy for OpenID Relying Party 161
13.3 Testing OIDC Single Sign-on Flow 166
13.3.1 Testing Hybrid Flow....166
13.3.2 Testing Authorization Code Flow 169
13.3.3 Testing Implicit Flow....171
14 Configuring OpenID Connect Flows to request for id_token and Userinfo claims 174
14.1 Configuring OIDC OP Attribute Mapping 174
14.2 Configuring Advanced Mapping rule OIDC RP Federation 175
14.3 Testing the OpenID Connect flow....179
15 Configuring OpenID Connect Dynamic client registration 182
15.1 Registering a client....182
15.1.1 Enabling Client Registration 182
15.1.2 Client Registration 183
15.2 Configuring OIDC Relying Party Partner 184
15.3 Testing the OpenID connect Single Sign-On flow 189
16 Introduction to the Security Token Service (STS) 193
16.1 Configuring the "STSUU to STSUU" Chain Template 194
16.2 Configuring the "STS Test" Module Chain....197
16.3 Allowing access to the STS via the ISAM Reverse Proxy 201
16.4 Updating the easuser password 202
16.5 Invoking the STS Test chain with cUrl 204
17 Advanced Federation Mapping Rules 205
17.1 Using HttpClient from Javascript mapping rules....205
17.2 Using the external http callout mapping module....208
17.3 Using an auxiliary STS chain for LDAP attribute lookup....213
17.3.1 Pre-LDAP javascript mapping module 214
17.3.2 Configuring the LDAP Attribute Map 214
17.3.2.1 Configuring a Server Connection 215
17.3.2.2 Configuring Attribute Sources....218
17.3.3 Post-LDAP javascript mapping module....220

17.3.4 Configuring the STS chain for LDAP Attribute Map 221
18 STS Tokens on Reverse Proxy Junctions 228
18.1 Create the ISAM Credential to SAML 2.0 STS Chain Template 228
18.2 Create the ISAM Credential to SAML 2.0 STS Module Chain 230
18.3 Update the reverse proxy configuration file 237
18.4 Create the /samljct Junction 238
18.5 Enable the demonstration application 239
18.6 Authorize Access to Mobile Demo Application 241
18.7 Test the /samljct Junction 242
19 Advanced External Authentication Interface Configuration for Service Providers 243
19.1 Patterns of EAI Authentication 243
19.1.1 USERNAME method of EAI Authentication 243
19.1.2 PAC method of EAI Authentication 243
19.1.3 EXTUSER method of EAI Authentication 244
19.2 Configuration of EAI Authentication Headers 244
19.2.1 EAI Authentication Headers in ISAM Reverse Proxy 244
19.2.2 EAI Authentication Headers in Federation Runtime 244
19.2.3 Point of Contact (PoC) Profiles Management 245
19.3 Scenario: “USERNAME” authentication 245
19.4 Scenario: “PAC” authentication 247
19.5 Scenario: “EXTUSER” authentication 249
20 Advanced OIDC: Configuring access policies to showcase prompt=login during OpenID Connect flow254
20.1 Activation Advanced Access Control 254
20.2 Configuring Reverse Proxy for Advanced Access Control 256
20.2.1 Configuring ACL 259
20.3 Configuring Username Password Authentication Mechanism 260
20.4 Configuring Definition with Access Policy 263
20.4.1 Configuring Access Policy 263
20.4.1 Updating Definition with Access Policy 265
20.5 Testing OpenID Connect flow with prompt=login 267
21 Reverse Proxy Native OIDC Relying Party 272
21.1 Configuring a Client 272
21.2 Configuring WebSEAL 273
21.3 Testing OIDC Single Sign-on flow 275
22 Appendix A: Troubleshooting and Workarounds 280
22.1 Troubleshooting 280
22.1.1 Enabling PD logs 280
22.1.2 Enable Federation logs 280
22.1.3 Time Sync error 282
23 Appendix B – Python Automation Project 284
24 Appendix C – Additional STS Examples 286
24.1 UsernameToken to SAML 2.0 286
24.1.1 Pre-requisites and Configuration 286
24.1.2 Testing 287
24.1.3 Further Details 287
24.2 SAML 2.0 to SAML 2.0 288
24.2.1 Pre-requisites and Configuration 288
24.2.2 Testing 289
24.2.3 Further Details 289
24.3 LTPA Junction 290
24.3.1 Pre-requisites and Configuration 290
24.3.2 Testing 291
24.3.3 Further Details 292
24.4 LTPA to STSUniversalUser 292
24.4.1 Pre-requisites and Configuration 292
24.4.2 Testing 293
24.4.3 Further Details 294

24.5 JWT to STSUniversalUser 294
24.5.1 Pre-requisites and Configuration 294
24.5.2 Testing 295
24.5.3 Further Details 295
24.6 STSUniversalUser to SAML 1.1 295
24.6.1 Pre-requisites and Configuration 296
24.6.2 Testing 296
24.6.3 Further Details 297
24.7 RACF PassTicket Junction 297
24.7.1 Pre-requisites and Configuration 297
24.7.2 Testing 298
25 Appendix D – Manual ISAM Configuration steps for IdP and SP 300
25.1 ISAM Configuration for the IdP 300
25.1.1 Load Federation Runtime SSL certificate into pdsrv trust store 300
25.1.2 Configure runtime junction for the IdP 302
25.1.3 Configure ACL policy for IdP 304
25.1.4 Configure the IdP reverse proxy 306
25.2 ISAM Configuration for SP 307
25.2.1 Load Federation Runtime SSL certificate into pdsrv trust store 307
25.2.2 Configure runtime junction for the SP 309
25.2.3 Configure ACL policy for SP 311
25.2.4 Configure the SP reverse proxy 313
26 Appendix E – Using cURL to call POC Configuration REST 315
26.1 Configuration for the IdP 315
26.1.1 Load Federation Runtime SSL certificate into pdsrv trust store 315
26.1.2 Federation auto-configuration endpoint 317
26.2 ISAM Configuration for SP 318
26.2.1 Load SSL certificates 318
26.2.2 Federation auto-configuration endpoint 320
27 Statement of Good Security Practices 329

1 Introduction

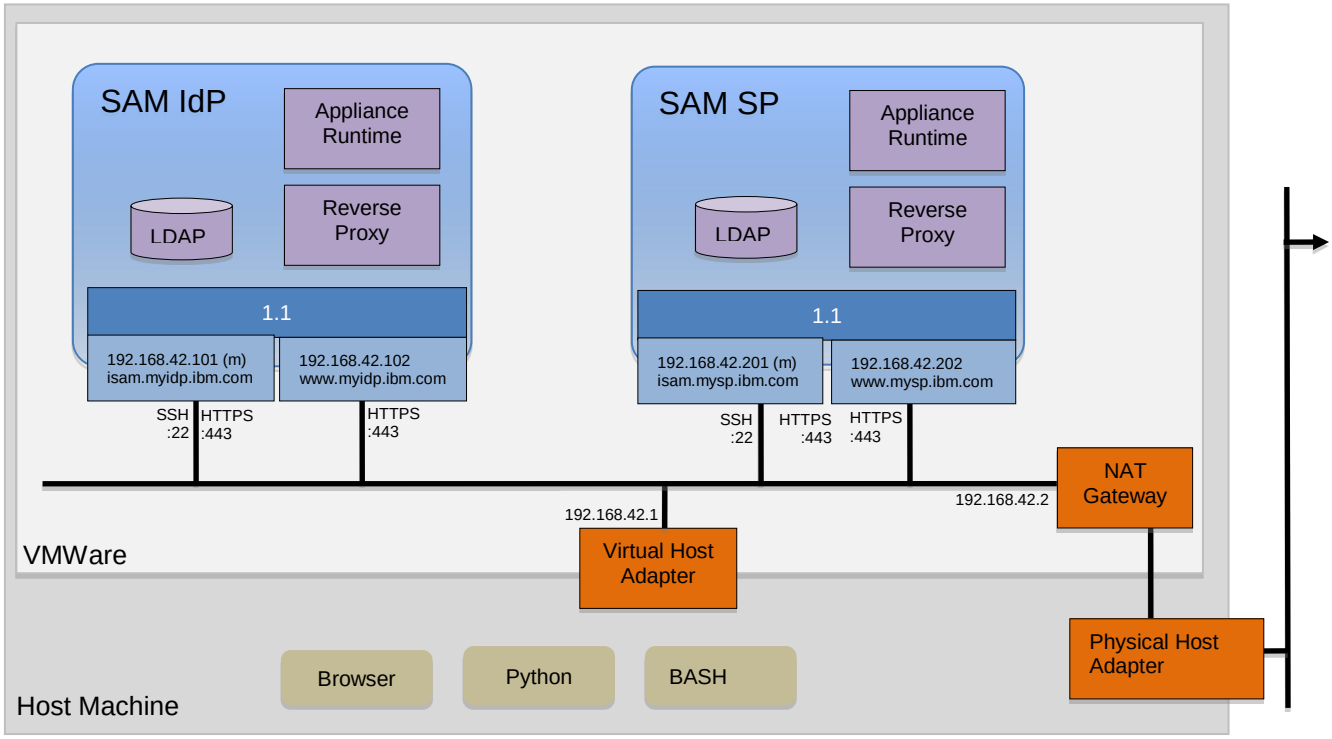
This cookbook provides a step-by-step guide to installing a pair of IBM Security Access Manager Virtual Appliances and then configuring them to demonstrate federation scenarios (such as federated single sign-on using the SAML 2.0 Browser POST profile) and Secure Token Service scenarios.

This cookbook is designed to work with IBM Security Access Manager 9.0.6.0.

The Python scripts written to accompany this document are designed to detect the installed version and perform the appropriate configuration.

1.1 High Level Architecture and Networking

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



1.2 Required Components

1.2.1 Access Manager Virtual Appliance ISO Image

The Access Manager Virtual Appliance installation ISO image is required in order to create a Virtual Appliance from an empty Virtual Machine. This image can be downloaded from IBM Software Sellers Workplace (IBMers), IBM PartnerWorld (Authorized Partners), or Passport Advantage (Entitled Customers). For SAM 9.0.6.0, search for Part **CNX5LML**.

1.2.2 Access Manager 9.0 Activation Codes

Access Manager functionality is enabled using Activation Codes. In order to use this cookbook, you will need the Activation Codes for the Platform and the Federation Add-on. Files containing these codes can be downloaded

from IBM Software Sellers Workplace (IBMers), IBM PartnerWorld (Authorized Partners), or Passport Advantage (Entitled Customers). Search for Parts **CNX5NML** and **CNX5QML**.

If you are planning to manually configure the appliance, you will need to select these files during activation so make sure they are available on the same machine as the browser you will use to access the appliances.

1.2.3 Host machine running VMWare

This guide assumes that the Hypervisor environment is VMWare Workstation. The host machine should have these minimum specifications:

- Good 64-bit processor (recommend dual core i5 or better)
- 12GB memory (4GB for host OS + 2 virtual machines each requiring 4GB)
- 20GB free disk space

1.2.4 VMWare Networking

This cookbook assumes NAT networking is used within VMWare and that the NAT network is configured for **192.168.42.0** subnet.

Internet connectivity is required for Network Time Protocol to be configured against an internet source.

1.2.5 Hosts file

The hosts file on the host machine must include the following entries to allow it to resolve the hostnames used in this lab guide:

```
192.168.42.101            isam.myidp.ibm.com
192.168.42.102            www.myidp.ibm.com
192.168.42.201            isam.mysp.ibm.com
192.168.42.202            www.mysp.ibm.com
```

> Some windows machines require that you run your text editor as administrator in order to be able to edit the %systemroot%\system32\drivers\etc\hosts file.

1.2.6 Required Files

The files required during the lab (mapping files, keys, scripts, etc.) are provided in a ZIP file which accompanies this document. This should be unpacked to a local directory on your host machine. In this guide it will be referred to as the …/**providedfiles** directory.

If you are planning to use the provided scripts to automate configuration of the appliances then you will need to copy the Platform and Federation activation codes (see section 1.2.2) into the *…/providedfiles/Automation/automation.ini* file as shown below:

```
[common]
platform-activation-code = xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx
federation-activation-code = yyyy-yyyy-yyyy-yyyy-yyyy-yyyy-yyyy-yyyy
```

1.3 Manual vs. Programmatic configuration

Once an appliance is installed and has been configured for basic IP connectivity, two methods of configuration are available:

- Manually via the LMI web console
- Programmatically via REST APIs

While manual configuration enables a more complete understanding of the steps required, programmatic configuration is preferred for quick and repeatable set up of appliances in a change-managed environment.

Where possible in this document, a Python script (which uses the REST APIs) is provided in addition to a step-by-step description of the manual steps. In this case you will see a notice that looks like this:

SCRIPT-START: A script is available for this section as an alternative to following the manual steps.
…

The scripts are found, under the providedfiles directory, in the following location:

…\providedfiles\Automation

Go to this directory in a Command Window and, from there, you can execute the scripts directly using the command indicated in the START-SCRIPT instructions.

If you decide to use the script, skip the manual steps until you see the corresponding end-of-script notice.

…..
SCRIPT-END:

Appendix B – Python Automation Project, at the end of this document, contains information about the Python project required for automatic configuration used throughout this document. You will likely need to follow the steps in that section to import additional modules into your python environment. Python 2.7 was used in the development of these automation scripts.

1.3.1 End-to-end SAML 2.0 use case configuration

If you want to quickly configure the SAML 2.0 federation use case, use the following commands after you have created the two SAM Appliances and given them management IP addresses:

1. SAMLIPConfig.py -configure All
2. SAMLSPConfig.py -configure All
3. ImportPartners.py -import IdP_Partner_Metadata
4. ImportPartners.py -import SP_Partner_Metadata

2 Virtual Machine creation and Appliance Install

This section describes the installation of two ISAM Virtual Appliances in VMWare Workstation.

> **This section needs to be completed twice**
> Once for the Identity Provider and once for the Service Provider.

2.1 Create a new virtual machine

The first step is to create a new VMWare virtual machine to host the virtual appliance.

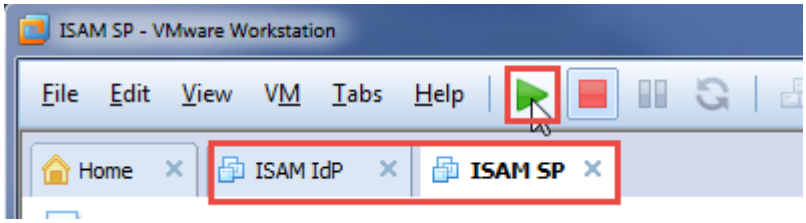
Set up the Virtual Machine with the following options:

Option	Recommended Value
Compatibility	Accept default unless you want to share with older versions
CD Drive	Use SAM 9.0.6.0 ISO image file (from Passport Advantage)
Guest Operating System	Other Linux 3.x kernel 64-bit
Virtual Machine Name	ISAM IdP **or** ISAM SP
Number of processors	1
Memory	4096 MB
Networking	Network Address Translation
Disk	New virtual disk
Disk Type	SCSI (default controller)
Disk Size	40 GB

If you like, you can delete any printer, sound card, and camera devices; they are not required.

2.2 Loading the Firmware Image onto the Virtual Appliance

Having now created the virtual machine, the next step is to load the ISAM virtual appliance firmware from the ISO image that we attached to the virtual machine when we created it.



With the new appliance tab (either ISAM IdP or ISAM SP) selected, press the green arrow button to start the virtual machine.

If you need to release your focus from the Virtual Machine, press **<Ctrl>** and **<Alt>** at the same time.

```
ISOLINUX 4.05 0x587a3765 ETCD Copyright (C) 1994-2011 H. Peter Anvin et al

Security Appliance Installer

Wait 10 seconds or press enter to boot the appliance installer.

Type "boothdd" to boot from the hard drive, or "interactive" to boot the
interactive appliance installer.


boot: _
```

Press ***Enter*** to start the appliance installer.

```
The signature of the installation image has been verified.
Partitioning the disk...
Formatting the boot partition on the disk...
Configuring the disk boot loader...
Formatting the swap partition...
Formatting the partition...
Installing the firmware image...
```

The installer automatically installs the appliance firmware to the Virtual Machine hard drive.

```
Installing the firmware image...
Creating the secondary partition...
Formatting the partition...
Installing the firmware image...
Installing the firmware image...
```

Two partitions are created each with a copy of the firmware.

When the firmware installation is complete, the Virtual Machine automatically shuts down.

Use setting in VMWare to disconnect the Virtual CD Drive from the Virtual Machine.

> Virtual Machines default to boot from the local hard disk so it is not a requirement to disconnect the virtual CD drive. However, doing so removes dependency on the ISO image being available which can generate unwanted warnings at start up.

3 Appliance Host and Networking Configuration

We will now perform host and networking configuration of the appliance so that the management interface is available on the network. This is done on the appliance console shown in the VMWare Workstation window.

> **This section needs to be completed twice**
> Once for the Identity Provider and once for the Service Provider.

3.1 Manual vs Silent Configuration

There are two ways that initial networking configuration can be applied to a new Access Manager appliance:

- Manual configuration via console
- Silent configuration using configuration ISO file

Silent configuration is designed for use when completely automated configuration of appliances is required; it allows networking to be configured so that the appliance management interface can be reached. Once this is done, all subsequent configuration can be performed via the LMI REST interfaces.

Both configuration methods are documented here; you can choose which to use.

3.2 OPTION 1: Silent Configuration

3.2.1 Use Configuration ISO to configure IP connectivity

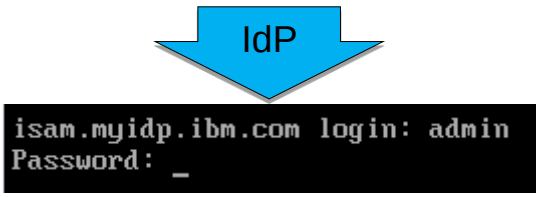
Silent configuration ISO files are available for this section as an alternative to following the manual steps. The ISO files contain configuration files that are used to perform a silent initial configuration of adapters, and networking. More details available at:
https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.6/com.ibm.isam.doc/admin/concept/con_silent_config.html

ISO files tailored for the environment documented in this document are available in the *…**/providedfiles/configuration_iso_files*** directory

For the IdP, boot from the IDP.iso configuration ISO after appliance image install
For the SP, boot from the SP.iso configuration ISO after appliance image install

In CD/DVD drive options under VM settings, select the appropriate ISO image, check the checkbox for **Connect at power on** and click **OK**.

We can now Power on the machines. Automatic configuration is performed. When complete, the configured hostname is displayed followed by a login prompt:



Return to the VMWare CD/DVD properties and clear the **Connect at power on** option.

Using the Boot ISO performs the minimum configuration required to provide IP connectivity; the appliance is given a hostname and a management IP address.

3.2.2 Complete "First-Steps" process

Before the appliance can be fully managed the "First Steps" process must be completed to confirm acceptance of the Software License Agreement (SLA).

SCRIPT-START:

For the IdP, run this script: **SAMLIPConfig.py -configure First_Steps**
For the SP, run this script: **SAMLSPConfig.py -configure First_Steps**

If you use this script, skip to the corresponding SCRIPT-END notice.

It is possible to complete the First Steps process manually by connecting to the LMI Web Console of the appliances and working through the First Steps wizard. This process is not detailed here because it is assumed that by choosing the "Silent Install" option a scripted approach is preferred.

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring the first steps for the appliance
INFO:BaseManager:Checking if the SLA has been accepted
INFO:BaseManager:Accepting the license
INFO:BaseManager:Checking if the appliance is configured
INFO:BaseManager:Configuring the appliance
INFO:BaseManager:Successfully configured the first steps for the appliance

If you use this method, skip the following manual configuration section and go to Section 3.4 - Check internet connectivity.

3.3 OPTION 2: Manual Configuration

Start the Virtual Appliance Virtual Machine.

While the appliance boots you will see a flashing cursor. After around 1 minute you should see the following:

```
MAC verified OK

unconfigured.appliance login: admin
Password:
```

Log in to the console using the administrator user id **`admin`** and the default password of **`admin`**.

```
Welcome
Welcome to the IBM Security Access Manager setup wizard.
Using this setup wizard, you can:
* View and accept the Software License Agreement
* Set the appliance password
* View and configure networking
Press Enter to continue.
```

During the first login after the initial firmware has been loaded onto the appliance, a wizard is automatically run to configure the firmware.

Press **Enter** to run the configuration wizard.

```
Software License Agreement
Currently selected language: English
1: Select language for license display
2: Read IBM terms
3: Read non-IBM terms
4: Proceed to acceptance

Select option: 4
```

Once you have read the Software License Agreement, enter ***4*** to proceed to acceptance of terms.

```
By choosing 'I agree,' you agree that (1) you have had the opportunity to
review the terms of both the IBM and non-IBM licenses presented above and (2)
such terms govern this transaction. If you do not agree, choose 'I do not
agree'.
1: I agree
2: I do not agree

Select option: 1
```

Enter **1** to agree to the license terms.

```
FIPS 140-2 Mode Configuration

You must enable FIPS mode in order to comply with FIPS 140-2 and NIST 800-131a.

If you select to enable FIPS mode, appliance will be rebooted immediately to
perform FIPS power-up integrity checks.
Do not choose to enable FIPS mode without reading the FIPS section in the user
guide.

If you choose to enable FIPS mode now, you cannot disable it later without
reinstalling the appliance.

FIPS 140-2 Mode is not enabled.
1: Enable FIPS 140-2 Mode
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

We don't want to enable FIPS mode so enter **n** to continue.

```
Appliance Password
Password changes are applied immediately.
Password has not been modified.
1: Change password
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

We don't want to change the password (we'll do that in a later step) so enter **n** to continue.

```
Host Name Configuration
Host name: unconfigured.appliance
1: Change the host name
x: Exit
p: Previous screen
n: Next screen

Select option: 1
```

Enter **1** to set the host name.

IdP

```
Change the Host Name
Enter the new host name: isam.myidp.ibm.com
```

Enter **`isam.myidp.ibm.com`** as the host name.

```
Host Name Configuration
Host name: isam.myidp.ibm.com
1: Change the host name
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

SP

```
Change the Host Name
Enter the new host name: isam.mysp.ibm.com
```

Enter **`isam.mysp.ibm.com`** as the host name.

```
Host Name Configuration
Host name: isam.mysp.ibm.com
1: Change the host name
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

Enter **n** to continue.

```
Network Interface Settings
1: Display device settings
2: Display policy
3: Configure an interface
4: Create a VLAN interface
5: Delete a VLAN interface
6: Set IPv4 default gateway
7: Set IPv6 default gateway
x: Exit
p: Previous screen
n: Next screen

Select option: 3
```

We now want to configure a management interface.

Enter **3** to configure an interface.

```
Configure an Interface
Select the interface to configure:
1: 1.1
Enter index: 1
```

Enter **1** to configure the 1.1 interface. This is the only interface available because we only defined one networking card for the Virtual Machine.

```
Enable this interface?
1: Yes
2: No
Enter index: 1
```

Enter **1** to enable this interface.

```
Select an IPv4 configuration mode:
1: Automatic
2: Manual
3: Automatic and Manual
Enter index: 2
```

Enter **2** for manual configuration - we want to specify a fixed IP address for the management interface.

```
Configure Static IPv4 Addresses
Select an action:
1: Show configured addresses
2: Add an address
3: Delete an address
4: Finish configuring addresses
Enter index: 2
```

Enter **2** to add a new IP address to the 1.1 interface

IdP

```
Enter the IPv4 address: 192.168.42.101
Enter the IPv4 prefix or subnet mask: 255.255.255.0
```

Enter the IPv4 configuration as follows:

- Address: **192.168.42.101**
- Subnet Mask: **255.255.255.0**

SP

```
Enter the IPv4 address: 192.168.42.201
Enter the IPv4 prefix or subnet mask: 255.255.255.0
```

Enter the IPv4 configuration as follows:

- Address: **192.168.42.201**
- Subnet Mask: **255.255.255.0**

```
Use this IP address for management?
1: Yes
2: No
Enter index: 1
```

Enter **1** to specify this IP address as a management address.

```
Enable this IP address?
1: Yes
2: No
Enter index: 1
```

Enter **1** to enable this IP address.

```
Configure Static IPv4 Addresses
Select an action:
1: Show configured addresses
2: Add an address
3: Delete an address
4: Finish configuring addresses
Enter index: 4
```

Enter **4** to finish configuring addresses.

> We could add other IP addresses here but configuration of the management address is the minimum required. With the management address configured, further addresses can be added later using the management console or REST APIs.

```
Select an IPv6 configuration mode:
1: Automatic
2: Manual
3: Automatic and Manual
Enter index: 2
```

We're not going to use IPv6 so we want to manually configure it with no addresses. Enter **2** to select this.

```
Configure Static IPv6 Addresses
Select an action:
1: Show configured addresses
2: Add an address
3: Delete an address
4: Finish configuring addresses
Enter index: 4
```

Enter **4** to finish (without creating any IPv6 addresses).

```
Network Interface Settings
1: Display device settings
2: Display policy
3: Configure an interface
4: Create a VLAN interface
5: Delete a VLAN interface
6: Set IPv4 default gateway
7: Set IPv6 default gateway
x: Exit
p: Previous screen
n: Next screen

Select option: 6
```

Enter **6** to set the IPv4 default gateway. This is required to give the appliance connectivity beyond the local 192.168.42.0 subnet.

```
Set IPv4 Default Gateway
Enter gateway IP address: 192.168.42.2
Select interface:
1: 1.1
Enter index: 1
```

Enter **192.168.42.2** as the Default Gateway.

Enter **1** to specify that the 1.1 interface should be used to reach the Default Gateway

> The 192.168.42.2 gateway is provided by VMWare. On a NAT-enabled subnet, this gateway will use Network Address Translation to route out from the host machine using its IP addresses and routing table.

```
Network Interface Settings
1: Display device settings
2: Display policy
3: Configure an interface
4: Create a VLAN interface
5: Delete a VLAN interface
6: Set IPv4 default gateway
7: Set IPv6 default gateway
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

We have now completed networking configuration so enter **n** to move on.

```
DNS Configuration
DNS is obtained from DHCP on 1.1
1: Set DNS server 1
2: Set DNS server 2
3: Set DNS server 3
4: Obtain DNS servers from DHCP
x: Exit
p: Previous screen
n: Next screen

Select option: 1
```

Since we're not using DHCP, we need to manually configure a DNS server.
Enter **1** to set DNS server 1.

```
Set DNS Server 1
Enter the DNS server IP address: 192.168.42.2
```

Enter **192.168.42.2** as the DNS server address.

> .2 is the DNS server provided by VMWare. It forwards DNS requests to the DNS servers configured for the host machine.

```
DNS Configuration
DNS server 1: 192.168.42.2
DNS server 2:
DNS server 3:
1: Set DNS server 1
2: Set DNS server 2
3: Set DNS server 3
4: Obtain DNS servers from DHCP
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

We have completed DNS configuration. Enter **n** to move on to the next screen.

```
Time Configuration
Time configuration changes are applied immediately.
Time: 10:11:41
Date: 07/07/2015
Time Zone: America/New_York
1: Change the time
2: Change the date
3: Change the time zone
x: Exit
p: Previous screen
n: Next screen

Select option: 3
```

Enter **3** to set the time zone.

```
Change the Time Zone
Select a continent or ocean:
1: Africa
2: Americas
3: Asia
4: Atlantic Ocean
5: Australia
6: Europe
7: Pacific Ocean
8: Etc
Enter index: 6
```

Enter the number associated with your geography

```
Select a timezone:
1: (UTC+00:00) Europe/Dublin
2: (UTC+00:00) Europe/Lisbon
3: (UTC+00:00) Europe/London
4: (UTC+01:00) Europe/Amsterdam
5: (UTC+01:00) Europe/Belgrade
6: (UTC+01:00) Europe/Berlin
7: (UTC+01:00) Europe/Bratislava
8: (UTC+01:00) Europe/Brussels
9: (UTC+01:00) Europe/Budapest
10: (UTC+01:00) Europe/Copenhagen
11: (UTC+01:00) Europe/Ljubljana
12: (UTC+01:00) Europe/Madrid
13: (UTC+01:00) Europe/Paris
14: (UTC+01:00) Europe/Prague
15: (UTC+01:00) Europe/Rome
16: (UTC+01:00) Europe/Sarajevo
17: (UTC+01:00) Europe/Skopje
18: (UTC+01:00) Europe/Stockholm
19: (UTC+01:00) Europe/Vienna
20: (UTC+01:00) Europe/Warsaw
21: (UTC+01:00) Europe/Zagreb
22: (UTC+02:00) Europe/Athens
23: (UTC+02:00) Europe/Bucharest
24: (UTC+02:00) Europe/Helsinki
25: (UTC+02:00) Europe/Istanbul
26: (UTC+02:00) Europe/Kiev
27: (UTC+02:00) Europe/Minsk
28: (UTC+02:00) Europe/Riga
29: (UTC+02:00) Europe/Sofia
30: (UTC+02:00) Europe/Tallinn
31: (UTC+02:00) Europe/Vilnius
32: (UTC+03:00) Europe/Moscow
Enter index: 3
```

and then enter the number associated with your time zone.

```
Applying policy changes.
Policy changes were successfully applied.



Time Configuration
Time configuration changes are applied immediately.
Time: 15:13:55
Date: 07/07/2015
Time Zone: Europe/London
1: Change the time
2: Change the date
3: Change the time zone
x: Exit
p: Previous screen
n: Next screen

Select option: n
```

Check the time and date displayed and, if necessary, use options 1 and 2 to modify. Once the date, time and time zone are set correctly, enter **n** to continue.

IdP

```
Summary
FIPS 140-2 Mode is not enabled.
Password has not been modified.
Host name: isam.myidp.ibm.com
Interface: 1.1
Policy:
        IPv4 Mode:      Manual
        IPv4 Manual Settings:
        IPv4 Address:   192.168.42.101/255.255.255.0
                        [Management]
Interface: 1.2
Policy:
Interface: 1.3
Policy:
Interface: 1.4
Policy:
The IPv4 default gateway is 192.168.42.2 on interface 1.1.
DNS server 1: 192.168.42.2
DNS server 2:
DNS server 3:
Time: 09:51:16
Date: 07/24/2015
Time Zone: Europe/London

1: Accept the configuration
2: Cancel the configuration
3: Modify the configuration

Select option: 1
```

SP

```
Summary
FIPS 140-2 Mode is not enabled.
Password has not been modified.
Host name: isam.mysp.ibm.com
Interface: 1.1
Policy:
        IPv4 Mode:      Manual
        IPv4 Manual Settings:
        IPv4 Address:   192.168.42.201/255.255.255.0
                        [Management]
        IPv6 Mode:      Automatic
        IPv6 Automatic Settings:
                        Management Address
Interface: 1.2
Policy:
Interface: 1.3
Policy:
Interface: 1.4
Policy:
The IPv4 default gateway is 192.168.42.2 on interface 1.1.
DNS server 1: 192.168.42.2
DNS server 2:
DNS server 3:
Time: 09:58:06
Date: 07/24/2015
Time Zone: Europe/London

1: Accept the configuration
2: Cancel the configuration
3: Modify the configuration

Select option: 1
```

Check the data displayed in the Summary. If it is correct, enter **1** to apply the specified configuration.

```
Applying policy changes.
Policy changes were successfully applied. Local Management Interface has been
restarted.
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.myidp.ibm.com> exit
```

The appliance firmware is now configured.

Enter **exit** to logout from the console interface.

3.4 Check internet connectivity

We will now test internet connectivity from our Virtual Appliance.

```
isam.myidp.ibm.com login: admin
Password: admin
```

Login with username **admin** and password **admin**

Note that hostname shown will be *isam.mysp.ibm.com* when configuring your SP appliance.

```
Last login: Wed Nov 11 06:35:49 2015
Welcome to the IBM Security Access Manager
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.myidp.ibm.com> tools
```

Enter **tools** to open the tools folder.

```
isam.myidp.ibm.com:tools> ping pool.ntp.org
PING pool.ntp.org (91.237.88.67) 56(84) bytes of data.
64 bytes from mail.qraftwerk.de (91.237.88.67): icmp_seq=1 ttl=128 time=45.5 ms
64 bytes from mail.qraftwerk.de (91.237.88.67): icmp_seq=2 ttl=128 time=42.1 ms
64 bytes from mail.qraftwerk.de (91.237.88.67): icmp_seq=3 ttl=128 time=42.0 ms
^C
--- pool.ntp.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2404ms
rtt min/avg/max/mdev = 42.096/43.270/45.566/1.632 ms
isam.myidp.ibm.com:tools>
```

Enter command **ping pool.ntp.org**. If ping is successful then this proves that our IP address is working, our DNS server is working, our default gateway is working, NAT connectivity to the internet is working, and that an NTP server can be reached.

The NTP server returned may differ depending on your location.

Press **Ctrl-C** to terminate the ping command.

If this ping command fails, debug of the networking configuration will be required. Check that the VMWare networking configuration of the default NAT network is set correctly and ensure that your host machine has connectivity to the internet.

```
isam.myidp.ibm.com:tools> exit
```

Enter **exit** to log out from the appliance console.

4 Basic Appliance Configuration

In this section we will perform basic configuration of the appliance. The following will be configured:

- Network Time Protocol
- Functionality Activation
- Additional IP addresses
- Static hosts
- Access Manager Runtime (local policy server and LDAP)

This section needs to be completed twice
Once for the Identity Provider and once for the Service Provider.

Reminder: scripts are provided for many of these steps. If you use the script, skip the manual steps until the end-of-script notice.

4.1 Login to the Local Management Interface (LMI) of the Appliance

The scripting options show here for resetting the LMI password may not be needed if you already set the password manually during initial license accept after iso installation. If however you have used the bootstrap ISO to boot up the appliance with a management interface, the admin password will still be at the default of “admin”, and this script will show you how to programmatically change it.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Admin_Password**
For the SP, run this script: **SAMLSPConfig.py -configure Admin_Password**

If you use this script, skip to the corresponding SCRIPT-END notice.

Open a browser on your host system. Firefox 38.1.0 ESR was used when writing this lab guide.

Open the LMI GUI for the ISAM Appliance via the URL:

IdP

`https://isam.myidp.ibm.com`

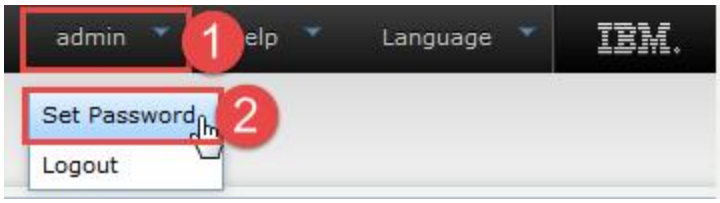
SP

`https://isam.mysp.ibm.com`

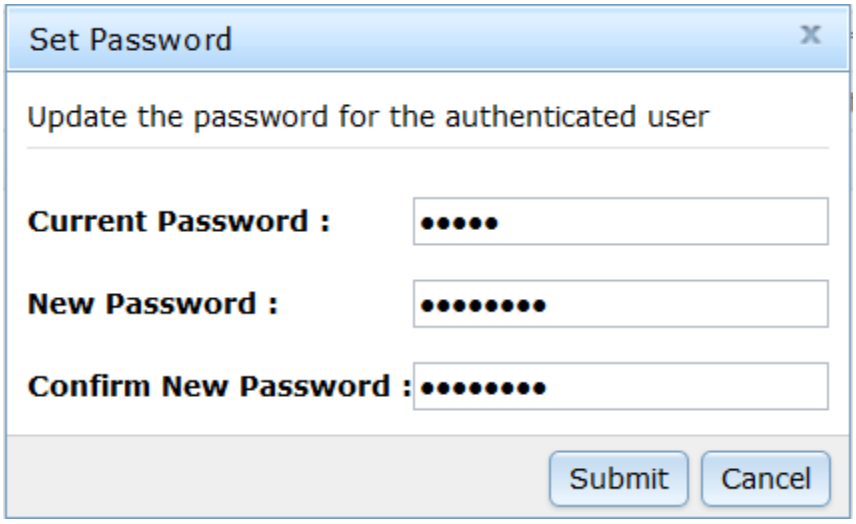
Accept any security exceptions related to the self-signed certificate presented by the Virtual Appliance. Ensure that the **Permanently store this exception** checkbox is selected to avoid seeing this certificate warning in the future.

The login page for the ISAM Appliance LMI is now displayed:

Login as user **`admin`** with password **`admin`**



Click on the **admin** username in the console title bar and select **Set Password** from the drop-down menu.



Enter **admin** as the *Current Password* and enter **Passw0rd** in the *New* Password and *Confirm New Password* boxes. Click Submit.

This is the password used for most users and administrator accounts in this guide.

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring the administrator password
INFO:BaseManager:Successfully configured the administrator password

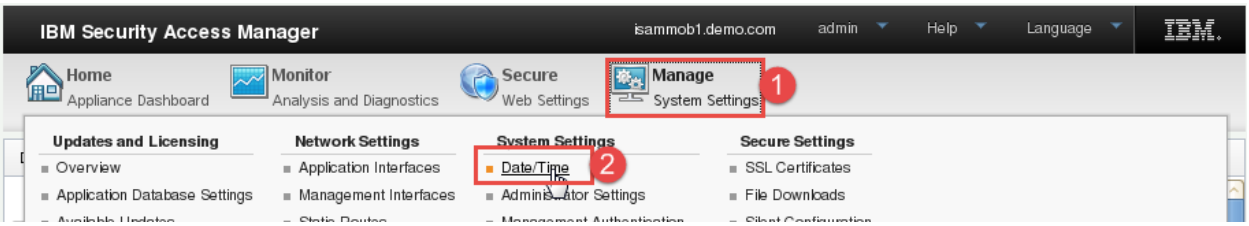
4.2 Enable NTP

You may notice that on the LMI dashboard there is a notification warning that "Local clock is not synchronized". We will now configure the appliance to use an internet NTP service to maintain clock synchronization.

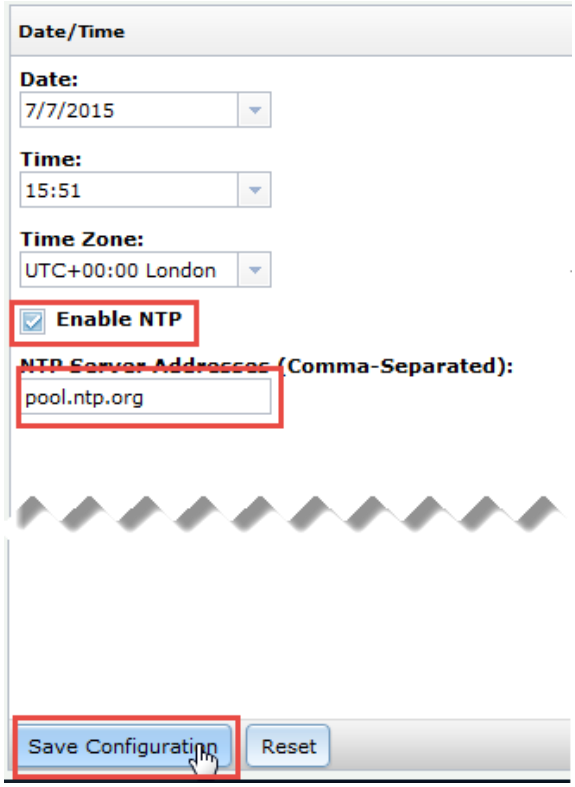
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure NTP**
For the SP, run this script: **SAMLSPConfig.py -configure NTP**

If you use this script, skip to the corresponding SCRIPT-END notice.



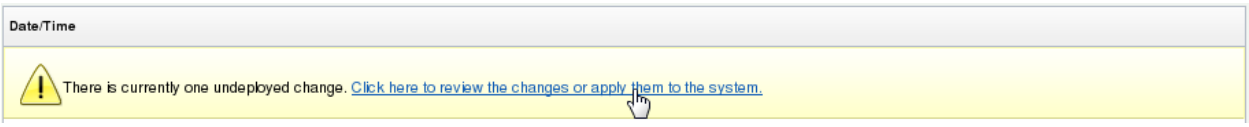
Click on the ***Manage System Settings*** icon to open the "mega-menu" and click the ***Date/Time*** item - as shown above.



Check the checkbox for ***Enable NTP*** and enter `pool.ntp.org` in the ***NTP Server Addresses*** entry box.

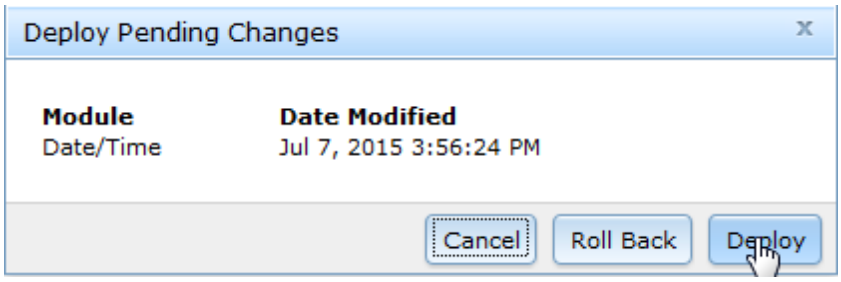
Click **Save Configuration** at the bottom of the window to save the changes.

Notice that a warning is now displayed at the top of the window:



This indicates that changes have been made to the appliance configuration but they are not yet active. Click the link to activate the configuration change you have just made.

A pop-up dialog is displayed showing the pending changes:



Click **Deploy** to deploy the changes to the appliance.

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring NTP server
INFO:BaseManager:Successfully configured NTP server

4.3 Product Activation

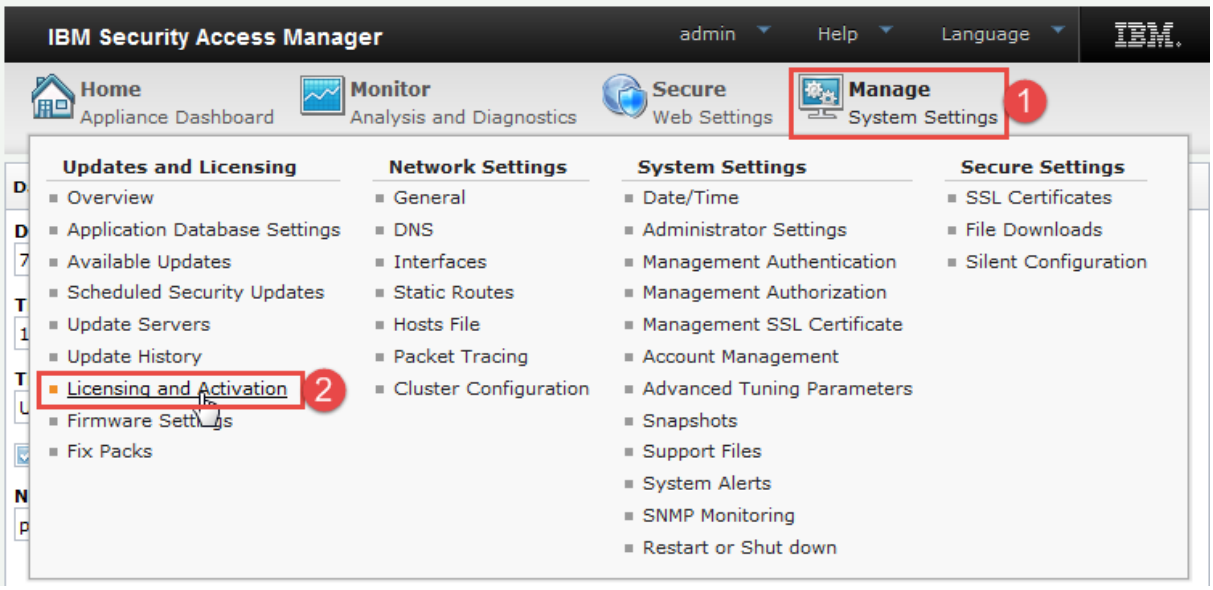
The Access Manager 9.0 Virtual Appliance firmware contains a number of functional modules. However, after initial installation, only basic management functions are available. Activation is required in order to enable the purchased modules.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

Before you run this script you must add the Access Manager Platform and Federation Activation Codes into the file *…/providedfiles/Automation/automation.ini.* See section 1.2.2.

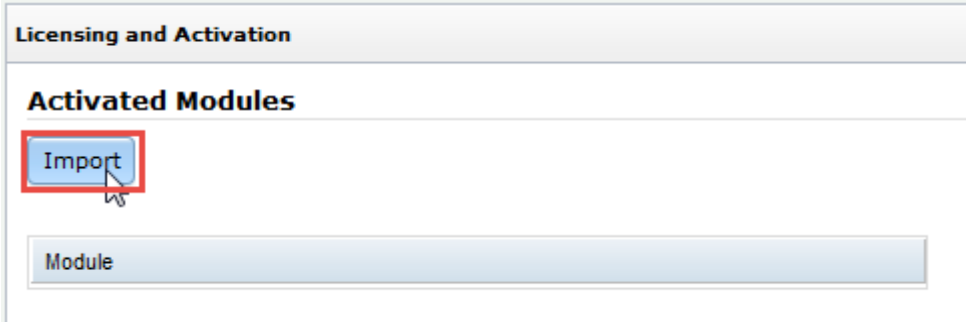
For the IdP, run this script: **SAMLIPConfig.py -configure Product_Activation**
For the SP, run this script: **SAMLSPConfig.py -configure Product_Activation**

If you use this script, skip to the corresponding SCRIPT-END notice.



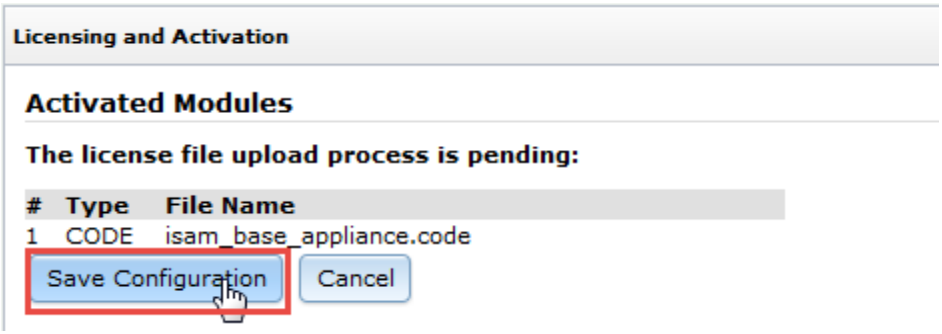
Click on the ***Manage System Settings*** icon to open the "mega-menu" and click the ***Licensing and Activation*** item - as shown above.

The licensing and Activation screen is displayed. Currently there are no activated modules.



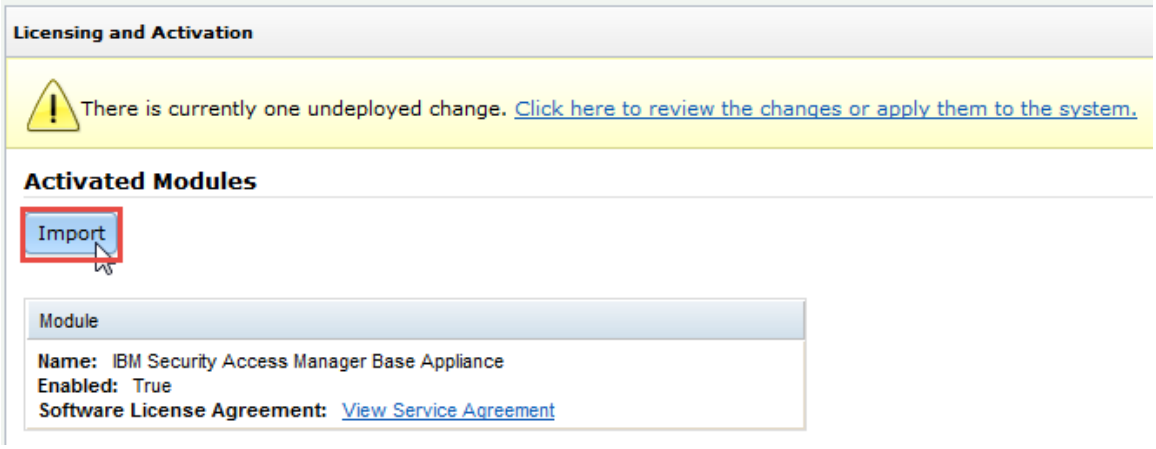
Click the ***Import*** button. A file selector dialog is displayed.

Select the ISAM 9.0 Platform Activation File that you downloaded from IBM (see section 1.2.2)



Click ***Save Configuration***.

The IBM Security Access Manager base activation code is processed and the module is listed. We won't deploy changes yet because we also want to activate Federation functionality.

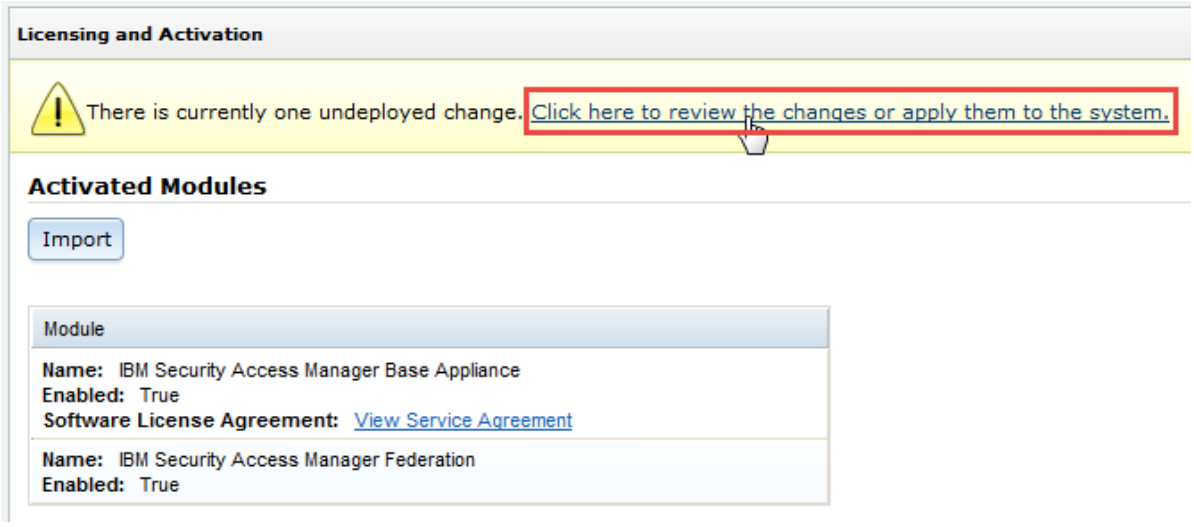


Click ***Import***. The file selection dialog opens again.

Select the ISAM 9.0 Federation Activation File that you downloaded from IBM (see section 1.2.2)

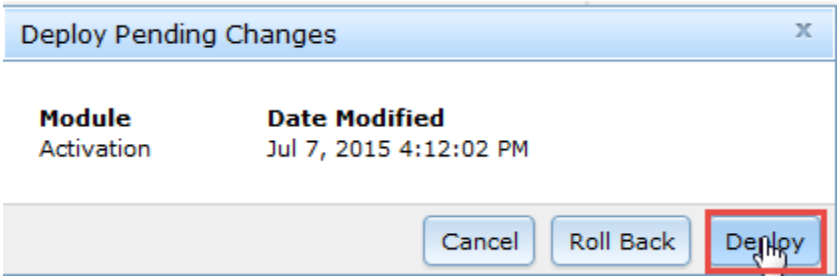
Click ***Save Configuration***.

The Federation activation code is processed. Now both IBM Security Access Manager Base Appliance and IBM Security Access Manager Federation Mobiles are listed:



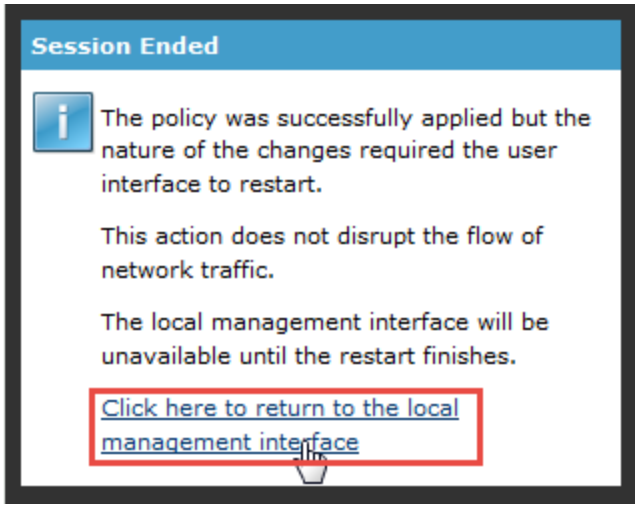
To complete the activation process we must deploy the changes we have made.

Click the ***Click here to review the changes or apply them to the system*** link in the warning message - as shown above.



Click ***Deploy*** to confirm the deployment of the changes.

The activation process can take a few minutes to complete because a number of new components are started and initialized within the appliance. Once it is complete, the following message is displayed:



Click on the link in the message to reconnect to the appliance management interface (it may take a few seconds for this to work).

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring product activation
INFO:BaseManager:Successfully activated platform
INFO:BaseManager:Successfully activated federation

You should now see that **Secure Federation** mega-menu is available in the LMI Web Console:

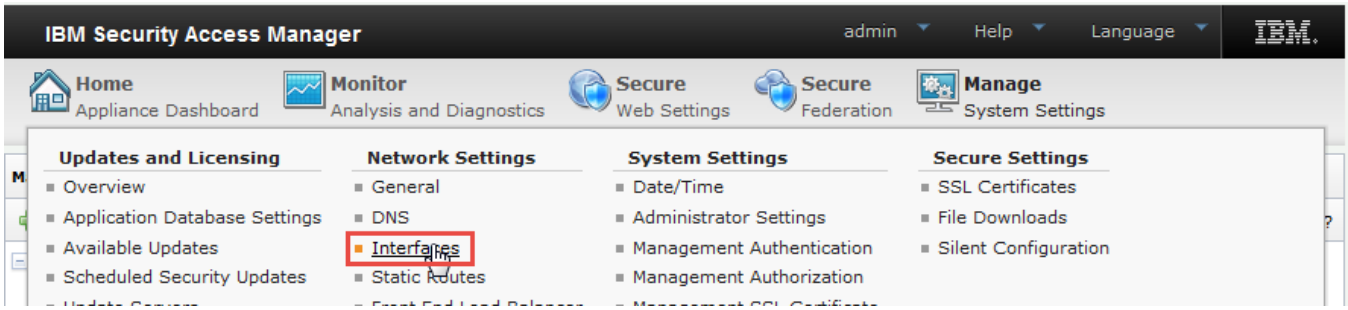
4.4 Configure Runtime Interfaces

We will now configure the Interfaces where appliance runtime components, such as the Reverse Proxy will listen.

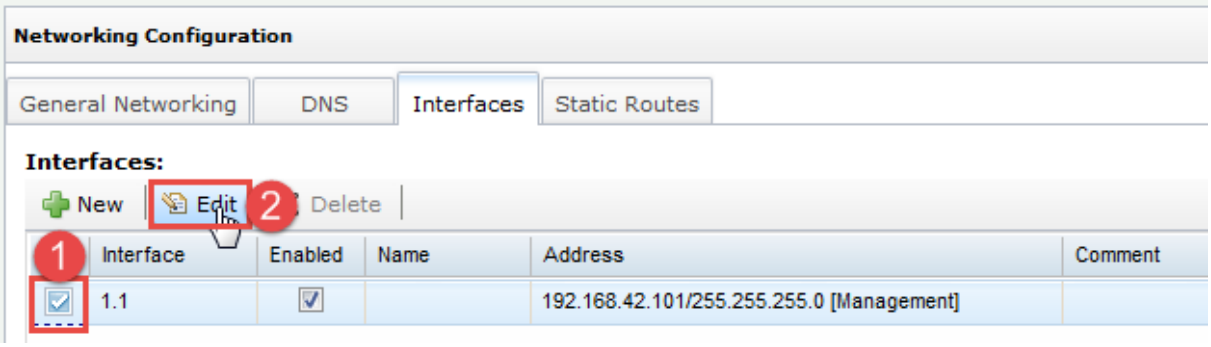
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Runtime_Interfaces**
For the SP, run this script: **SAMLSPConfig.py -configure Runtime_Interfaces**

If you use this script, skip to the corresponding SCRIPT-END notice.



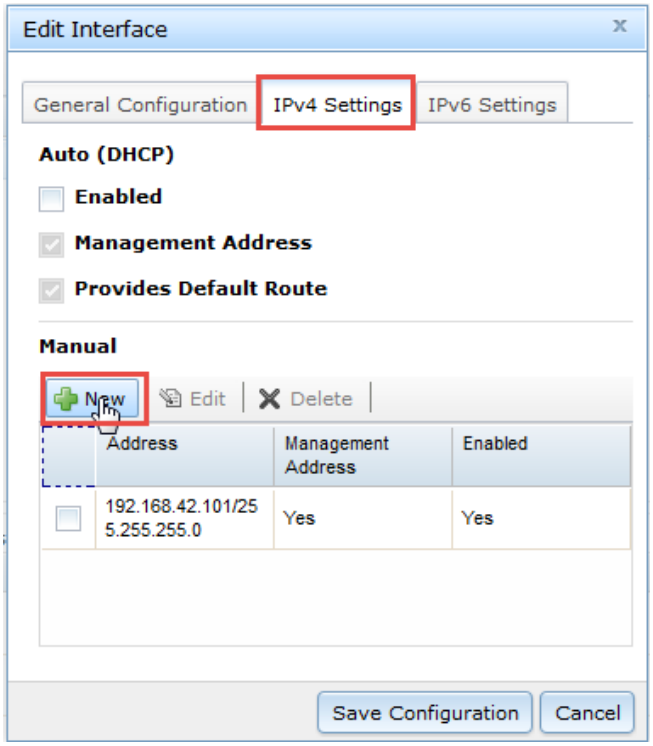
In the top menu panel, select ***Manage System Settings*** → ***Network Settings: Interfaces***" - as indicated above.



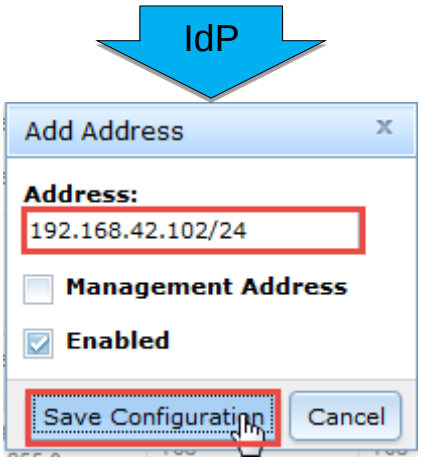
The configuration shows our only interface (1.1) and the single management IP address that we are connected to.

We need to edit this interface configuration in order to add an additional (non-management) IP address.

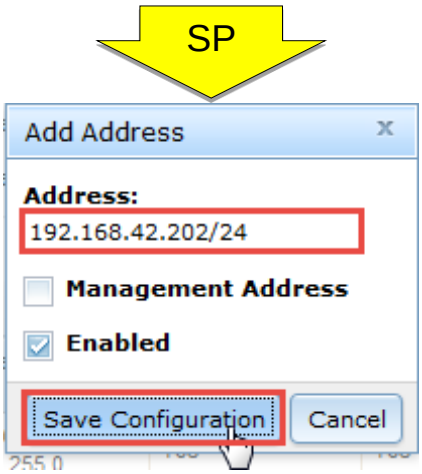
Select the checkbox next to the **1.1** interface and click **Edit** - as shown above.



Select the **IPv4 Settings** tab and then click the **New** button to add a new IP address.



Enter **192.168.42.102/24** in the *Address* field.

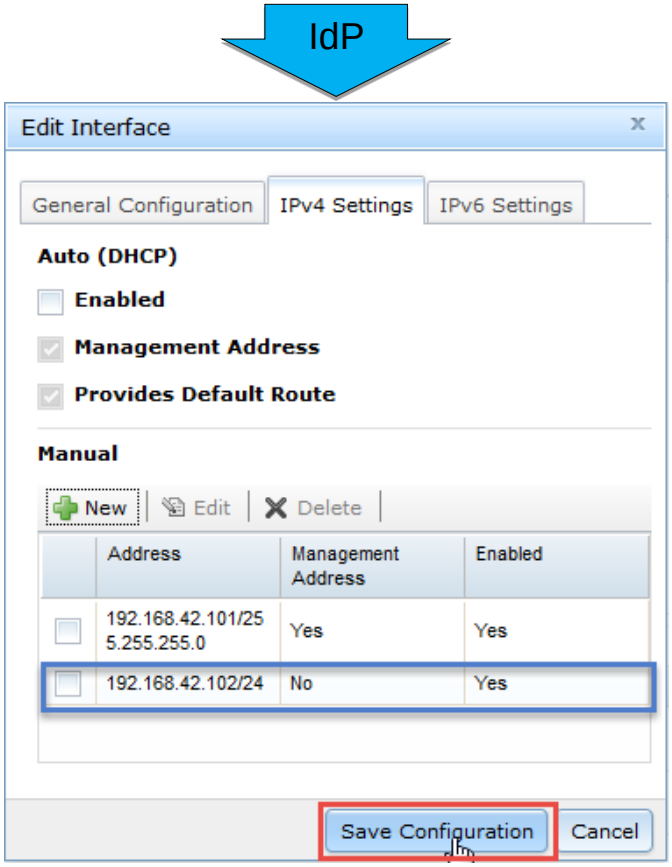


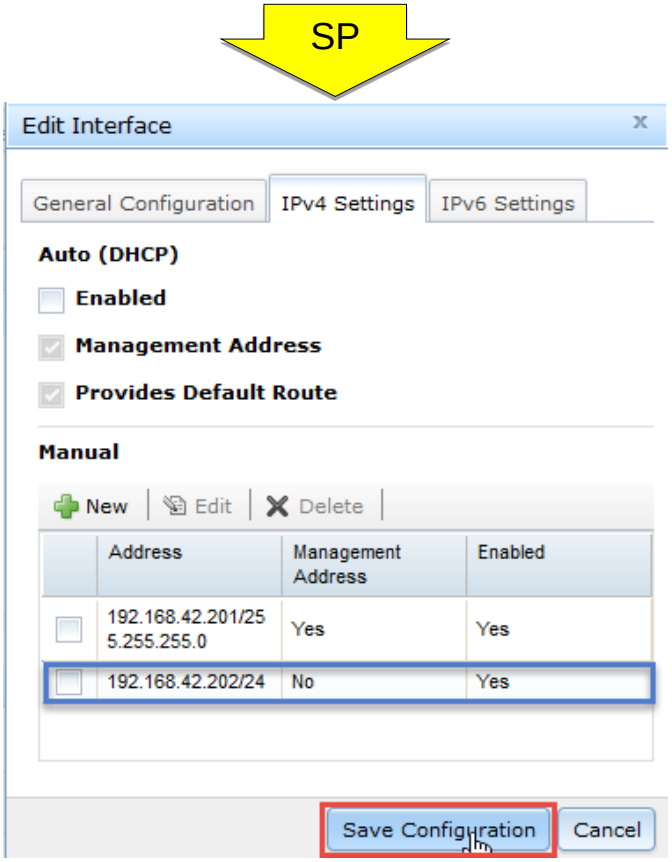
Enter **192.168.42.202/24** in the *Address* field.

This is CIDR notation; the /24 means there are 24 bits in the subnet mask (i.e. 255.255.255.0).

Click **Save Configuration**.

The new IP address is now listed:





Click **Save Configuration** to save the new interface configuration.

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

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring Runtime Interfaces
INFO:BaseManager:Successfully configured Runtime Interfaces

Open a command window on your host machine and ping the new IP address you just created to check that the address is active and reachable.

IdP

```
# ping 192.168.42.102
Pinging 192.168.42.102 with 32 bytes of
data:
Reply from 192.168.42.102: bytes=32
time<1ms TTL=64
Reply from 192.168.42.102: bytes=32
time<1ms TTL=64
```

SP

```
# ping 192.168.42.202
Pinging 192.168.42.202 with 32 bytes of
data:
Reply from 192.168.42.202: bytes=32
time<1ms TTL=64
Reply from 192.168.42.202: bytes=32
time<1ms TTL=64
```

4.5 Update Hosts File on the Appliance

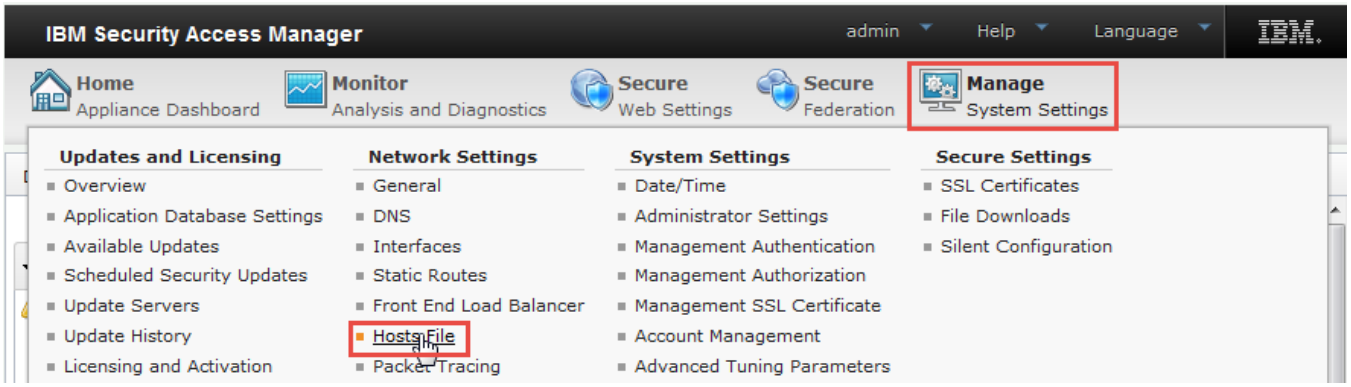
Since we don't have access to a DNS server that we can modify, we will now add a couple of host aliases to the appliance.

SCRIPT-START:

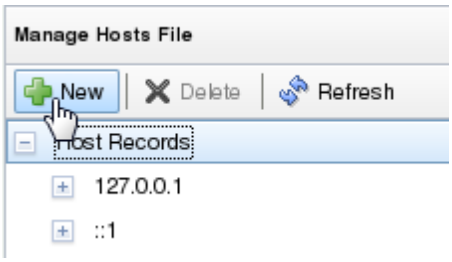
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Hosts_File**
For the SP, run this script: **SAMLSPConfig.py -configure Hosts_File**

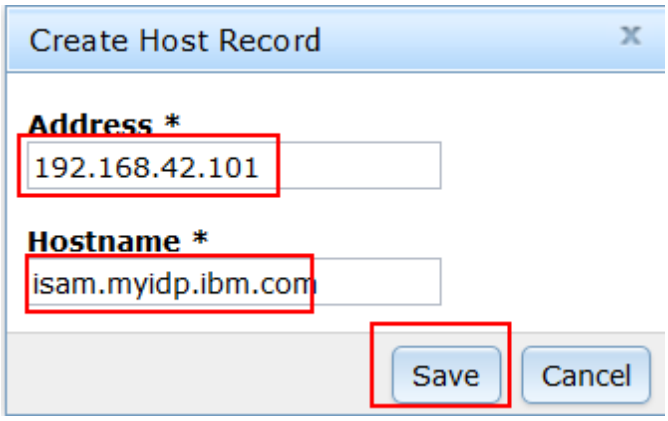
If you use this script, skip to the corresponding SCRIPT-END notice.



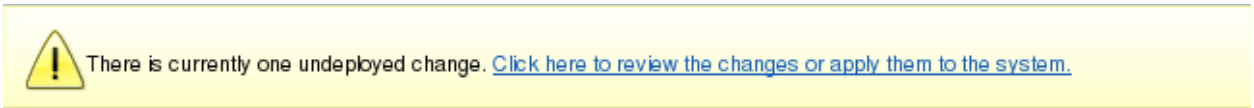
In the top menu panel, select ***Manage System Settings*** → ***Network Settings: Hosts File*** - as indicated above.



Select the "*Host Records*" entry and press the ***New*** button .



Enter "***192.168.42.101***" as the *Address* and "***isam.myidp.ibm.com***" as the *Hostname*. Press ***Save*** to create the hosts file entry.

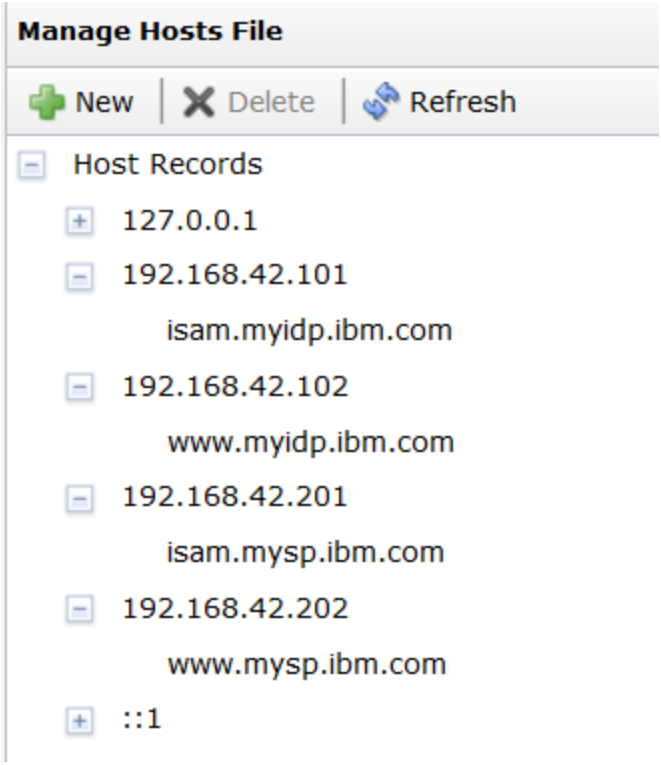


You can see at this point that there is now an undeployed change in the system. The change to the hosts configuration is pending at this point.

Before we deploy the change, add these additional host entries:

Address	Hostname
192.168.42.102	www.myidp.ibm.com
192.168.42.201	isam.mysp.ibm.com
192.168.42.202	www.mysp.ibm.com

The hosts configuration should now look like this:



Click the ***Click here to review the changes or apply them to the system*** link shown above and click **Deploy** to confirm the changes.

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Configuring HOSTS file entries
INFO:BaseManager:Successfully configured HOSTS file entries

4.6 Configure ISAM Runtime Component on the Appliance

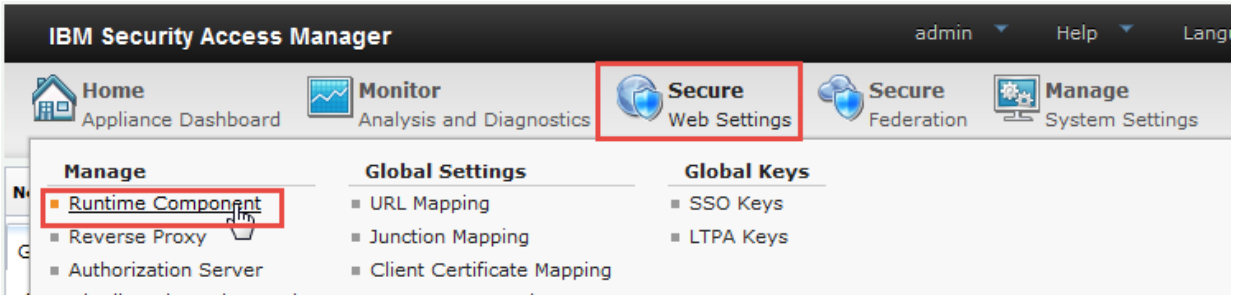
In this section we will now configure the ISAM Runtime component of the appliance.

For this lab, we will configure the ISAM appliance to run with a local ISAM Policy Server and a local LDAP server.

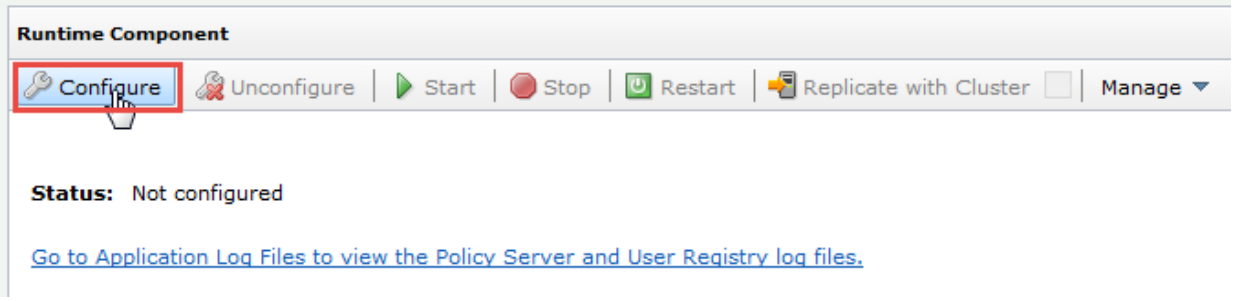
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Runtime_Component**
For the SP, run this script: **SAMLSPConfig.py -configure Runtime_Component**

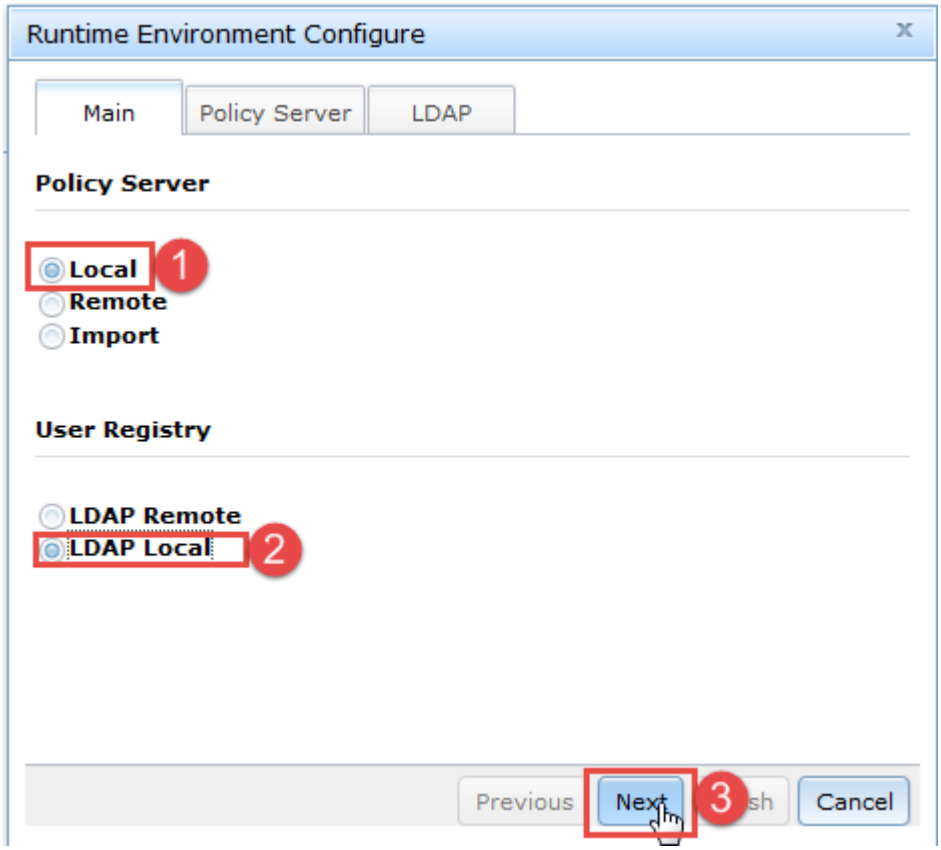
If you use this script, skip to the corresponding SCRIPT-END notice.



In the top menu panel, select ***Secure Web Settings*** → ***Manage: Runtime Component*** – as indicated above.

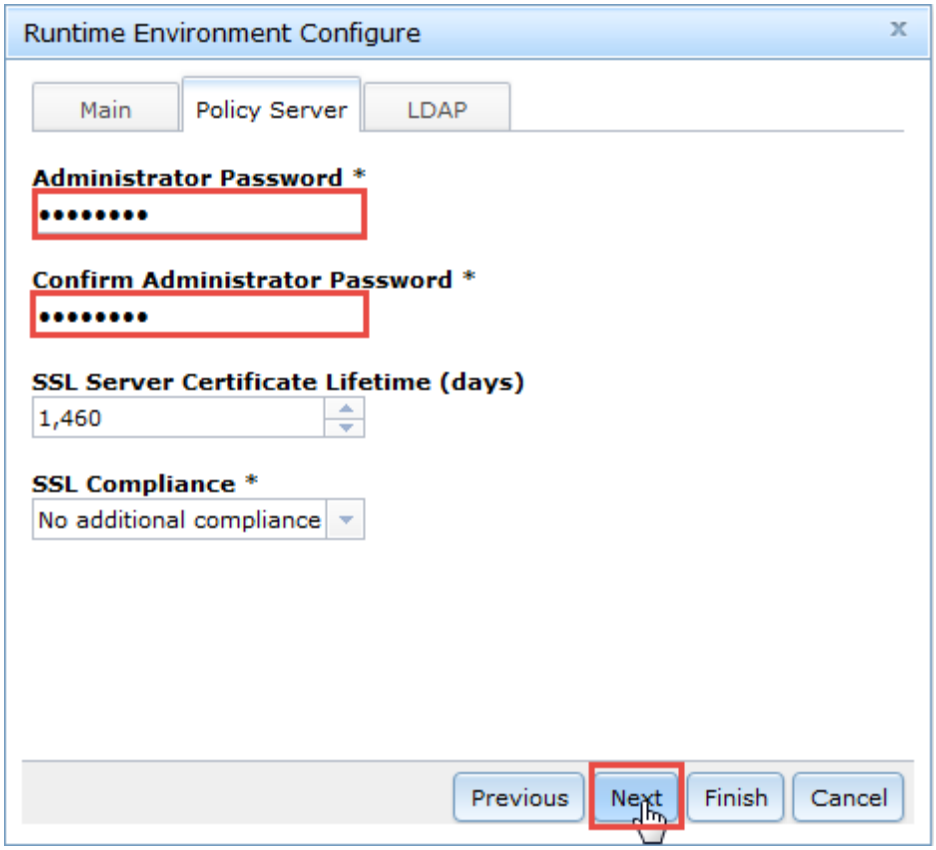


Click the ***Configure*** button to initiate the runtime configuration dialog.

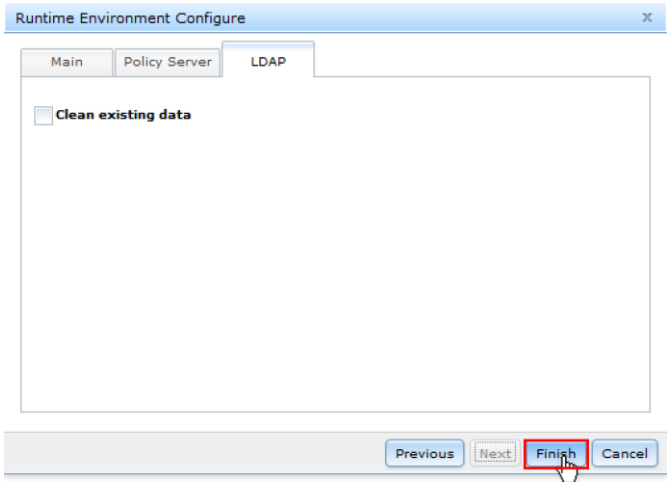


Select the radio buttons for a "***Local***" Policy Server and an "***LDAP Local***" User Registry.

Click ***Next*** to move to the next configuration tab.

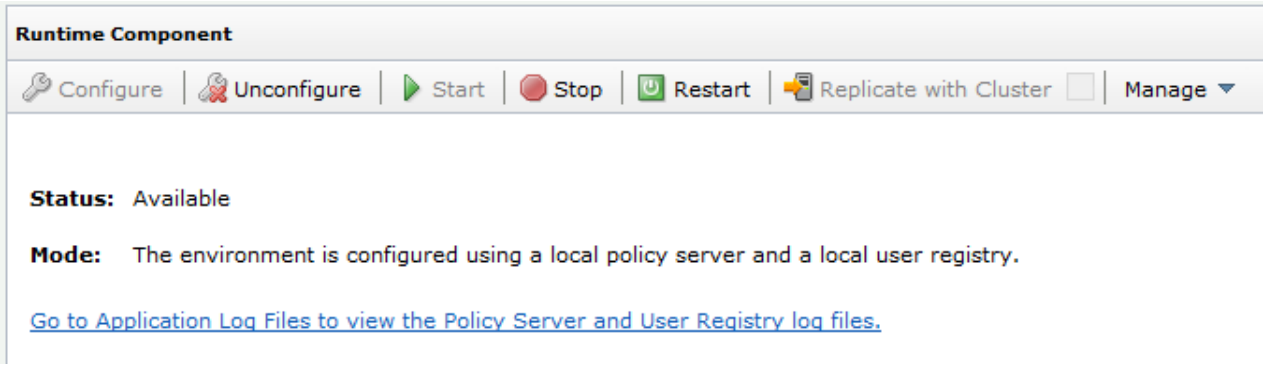


Enter "***Passw0rd***" as the "*Administrator Password*" and "*Confirm Administrator Password".* Ensure the other fields left as default. Press ***Next*** to progress to the next tab.



On the LDAP tab, press ***Finish*** to perform the runtime configuration.

> Note that you are using the default password (passw0rd) for the built-in LDAP directory (which should obviously be changed in any non-trivial environment). We won't change the password in this environment, but it can be changed under **Manage→Embedded LDAP→Change Password**.



After a short time, during which the Policy Server is configured and entries are created in the LDAP, you should see a message indicating that the ISAM runtime component is configured using a local policy server and a local user registry – as shown above.

SCRIPT-END:

The script should display the following:

INFO:WGAManager:Configuring Runtime component

INFO:WGAManager:Successfully configured Runtime component

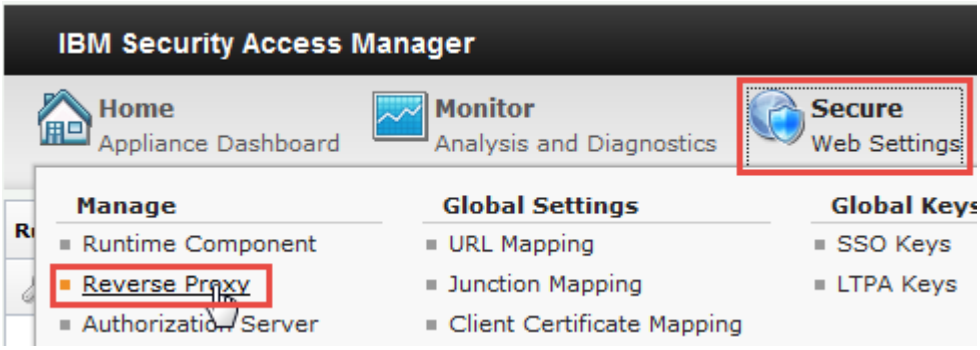
5 Create Reverse Proxy instance

In this lab we will create a Reverse Proxy instance on our ISAM Appliance. This will authenticate users at the Identity Provider and protect services at the Service Provider.

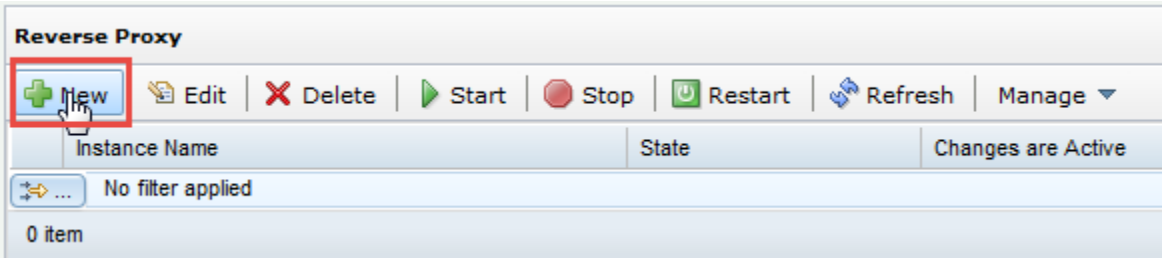
> **This section needs to be completed twice**
> Once for the Identity Provider and once for the Service Provider

5.1 Create new Reverse Proxy

> **SCRIPT-START:**
> A script is available for this section as an alternative to following the manual steps.
>
> **For the IdP**, run this script: **SAMLIPConfig.py -configure WebSEAL_Instance**
> **For the SP**, run this script: **SAMLSPConfig.py -configure WebSEAL_Instance**
>
> If you use this script, skip to the corresponding SCRIPT-END notice.

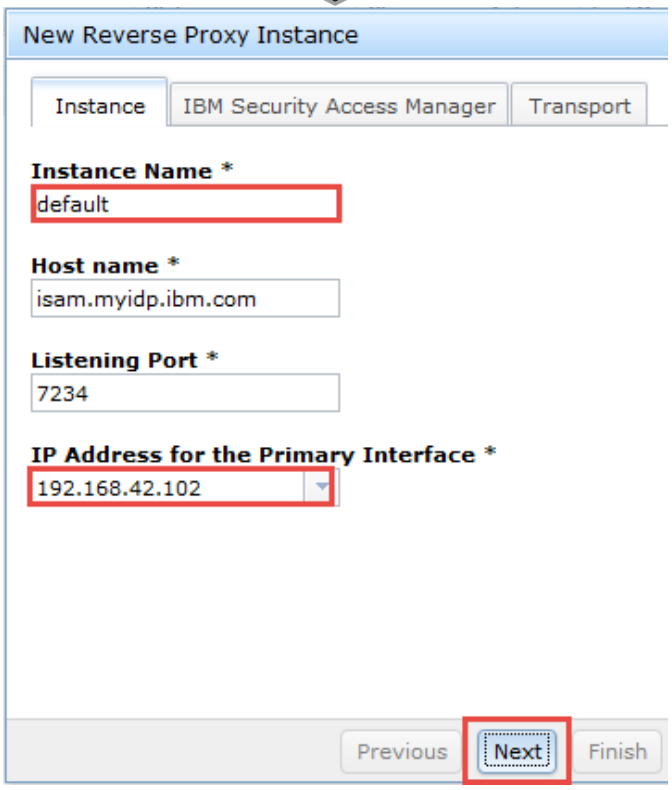


In the top menu panel, select ***Secure Web Settings*** → ***Manage: Reverse Proxy***, as indicated above.



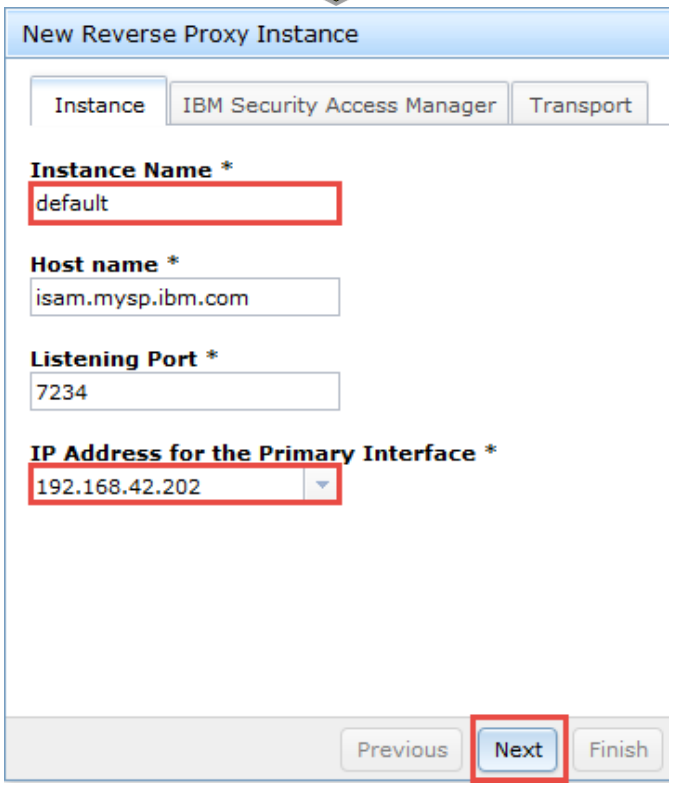
Click the ***New*** button to open the Reverse Proxy creation dialog.

IdP



Enter **default** as the *Instance Name* and select the IP address associated with the non-management interface we configured earlier (`192.168.42.102`) from the *IP Address for the Primary Interface* pull-down list.

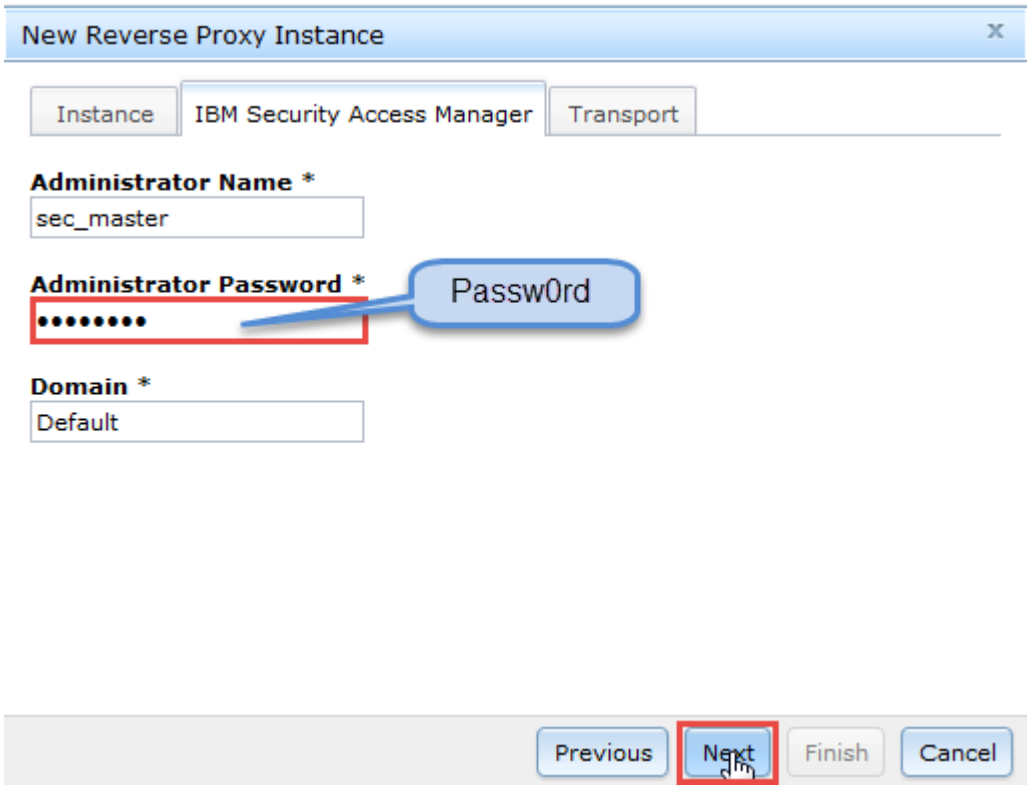
SP



Enter **default** as the *Instance Name* and select the IP address associated with the non-management interface we configured earlier (`192.168.42.202`) from the *IP Address for the Primary Interface* pull-down list.

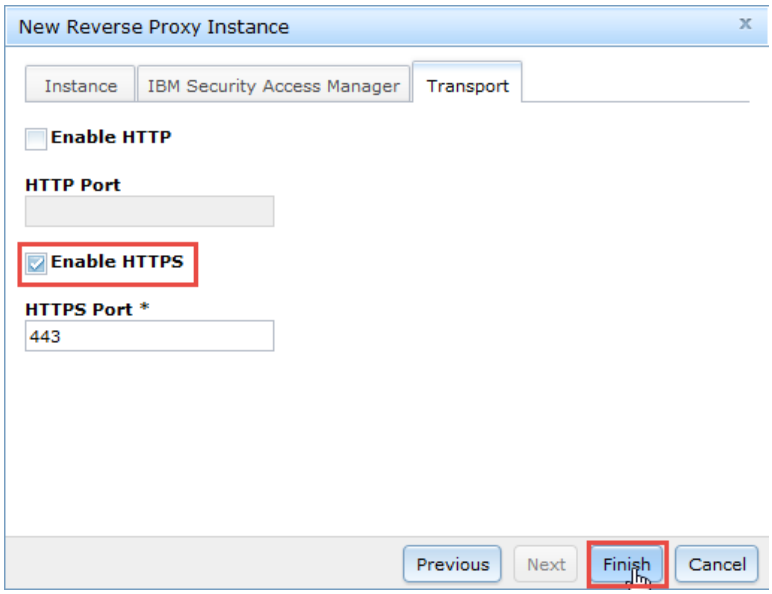
Ensure the *Host name* and *Listening Port* default correctly to the values shown above.

Click ***Next*** to progress to the next configuration panel.



Enter **`Passw0rd`** as the (ISAM) *Administrator Password*. Ensure the other fields default correctly as shown above.

Click ***Next*** to progress to the next configuration panel.

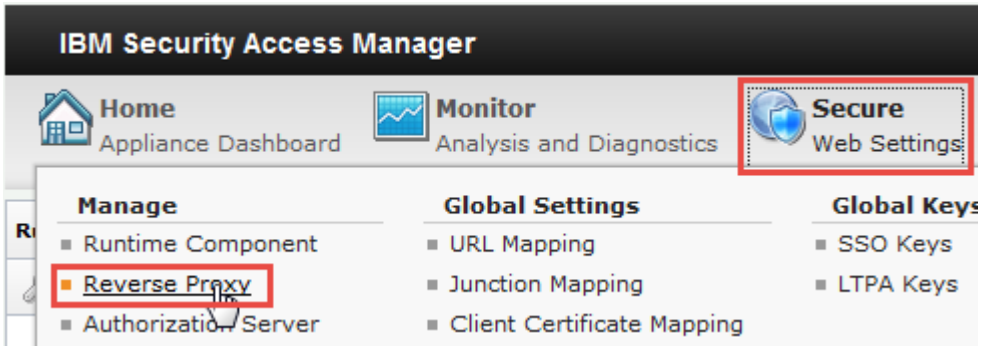


Select the checkbox for *HTTPS* and ensure the "*HTTP Port*" is set to **`443`**. Click ***Finish*** to create the Reverse Proxy instance.

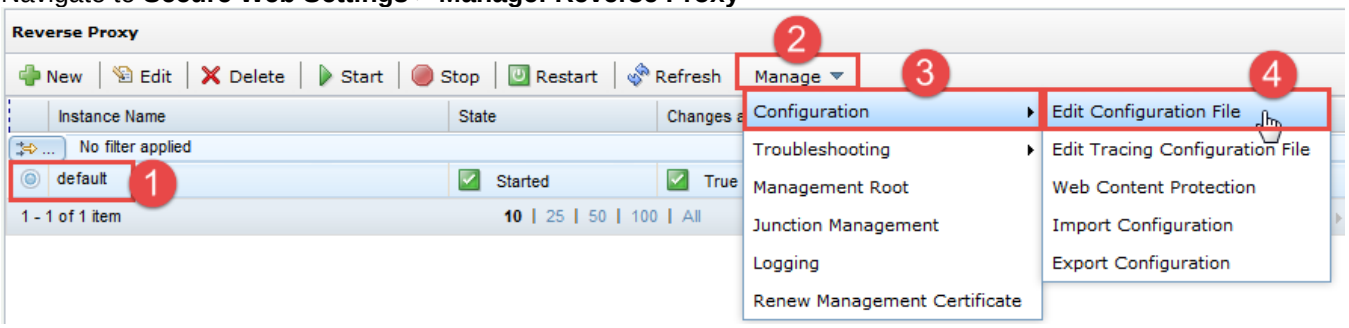
The Reverse Proxy instance is now configured and started.

5.2 Modify Reverse Proxy Instance Settings

In this section we will modify the configuration for the Reverse Proxy instance.



Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

▌ To find a location in this file, use the browsers search function. On Firefox this is activated using **Ctrl-f**.

In the [server] stanza set the following entry highlighted in red:

IdP

```
[server]

...

# web-host-name = www.webseal.com
web-host-name = www.myidp.ibm.com
```

SP

```
[server]

...

# web-host-name = www.webseal.com
web-host-name = www.mysp.ibm.com
```

In the [step-up] stanza, change *verify-step-up-user* to no, as highlighted in red:

```
[step-up]
…
#
# The following entry determines, in the event of a step-up operation,
# whether the new user ID must match the user ID from the previous
# authentication. In the situation where verify-step-up-user = yes,
# and the user IDs do not match, an error will be presented to the user.
#
verify-step-up-user = no
```

In the [session] stanza, set the following entries highlighted in red:

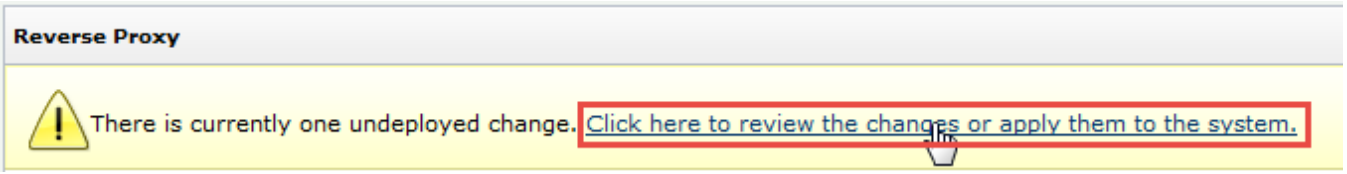
```
[session]
…
user-session-ids = yes
…
inactive-timeout = 1800
…
create-unauth-sessions = yes
```

Save changes.

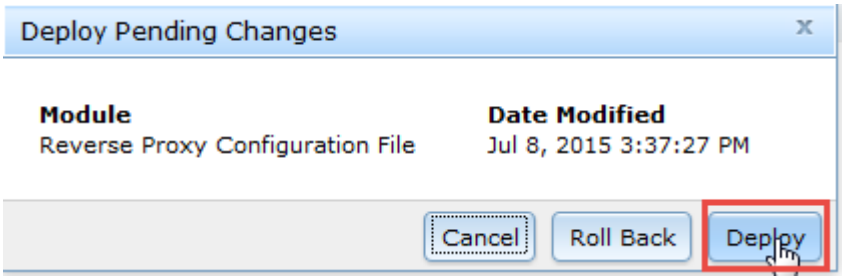
Note that you are now warned about an undeployed change. The configuration changes are not active yet.

5.3 Deploy the Changes and Restart the Reverse Proxy Instance

We are now ready to deploy the configuration changes and restart the Reverse Proxy instance so the changes come into effect.

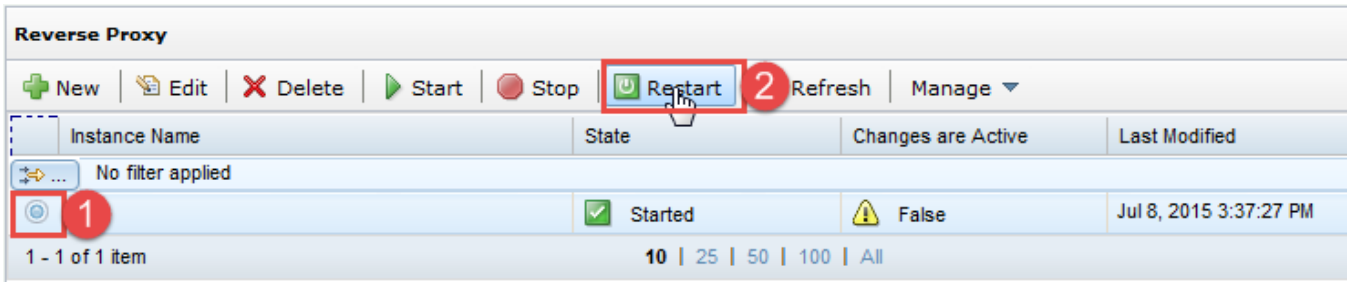


The configuration file settings we just changed were performed on a copy of the real configuration files. Press the link in the yellow warning bar to apply (or discard) the changes.



Press the ***Deploy*** button to deploy the changes to the master copy of the configuration files.

A warning message is displayed advising that the Reverse Proxy instance will need to be restarted in order for the changes to come into effect. The **Changes are Active** shows as False.



Select the checkbox next to the Reverse Proxy instance and press the ***Restart*** button – as shown above, to restart the server.

A blue message box should briefly appear once the instance has restarted. **Changes are Active** shows as **True** to reflect that the deployed configuration changes are now active.

SCRIPT-END:
The script should display the following:
INFO:WGAManager:Configuring WebSEAL Instance
INFO:WGAManager:Successfully configured WebSEAL Instance

6 Create SAML 2.0 Identity Provider federation

This section is completed only for the Identity Provider.
You will configure the Service Provider in a later section.

6.1 Upload keystore files

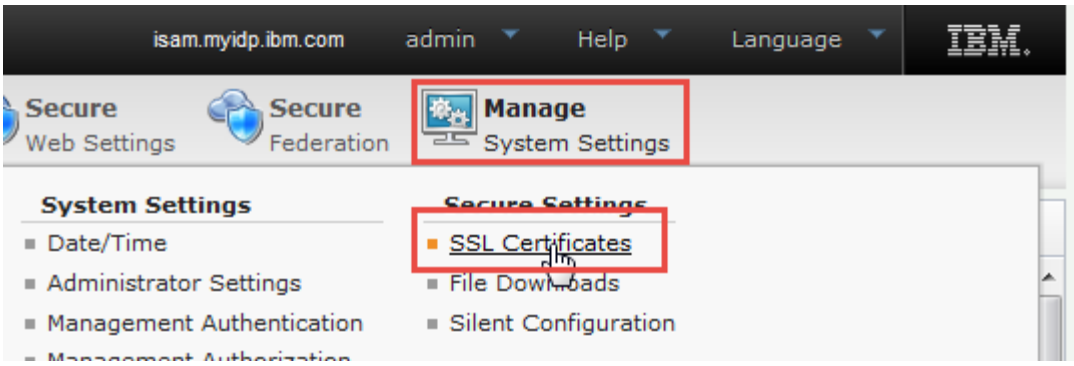
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Keystore**

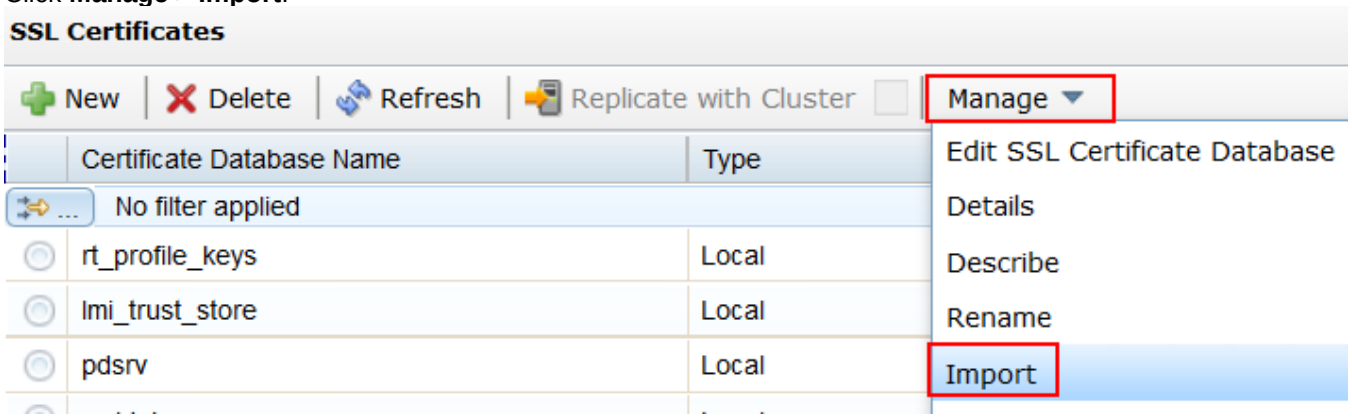
If you use this script, skip to the corresponding SCRIPT-END notice

A sample keystore and stash file for the IdP is available in the *…/provided_files/myidpkeys* directory. The keystore contains all the certificates required for a SAML flow to work based on the configuration used in this document.

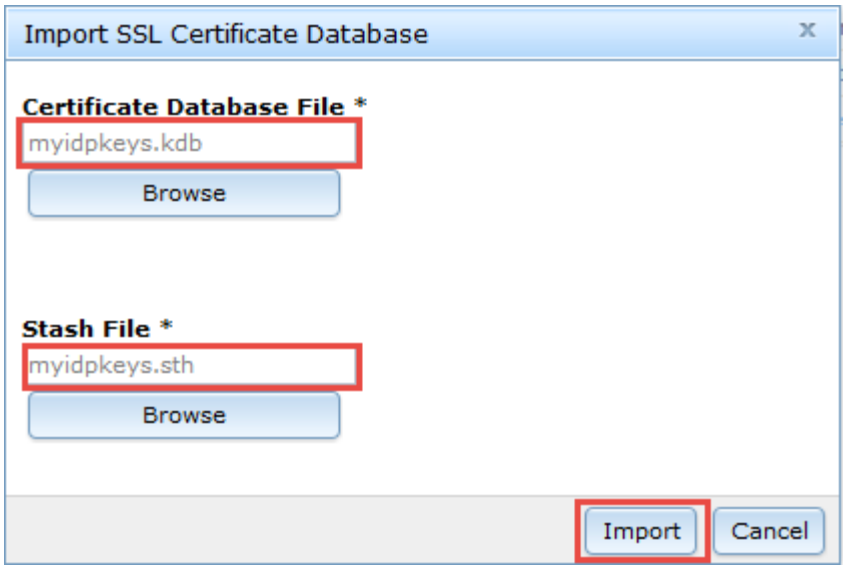
Navigate to **Manage System Settings > Secure Settings: SSL Certificates**.



Click **Manage > Import**.

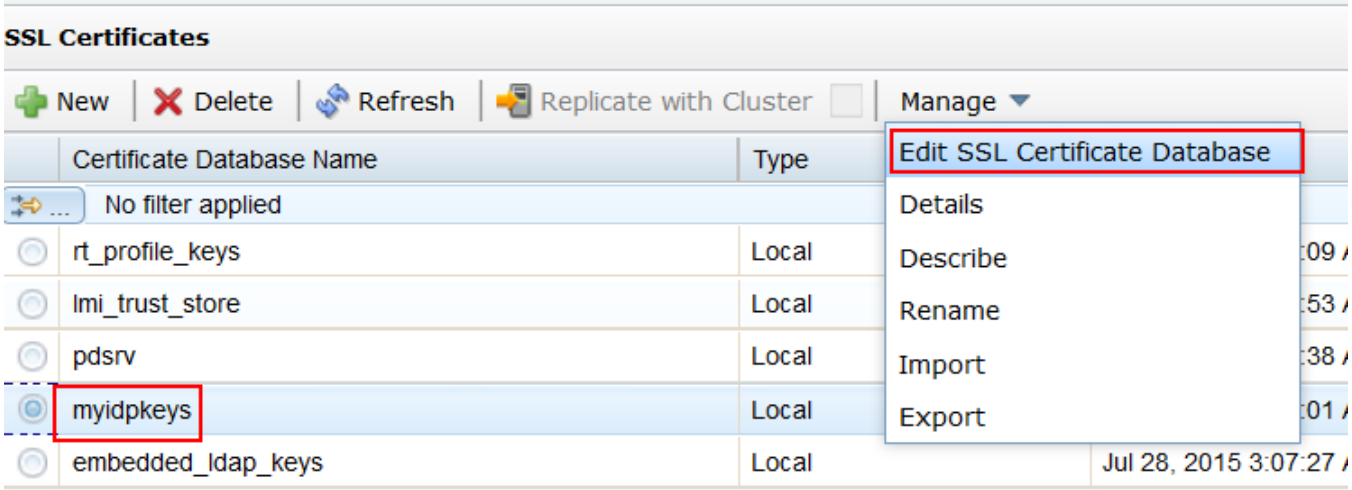


Select the certificate database and stash file from the *…/provided_files/myidpkeys* directory and click **Import**.

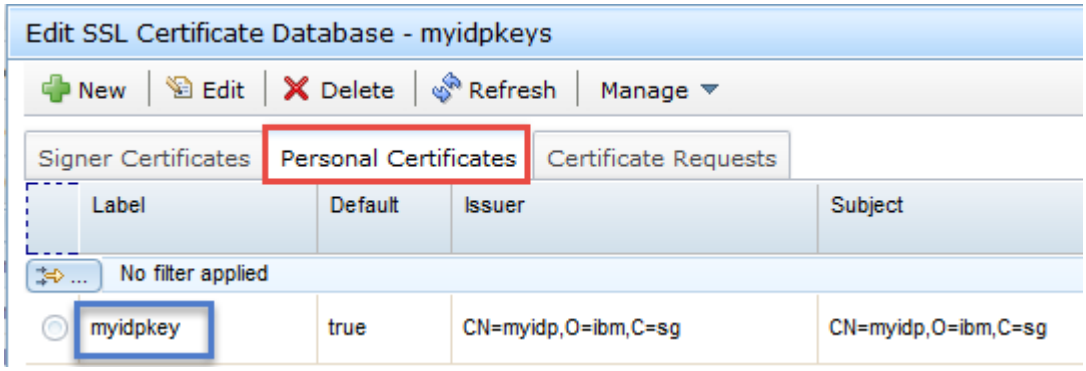


A warning will be displayed at the top of the window. Click the link to activate the configuration change you have just made. A pop-up dialog is displayed showing the pending changes. **Deploy** the changes.

Select the keystore. Click **Manage > Edit SSL Certificate Database.**



Verify that the **Personal Certificate** is present.



Once verified, close the dialog.

SCRIPT-END:
The script should display the following
INFO:WGAManager:Configuring keystore for IdP
INFO:WGAManager:Successfully uploaded and configured keystore

6.2 Upload mapping rules

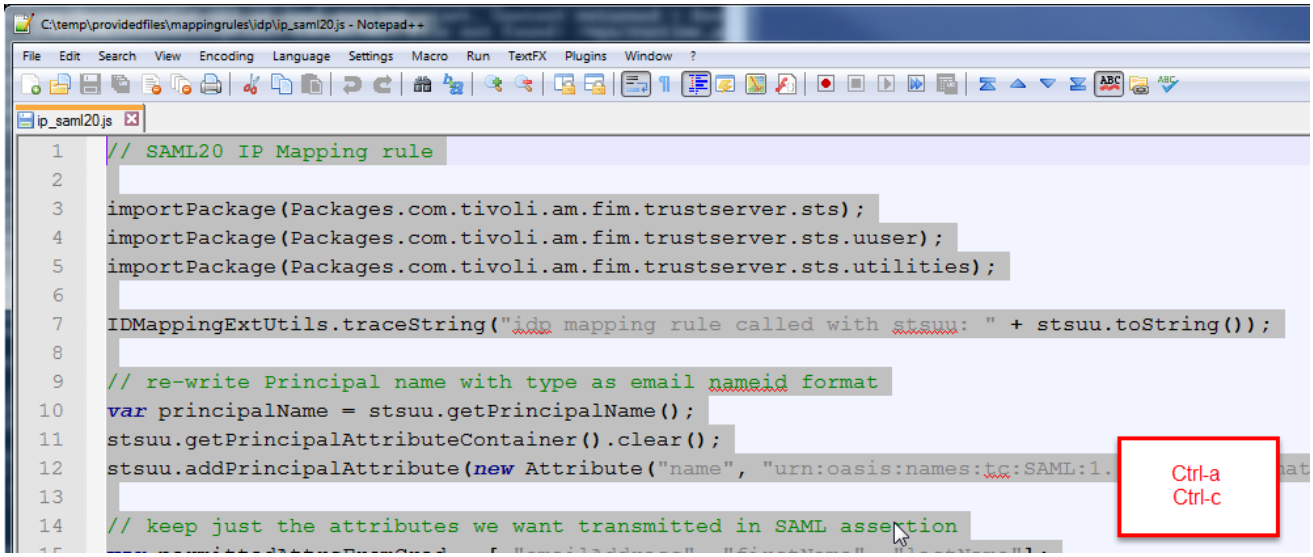
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py –configure Upload_Mapping_Rules**
If you use this script, skip to the corresponding SCRIPT-END notice

This document makes use of a number of JavaScript mapping rules. These need to be created on the Identity Provider appliance. We will actually create quite a few mapping rules at this time although the SAML federation will initially use only the first of these rules.

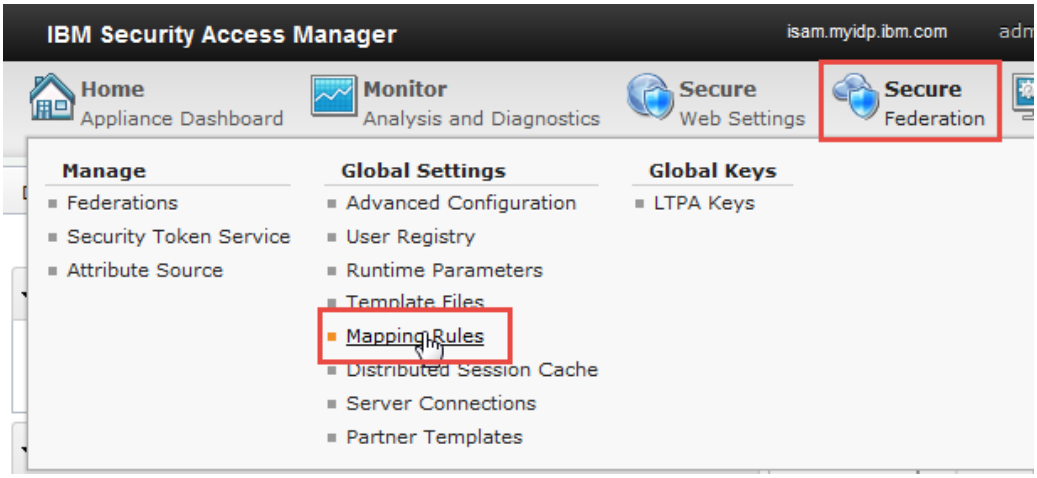
When using the appliance console to create Mapping Rules, cut-and-paste is used to load the JavaScript content of the rules. Before we get started, we need to open our first rule in a text editor so we can copy it.

Go to the **…/providedfiles/mappingrules/idp** directory and open the **ip_saml20.js** file in a text editor.

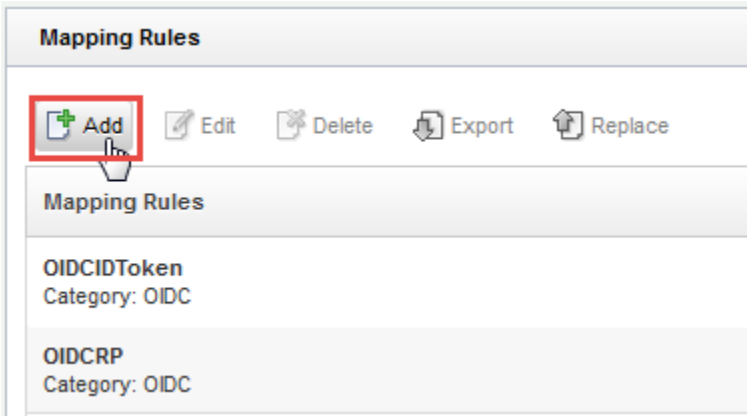


Select all the text in the file and then copy it. On Windows you can use **Ctrl-a** to select all and **Ctrl-c** to copy.

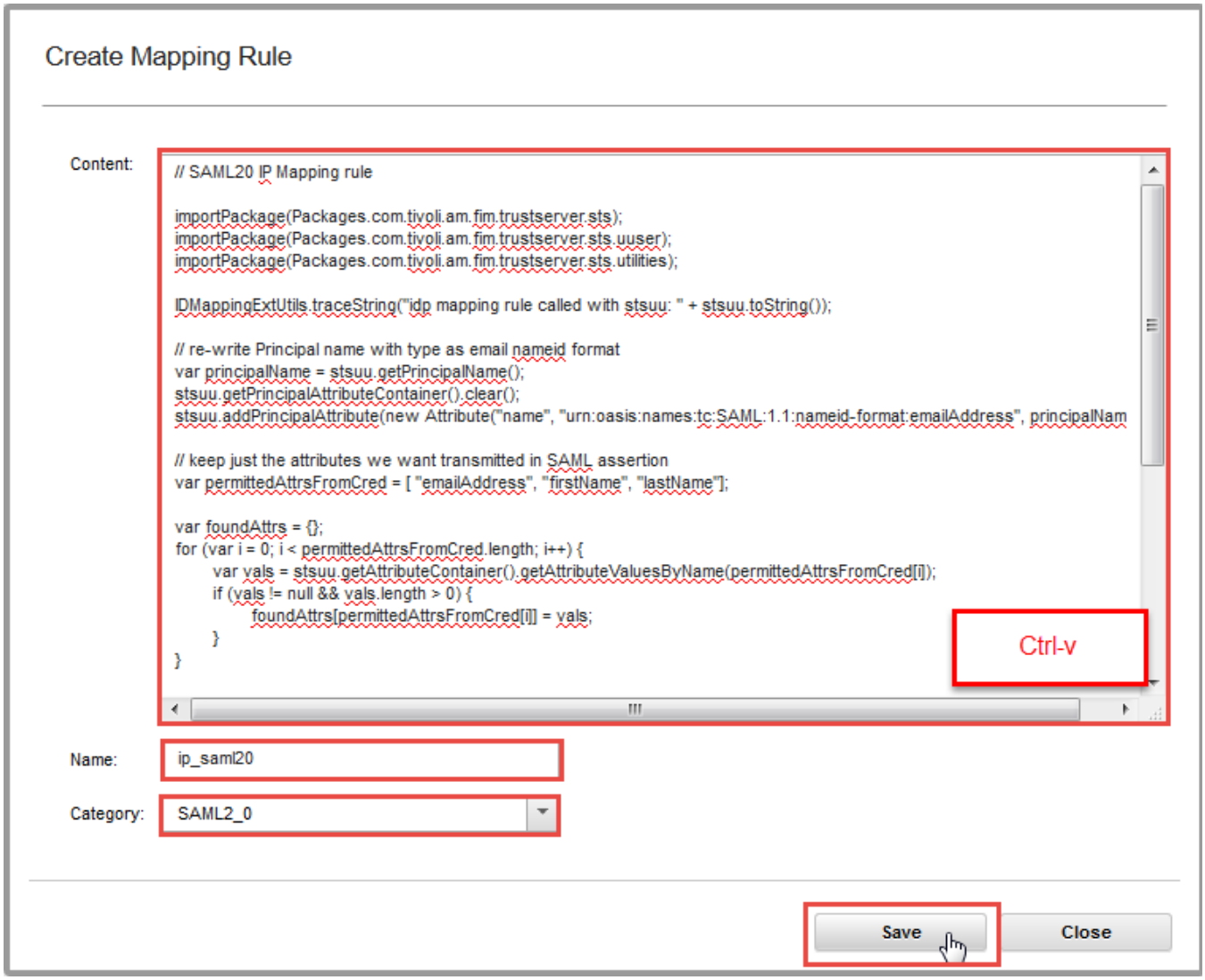
Now we're ready to create a Mapping Rule on the appliance with this content.



In the LMI Administration console, navigate to **Secure Federation→Global Settings: Mapping Rules**.



Click **Add** to add a new mapping rule.



Paste the rule text into the *Content* box. On Windows you can use **Ctrl-c** to paste.

Enter **ip_saml20** as the rule *Name* and select **SAML2_0** as the *Category*.

Click **Save** to save the new Mapping Rule.

Repeat the process above for all of the files in the *…/providedfiles/mappingrules/idp* directory

Once all Mapping Rules are loaded, **deploy** the pending changes.

SCRIPT-END:

The script should display the following

```
INFO:FederationManager:Upload all mapping rules
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
```

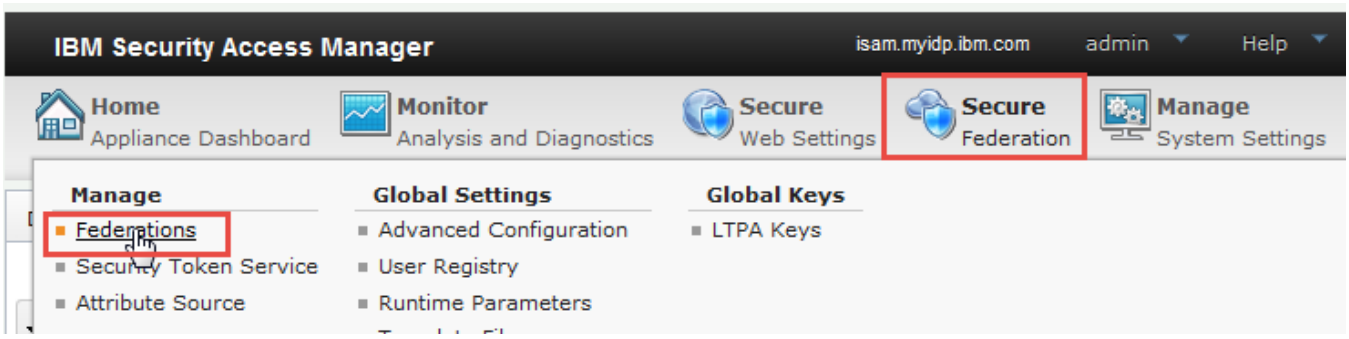
6.3 Create federation

SCRIPT-START:

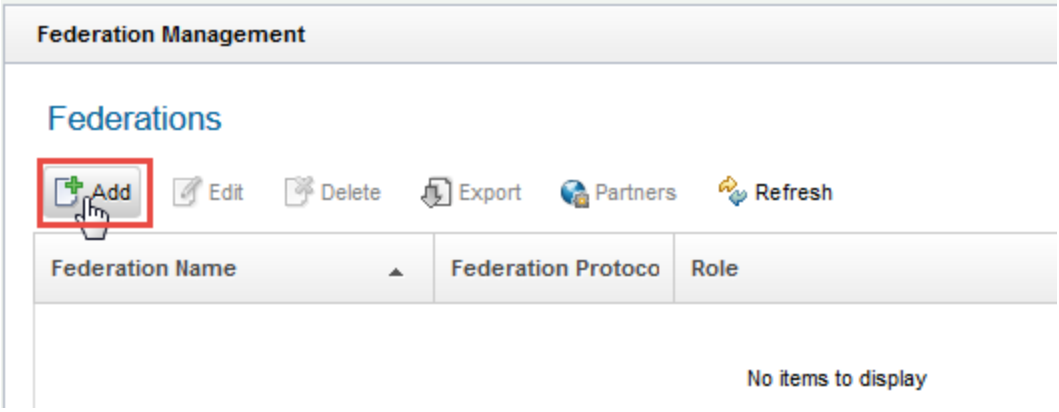
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Federation**

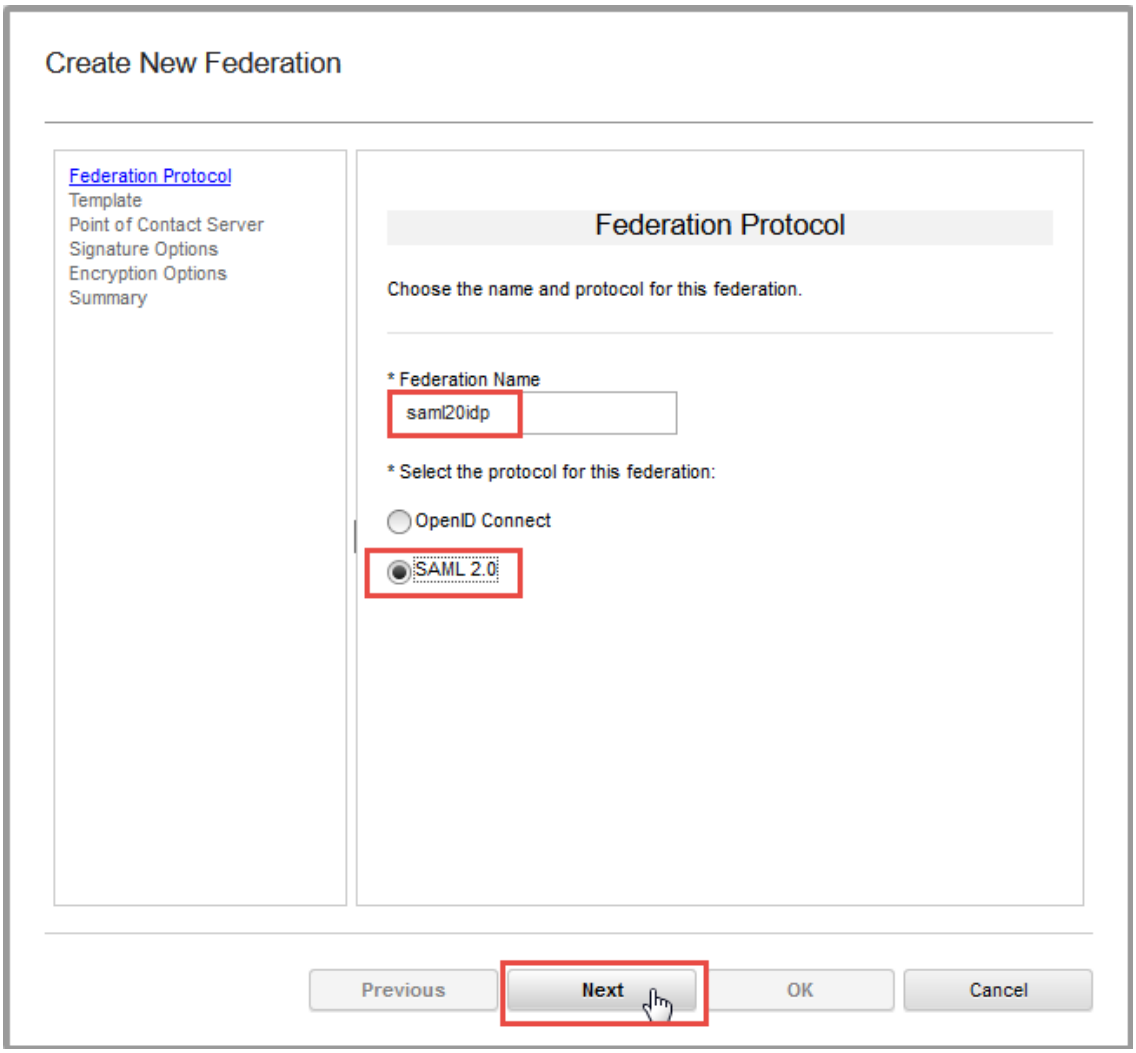
If you use this script, skip to the corresponding SCRIPT-END notice



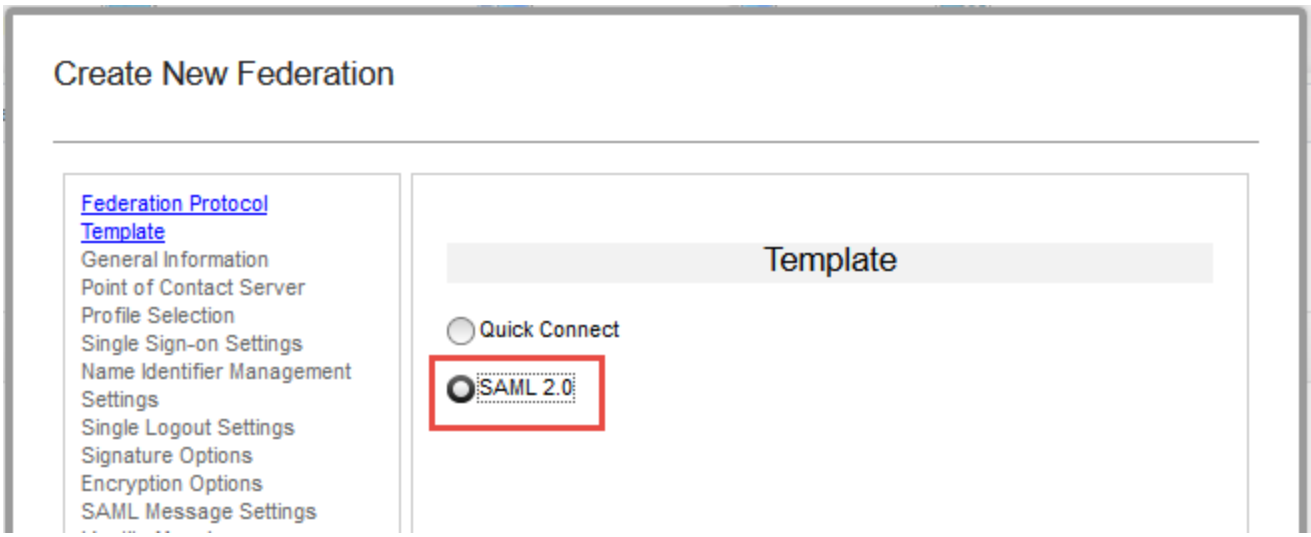
Using the administration console, navigate to **Secure Federation→Manage: Federations**.



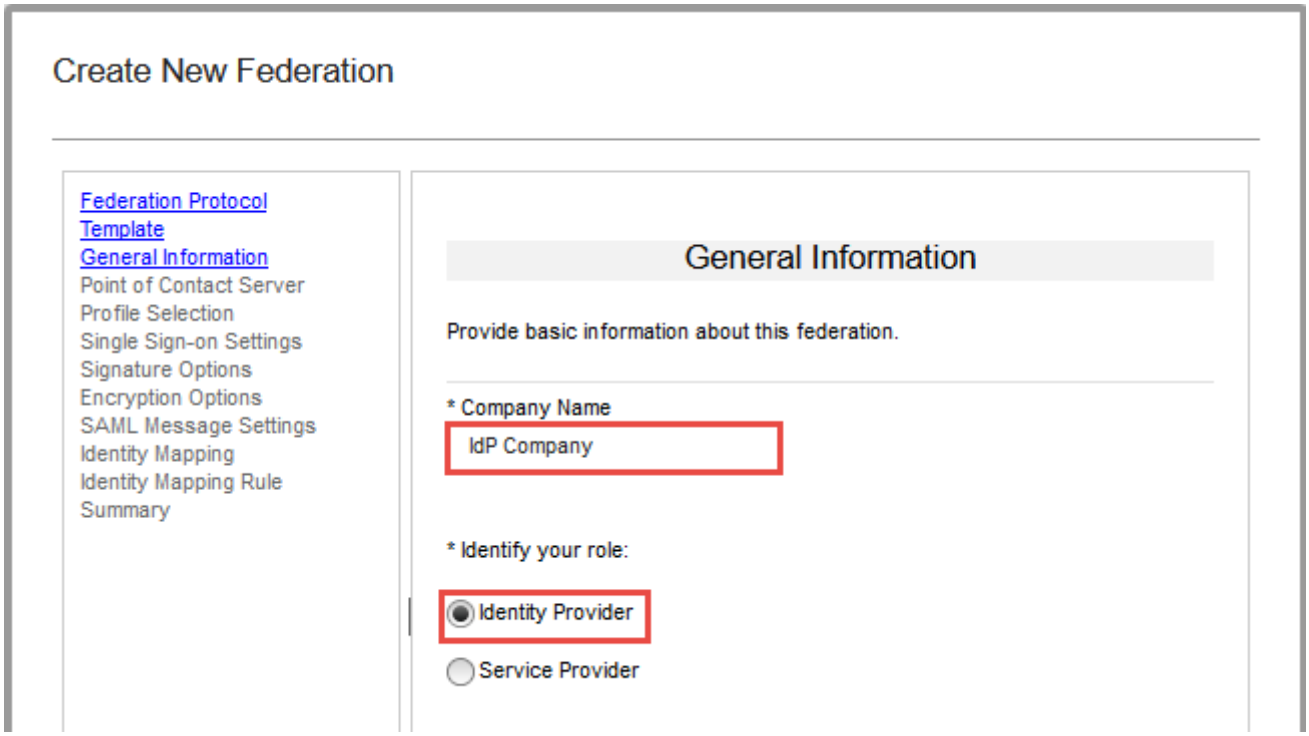
Click **Add** to create a new federation.



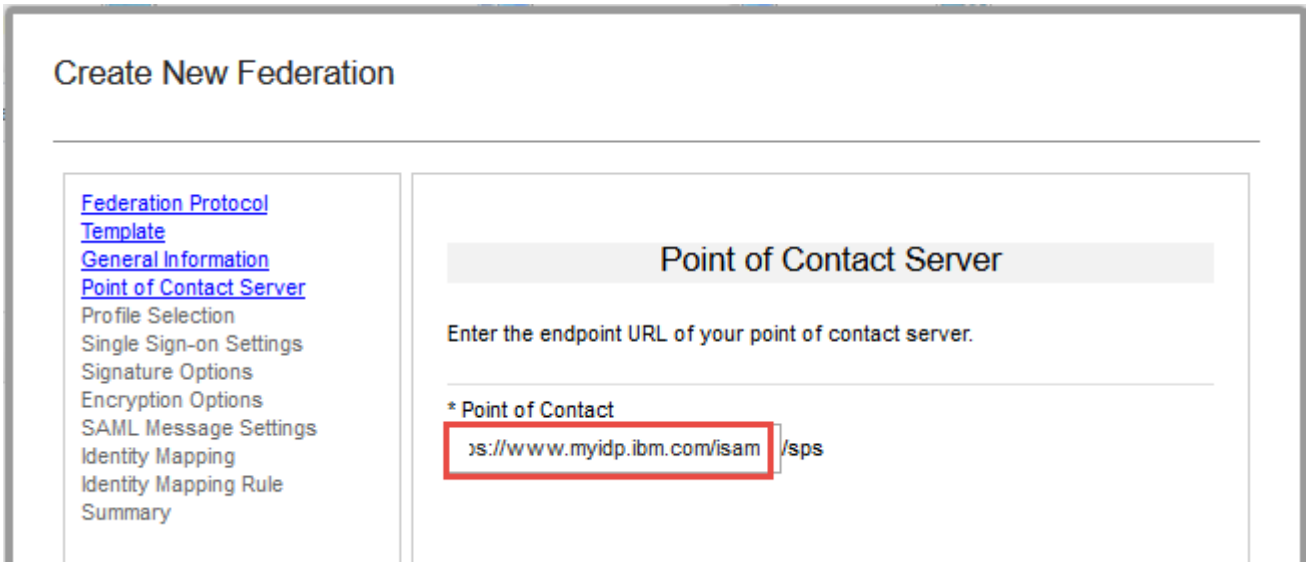
Create a new **SAML 2.0** federation named **saml20idp** as shown and click **Next**.



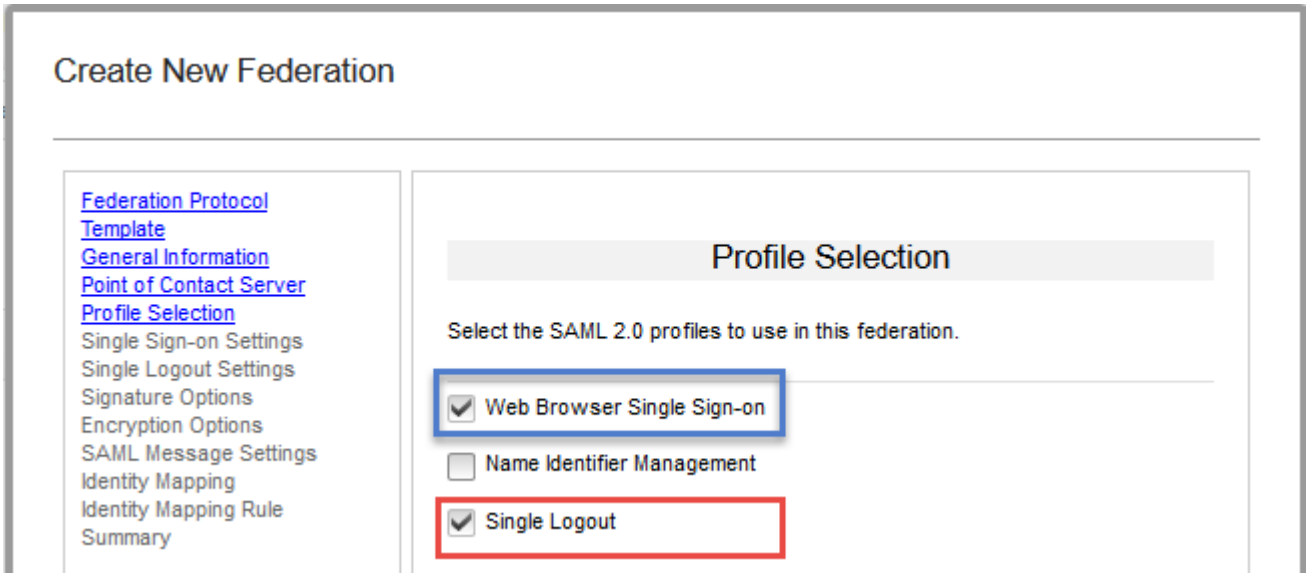
On the template page, select **SAML 2.0** and click **Next**.



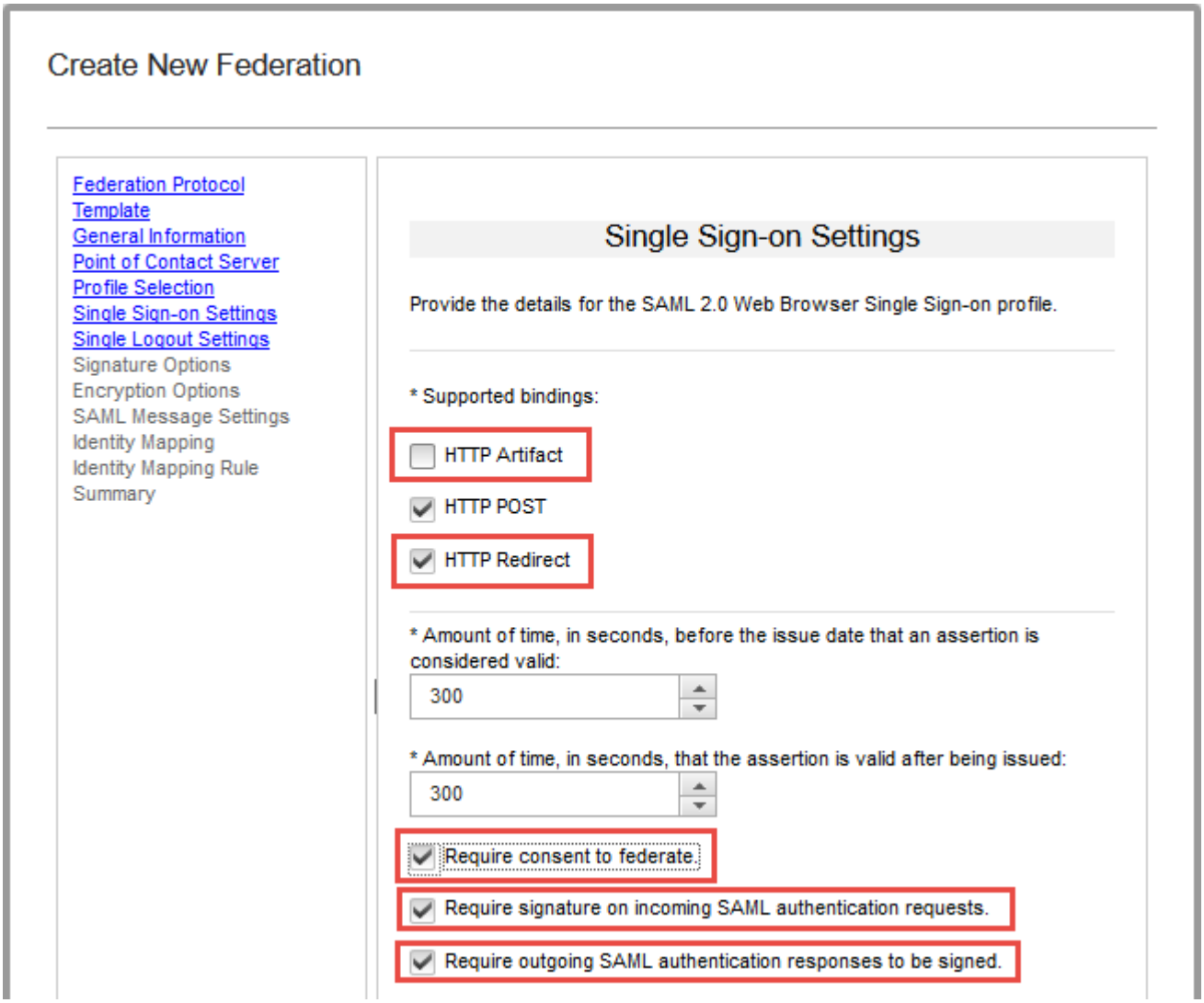
On the General Information panel, enter **IdP Company** as the Company Name, select **Identity Provider** as the role, and click **Next**.



On the Point of Contact Server panel, enter **https://www.myidp.ibm.com/isam** and click **Next**.



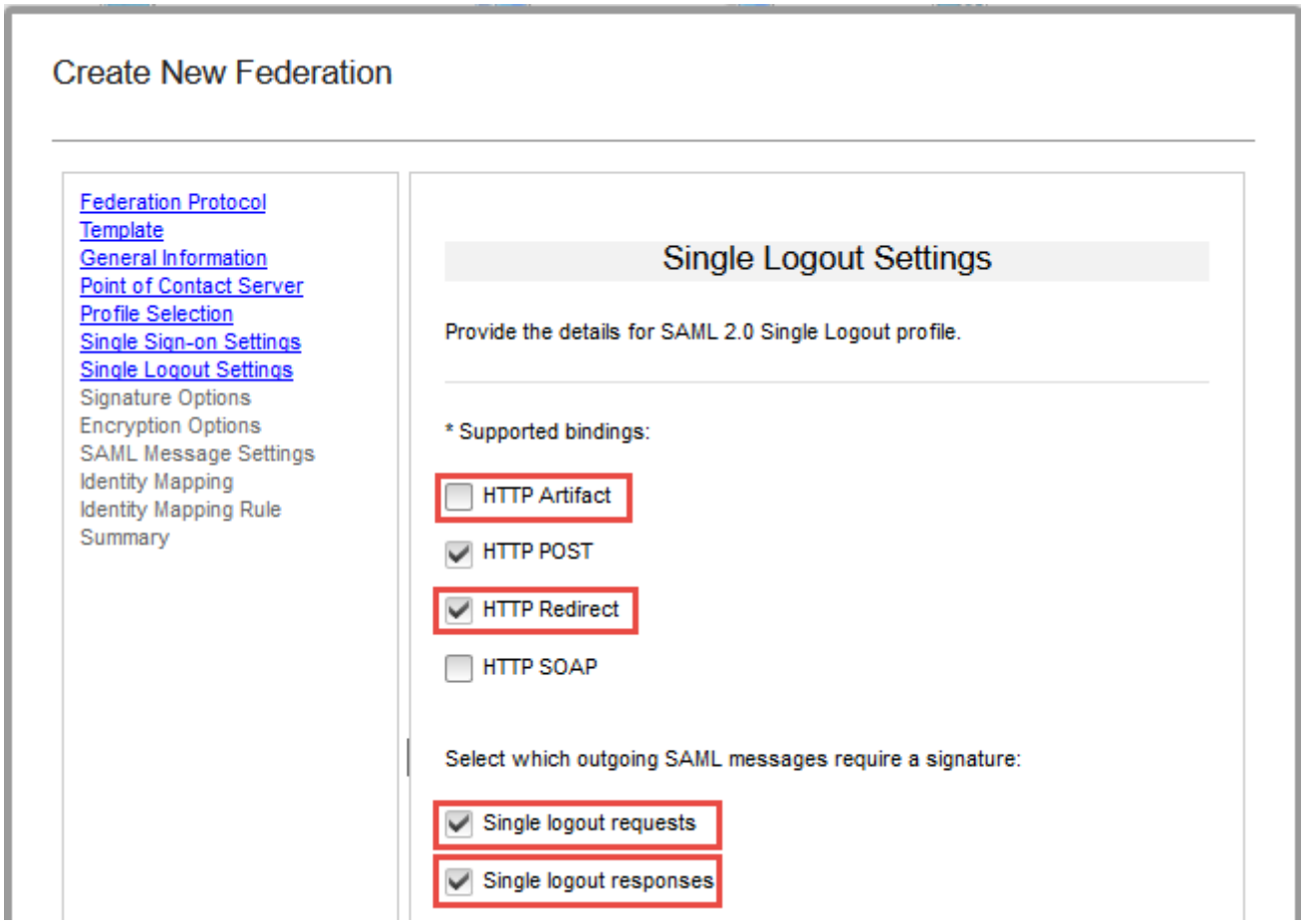
On the profile selection panel, leave **Web Browser Single Sign-on** selected and also select **Single Logout**. Then press **Next**.



On the Single Sign-on settings panel, deselect **HTTP Artifact** and select **HTTP Redirect**.

Select checkboxes for **Require Consent to Federation**, **Require signature on incoming SAML authentication requests** and **Require outgoing SAML authentication responses to be signed**.

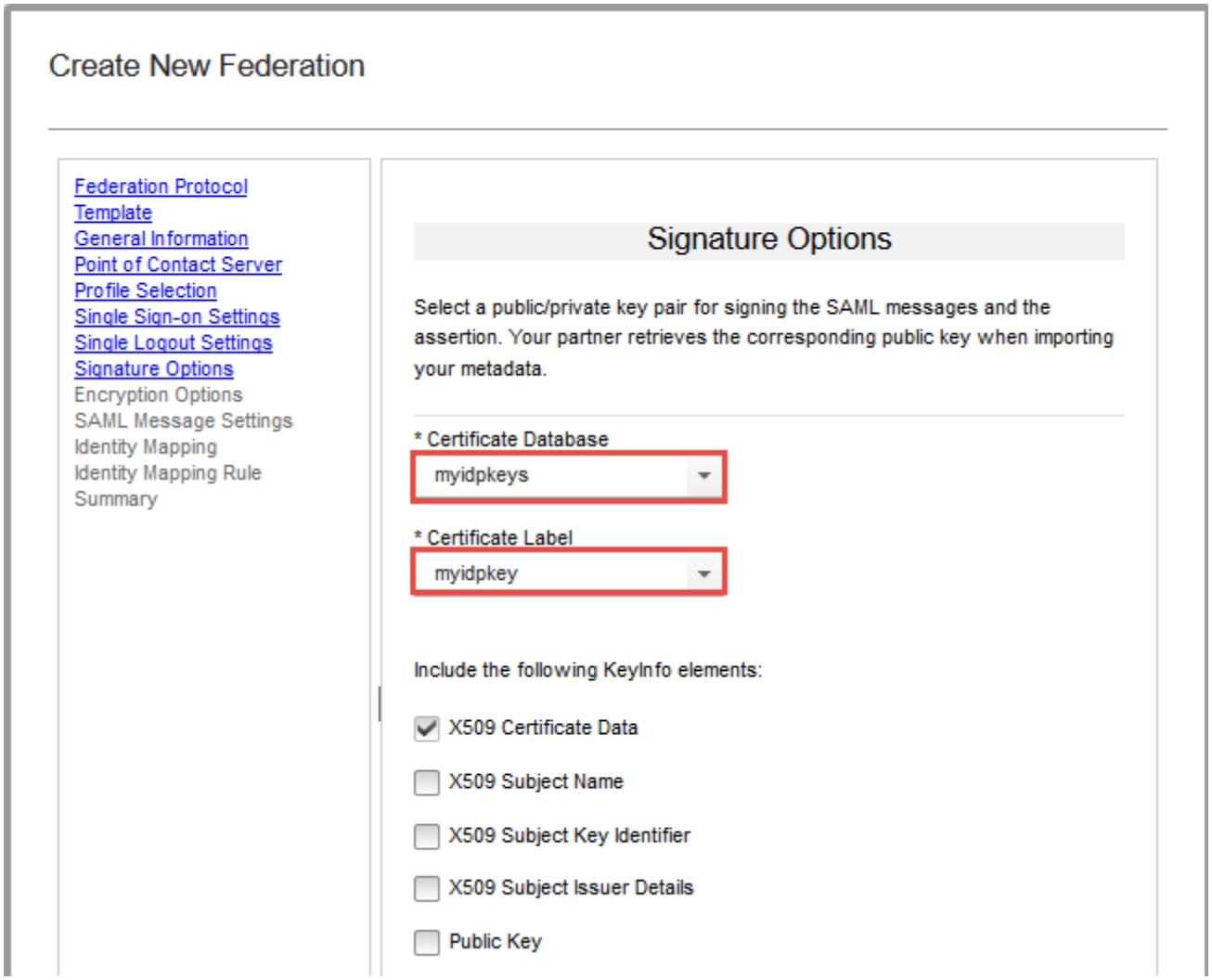
Then click **Next**.



On the Single Logout Settings panel, deselect **HTTP Artifact** and select **HTTP Redirect**.

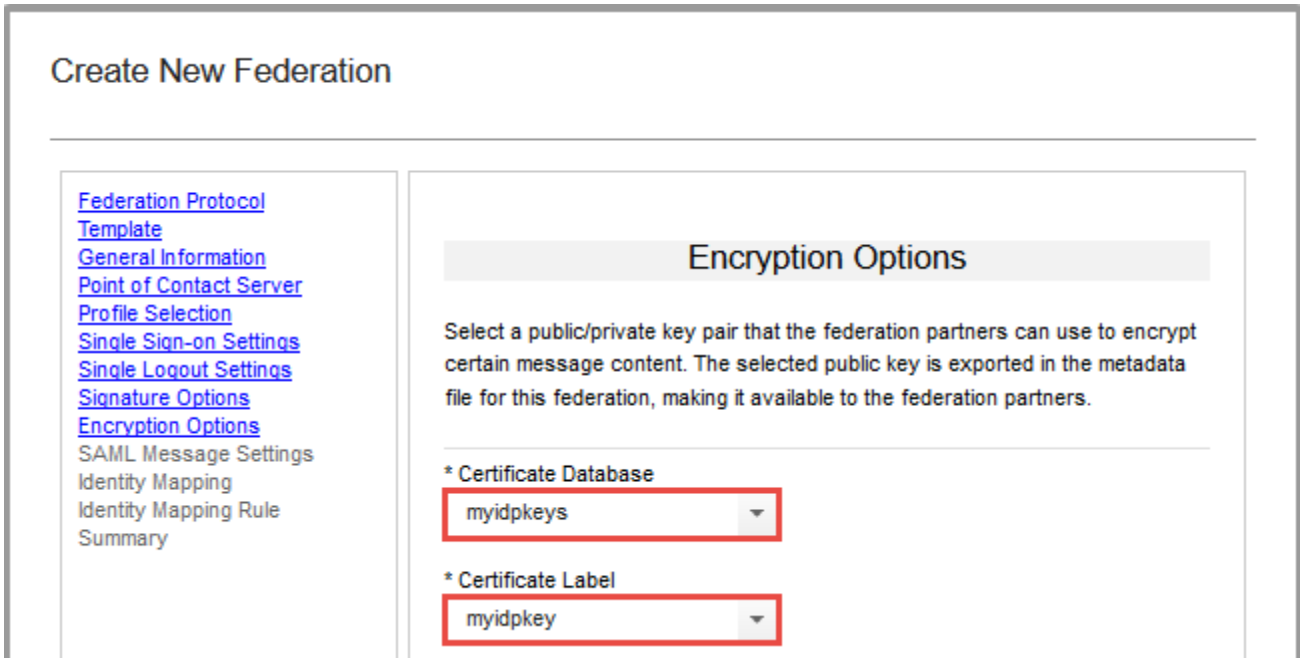
Select checkboxes to require signatures for **Single logout requests** and **Single logout responses**.

Then click **Next**.



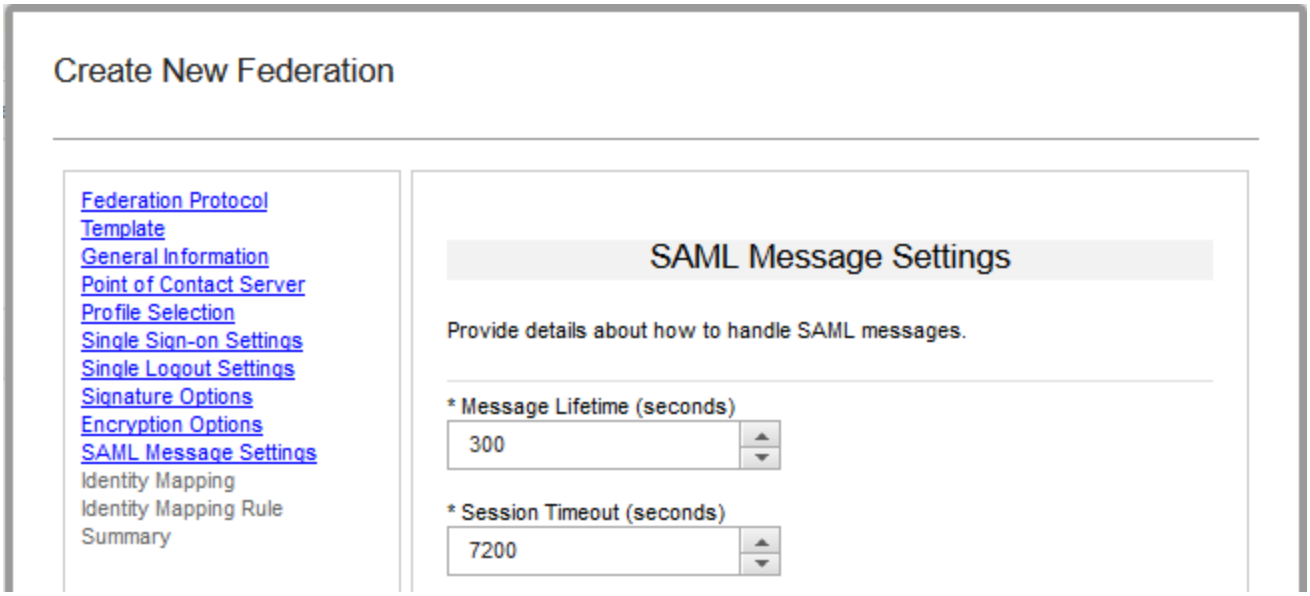
On the Signature Options panel, select the **myidpkeys** Certificate Database, and the **myidpkey** Certificate Label.

Then click **Next**.

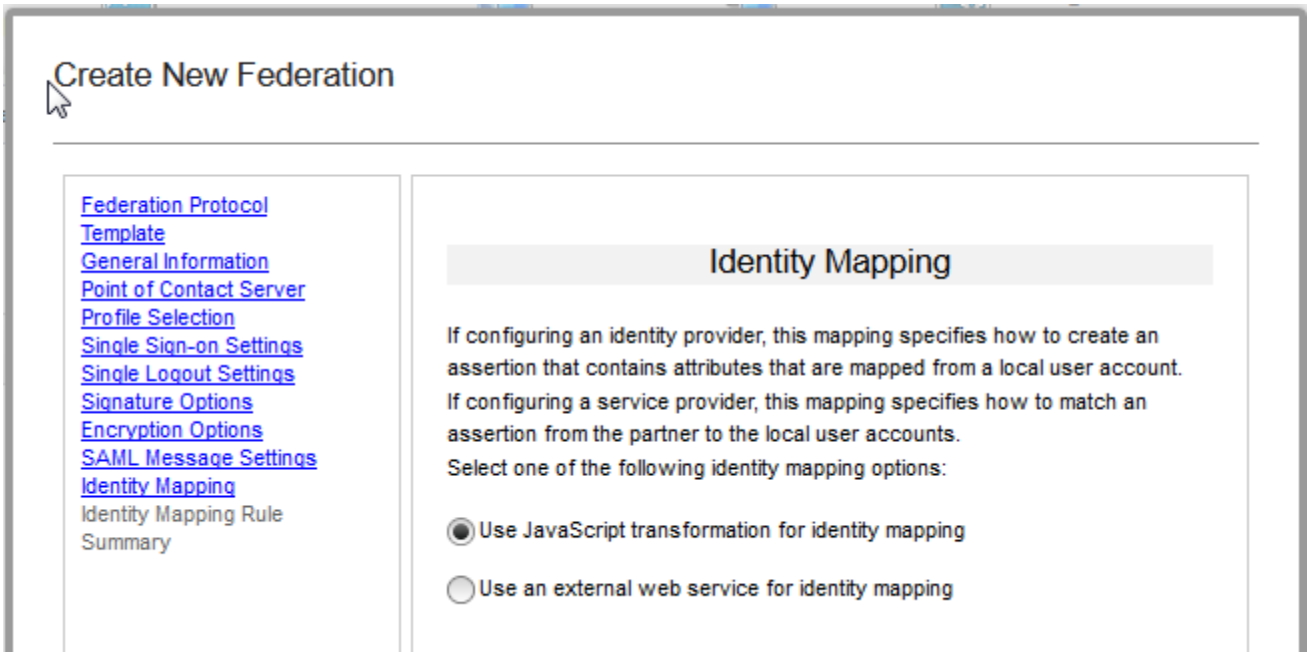


On the Encryption Options panel, select the **myidpkeys** Certificate Database and the **myidpkey** Certificate Label.

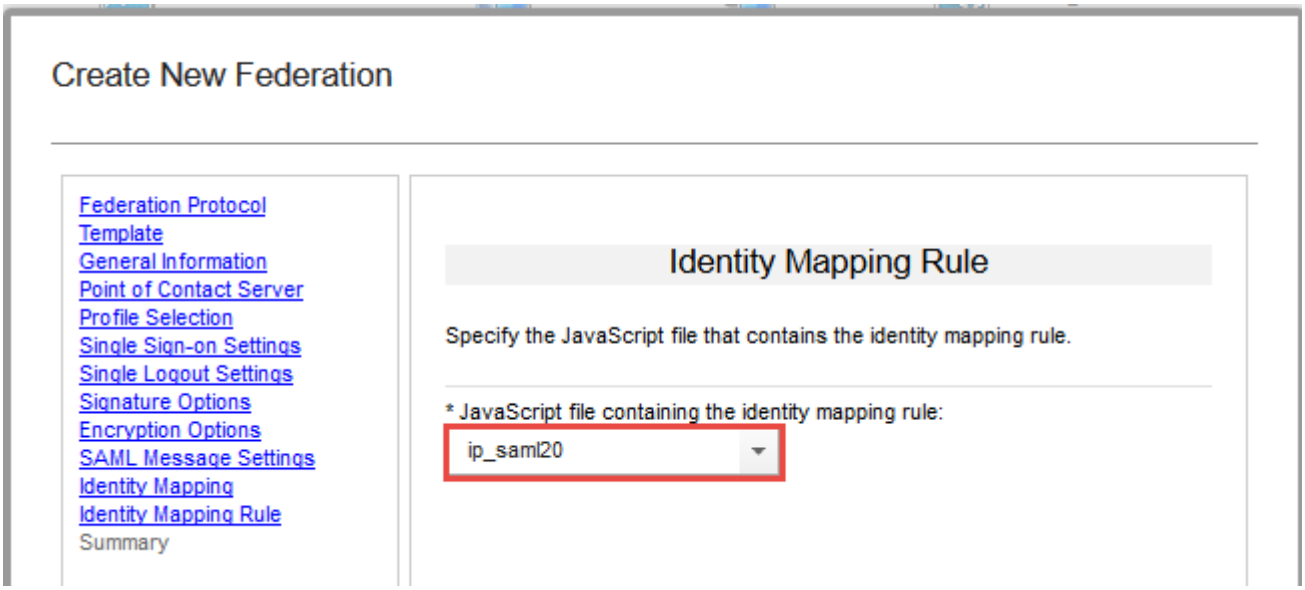
Then click **Next**.



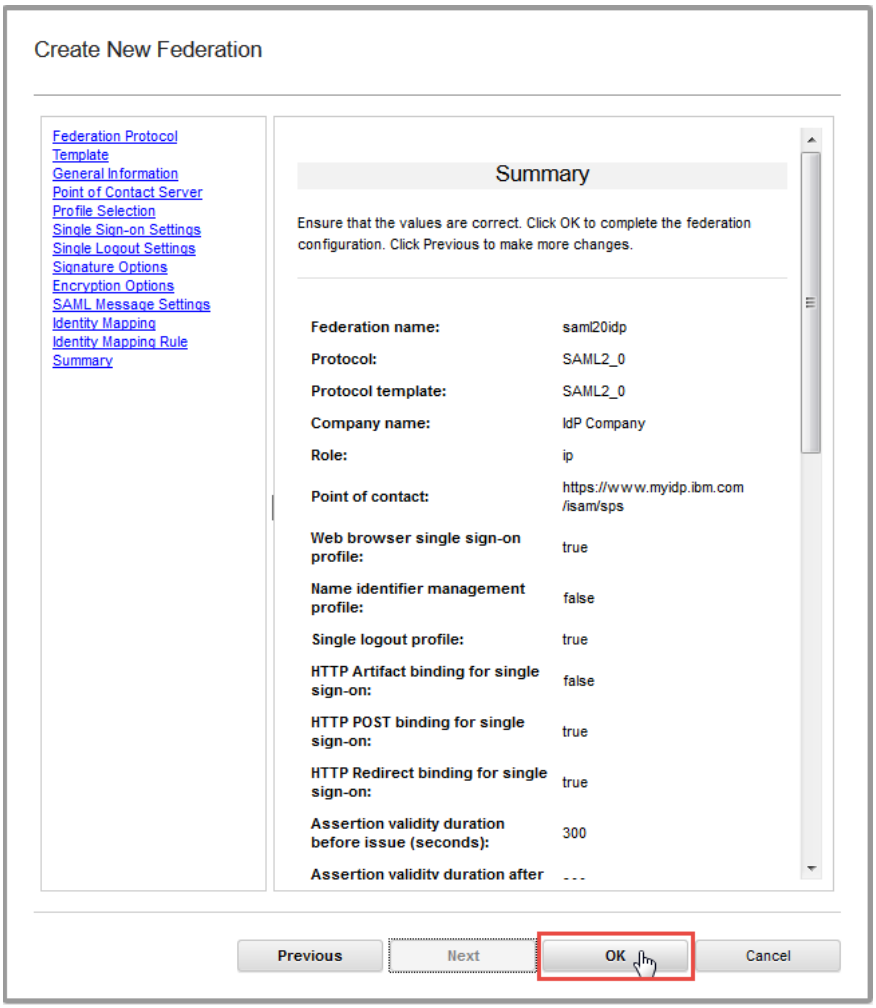
On the SAML Message Settings panel, leave values at their defaults and click **Next**.



On the Identity Mapping panel, we will use the default of **Use Javascript transformation for identity mapping** so just click **Next**.



On the Identity Mapping Rule panel, select **ip_saml20** from the drop-down list and click **Next**.



On the Summary panel, click **OK** to create the federation.

Follow on-screen instructions to **deploy** pending changes.

SCRIPT-END:
The script should display the following
INFO:FederationManager:Configuring the IdP Federation
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the IdP
FederationINFO:FederationManager:Successfully configured the IdP Federation

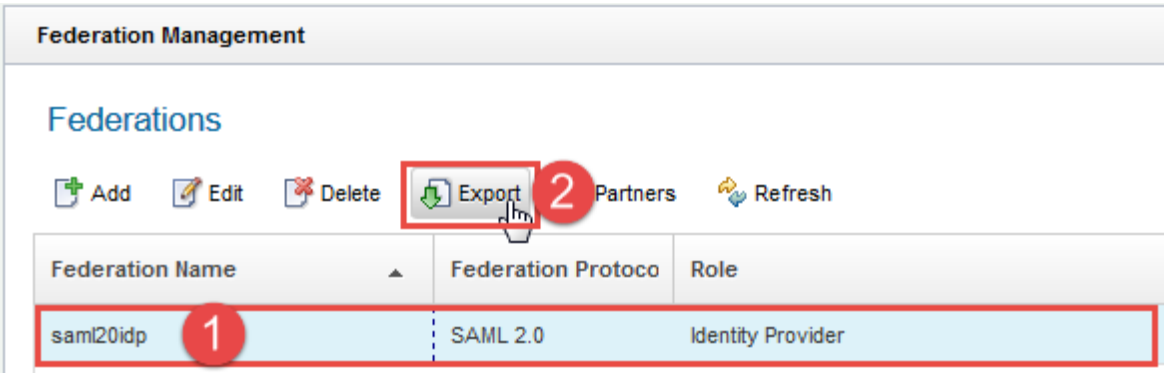
6.4 Export meta-data

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the IdP, run this script: **SAMLIPConfig.py -configure Export_Metadata**

If you use this script, skip to the corresponding SCRIPT-END notice

Using the administration console, navigate to **Secure Federation -> Manage: Federations**.



Click on the **saml20idp** federation and click **Export**.

This will start the download of the federation metadata. Save the file to:
…/providedfiles/Automation/tmp/ipmetadata.xml

> It is important to save the metadata file to this exact location if you are planning to use the automated scripts to import this metadata to the partner.

SCRIPT-END:
The script should display the following
INFO:FederationManager:Exporting Metadata
INFO:FederationManager:Successful export of metadata

The metadata file will be exported to *…/providedfiles/Automation/tmp/ipmetadata.xml*

7 Create SAML 2.0 Service Provider federation

This section is completed only for the Service Provider.
The Identity Provider creation is described in the previous section.

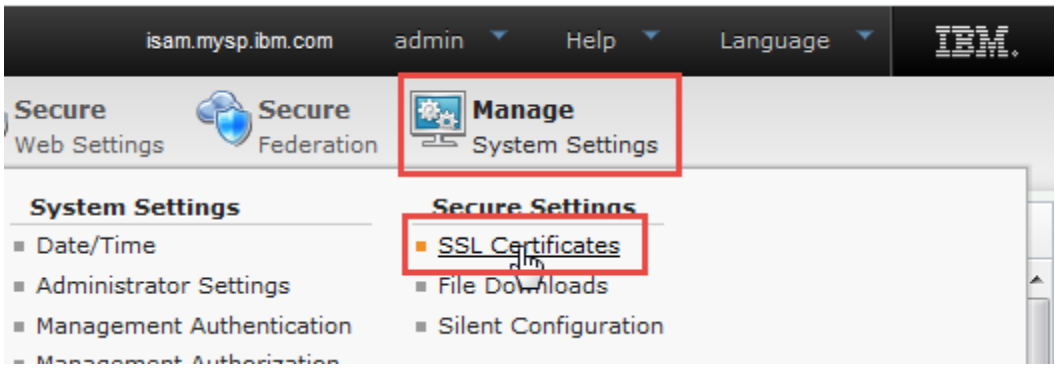
7.1 Uploading keystore files

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

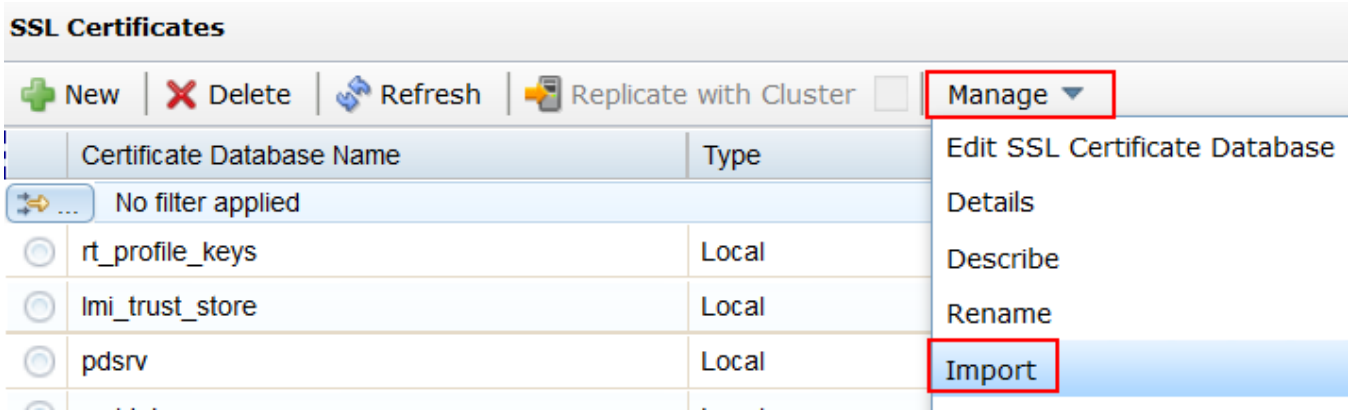
For the SP, run this script: **SAMLSPConfig.py -configure Keystore**

If you use this script, skip to the corresponding SCRIPT-END notice

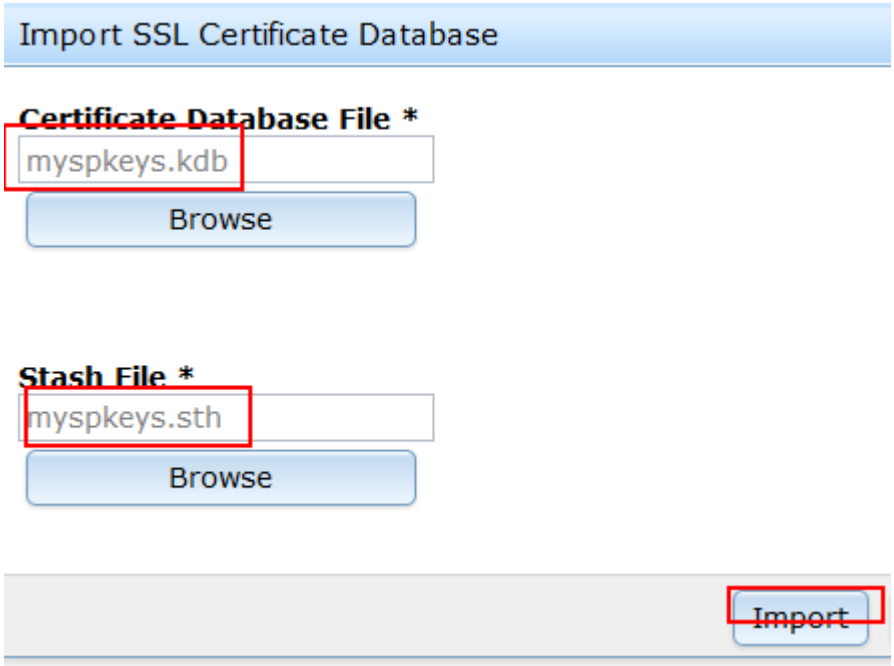
A sample keystore and stash file for the SP are available in the *…/providedfiles/myspkeys* directory. The keystore contains all the certificates required for a SAML flow to work based on the configuration described in this document.



Navigate to **Manage System Settings > Secure Settings: SSL Certificates**

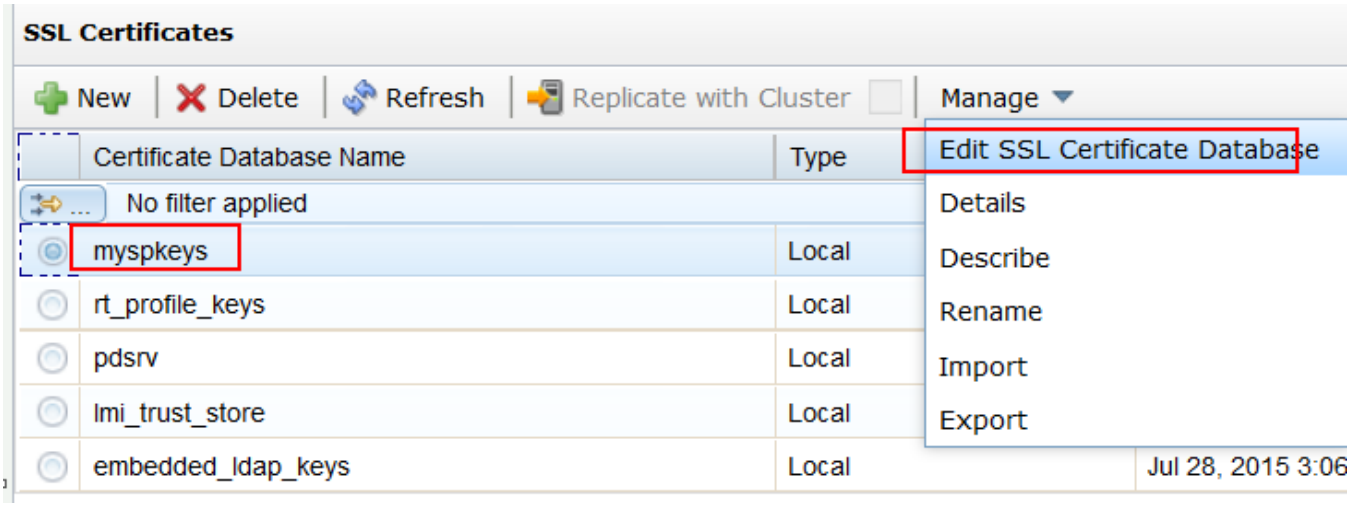


Click **Manage > Import**.

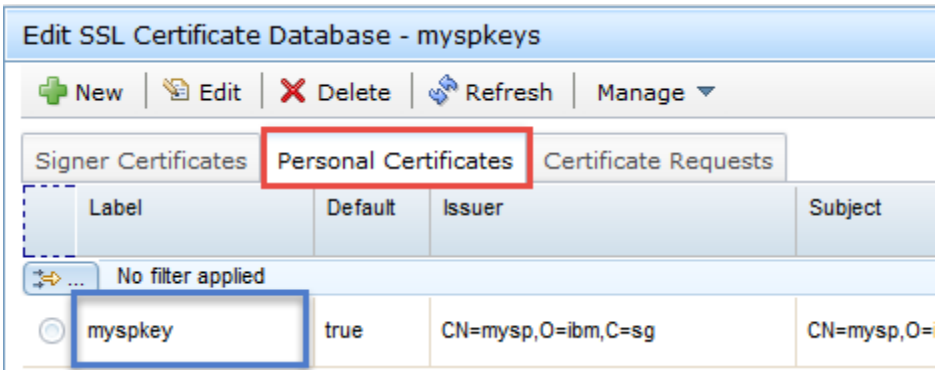


Select the certificate database and stash file. Click **Import**.

A warning will be displayed at the top of the window. Click the link to activate the configuration change you have just made. A pop-up dialog is displayed showing the pending changes. Click deploy.



Select the keystore. Click **Manage > Edit SSL Certificate Database.**



Verify that the **Personal Certificate** is present.

Once verified, close the dialog.

SCRIPT-END:
The script should display the following
INFO:WGAManager:Configuring keystore for SP
INFO:WGAManager:Successfully uploaded and configured keystore

7.2 Upload mapping rules

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

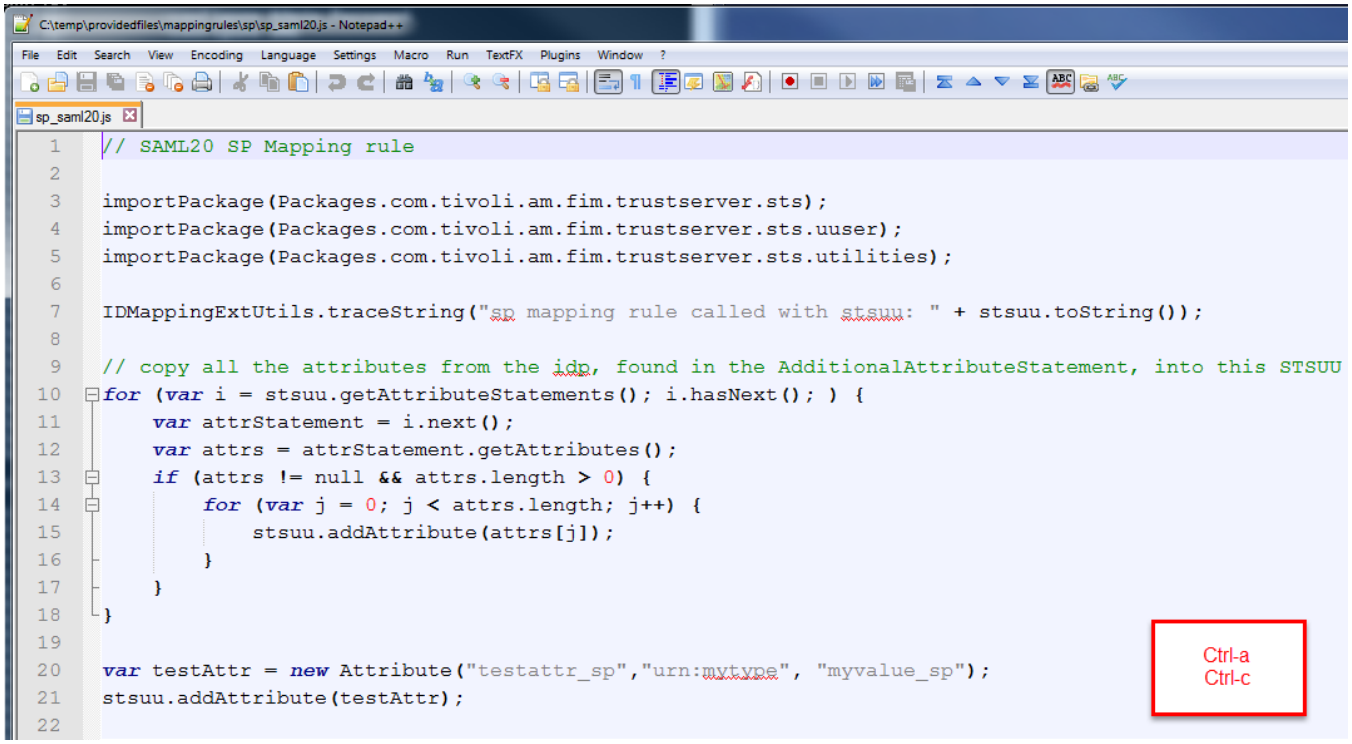
For the SP, run this script: **SAMLSPConfig.py -configure Upload_Mapping_Rules**

If you use this script, skip to the corresponding SCRIPT-END notice

This document makes use of a number of JavaScript mapping rules. These need to be created on the Identity Provider appliance. We will actually create a few mapping rules at this time although the SAML federation will initially use only the first of these rules.

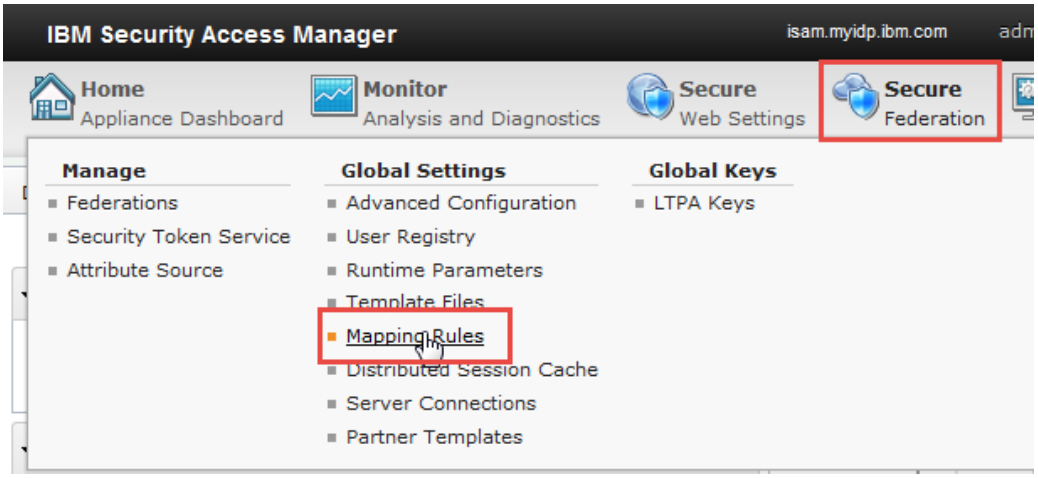
When using the appliance console to create Mapping Rules, cut-and-paste is used to load the JavaScript content of the rules. Before we get started, we need to open our first rule in a text editor so we can copy it.

Go to the **…/providedfiles/mappingrules/sp** directory and open the **sp_saml20.js** file in a text editor.

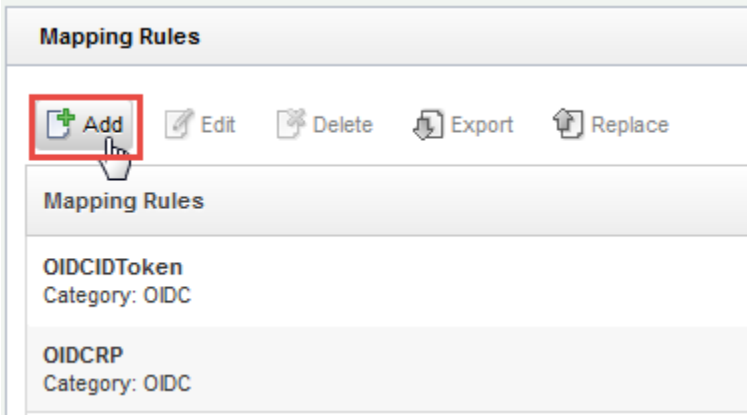


Select all the text in the file and then copy it. On Windows you can use **Ctrl-a** to select all and **Ctrl-c** to copy.

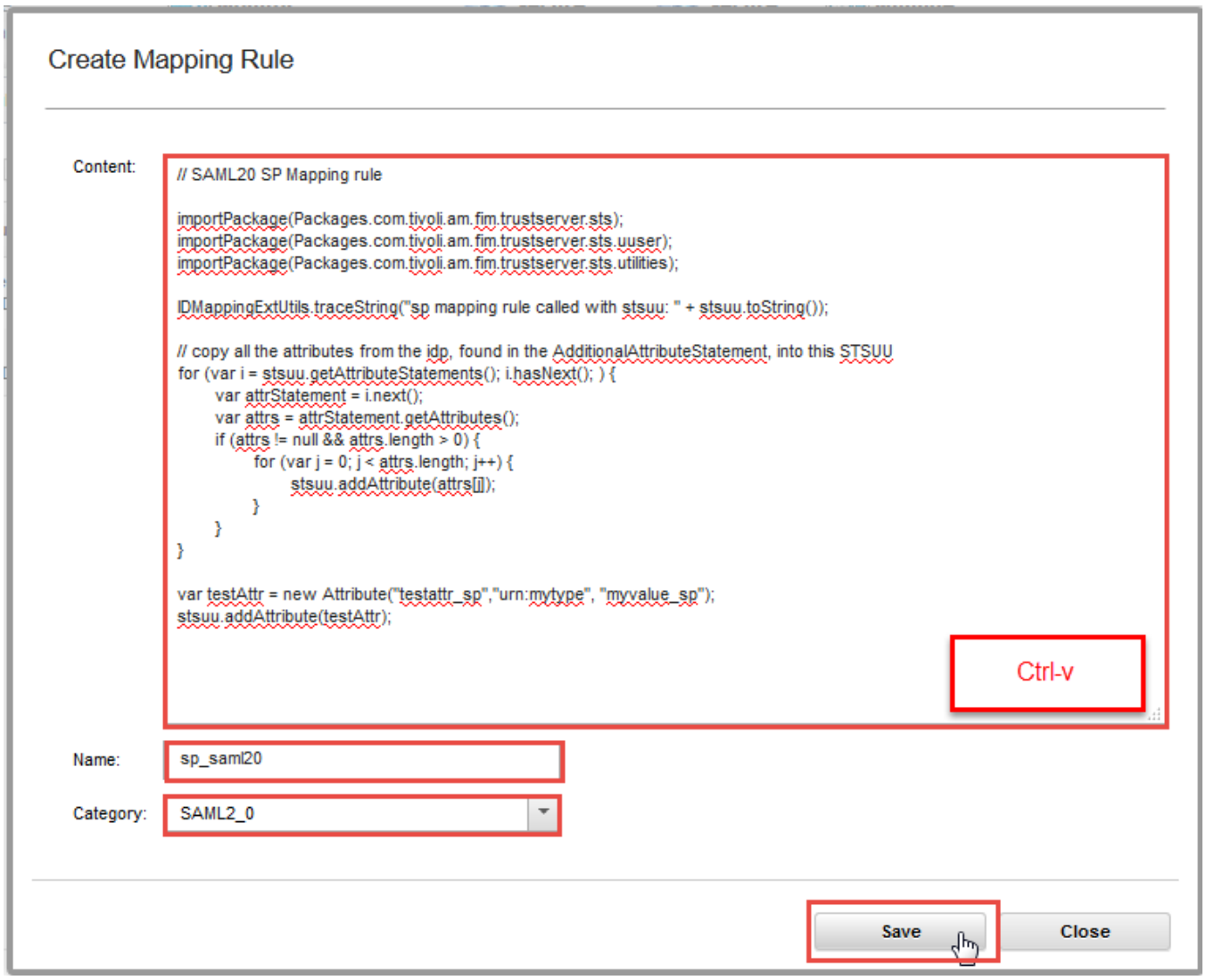
Now we're ready to create a Mapping Rule on the appliance with this content.



In the LMI Administration console, navigate to **Secure Federation→Global Settings: Mapping Rules**.



Click **Add** to add a new mapping rule.



Paste the rule text into the *Content* box. On Windows you can use **Ctrl-c** to paste.

Enter **sp_saml20** as the rule *Name* and select **SAML2_0** as the *Category*.

Click **Save** to save the new Mapping Rule.

Repeat the process above for all of the files in the *…/providedfiles/mappingrules/sp* directory

Once all Mapping Rules are loaded, **deploy** the pending changes.

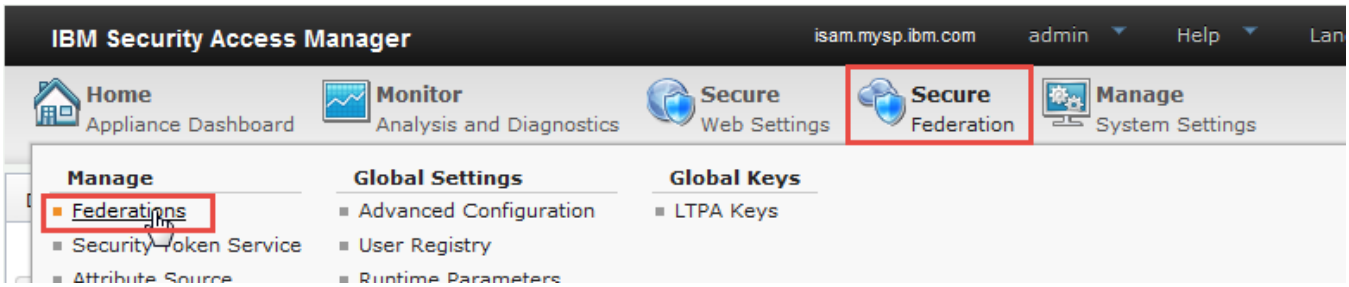
SCRIPT-END:
The script should display the following
INFO:FederationManager:Upload all mapping rules
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule

7.3 Create federation

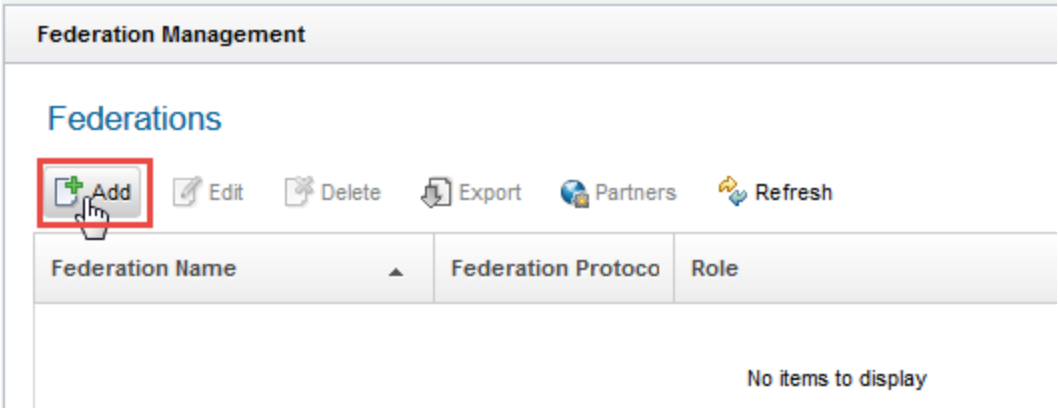
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the SP, run this script: **SAMLSPConfig.py -configure Federation**

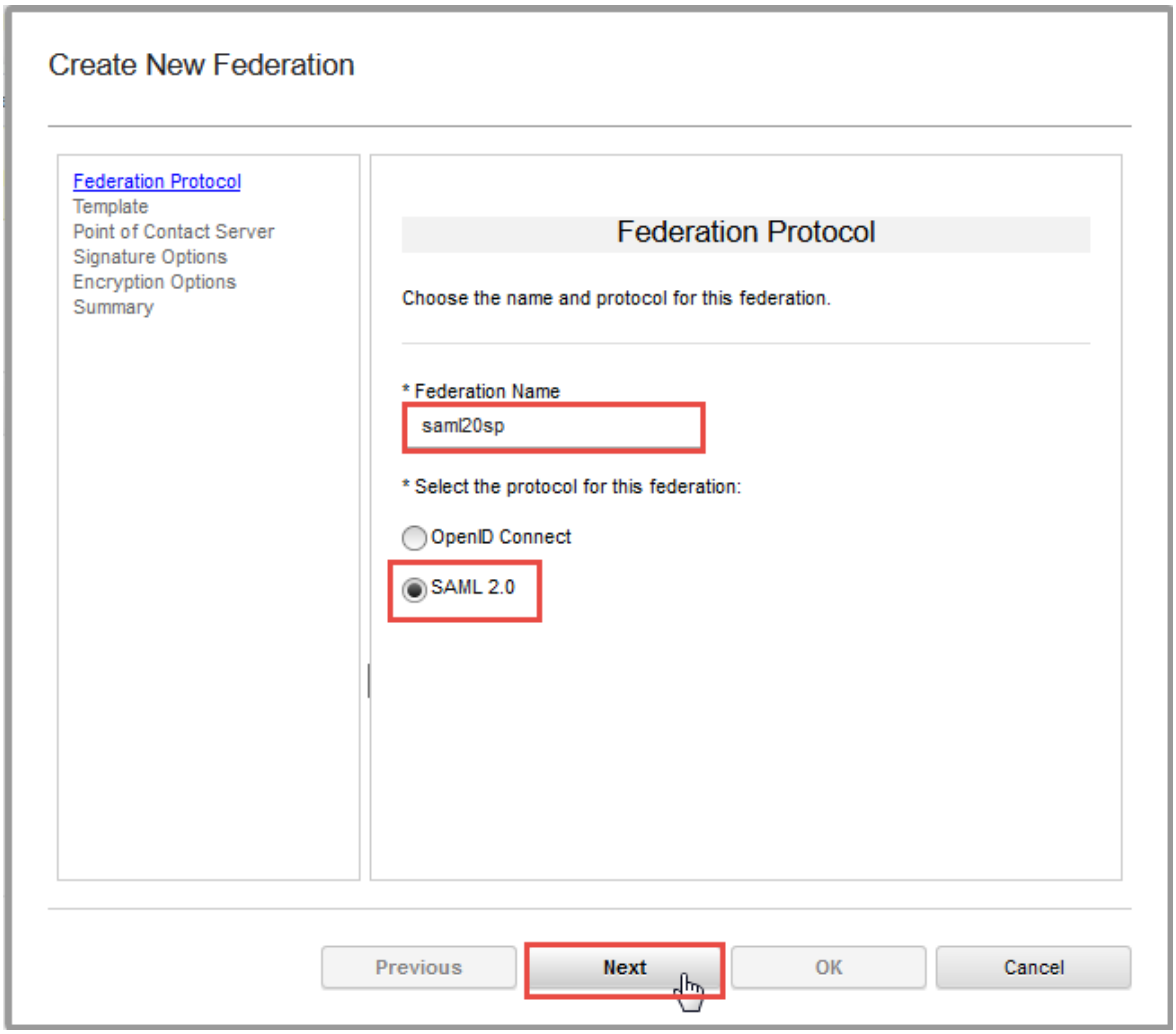
If you use this script, skip to the corresponding SCRIPT-END notice



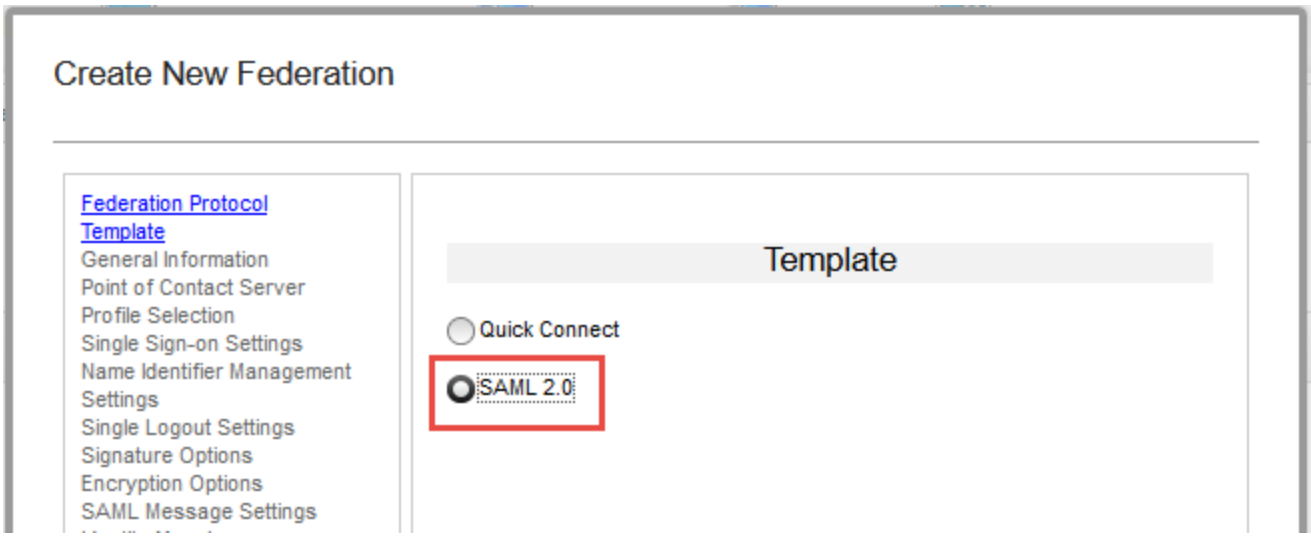
Using the administration console, navigate to **Secure Federation→Manage: Federations**.



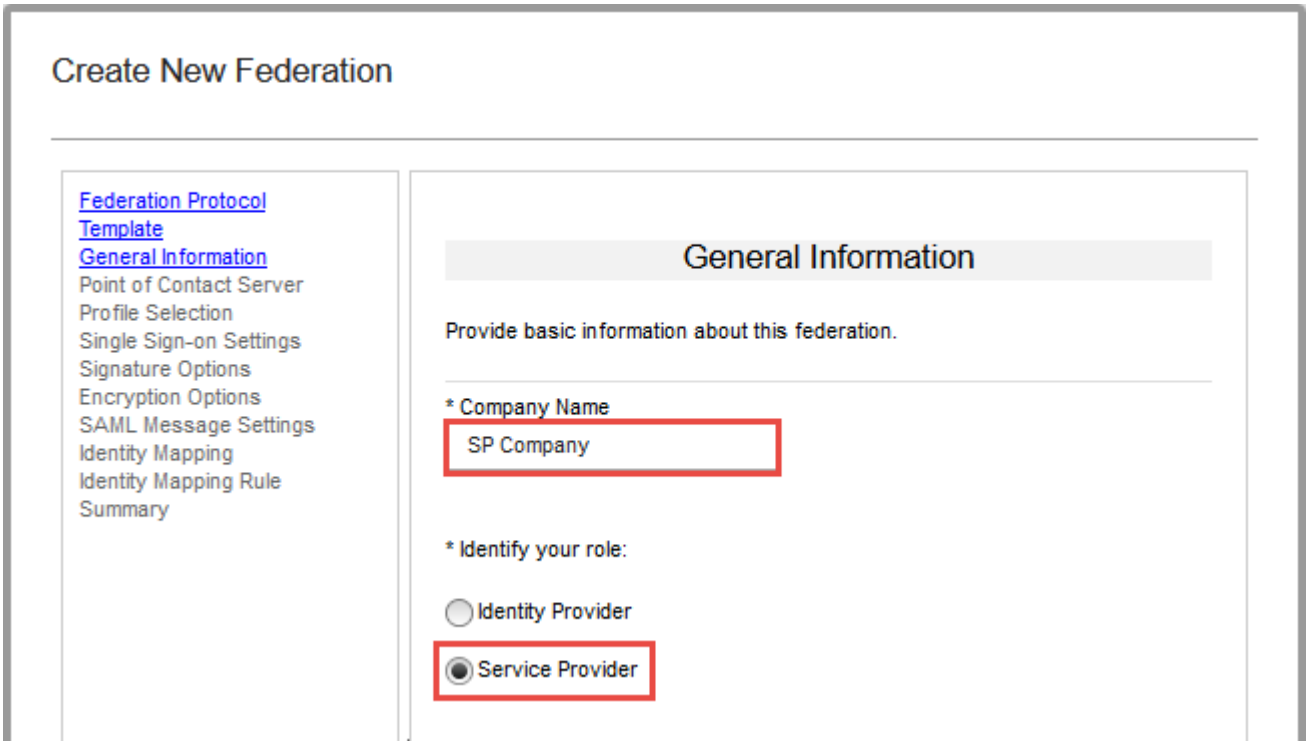
Click **Add** to create a new federation.



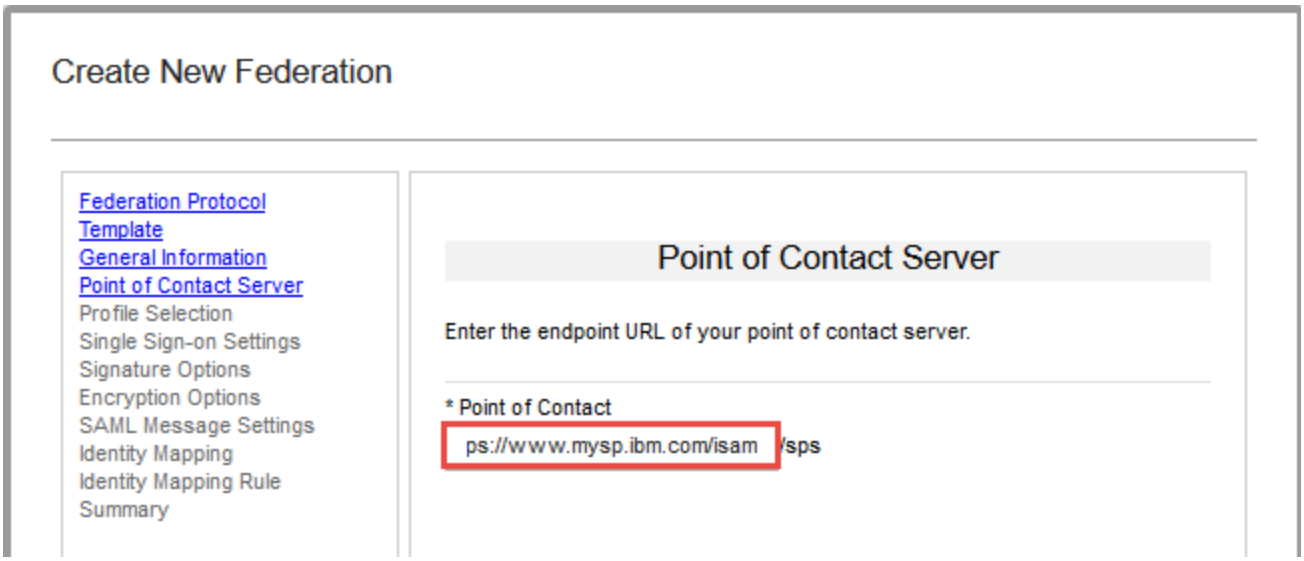
Create a new **SAML 2.0** federation named **saml20sp** as shown and click **Next**.



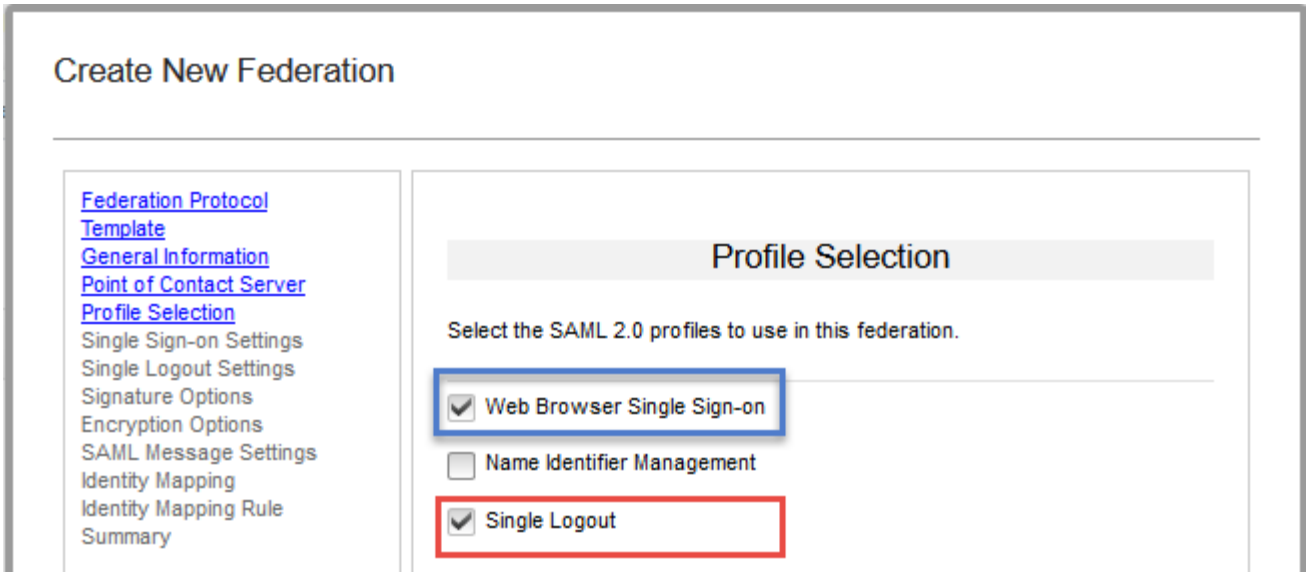
On the template page, select **SAML 2.0** and click **Next**.



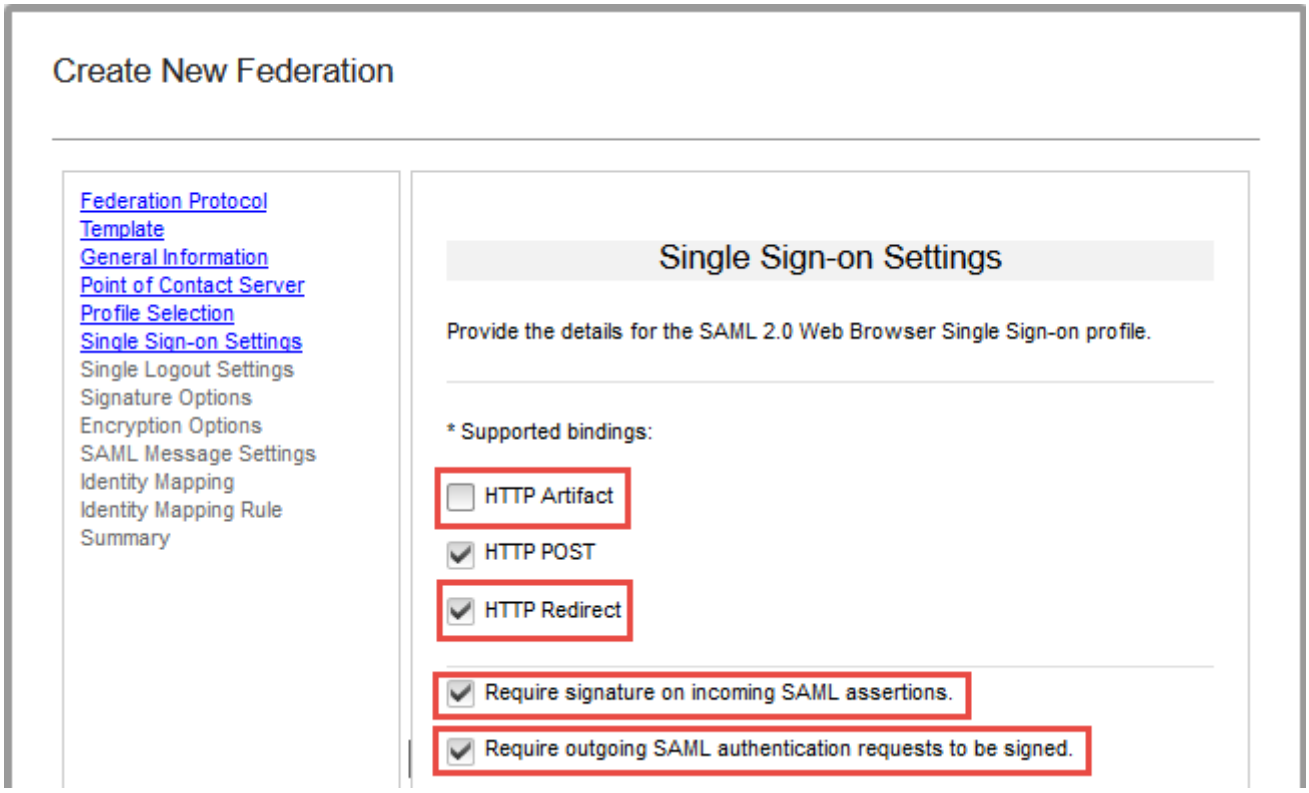
On the General Information panel, enter **SP Company** as the Company Name, select **Service Provider** as the role, and click **Next**.



On the Point of Contact Server panel, enter **https://www.mysp.ibm.com/isam** and click **Next**.



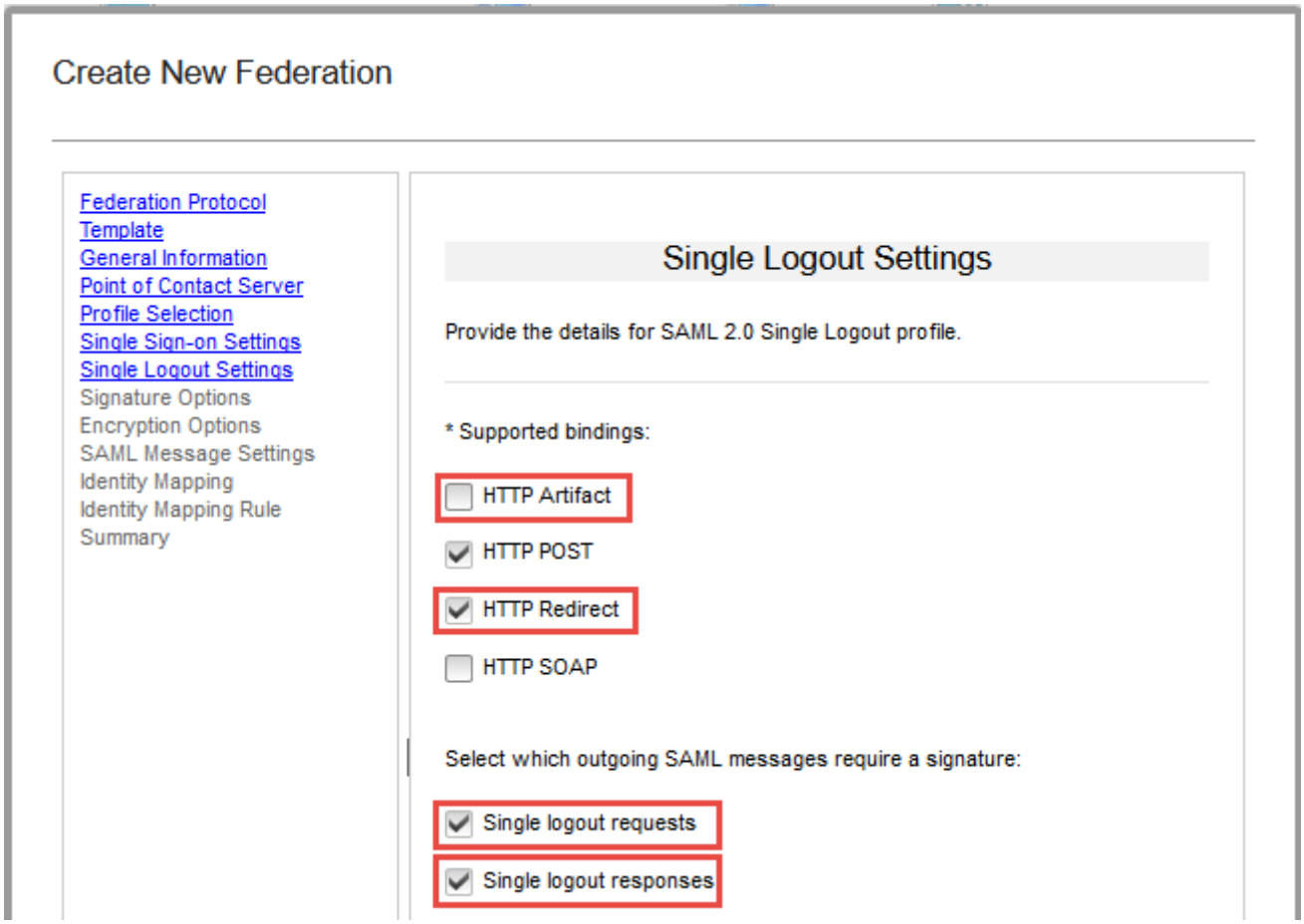
On the profile selection panel, leave **Web Browser Single Sign-on** selected and also select **Single Logout**. Then press **Next**.



On the Single Sign-on settings panel, deselect **HTTP Artifact** and select **HTTP Redirect**.

Select checkboxes for **Require signature on incoming SAML assertions** and **Require outgoing SAML authentication requests to be signed**.

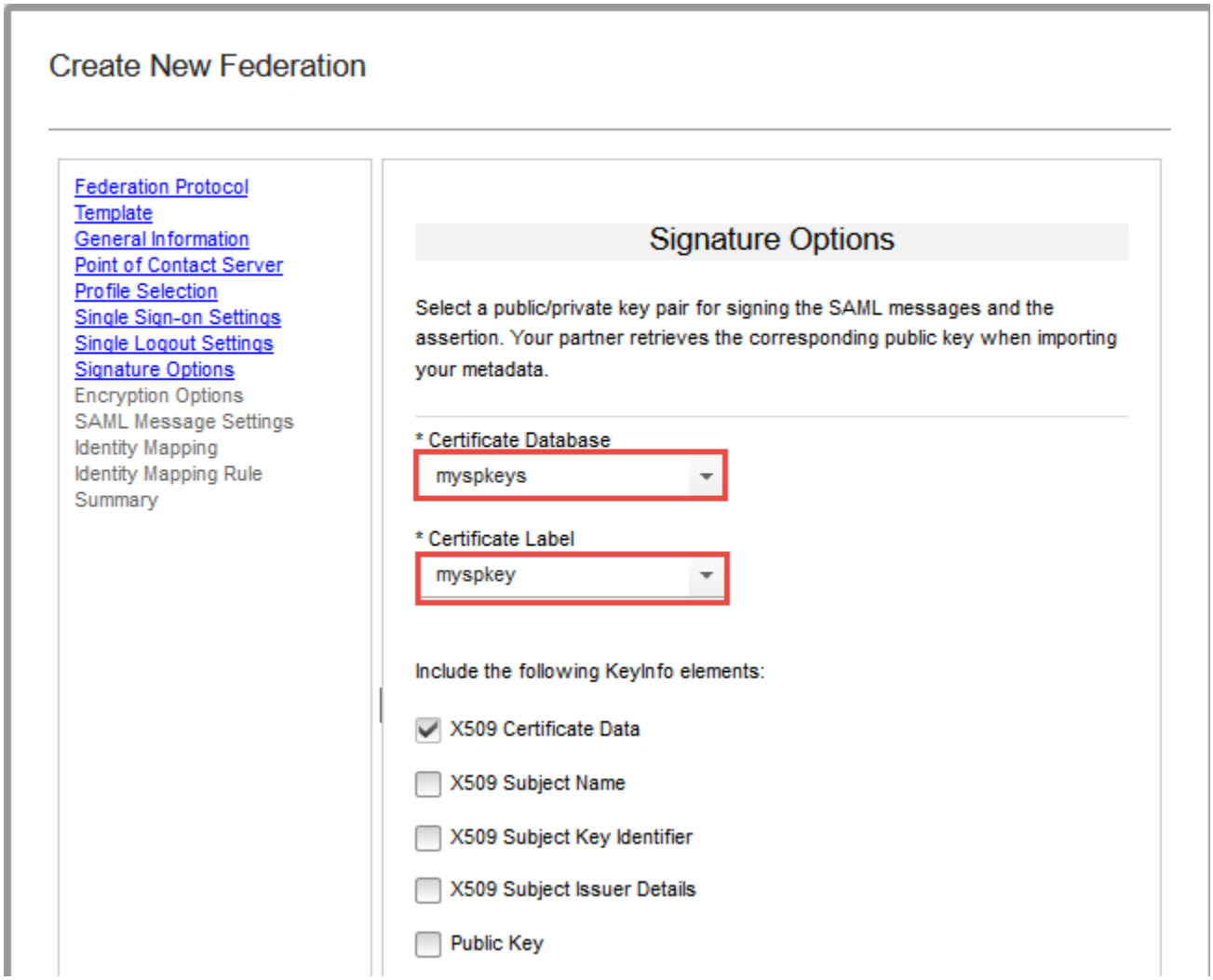
Then click **Next**.



On the Single Logout Settings panel, deselect **HTTP Artifact** and select **HTTP Redirect**.

Select checkboxes to require signatures for **Single logout requests** and **Single logout responses**.

Then click **Next**.

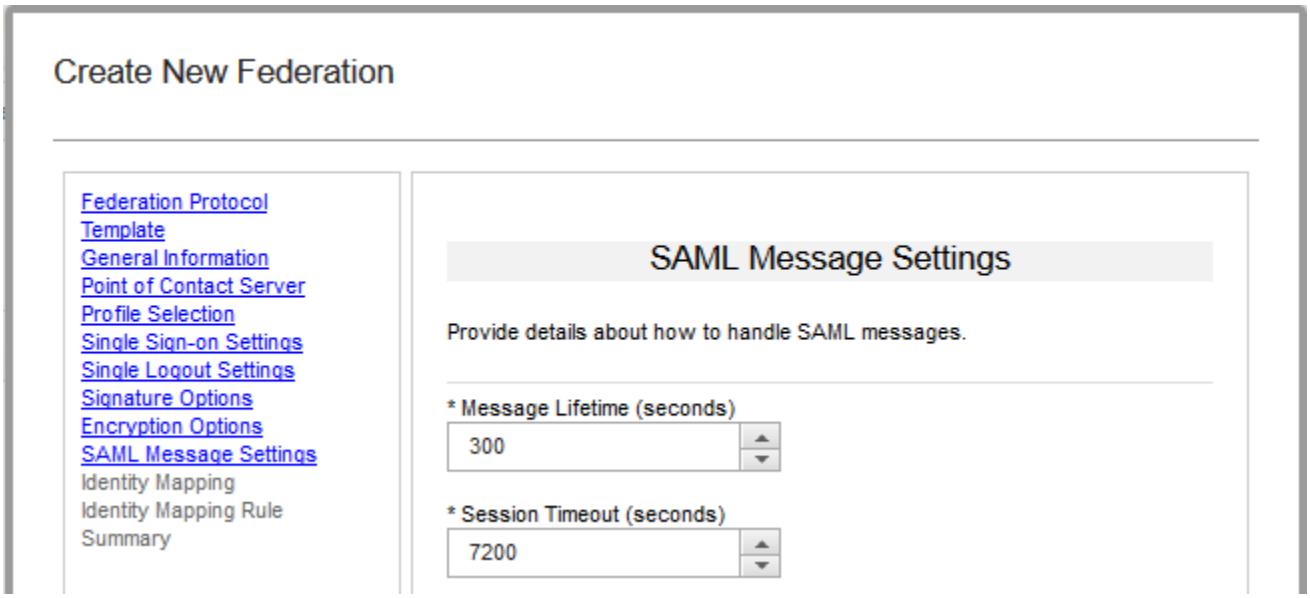


On the Signature Options panel, select the **myspkeys** Certificate Database, and the **myspkey** Certificate Label.

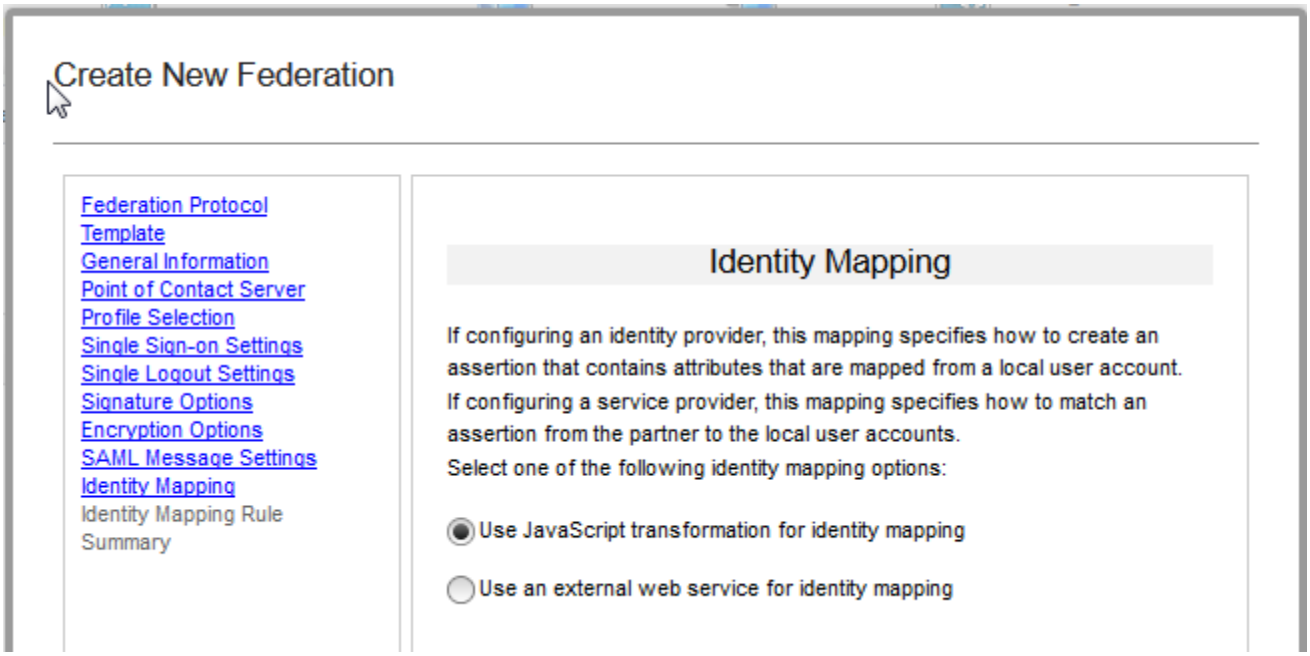
Then click **Next**.

On the Encryption Options panel, select the **myspkeys** Certificate Database and the **myspkey** Certificate Label.

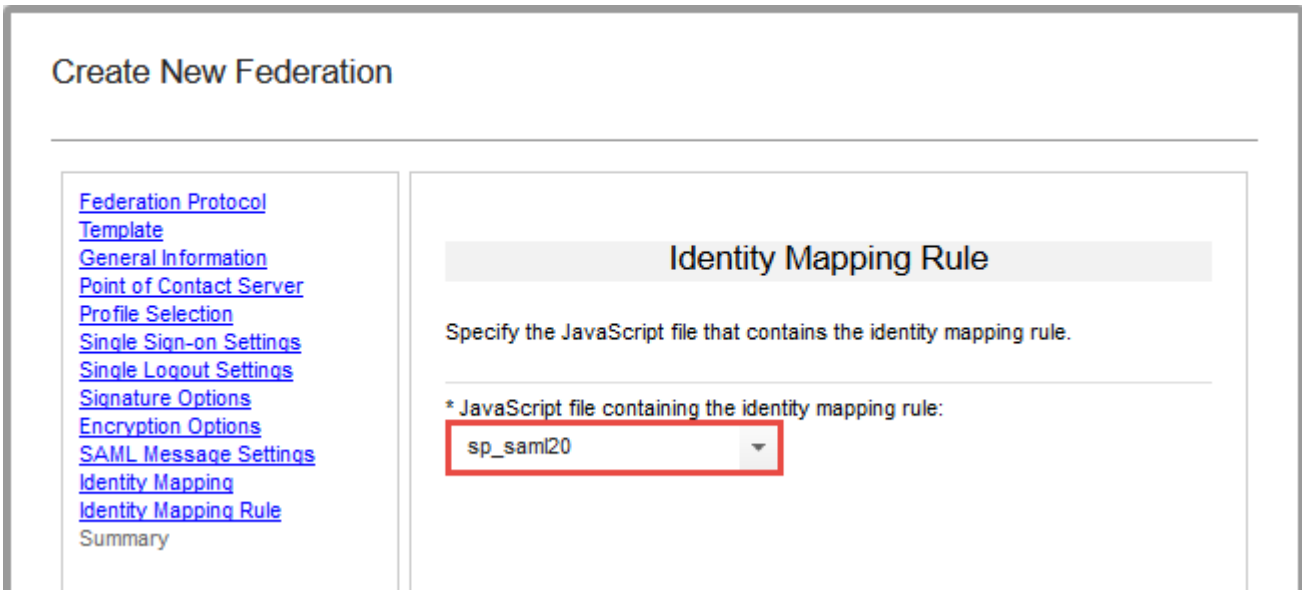
Then click **Next**.



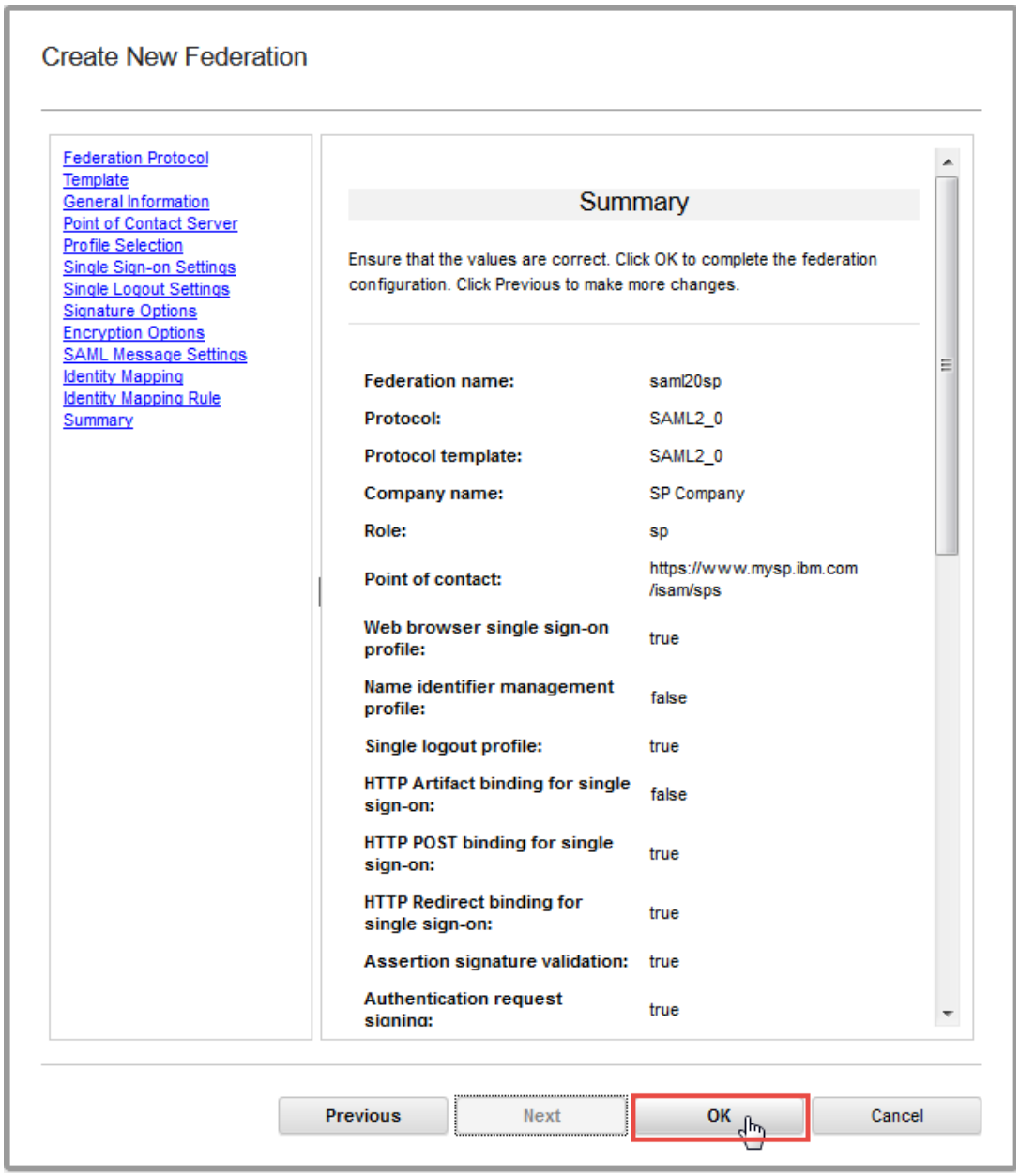
On the SAML Message Settings panel, leave values at their defaults and click **Next**.



On the Identity Mapping panel, we will use the default of **Use Javascript transformation for identity mapping** so just click **Next**.



On the Identity Mapping Rule panel, select **sp_saml20** from the drop-down list and click **Next**.



On the Summary panel, click **OK** to create the federation.

Follow on-screen instructions to **deploy** pending changes.

SCRIPT-END:
The script should display the following
INFO:FederationManager:Configuring the SP Federation
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the SP Federation

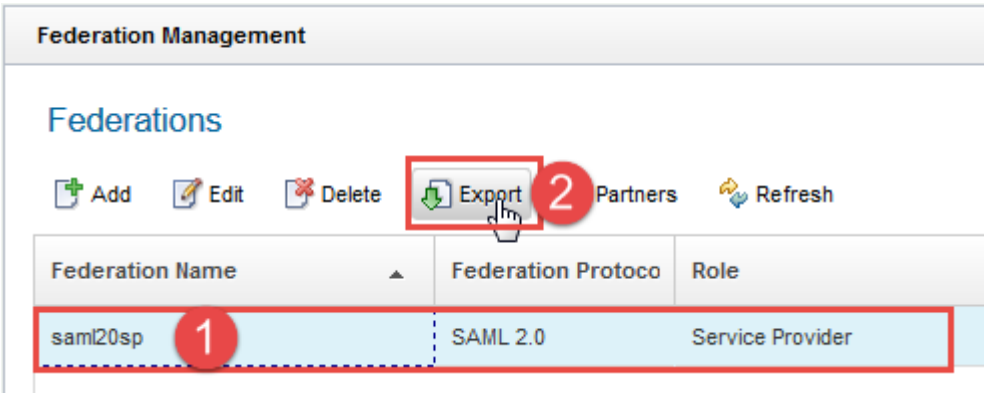
7.4 Export meta-data

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the SP, run this script: **SAMLSPConfig.py -configure Export_Metadata**

If you use this script, skip to the corresponding SCRIPT-END notice

Using the administration console, navigate to **Secure Federation -> Manage: Federations**.



Click on the **saml20sp** federation and click **Export**.

This will start the download of the federation metadata. Save the file to:
…/providedfiles/Automation/tmp/spmetadata.xml

> It is important to save the metadata file to this exact location if you are planning to use the automated scripts to import this metadata to the partner.

SCRIPT-END:
The script should display the following
INFO:FederationManager:Exporting Metadata
INFO:FederationManager:Successful export of metadata

The metadata file will be exported to *…/providedfiles/Automation/tmp/spmetadata.xml*

8 Configure Reverse Proxy for Federation

After a new federation has been configured, the Reverse Proxy needs to be configured for it. There are some general items that need to be completed for any federation:

- Load Federation Runtime certificate to Reverse Proxy keystore
- Create Junction to Federation Runtime
- Enable sending of session id in HTTP header

and there are some federation-specific items:

- Create and attach ACLs to protect federation endpoints
- Set up EAI trigger URLs

A REST service and a corresponding UI are available via the Access Manager LMI Management interface which triggers all of the required actions for a federation. This is the recommended way to perform the configuration. The manual steps are also documented in the Appendix D should you wish to follow them.

8.1 ISAM Configuration for the IdP

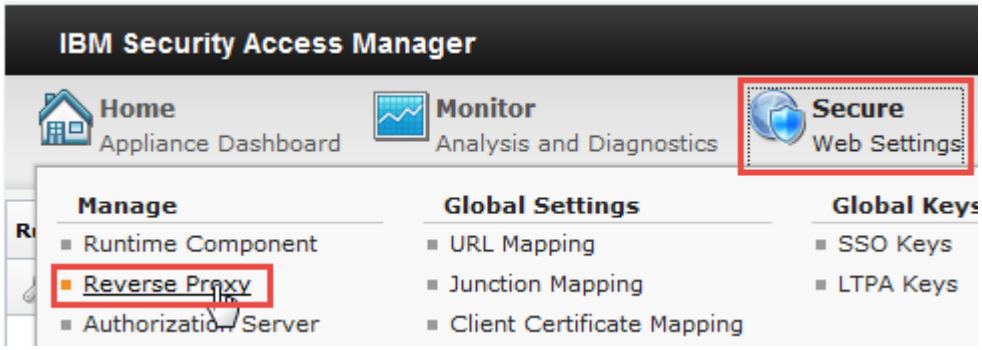
This section is completed only for the Identity Provider.
You will configure the Service Provider in a later section.

8.1.1 Configure Reverse Proxy for IdP

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

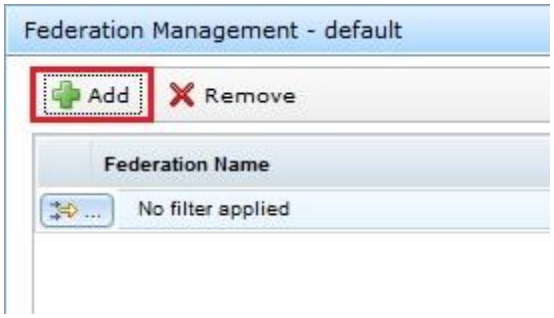
For the IdP, run this script: **SAMLIPConfig.py -configure POC_For_Federation**

For SAM versions before 9.0.1.0 there is no way to trigger the POC configuration from the LMI Web Console. You must either use the script to directly call the REST service (recommended) or follow the manual steps documented in Section 25.1 - ISAM Configuration for the IdP.



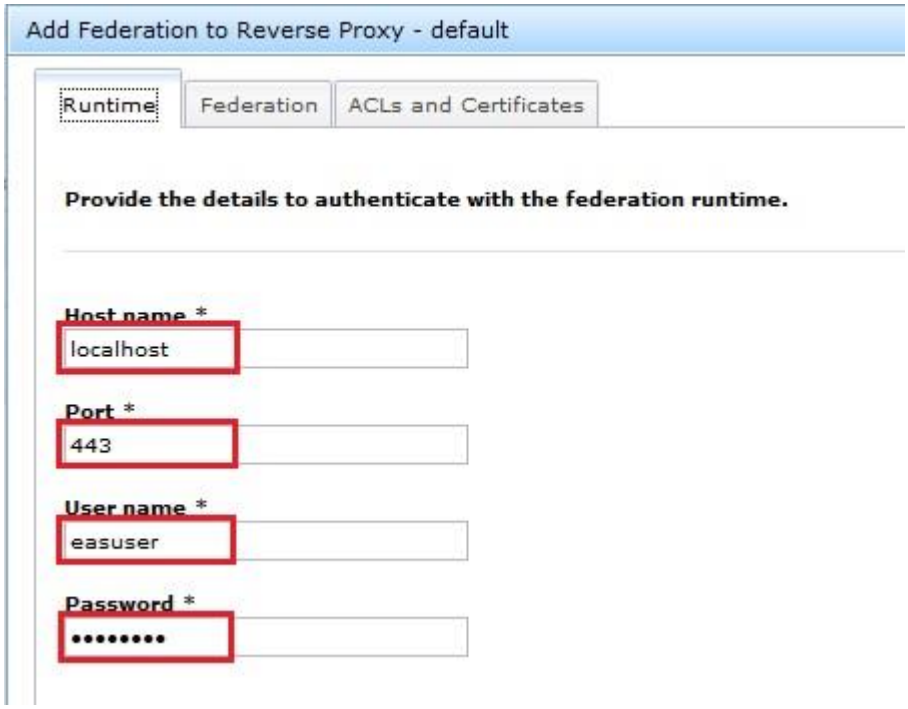
In the mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.

Select the Reverse Proxy instance, and click on **Manage -> Federation Management**.



To add the IdP federation, click on the **Add** button.

There are three panels which need to be filled out.



Inside the **Runtime** pane, user has to provide the details to authenticate with federation runtime. The details include the host, port, user name and password. All of them are required. When you move to the next pane, these details are used to connect to the Federation Runtime to retrieve a list of configured federations.

On the Federation Tab, Select the federation created in the previous section.

The next tab is the ACLs and Certificates panel, you can choose to reuse ACLs and Certificates if they exist or create new ones.

Once all the panels are done, click on **Submit** and then **Deploy** the Pending changes.

SCRIPT-END:

The script should display the following for IdP:

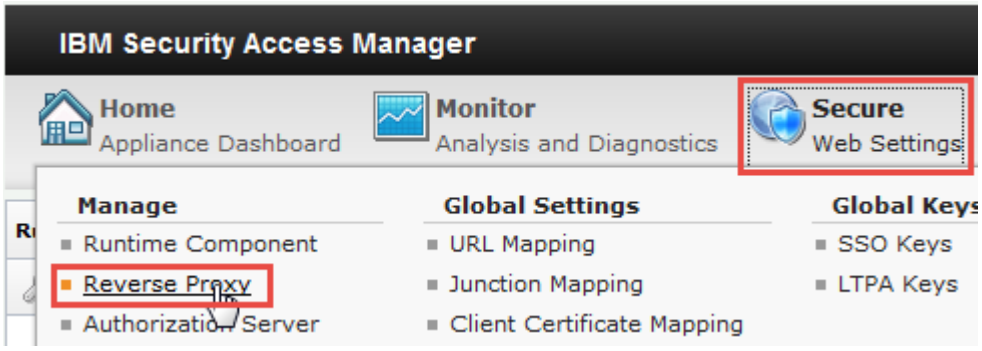
INFO:FederationManager:Configuring Reverse Proxy for federation saml20idp

INFO:FederationManager:Successfully configured Reverse Proxy for federation

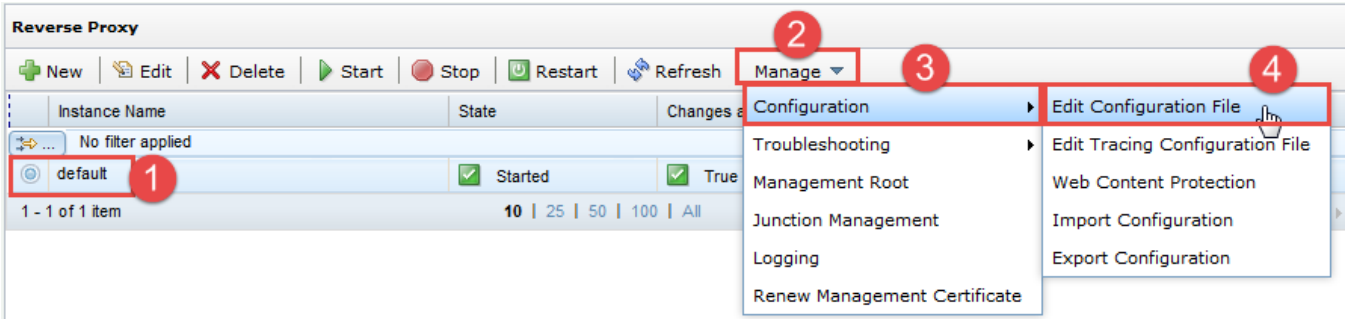
8.1.2 Environment-specific configuration

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

For the IdP, run this script: **SAMLIPConfig.py -configure WebSEAL_Configfile**



Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

At the end of the configuration file, create a new stanza as follows:

```
[junction:/isam]
reset-cookies-list = *ac.uuid,*JSESSIONID
```

Add the TAM_CRED_ATTRS_SVC stanza and TAM_CRED_ATTRS_SVC:eperson as shown below. Add these at the end of the file to ensure that they do not interfere with any existing stanza data.

```
[TAM_CRED_ATTRS_SVC]
eperson = azn_cred_registry_id

[TAM_CRED_ATTRS_SVC:eperson]
emailAddress = mail
firstName = cn
lastName = sn
```

Save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

SCRIPT-END:
The script should display the following for IdP:
INFO:WGAManager:Configuring WebSEAL.conf file for SAML IdP
INFO:WGAManager:Successfully configured WebSEAL.conf file for SAML IdP

8.2 ISAM Configuration for SP

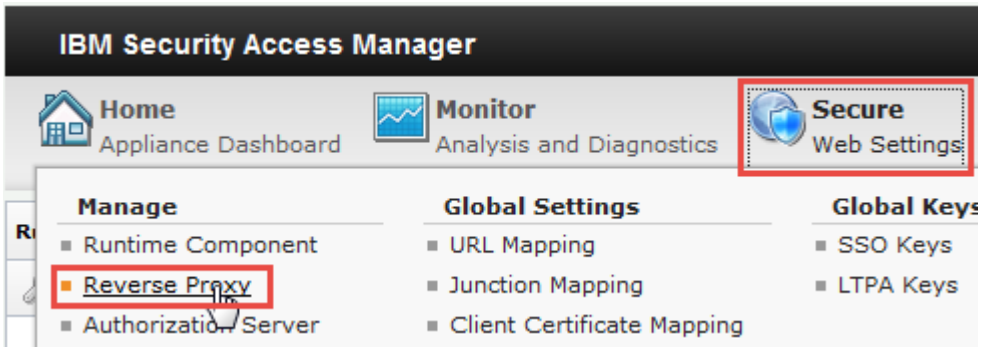
This section is completed only for the Service Provider.
You should have configured the Identity Provider in the previous section.

8.2.1 Configure Reverse Proxy for SP

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

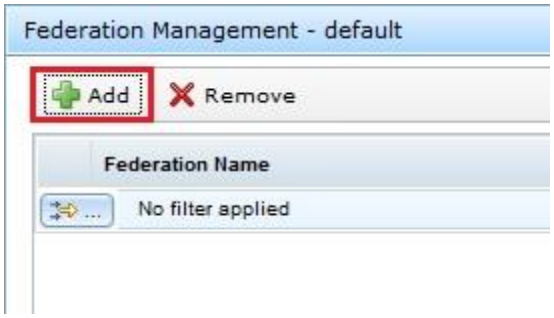
For the SP, run this script: **SAMLSPConfig.py -configure POC_For_Federation**

For SAM versions before 9.0.1.0 there is no way to trigger the POC configuration from the LMI Web Console. You must either use the script to directly call the REST service (recommended) or follow the manual steps documented in Section 25.2 - ISAM Configuration for SP.



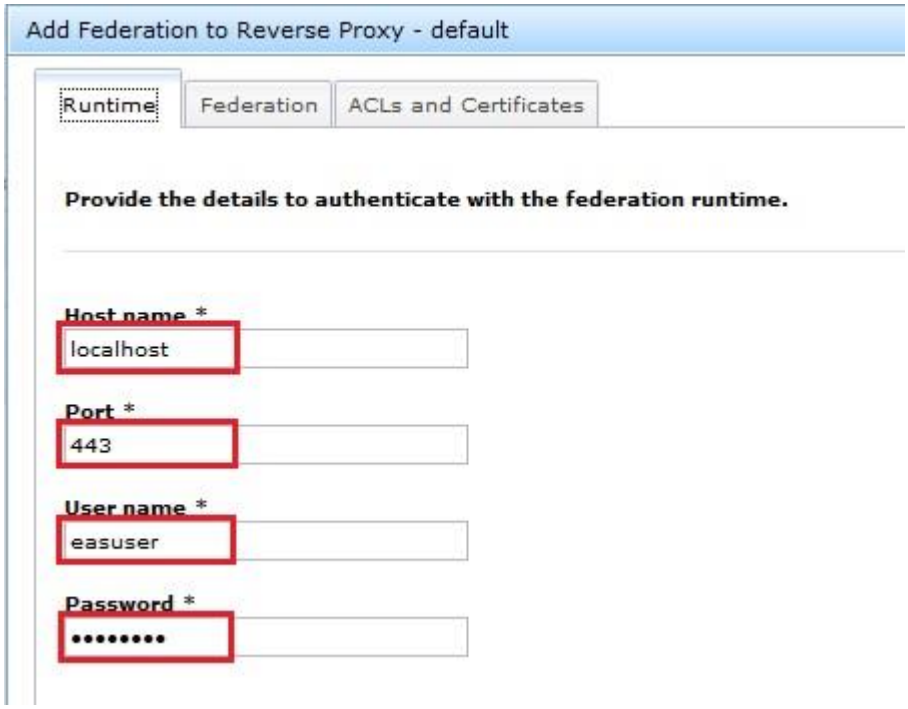
In the Mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.

Select the Reverse Proxy instance, and click on Manage -> Federation Management.



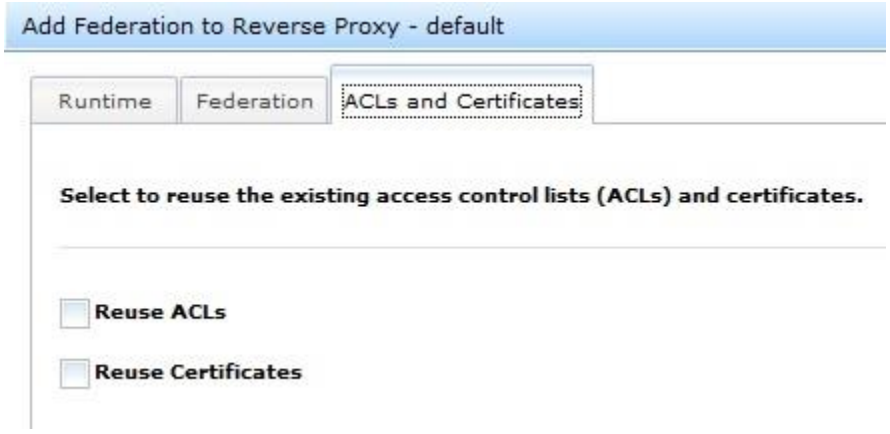
To add the SP federation, click on the **Add** button.

There are three panels which need to be filled out.



Inside the **Runtime** pane, user has to provide the details to authenticate with federation runtime. The details include the host, port, user name and password. All of them are required. When you move to the next pane, these details are used to connect to the Federation Runtime to retrieve a list of configured federations.

On the Federation Tab, Select the SP federation created in the previous section.



The next tab is the ACLs and Certificates panel. You can choose to reuse ACLs and Certificates if they exist or create new ones.

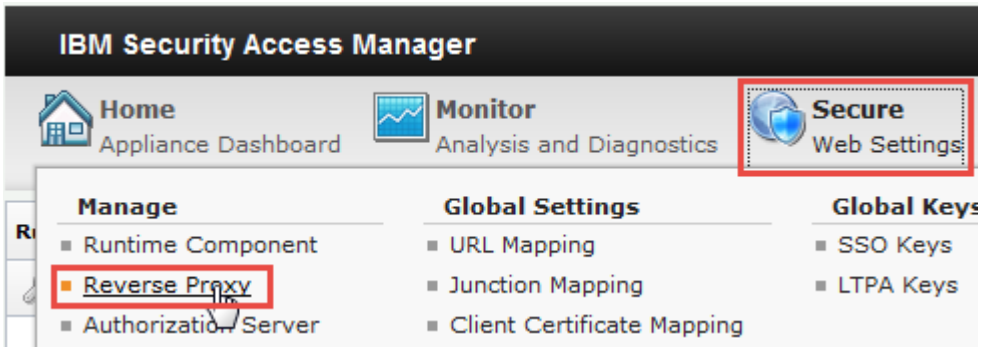
Once all the panels are done, click on **Submit** and then **Deploy** the Pending changes.

SCRIPT-END:
The script should display the following for SP:
INFO:FederationManager:Configuring Reverse Proxy for federation saml20sp
INFO:FederationManager:Successfully configured Reverse Proxy for federation

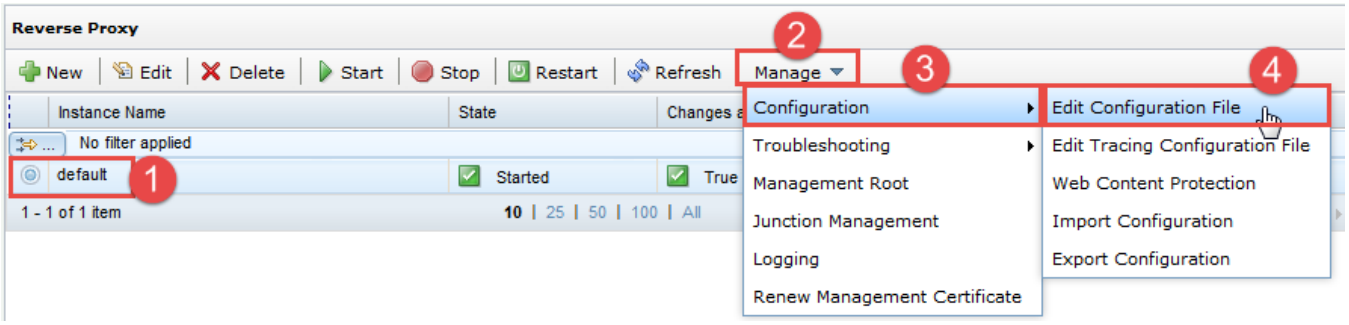
8.2.2 Environment-specific configuration

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

For the SP, run this script: **SAMLSPConfig.py -configure WebSEAL_Configfile**



Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

At the end of the configuration file, create a new stanza as follows:

```
[junction:/isam]
reset-cookies-list = *ac.uuid,*JSESSIONID
```

Save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

8.2.3 Add anonymous user

When importing the IdP Partner information to our Service Provider a User ID was specified which will be used when a transient NameIdentifier is passed by that IdP. The (default) User ID specified was **anonymous**.

We will now create that user at the Service Provider so that the Reverse Proxy can build a credential for this user when it is returned by the Federation Runtime.

Open an SSH session to the appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

SSH to **isam.mysp.ibm.com** and authenticate using the administrator credentials (admin/Passw0rd). The welcome message is displayed:

```
Welcome to the IBM Security Access Manager
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.mysp.ibm.com>
```

Using commands below, navigate to **isam** and start the **admin** utility. Then login to the pdadmin console**:**

```
isam.mysp.ibm.com> isam admin
pdadmin> login -a sec_master -p Passw0rd
pdadmin sec_master>
```

Create an unauth ACL using the commands and attach it to SAML endpoints:

```
user create anonymous cn=anonymous,dc=iswga anonymous anonymous Passw0rd
user modify anonymous account-valid yes
```

Enter **exit** twice to exit from the session.

SCRIPT-END:

The script should display the following for SP:

INFO:WGAManager:Configuring WebSEAL.conf file for SAML SP
INFO:WGAManager:Successfully configured WebSEAL.conf file for SAML SP
INFO:WGAManager:Configuring user: anonymous
INFO:WGAManager:Successfully configured user: anonymous

9 Configure Partners

In earlier sections, the metadata files corresponding to identity provider and service provider were exported. In this section, the respective metadata files will be imported to the each provider federation partners. Besides exchanging of metadata, the IdP and the SP partners are updated such that Signing is enabled and Encryption algorithms are configured.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

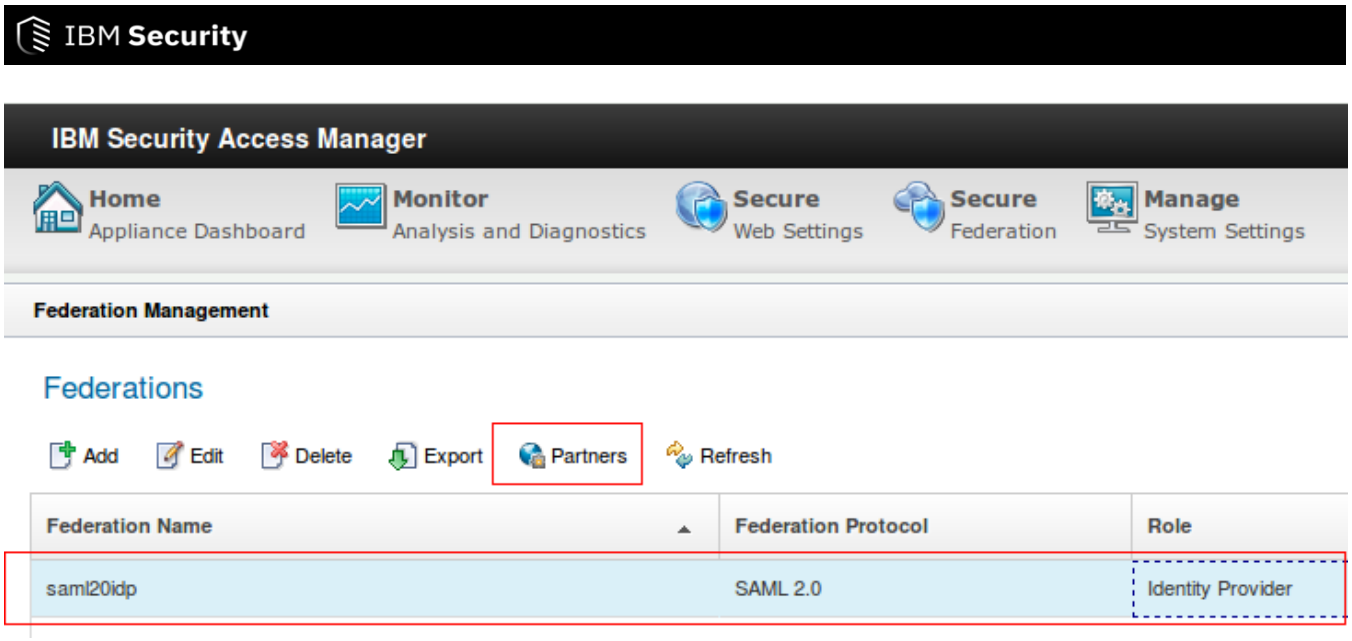
For the IdP, run this script: **ImportPartners.py -import IdP_Partner_Metadata**
For the SP, run this script: **ImportPartners.py -import SP_Partner_Metadata**

Note: These scripts look for partner metadata in the following locations:
…/providedfiles/Automation/tmp/ipmetadata.xml
…/providedfiles/Automation/tmp/spmetadata.xml

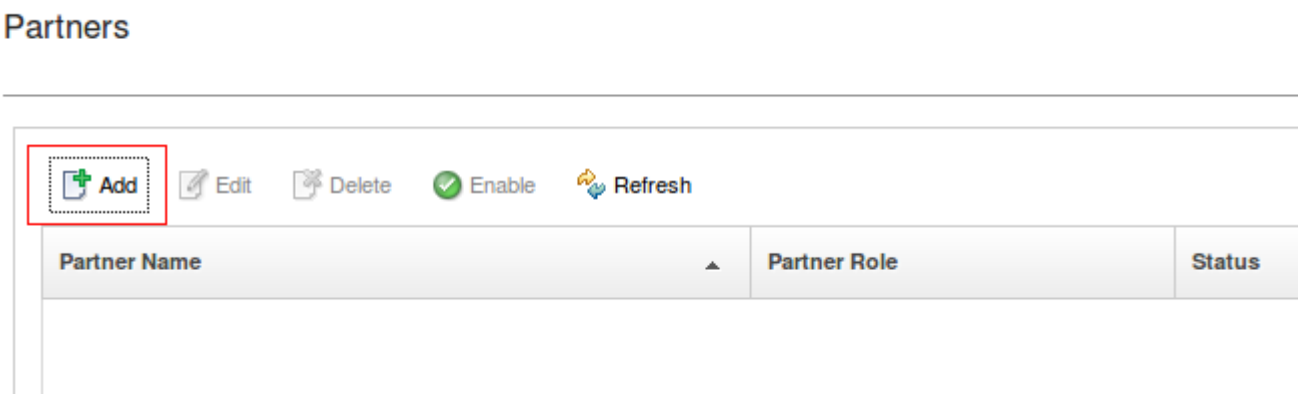
If you use this script, skip to the corresponding SCRIPT-END notice

9.1 Configuring Partner for the IdP

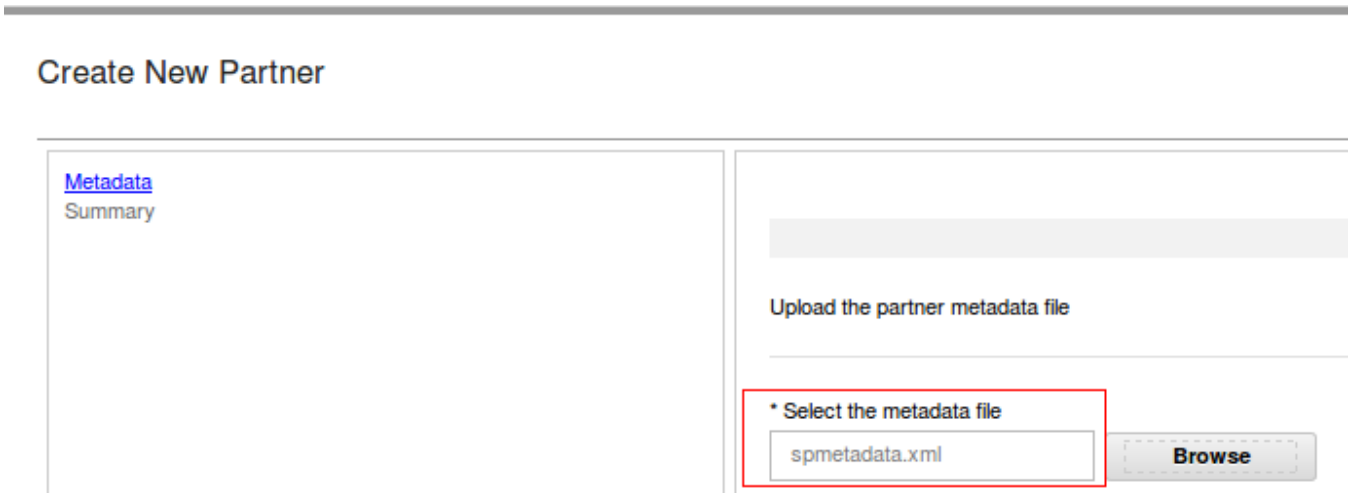
Under **Secure Federation** menu, click **on Manage: Federations**.



Select the **saml20idp** federation and click **Partners**.



Click on **Add** to import SP as partner.

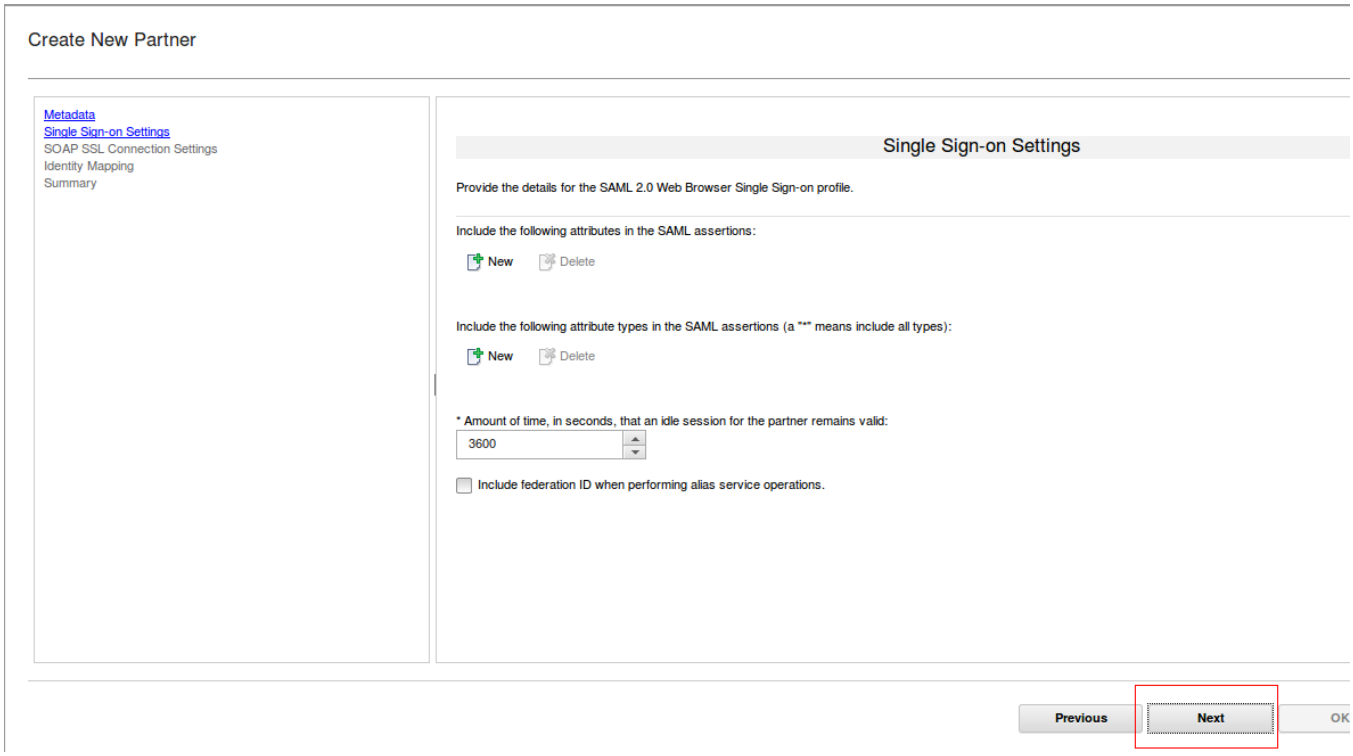


Select the metadata file that was exported from SP earlier.

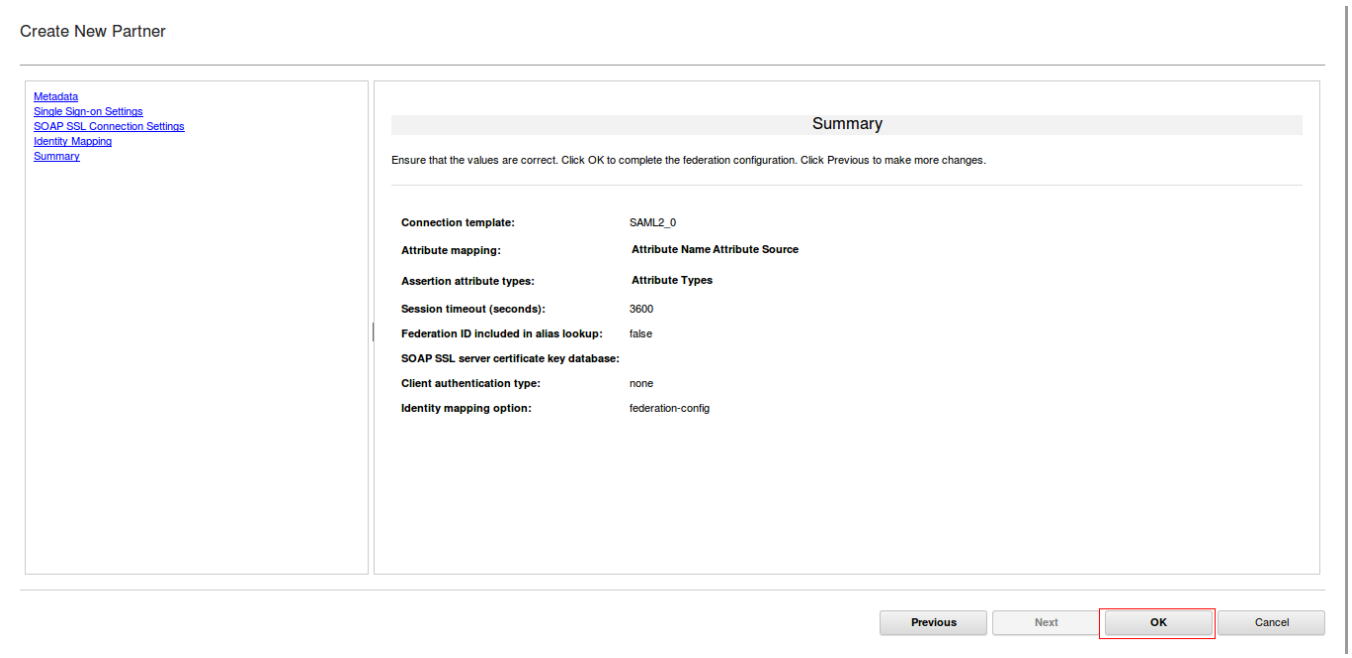
> If you used the provided Python script to export the metadata you'll find at *.../providedfiles/Automation/tmp/spmetadata.xml*

Click **Next**.

A message is briefly displayed to indicate that the partner has been created. You can now configure the partner.

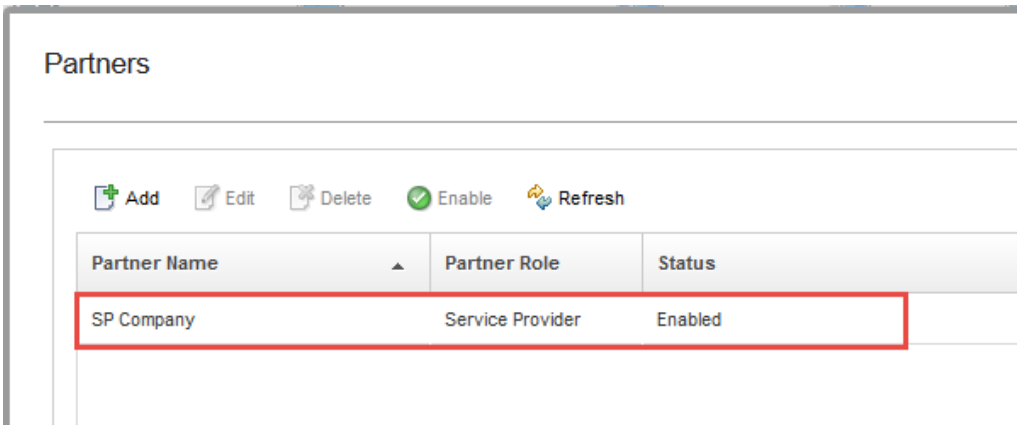


We don't need to change Single Sign-On settings so click **Next**.



Keep clicking **Next** in each screen to summary screen and then click **OK**.

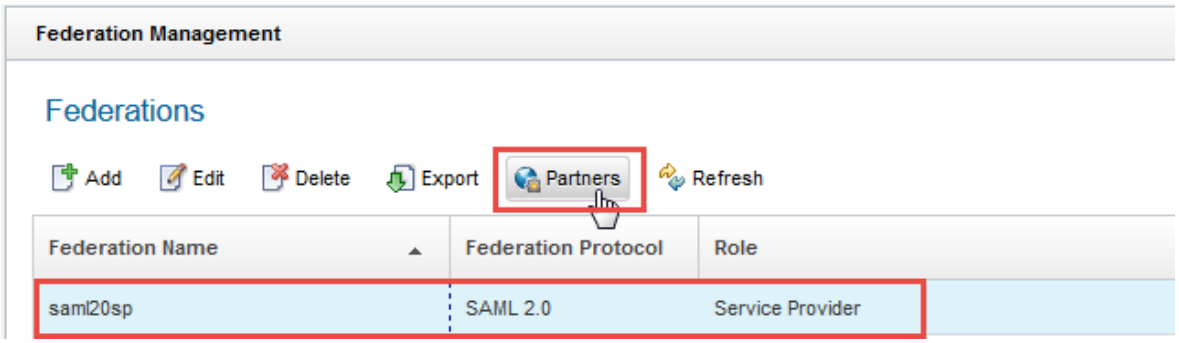
You can now see the new partner in the Partner list for the SAML 2.0 federation:



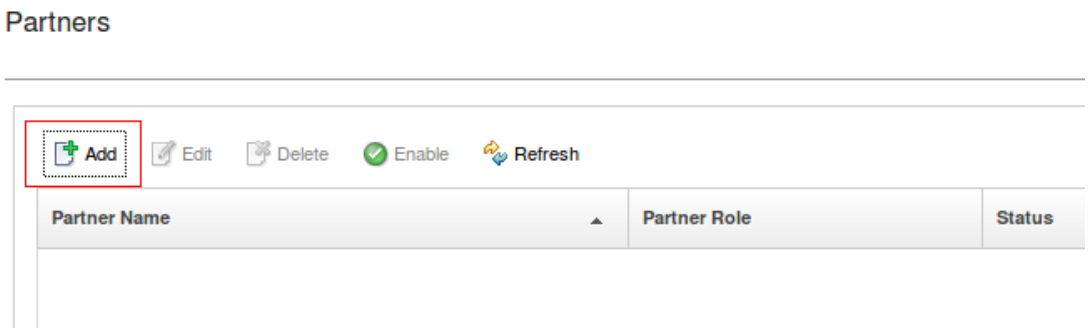
You may click “Edit” to add other optional and advanced configurations if needed

9.2 Configuring Partner for the SP

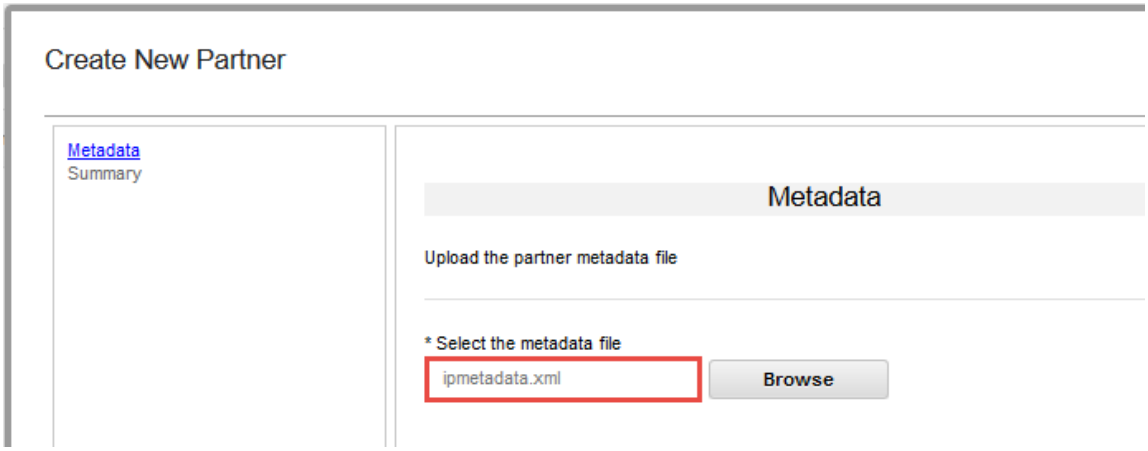
Under **Secure Federation** menu, click **on Manage: Federations**.



Select the **saml20sp** federation and click **Partners**.



Click on **Add** to import SP as partner.

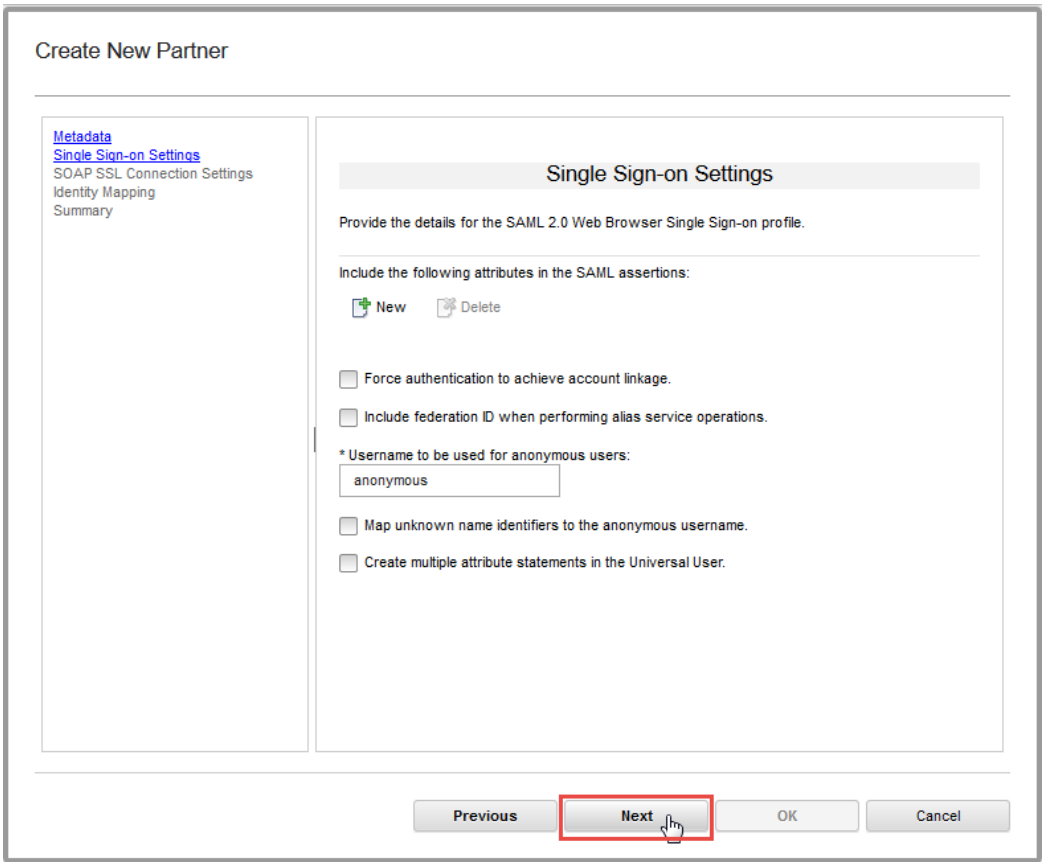


Select the metadata file that was exported from IdP earlier.

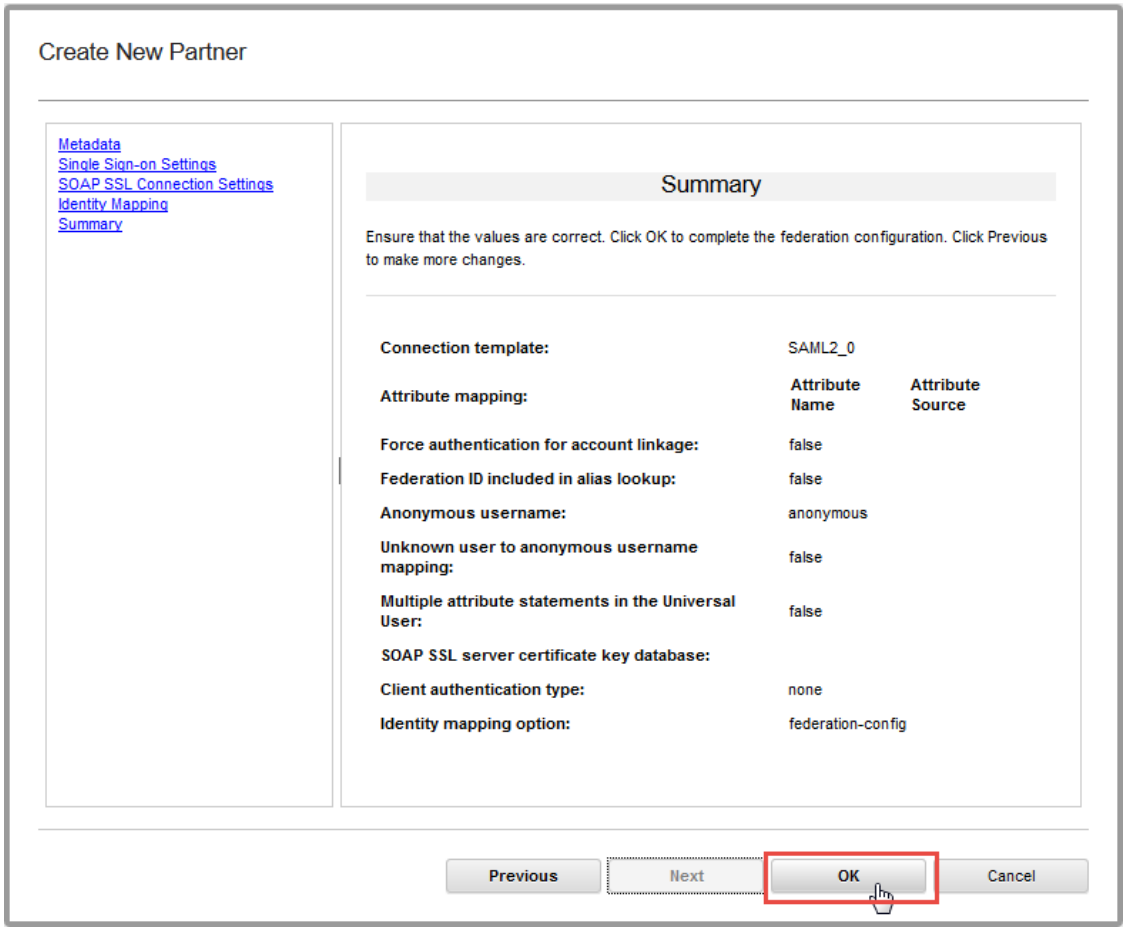
> If you used the provided Python script to export the metadata you'll find at *…/providedfiles/Automation/tmp/ipmetadata.xml*

Click **Next**.

A message is briefly displayed to indicate that the partner has been created. You can now configure the partner.

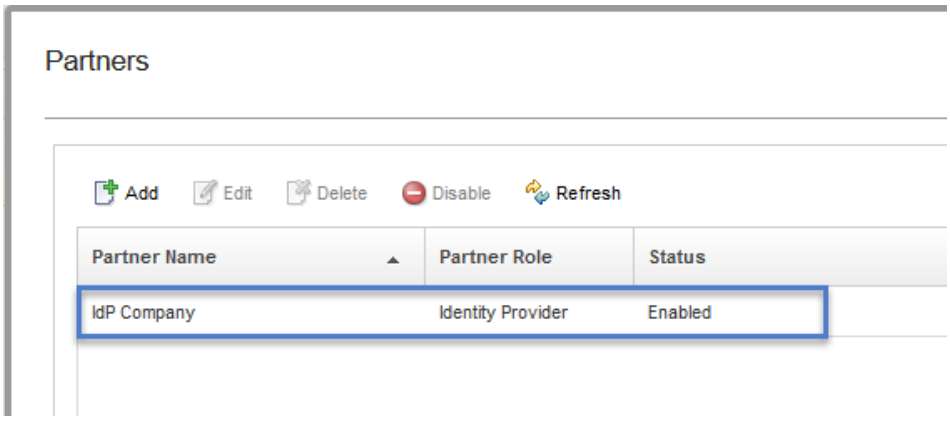


We don't need to change Single Sign-On settings so click **Next**.



Keep clicking **Next** in each screen to summary screen and then click **OK**.

You can now see the new partner in the Partner list for the SAML 2.0 federation:



- You may click “Edit” to add other optional and advanced configurations if needed

SCRIPT-END:

The script should display the following:

INFO:FederationManager:Importing Metadata
INFO:FederationManager:Successful import of metadata
INFO:FederationManager:Modifying Partner JSON to enable signing and encryption
INFO:FederationManager:Successfully modified the partner using PUT

10 Configure test application and test user

10.1 Configure test application

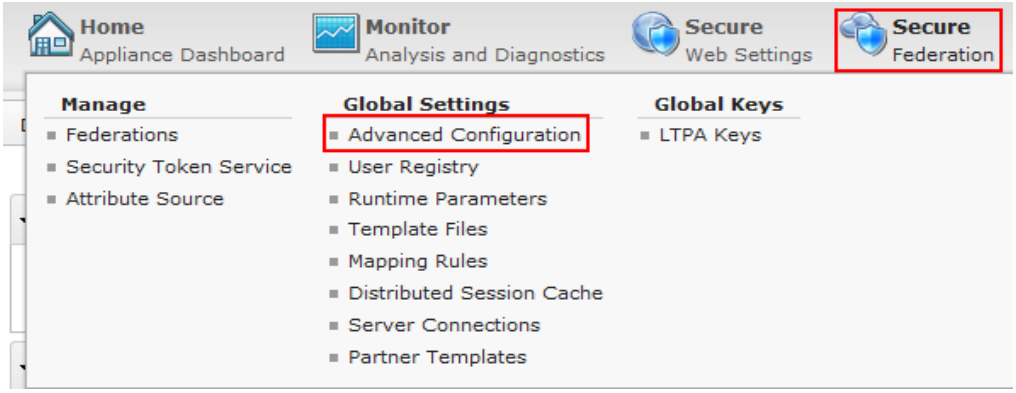
This section is only completed on the Service Provider

ISAM Appliances have a built-in "live demo" application that can be used as the target page after a successful SAML flow is completed. We will configure this demo application on the SP side so that we can use it as the target landing page on the SP site.

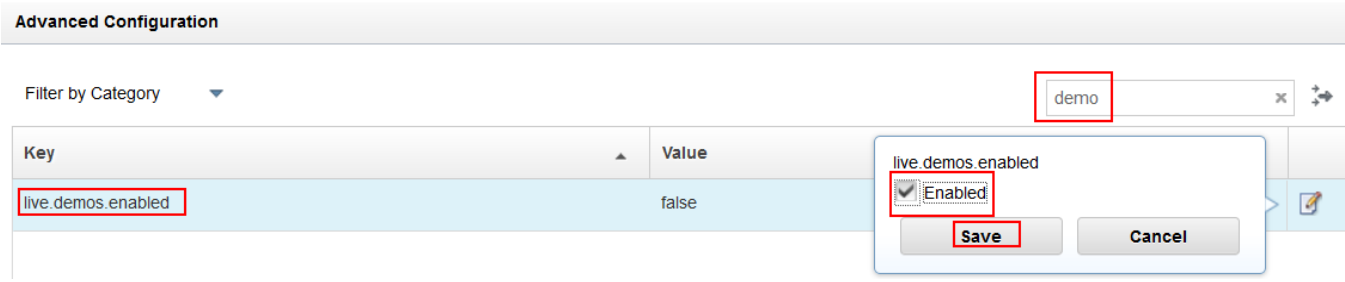
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

This is only for the SP, run this script: **SAMLSPConfig.py -action Enable_Demo_Application**

If you use this script, skip to the corresponding SCRIPT-END notice



Navigate to **Secure Federation > Manage > Advanced Configuration**

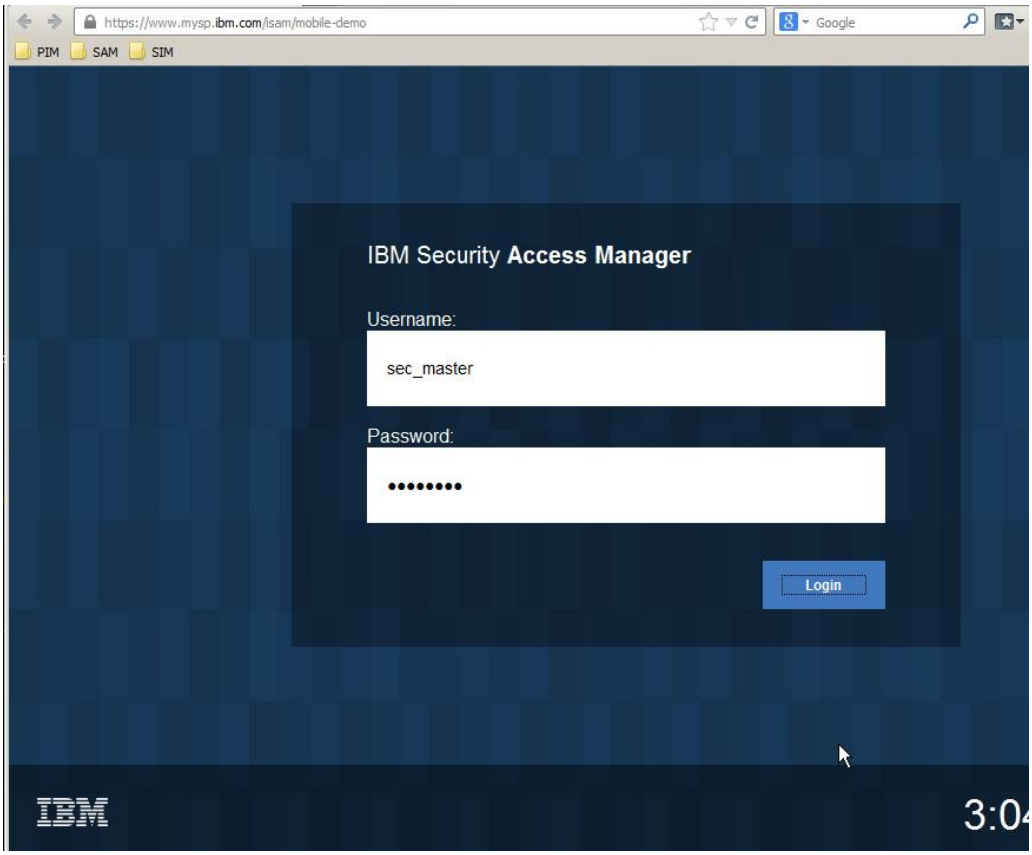


In the filter box seach for *demo*. Enable the **live.demos.enabled** key as shown above. Click **Save**.

A warning will be displayed at the top of the window. Click the link to activate the configuration change you have just made. A pop-up dialog is displayed showing the pending changes. **Deploy** the changes.

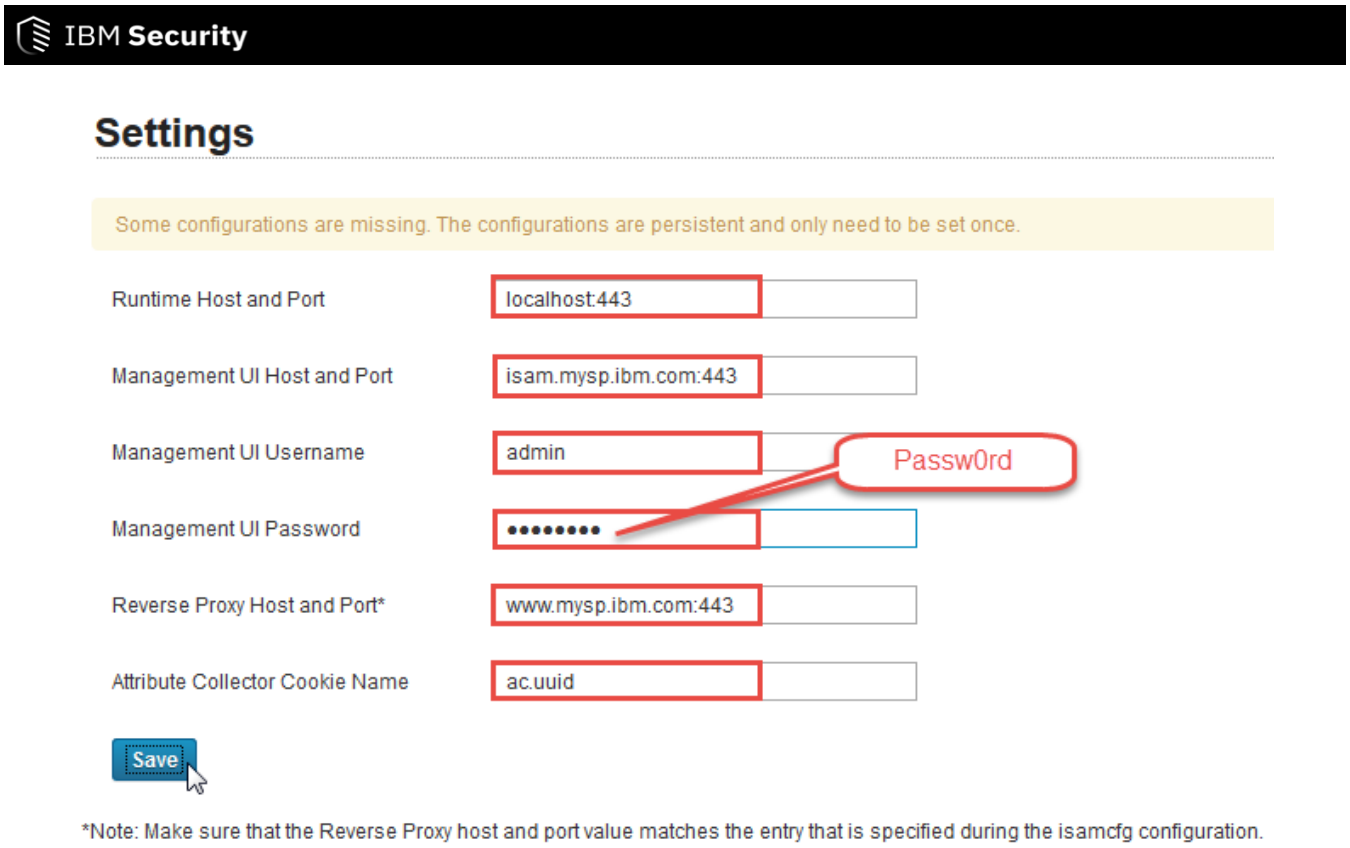
Once deployment is complete, navigate to:

https://www.mysp.ibm.com/isam/mobile-demo



Login with username **sec_master** and password **Passw0rd**

You will see a settings screen. This screen will be shown only for the first time during demo application configuration.



Enter the details that are shown in the form above and click **Save**. A success message as shown.

10.2 Authorize Access to Mobile Demo Application

The demo application is located on the /isam junction which, by default, only allows access to specified resources. We need to modify the Access Manager authorization policy to grant authenticated users access to the demo application (at /isam/mobile-demo).

Open an SSH session to the SP appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

Authenticate with **admin** and **Passw0rd**.

Navigate to **isam** and start the **admin** utility:

```
isam.mysp.ibm.com> isam
isam.mysp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Enter the following command to attach the *default-webseal* ACL to the demo application. This ACL grants access to all authenticated users:

```
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/mobile-demo default-webseal
```

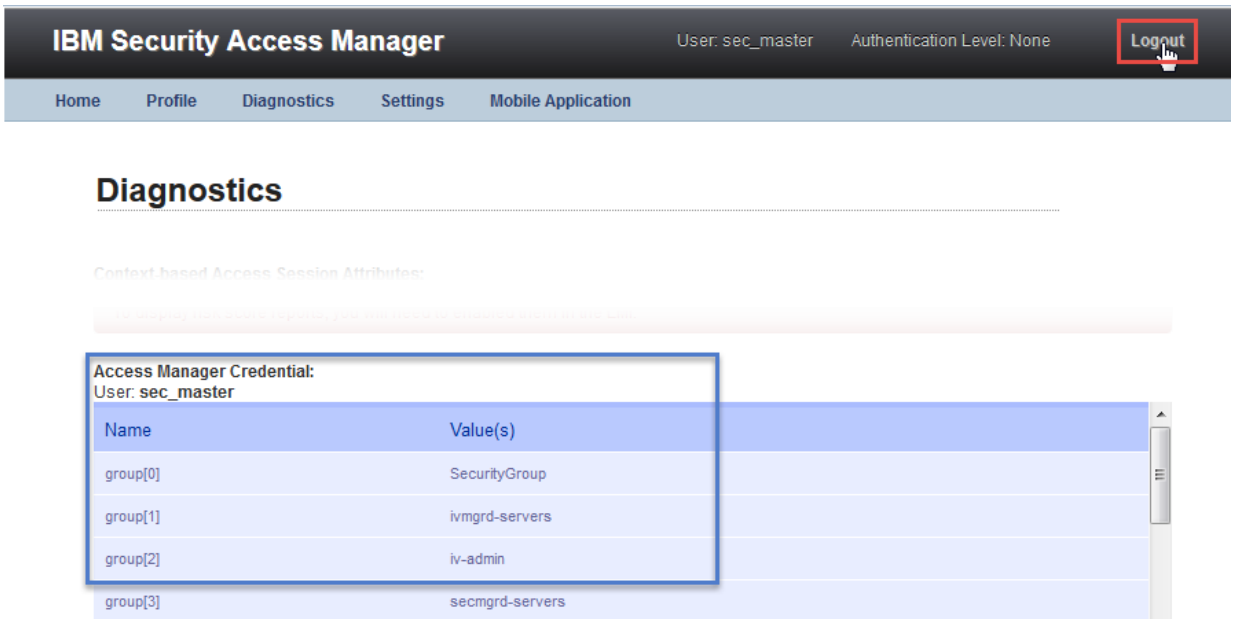
Type **exit** twice to end the session.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Setting demo application settings
INFO:FederationManager:Successfully set demo application settings

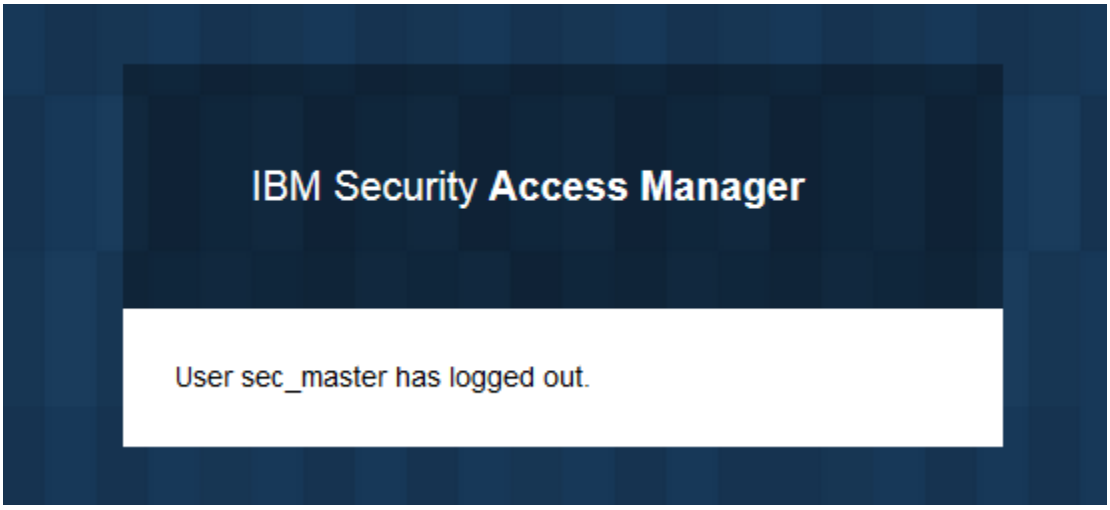
10.2.1 Test Access to Demo Application

Access https://www.mysp.ibm.com/isam/mobile-demo/diag/ and authenticate with **sec_master** and **Passw0rd**.

You can see Session attributes, credential details and HTTP Headers. This page will be the target page during a SAML flow.



Click the **Logout** link at the top of the page to logout of the Access Manager session.



10.3 Configure test user

This section is to be completed on both Identity Provider and Service Provider.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

Run this script: **SAMLIPConfig.py -configure Test_User**
Run the script: **SAMLSPConfig.py -configure Test_User**

If you use this script, skip to the corresponding SCRIPT-END notice

In order to run the SAML flow, a user needs to be created at both the IdP and the SP

Open an SSH session to the appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

Authenticate with **admin** and **Passw0rd**.

Navigate to **isam** and start the **admin** utility:

```
isam.myxx.ibm.com> isam
isam.myxx.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Enter the following commands to create a test user **testuser** with password **Passw0rd**:

```
user create testuser cn=testuser,dc=iswga Test User Passw0rd
user modify testuser account-valid yes
```

Type **exit** twice to end the session.

Repeat the steps above on the SP appliance.

On the IdP only, in addition to creating the user in ISAM, we need to set a number of LDAP attributes on the user's registry object. These attributes are used later in the cookbook to show attribute acquisition and propagation.

The easiest way to do this is to use the automated script for this section. If you want to perform the modifications manually then this will require the use of an LDAP tool (e.g. ldapmodify) to connect to the appliance registry and modify the following attributes:

```
mail: testuser@mailinator.com
homePhone: 555-12345
displayName: Test User
```

This is beyond the scope of this document.

SCRIPT-END:

The script should display the following:

INFO:WGAManager:Configuring a test user
INFO:WGAManager:Successfully configured a test user
INFO:WGAManager:Configuring test user to add extra attributes
INFO:LDAP Manager:Successfully added extra attributes to testuser

11 Test Federation

We are now ready to test the SAML 2.0 Federation that we have configured.

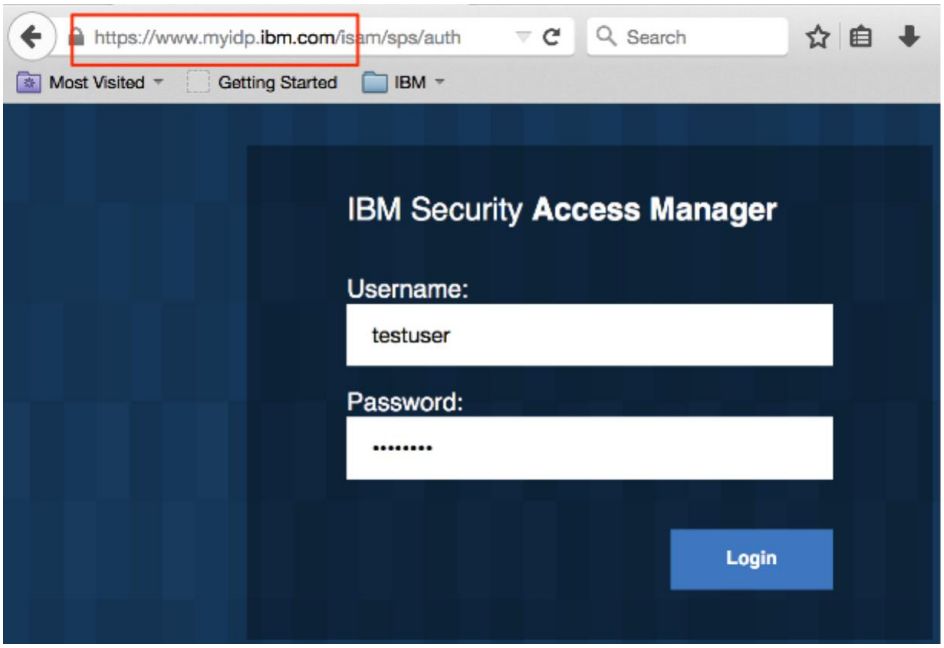
> **Note:** It is recommended that unless you are performing a single-logout after each single sign-on you restart your browser to remove all session cookies at both IdP and SP between each of the tests below.

An IdP initiated SAML flow which uses HTTP POST binding can be triggered using

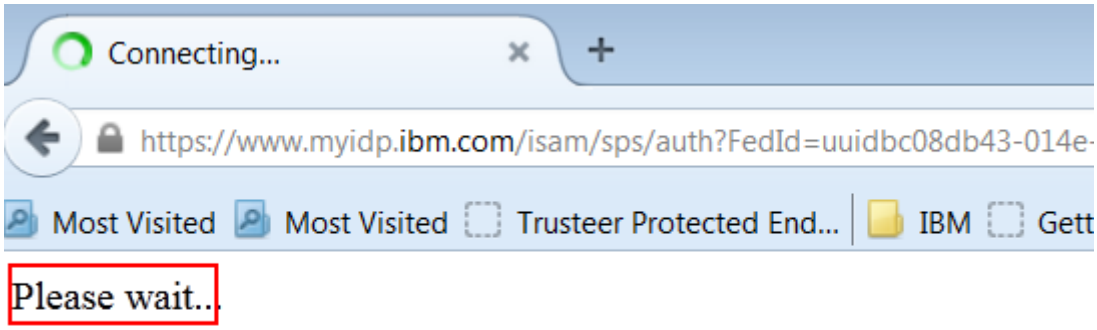
https://<IdP reverse proxy:port>/<junction name>/sps/<identity provider federation name>/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https://<SP reverse proxy:port/<junction name>/sps/<service provider federation name>/saml20&NameIdFormat=Email&Target=https://<TargetURL>

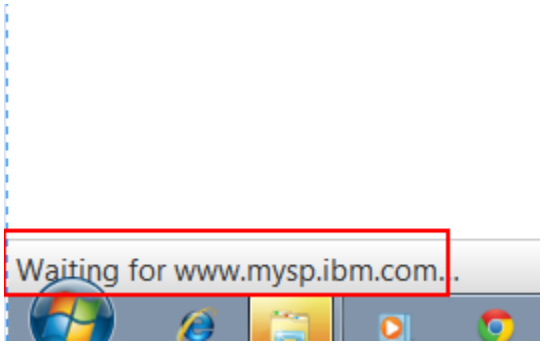
Based on values previously set by following this document, the URL will be:
https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Fsaml20sp%2Fsaml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

Trigger the flow using a browser. You will be asked to log in at the IdP. Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.

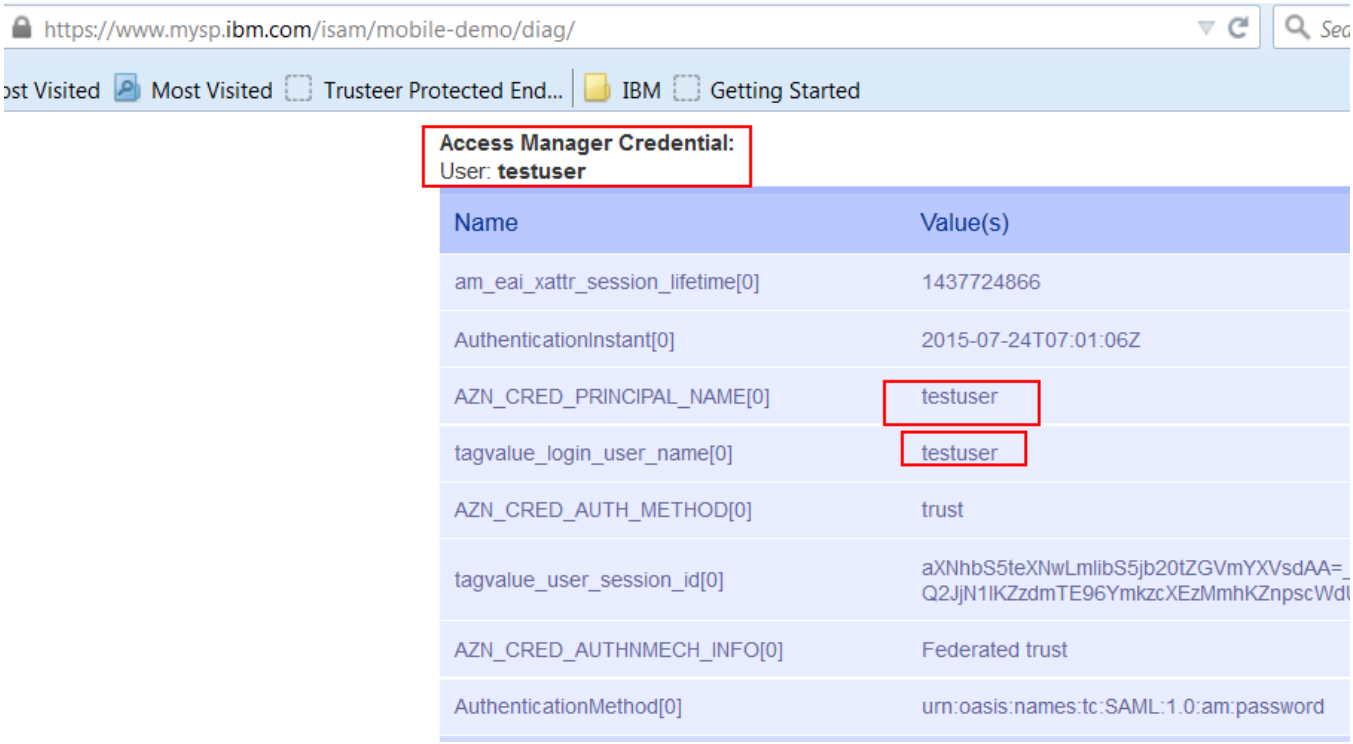


If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.



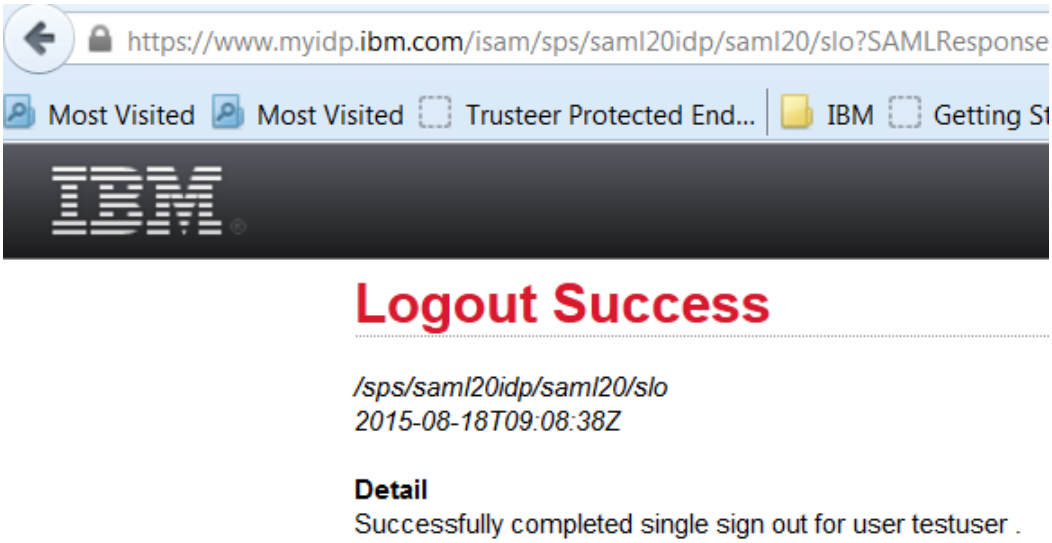


At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:



Single Logout (SLO) scenarios can be triggered from IdP or SP. An IdP initiated SLO can be triggered using https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/sloinitial?RequestBinding=HTTPRedirect

At the end of the IdP initiated SLO flow, a success page as shown below is displayed.



An SP initiated SAML flow which uses HTTP POST can be triggered using:

https://<SP reverse proxy:port>/<junction name>/sps/<service provider federation name>/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https://<IdP reverse proxy:port/<junction name>/sps/<identity provider federation name>/saml20&NameIdFormat=Email&Target=https://<TargetURL>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/saml20sp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https://www.myidp.ibm.com/isam/sps/saml20idp/saml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

At the end of the flow, the landing page will be displayed as shown above.

An SP initiated SLO can be triggered using
https://www.mysp.ibm.com/isam/sps/saml20sp/saml20/sloinitial?RequestBinding=HTTPRedirect

At the end of the SP initiated SLO flow, a success page is shown.

An SP initiated SAML flow which uses HTTP-Redirect can be triggered using:
https://www.mysp.ibm.com/isam/sps/saml20sp/saml20/logininitial?RequestBinding=HTTPRedirect&ResponseBinding=HTTPPost&PartnerId=https://www.myidp.ibm.com/isam/sps/saml20idp/saml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

At the end of the flow, the landing page will be displayed as shown above.

An SP initiated SLO using HTTPPost can be triggered using
https://www.mysp.ibm.com/isam/sps/saml20sp/saml20/sloinitial?RequestBinding=HTTPPost

An IdP initiated SAML flow which uses HTTP-POST and NameIdFormat as Transient can be triggered using:
https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Fsaml20sp%2Fsaml20&NameIdFormat=Transient&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

At the end of the flow, the landing page will be displayed as shown above but note that the authenticated principal is *anonymous* rather than *testuser*:

AZN_CRED_PRINCIPAL_NAME[0]	anonymous
tagvalue_login_user_name[0]	anonymous
tagvalue_max_concurrent_web_sessions[0]	unset
testattr[0]	myvalue
AZN_CRED_NETWORK_ADDRESS_STR[0]	192.168.42.1
NotOnOrAfter[0]	2015-11-17T10:31:06Z
firstName[0]	Test, testuser

An IdP initiated SLO (using POST) can be triggered using
https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/sloinitial?RequestBinding=HTTPPost

This concludes the testing of IBM Security Access Manager SAML 2.0 Federation capability.

12 Customizing IDP Login Screens with Authentication Macros

This section will document a scenario at the identity provider where information about the “in-flight” SSO can be used to customize the login process. This can be useful for several reason – for example to provide a different login page for different SP-partners, or to prompt for a different method of authentication based on information in the SSO request from the service provider.

This capability is documented in the ISAM Knowledge Center:
https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.0/com.ibm.isam.doc/config/concept/CustomizingAuthnLoginForm.html?cp=SSPREK_9.0.0

> In ISAM 9 only the ISAM reverse proxy point of contact type is supported, so authentication macro’s will always only be available as query string parameters to the %URL% macro in the ISAM login.html page.

To complete this scenario we will:

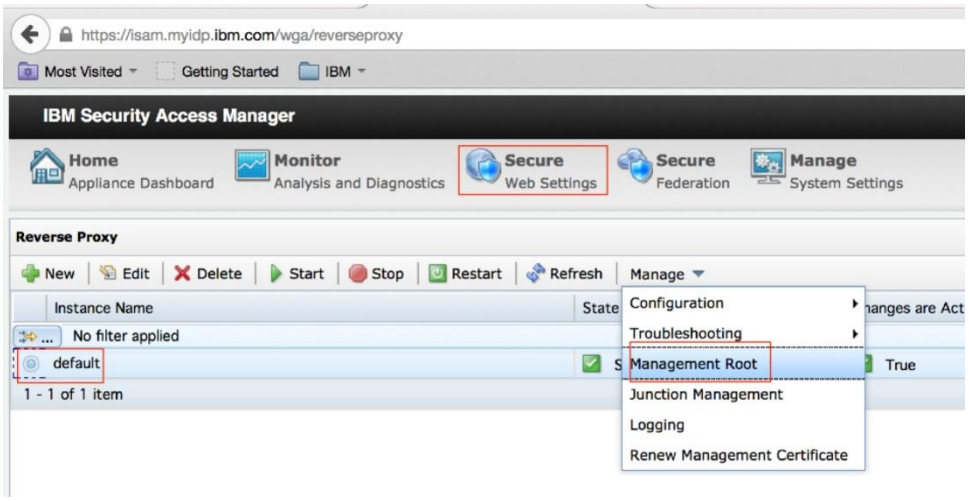
- Replace the login.html at the identity provider with a new one that interprets the %URL% macro to parse out and display authentication macros.
- Configure the federation runtime to supply the %PARTNERID% authentication macro for display.

12.1 Replace login.html on the identity provider

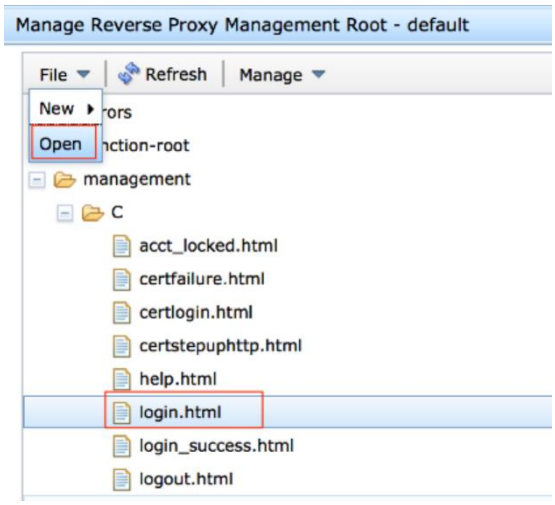
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script **SAMLIPConfig.py –configure Upload_pages**

Using the administration console on the Identity Provider, navigate to **Secure Web Settings -> Manage: Reverse Proxy**.



Select the default reverse proxy, then **Manage -> Management Root**, as shown.



Under management/C (this may vary depending on your machine locale), select the login.html, then File -> Open:

Replace the entire contents of login.html with the text contents found in the *…/providedfiles/pages/saml20/login.html* file.

Notice that this file contains the following additional HTML/Javascript code:

```
        <!--  START ADDED FOR AUTHENTICATION MACROS DEMO -->
                <div id="partnerid" style="display:none">
                </div>
        <script>
         function queryParams(u) {
                var result = {};
                var qp = u.split('?');
                if (qp.length == 2) {
                        qps = qp[1].split('&');
                        for (i in qps) {
                                qpn = qps[i].split('=');
                                if (qpn.length == 2) {
                                        result[decodeURIComponent(qpn[0])] = decodeURIComponent(qpn[1]);
                                }
                        }
                }
                return result;
         }
                        function htmlEncode(value){
                            if (value) {
                                return value.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/>/g,
'&gt;').replace(/"/g, '&quot;');
                            } else {
                                return '';
                            }
                        }

         var queryParameters = queryParams('%URL%');
         var partnerID = queryParameters['PartnerId'];

         if (partnerID != null) {
                document.getElementById("partnerid").innerHTML = "<br/>Partner: " + htmlEncode(partnerID) + "<br/>";
                document.getElementById("partnerid").style.display = "block";
         }
                </script>
        <!--  END ADDED FOR AUTHENTICATION MACROS DEMO -->
```

This code adds a hidden div that is only populated with (html encoded) content and displayed if the %URL% macro includes a query string parameter called PartnerId.

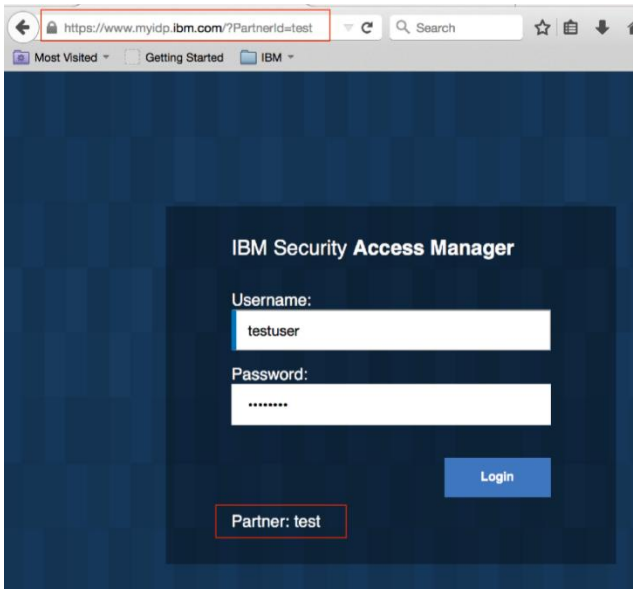
Close the Management Root dialog.

Deploy Pending Changes.
Restart the default reverse proxy.

SCRIPT-END:
The script should display the following:
INFO:WGAManager:Configuring login HTML
INFO:WGAManager:Successfully configured login HTML

You can test the login.html page simply by using this URL in your browser (without a current session at the IdP):

https://www.myidp.ibm.com?PartnerId=test



12.2 Configure authentication macros in the federation runtime

The authentication macros are configured on the federation runtime using an Advanced Configuration property:

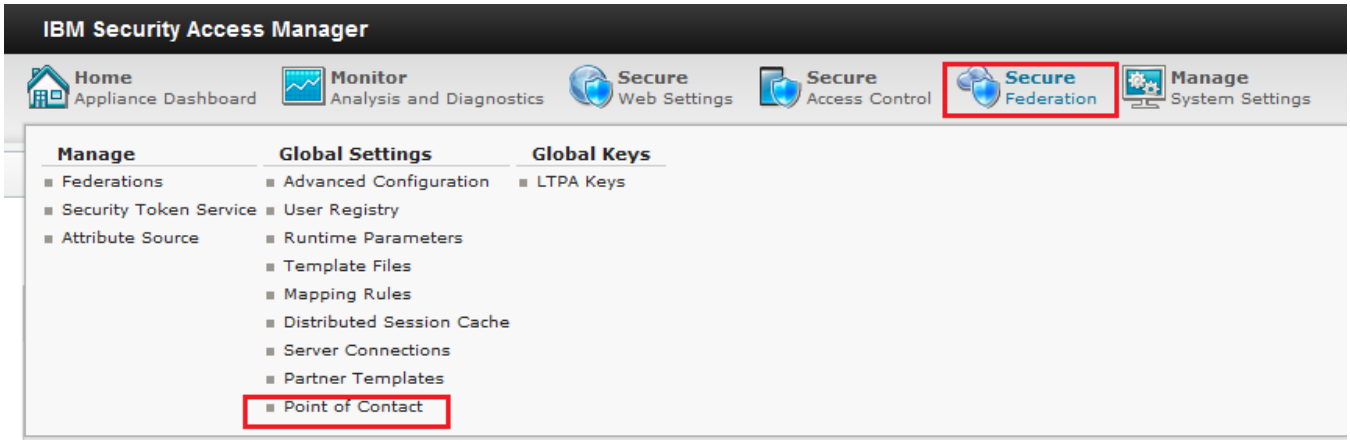
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.
Run this script: **SAMLIPConfig.py –configure Macros**

If you use this script, skip to the corresponding SCRIPT-END notice

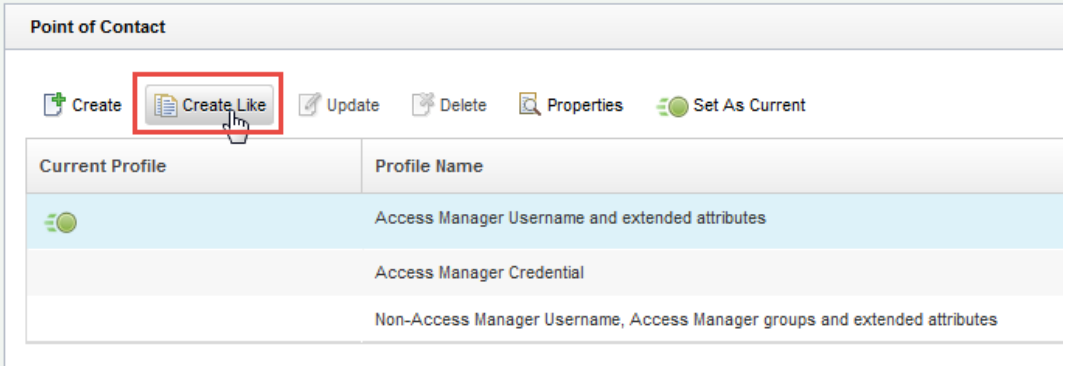
The following steps are only valid for SAM 9.0.1.0 onwards. For SAM 9.0.0.0 and 9.0.0.1, see Section 0 - .

Go to the LMI Admin console of the IdP using URL: https://isam.myidp.ibm.com

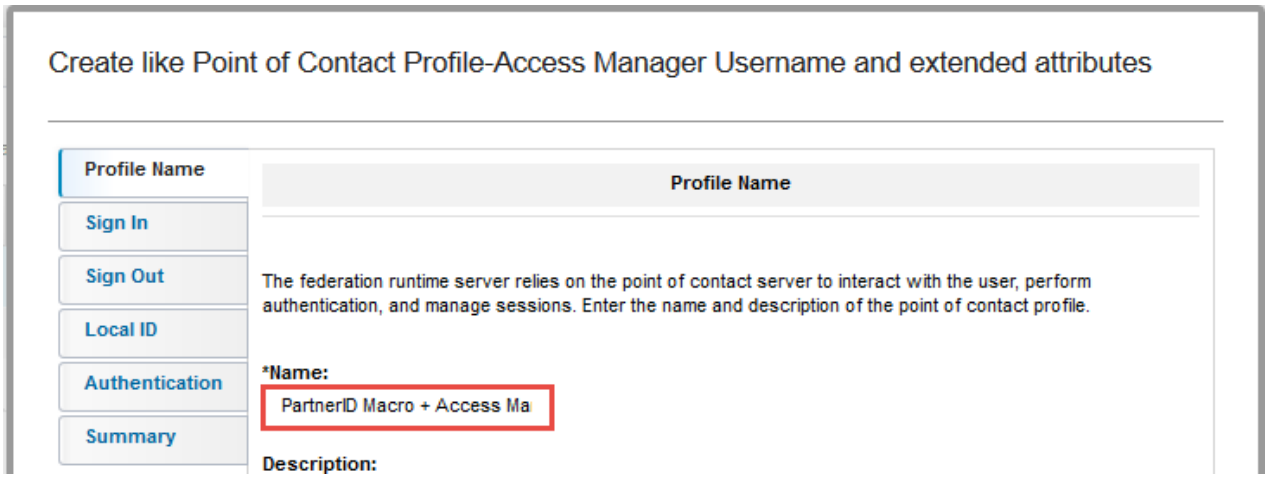
Authenticate with **admin** and **Passw0rd**.



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

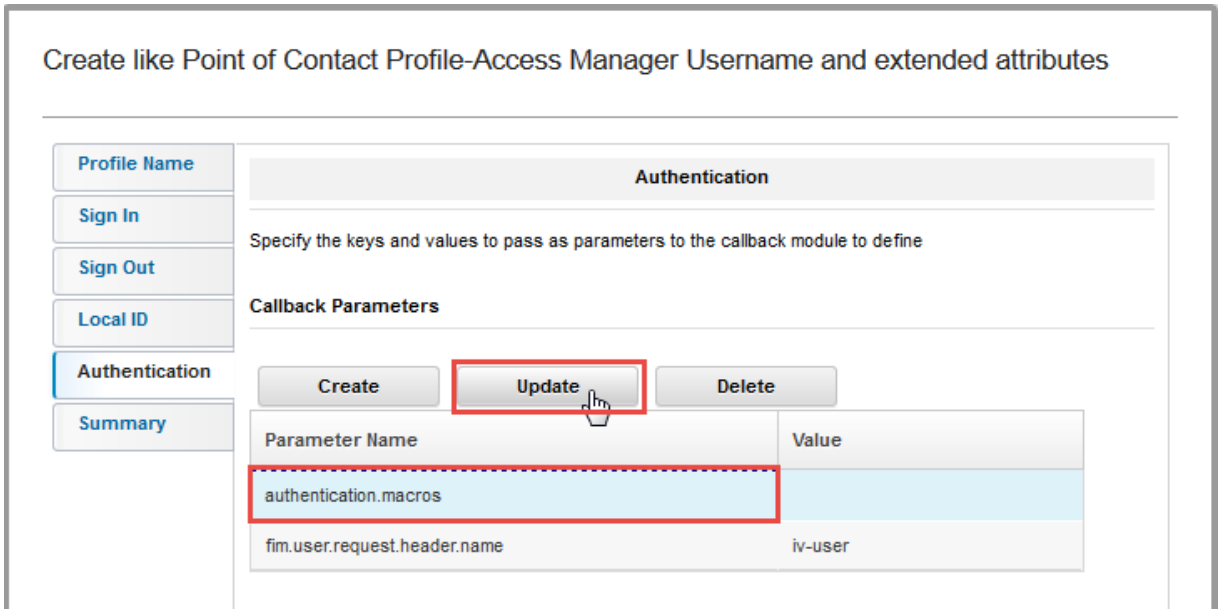


Select the current profile and click **Create Like**.

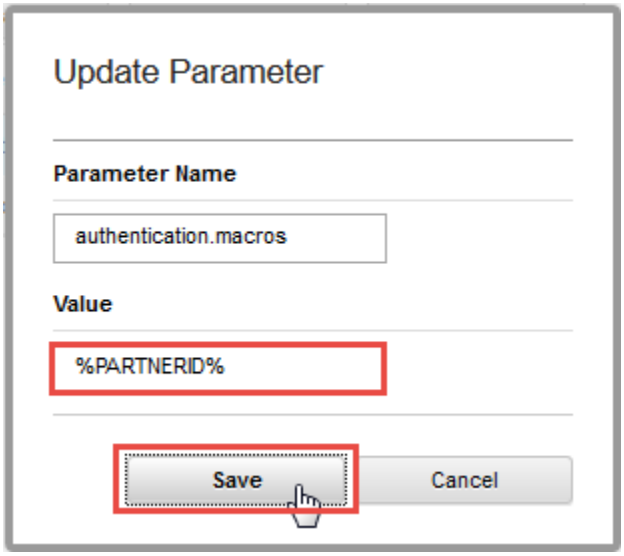


Pre-pend **PartnerID Macro +** to the front of the profile *Name*.

Click **Next** four times to get to the **Authentication** tab.

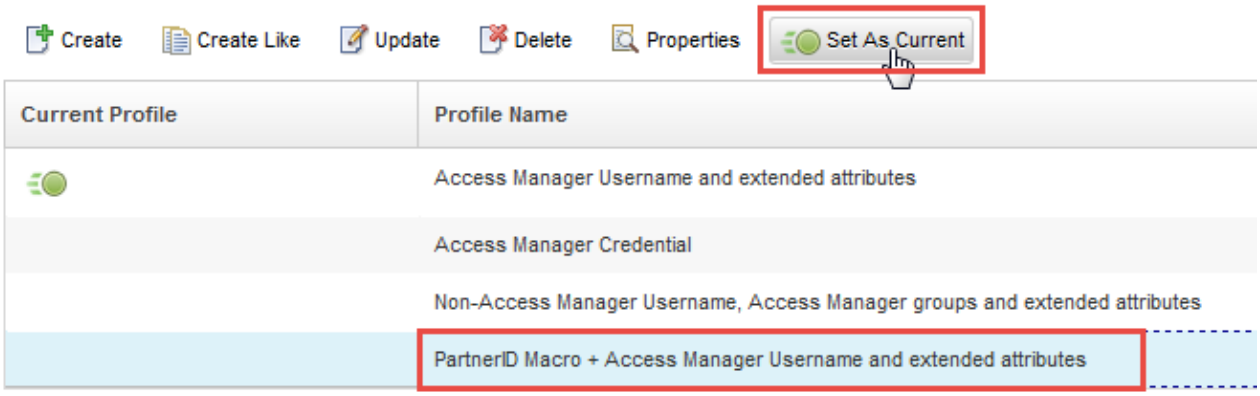


Select the **authentication.macros** row and click **Update**.

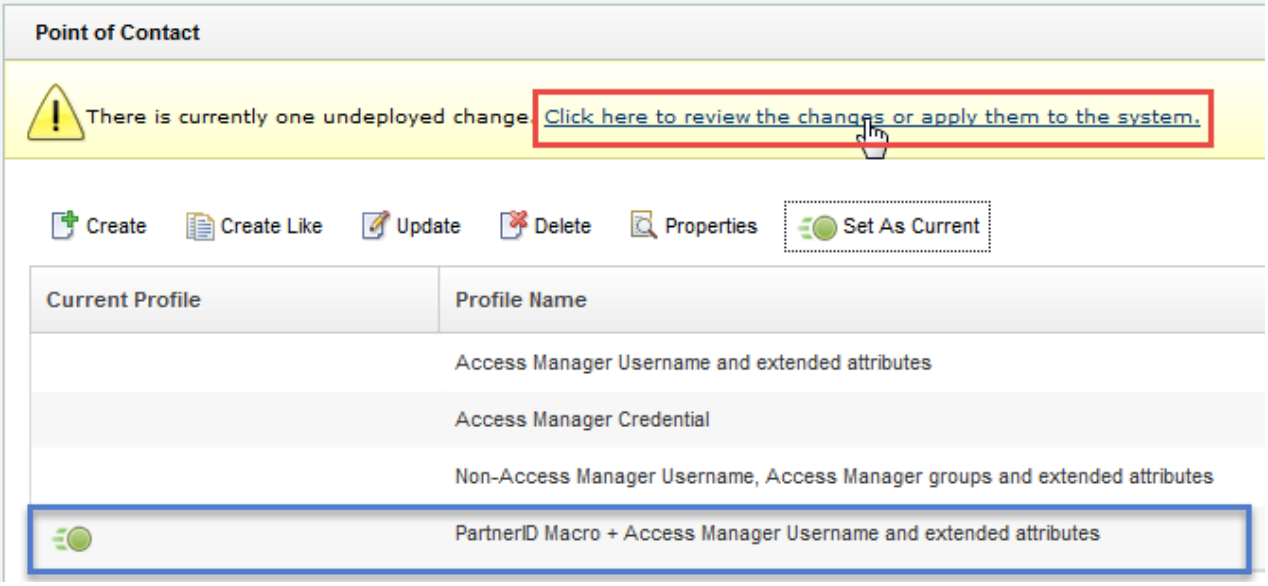


Enter **%PARTNERID%** as the *Value* of the parameter and click **Save**.

Click **Next** and then **Finish** to save the new POC Profile.



Select the new profile and click **Set As Current**.

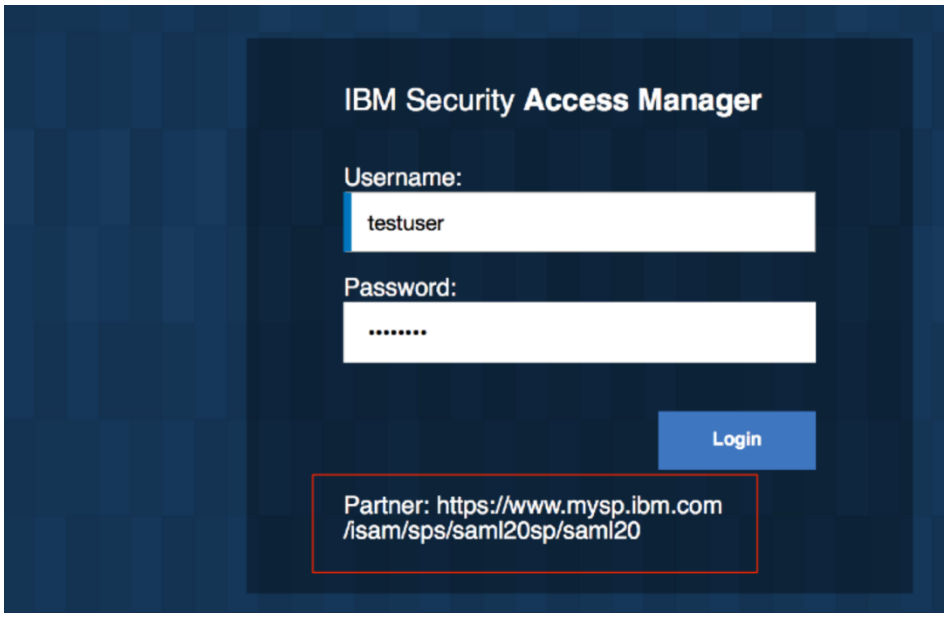


Check that the profile is now marked as current then **Deploy** the changes.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Setting Macros in the login page
INFO:FederationManager:Successfully set Macros in the login page

12.3 Testing the authentication macros

Performing a new SAML SSO without an authenticated session at the identity provider should display the Partner entity id in the IDP's login page:



Advanced exercise: Try changing the poc.websealAuth.authenticationMacros parameter to :

%PARTNERID%,%SSOREQUEST%

Perform SAML SSO again (without an existing authenticated session at the IdP), and notice the additional query string parameter on the ISAM login URL bar:

SSORequest

The value of SSORequest is the signed base64-encoded version of the entire SSO message. Understanding this **could** be changed by a malicious browser-user, you could use Javascript base64 decoding to unencode this value and determine other attributes of the SSO request. For example if this was an SP-initiated SSO, and the required included AuthenticationContext properties, it would be possibly to customize the login process that takes place programmatically within the login page.

This concludes exercises in this cookbook for authentication macros.

13 OpenID Connect

In this section, a single API Definition is created which support OpenID Connect (OIDC), and multiple relying party federations are created, supporting the Implicit, Authorization Code and Hybrid flows.

It is assumed that your IDP and SP image already has the basic set up completed.

13.1 Open ID Connect Provider

This section is completed only for the Identity Provider.
You will configure the Service Provider in a later section.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

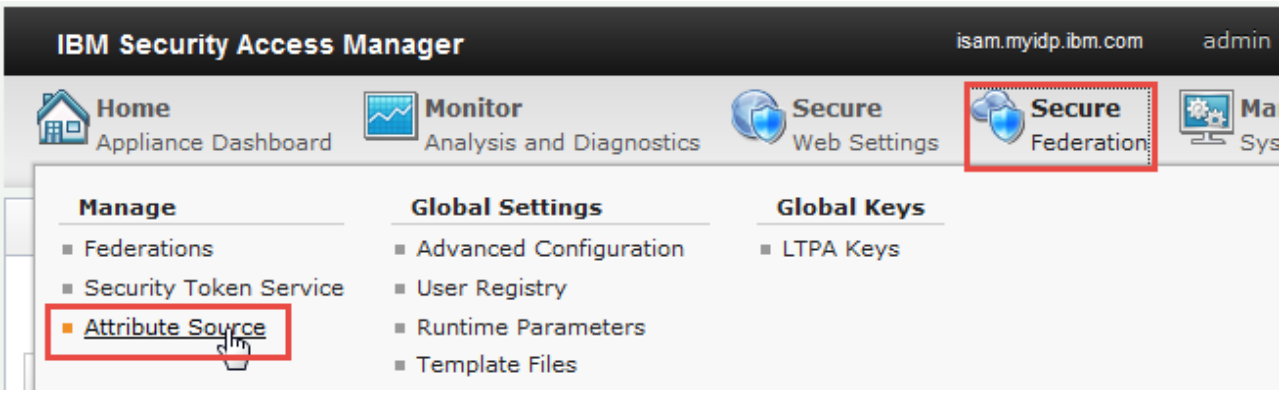
Run this script **OIDCOPConfig.py –configure All**

13.1.1 Configuring Attribute Sources

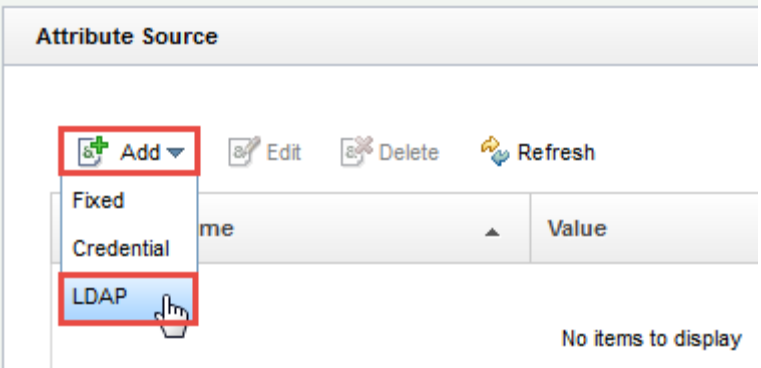
Attribute sources define where a particular attribute comes from, along with any configuration required to obtain that attribute. Attribute sources are referenced from the API Definition.

In this cookbook we will create two attribute sources – both are attributes read from the local LDAP server. The first represents the “displayName” attribute, and the second a “phone” attribute.

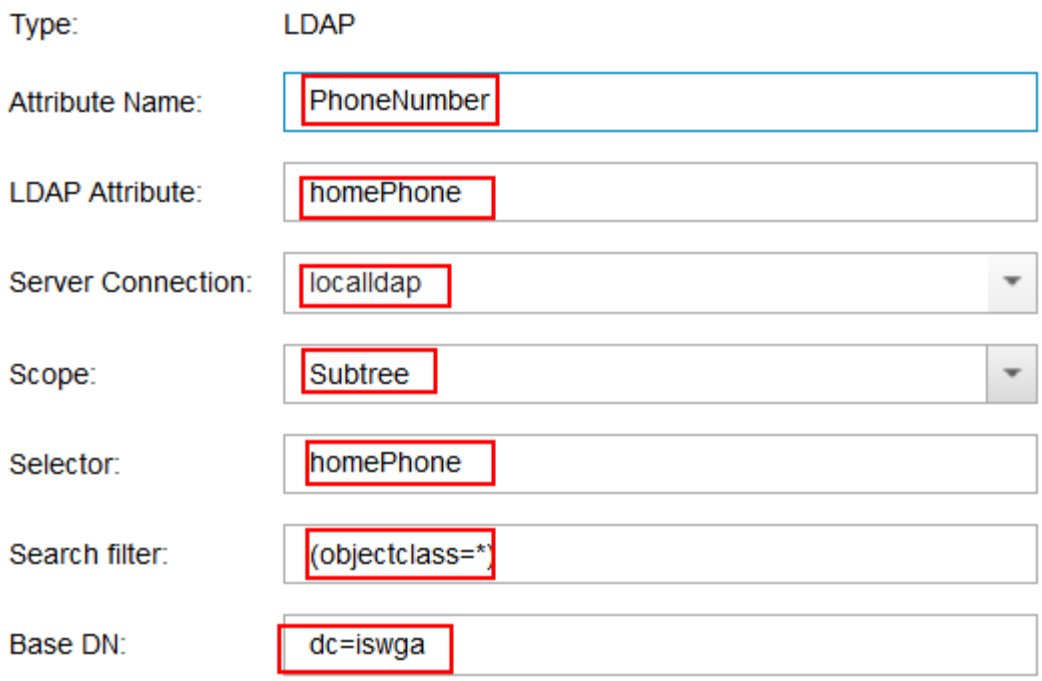
The local ldap server connection configuration is in section 18.3.2



Navigate to **Secure Federation -> Manage: Attribute Source**.



Click on the **Add** button and select **LDAP** from the drop-down list.



Complete the following properties:

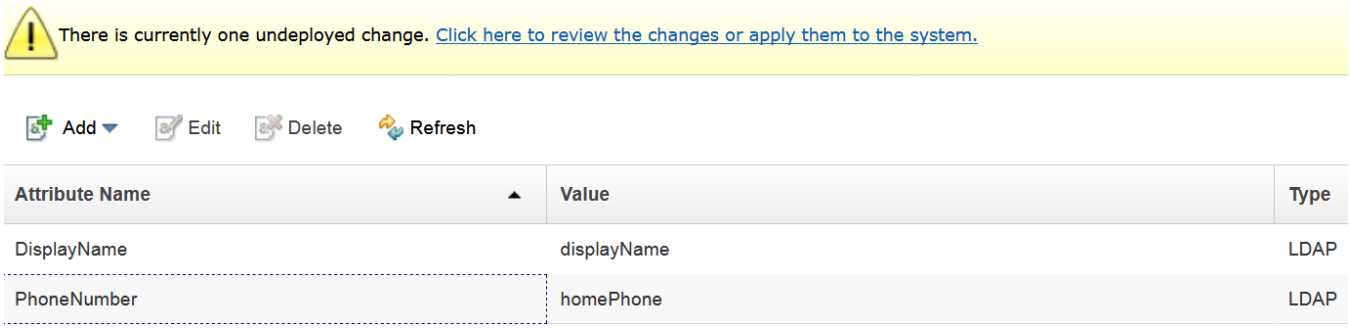
and then click **Add**.

Property	Value
Attribute Name	**`PhoneNumber`**
LDAP Attribute	**`homePhone`**
Server Connection	**`localldap`**
Scope	**`Subtree`**
Selector	**`displayName`**
Search filter	**`(objectclass=*)`**
Base DN	**`dc=iswga`**

Repeat the previous steps to create another attribute source for DisplayName:

Property	Value
Attribute Name	**DisplayName**
LDAP Attribute	**displayName**
Server Connection	**localldap**
Scope	**Subtree**
Selector	**homePhone**
Search filter	**(objectclass=*)**
Base DN	**dc=iswga**

After adding both LDAP attributes they appear in the table as follows:



Deploy Changes.

This completes the configuration of LDAP attribute sources. Later these will be referenced in the AttributeMap module configuration.

In this section, a single API Definition is created which supports OpenID Connect.

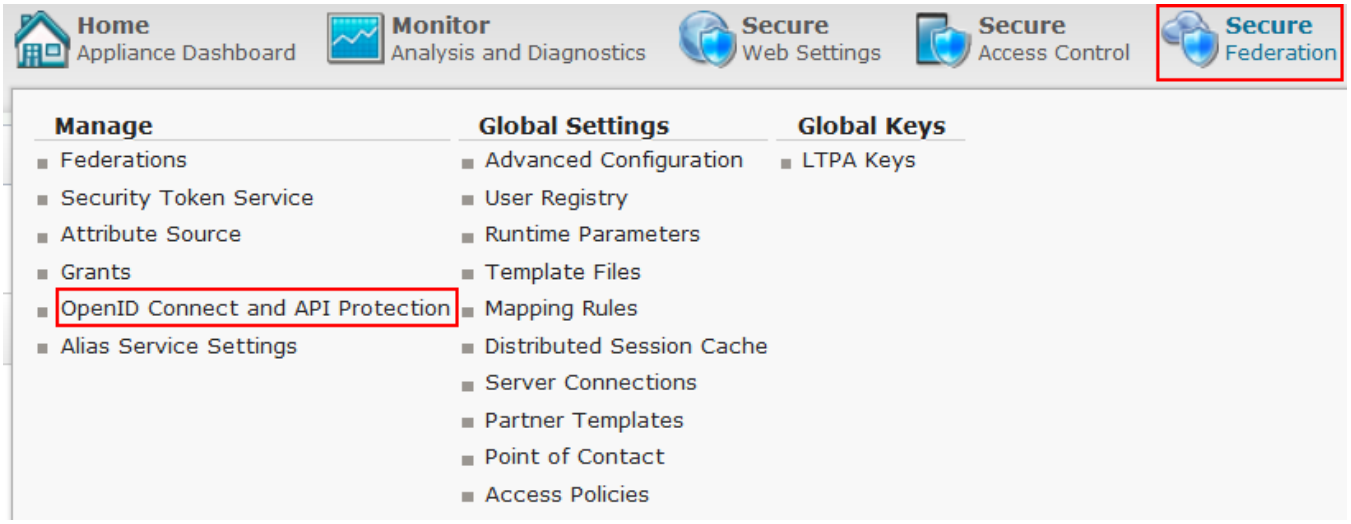
13.1.2 Create API Definition with OIDC Enabled

SCRIPT-START:

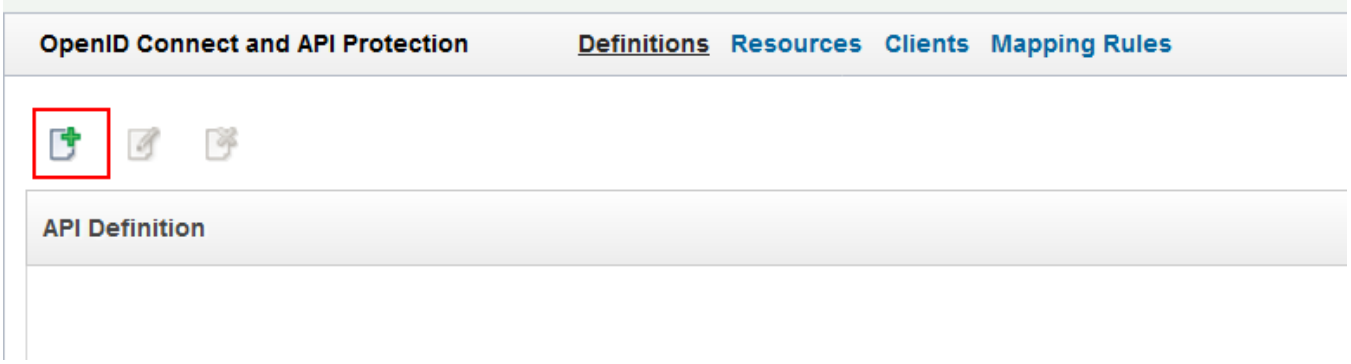
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

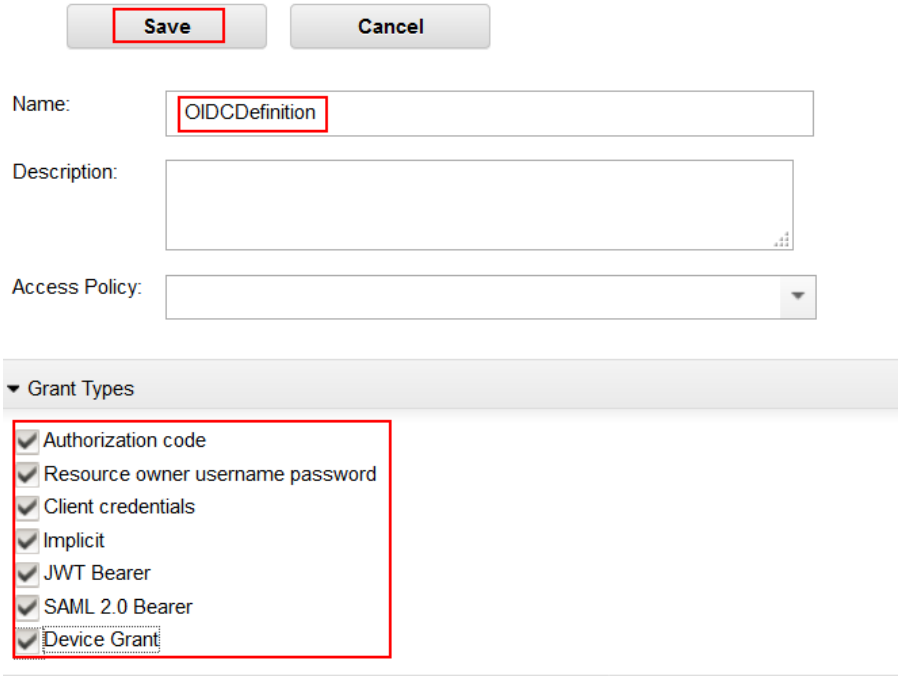
Run this script **OIDCOPConfig.py –configure Definition**

Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**

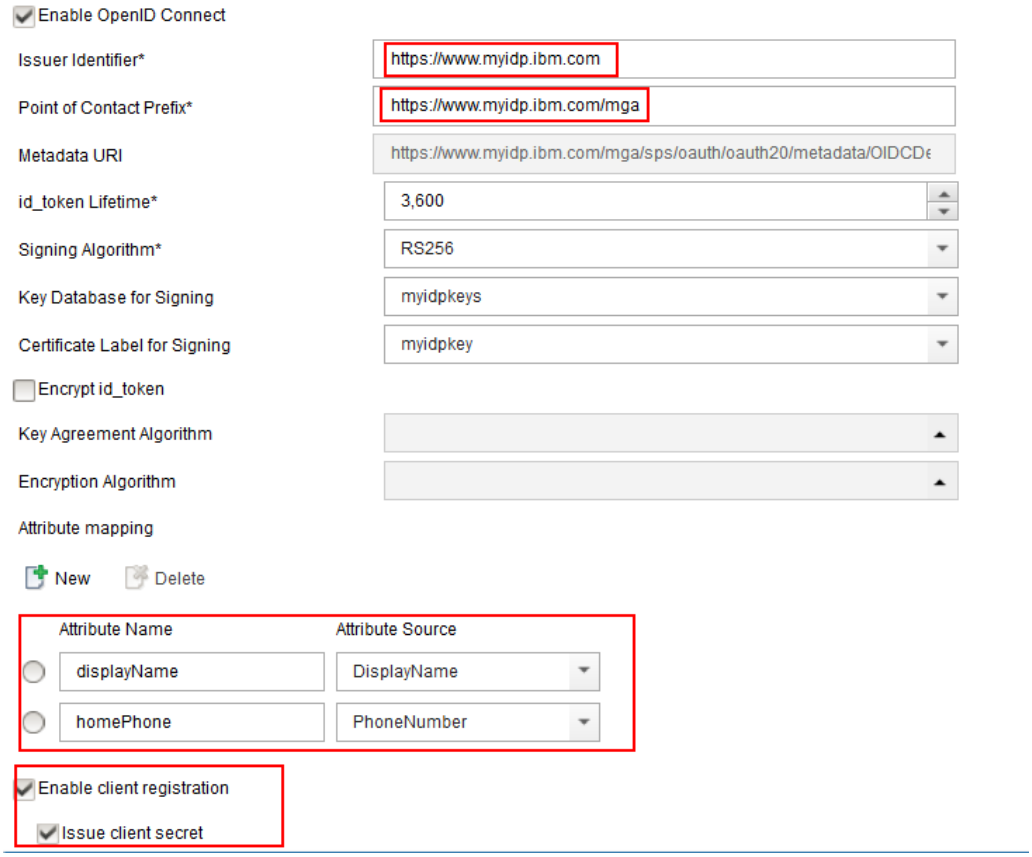


Click **Add** to create a new API Definition





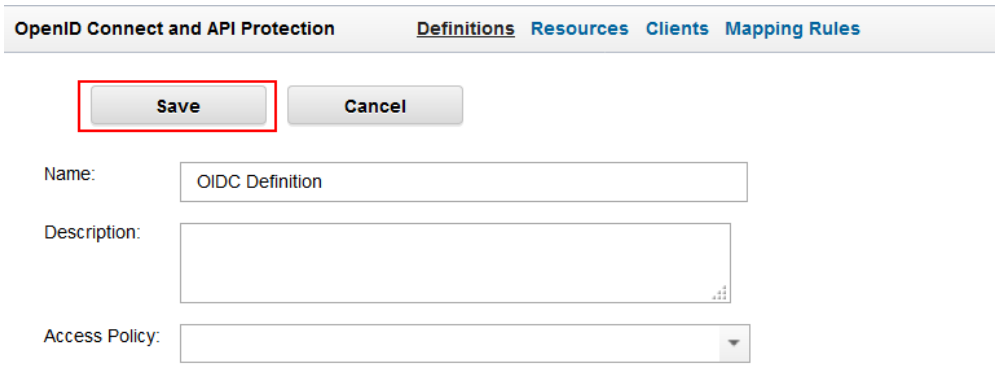
Create a new API Definition named **OIDCDefinition** as shown and enable all the **Grant Types.**



Leave the **Token Management** and **Trusted Clients and Consent** as default. Enable OpenID Connect by enabling the checkbox **Enable OpenID Connect**.

Set Issuer Identifier to https://www.myidp.ibm.com**,** set Point of Contact Prefix to https://www.myidp.ibm.com/mga**.** The pre-requisites for attribute mapping is explained in detail in Section 18.3.2

We are configuring attribute mapping and client registration for Section 24 and 25 respectively.
Configure Attribute Mapping set Attribute Name to **displayName** and Attribute Source to **LDAPDisplayName**, configure Attribute Mapping set Attribute Name to **homePhone** and Attribute Source to **LDAPPhoneNumber** and select the checkbox **Enable client registration** and **Issue client secret.**



Click **Save** to create the API Definition

Follow on-screen instructions to **deploy** pending changes.

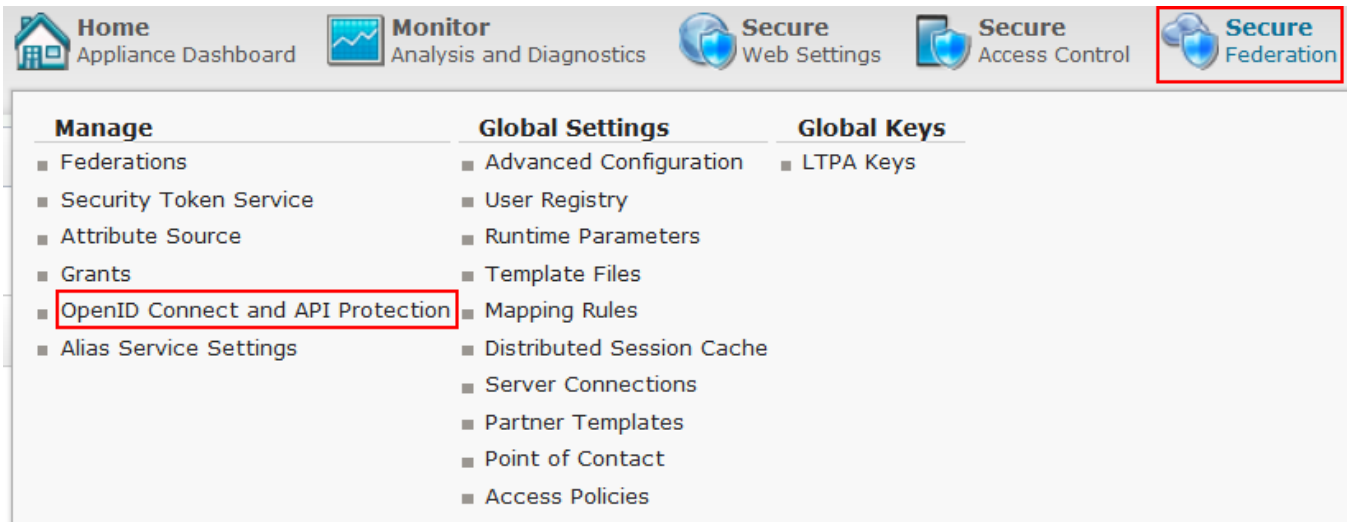
SCRIPT-END:
The script should display the following:
INFO:FederationManager:Configuring the easuser password
INFO:FederationManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the server connection
INFO:FederationManager:Successfully configured the server connection
INFO:FederationManager:Configuring Attribute sources
INFO:FederationManager:Successfully configured attribute sources
INFO:FederationManager:Configuring the OIDC Definition
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Successfully configured the OIDC Definition

13.1.3 Configuring Clients

In this section, a client is created.

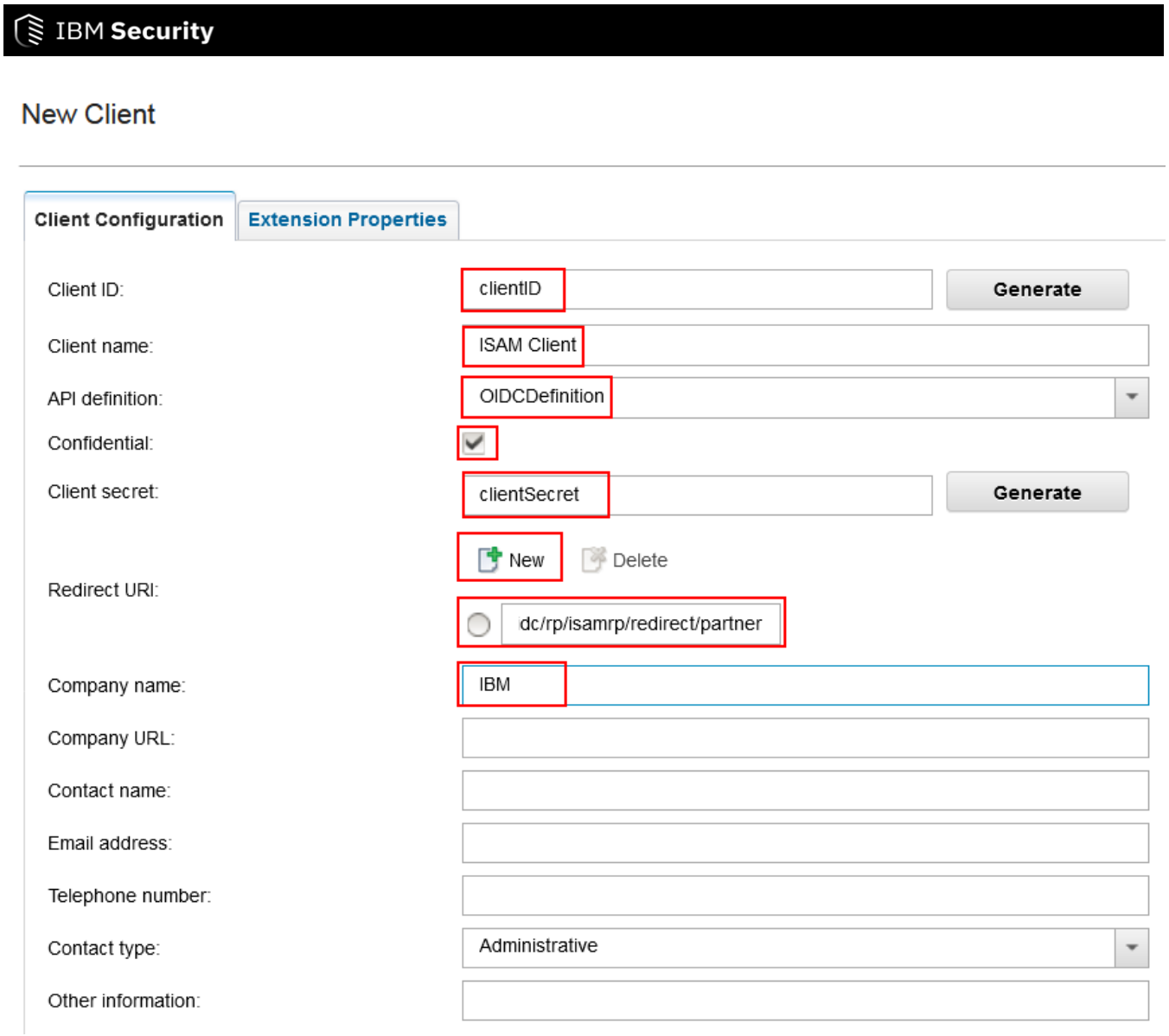
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script **OIDCOPConfig.py –configure Client**



Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**

Navigate to **Clients** and click on **Add** to create a New Client



On the Client Configuration panel, enter the **clientID** as Client ID, enter **ISAM Client** as the Client Name and select **OIDCDefinition** as the API Definition, select checkbox **Confidential** and enter **clientSecret** as Client Secret**.**

Since we three relying party federations to support hybrid, implicit and code flows, enter https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/redirect/partner**,** https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp_implicit/redirect/partner and https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp_code/redirect/partner as a Redirect URI by clicking New**,** enter **IBM** as the Company Name**.**

Click on **OK** to create a Client.
Follow on-screen instructions to **deploy** pending changes

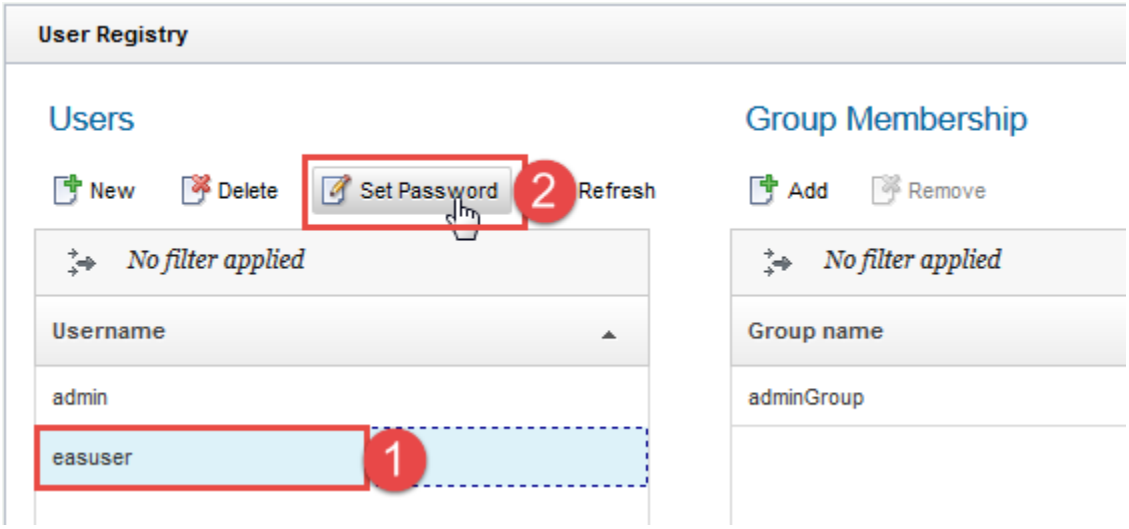
SCRIPT-END:
The script should display the following:
INFO:FederationManager:Configuring the Client
INFO:FederationManager:Successfully configured Client

13.1.4 Updating easuser password

The federation runtime has its own authentication requirements for access to the STS endpoints, and this is provided by the federation runtime user registry. This built-in registry (which is independent to the LDAP registry used by the Reverse Proxy) includes a preconfigured user called “easuser” which has a default password of “passw0rd”. The easuser is typically used in ISAM reverse proxy configuration to allow it to be a client of the STS, and we will see this in action later in the document.

For now, we will change the easuser password to “Passw0rd”, for consistency with other passwords used throughout this cookbook, and so that you can see where and how this is done.

In the LMI, navigate to **Secure Federation -> User Registry**.



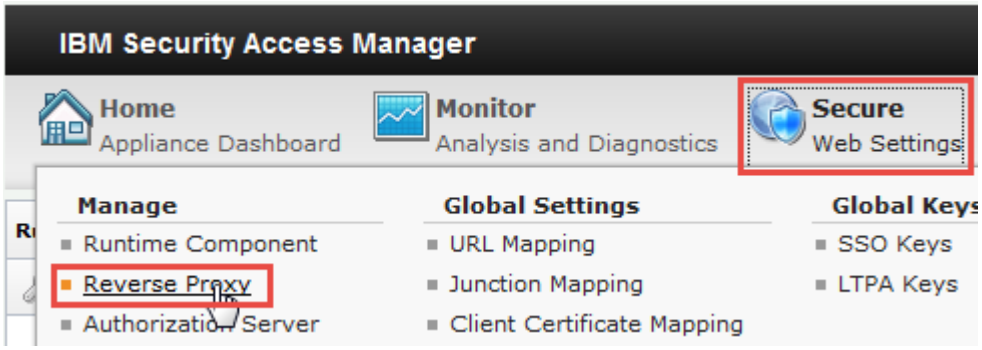
Select **easuser** and click **Set Password**

13.1.5 Configuring Reverse Proxy for OpenID Connect Provider

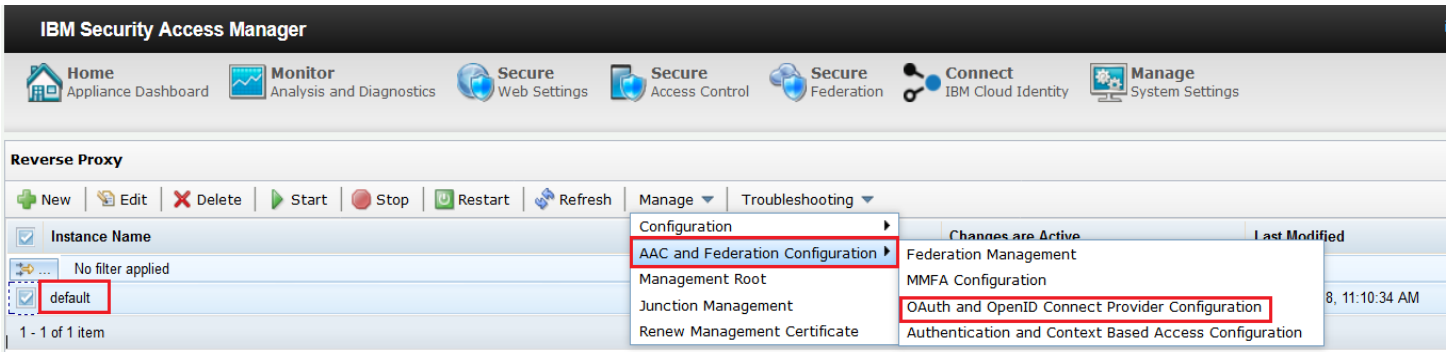
In this section we are configuring the reverse proxy instance for OAuth and OpenID connect provider.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

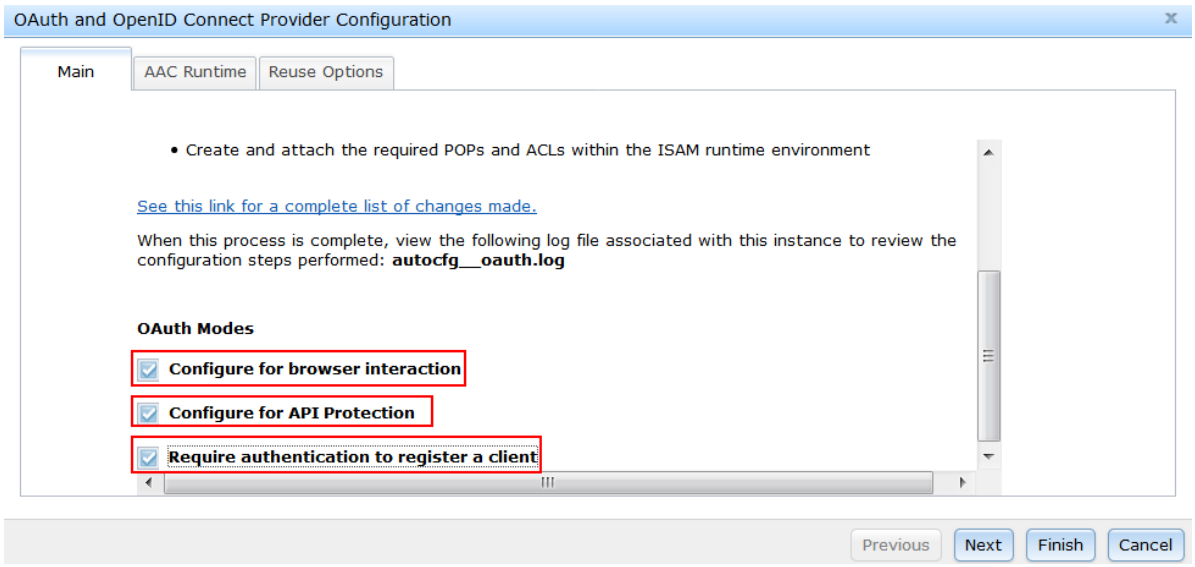
For the IdP, run this script: **OIDCOPConfig.py -configure WebSEAL_Conf_OIDC_OP**



In the mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.

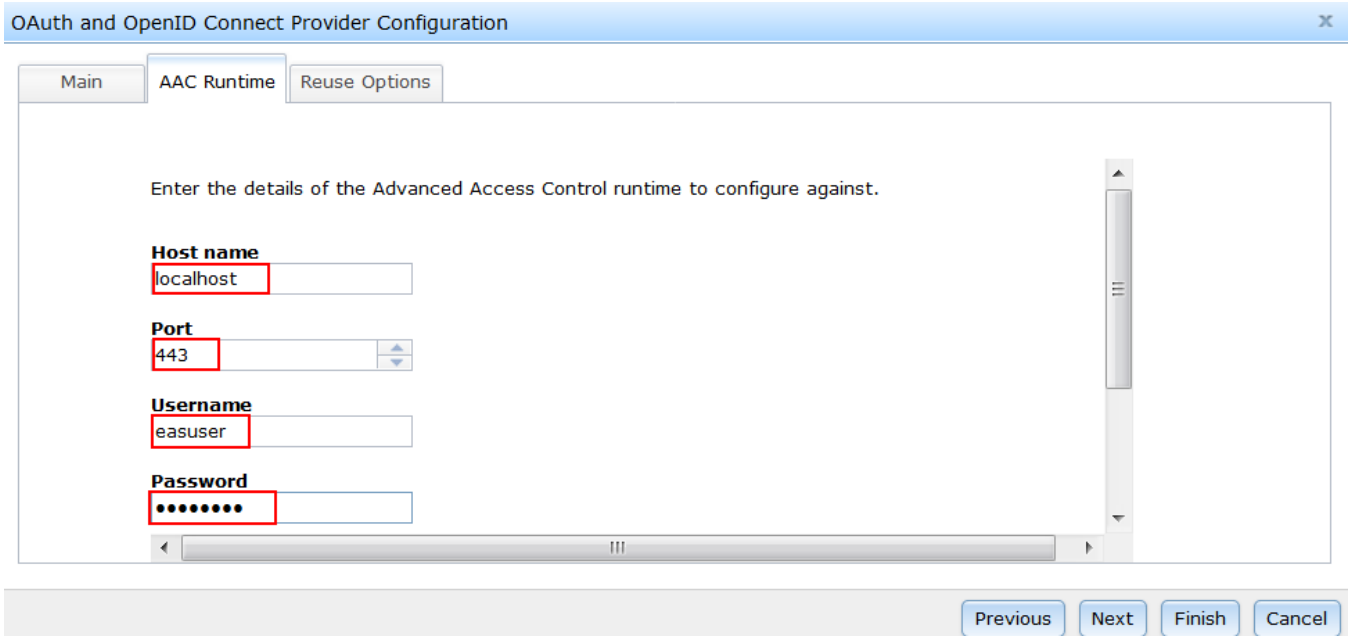


Select the Reverse Proxy instance and click on **Manage -> AAC and Federation Configuration -> OAuth and OpenID Connect Provider Configuration**.

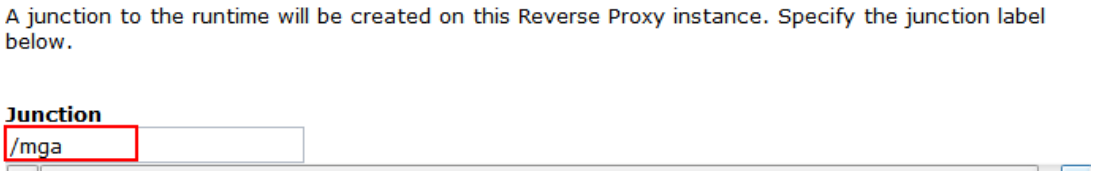


There are three panels which need to be filled out. In the **Main** panel, select all the checkboxes **Configure for browser interaction** – the /authorize and the /session endpoints are made accessible, **Configure for API Protection** – this configures the oauth-auth and oauth-cluster stanza, **Require authentication to register a client** – this sets an anyauth ACL to the client registration endpoint.

Click **Next.**

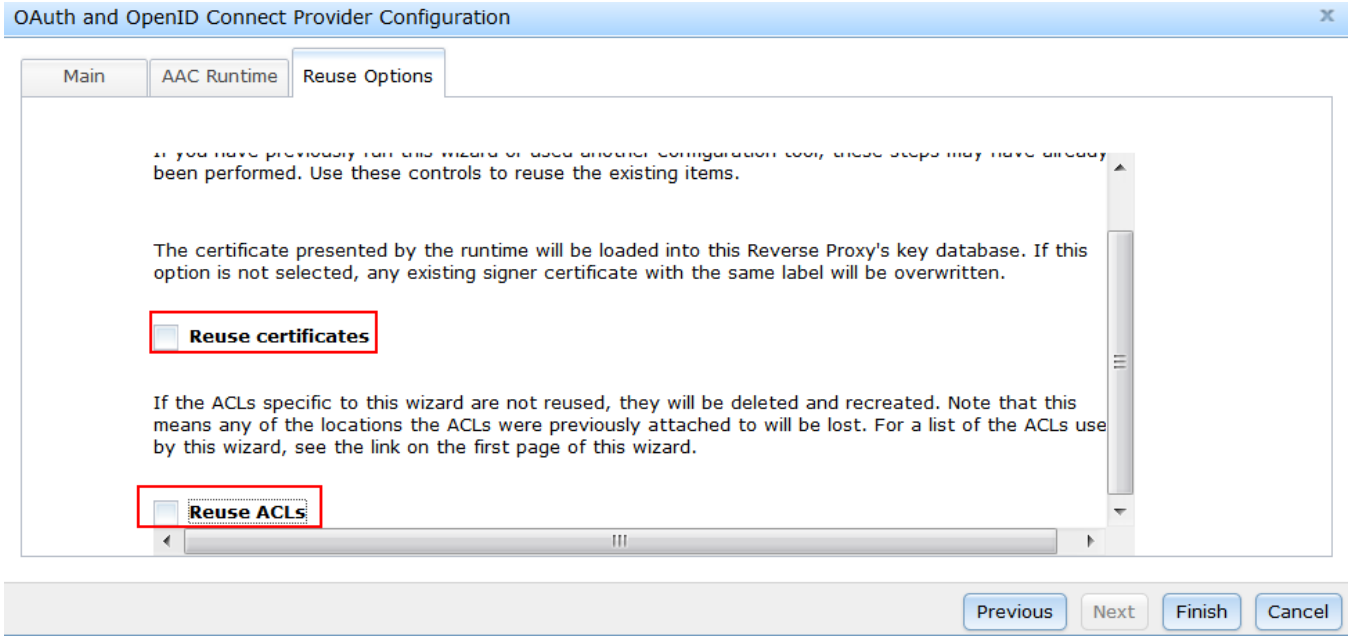


Inside the **AAC Runtime** pane, provide the details to authenticate with federation runtime. The details include the host, port, user name and password. All of them are required. When you move to the next pane, these details are used to connect to the Runtime.



The default junction name used is **/mga**.

Click **Next.**



The next tab is the ACLs and Certificates panel, you can choose to reuse ACLs and Certificates if they exist or create new ones.

Once all the panels are done, click on **Finish** and then **Deploy** the Pending changes.

SCRIPT-END:

The script should display the following for IdP:

INFO:WGAManager:Configure WebSEAL for OIDC OP

INFO:WGAManager:Successfully configured WebSEAL for OIDC OP

If the configure -All option was used the script end should look like this

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Configuring the easuser password
INFO:FederationManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the server connection
INFO:FederationManager:Successfully configured the server connection
INFO:FederationManager:Configuring Attribute sources
INFO:FederationManager:Successfully configured attribute sources
INFO:FederationManager:Configuring the OIDC Definition
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Successfully configured the OIDC Definition
INFO:FederationManager:Configuring the Client
INFO:FederationManager:Successfully configured Client
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:WGAManager:Configuring WebSEAL for OIDC OP
INFO:WGAManager:Successfully configured WebSEAL for OIDC OP

13.2 OpenID Connect Relying Party

This section is completed only for the Service Provider.

In this section we configure the relying party (SP) to create multiple federation which support hybrid, implicit and authorization code flows and their respective partners for OpenID connect relying party configuration.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

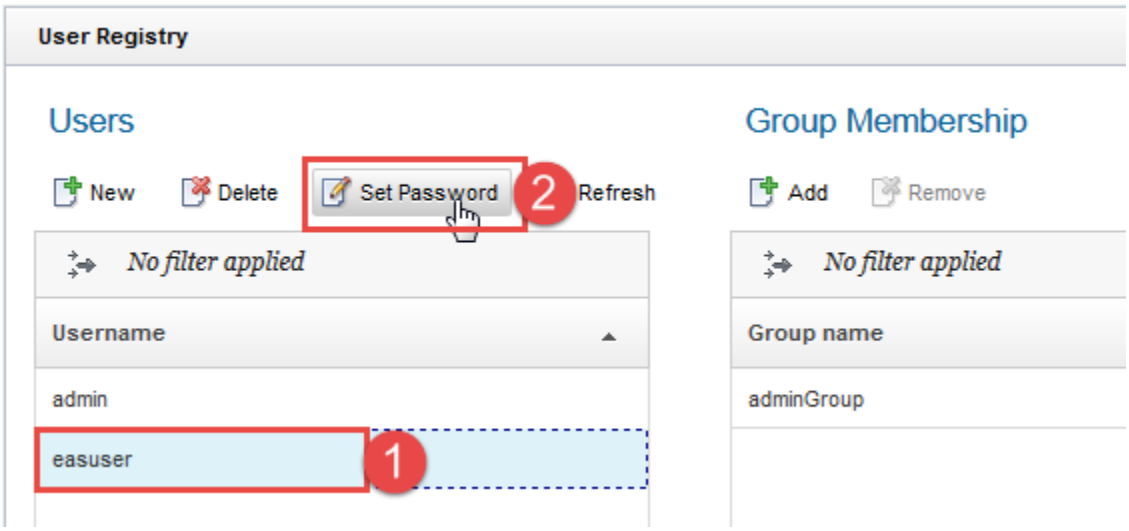
For the SP, run this script: **OIDCRPConfig.py -configure All**

13.2.1 Updating easuser password

The federation runtime has its own authentication requirements for access to the STS endpoints, and this is provided by the federation runtime user registry. This built-in registry (which is independent to the LDAP registry used by the Reverse Proxy) includes a preconfigured user called “easuser” which has a default password of “passw0rd”. The easuser is typically used in ISAM reverse proxy configuration to allow it to be a client of the STS, and we will see this in action later in the document.

For now, we will change the easuser password to “Passw0rd”, for consistency with other passwords used throughout this cookbook, and so that you can see where and how this is done.

In the LMI, navigate to **Secure Federation -> User Registry**.



Select **easuser** and click **Set Password**

13.2.2 Uploading mapping rules

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps.

For the SP, run this script: **OIDCRPConfig.py –configure Upload_Mapping_Rules**
If you use this script, skip to the corresponding SCRIPT-END notice

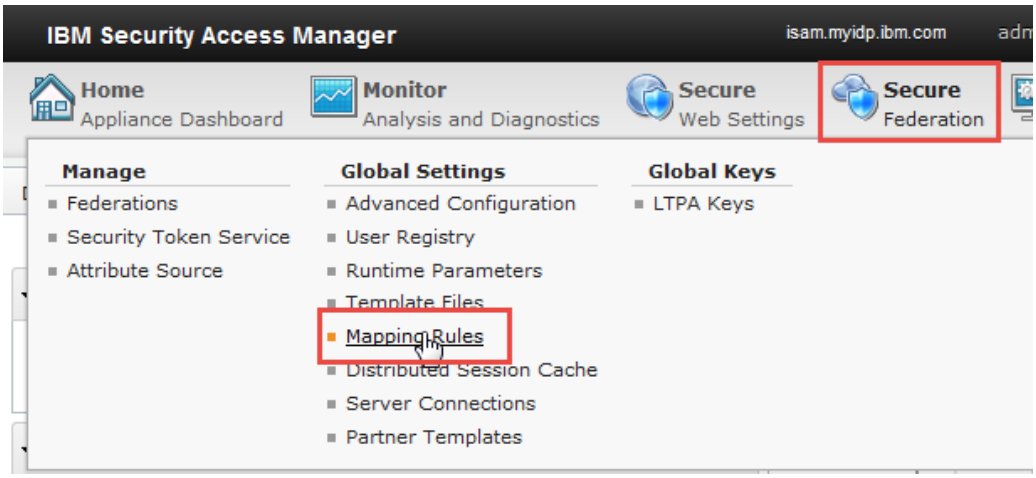
This document makes use of a number of JavaScript mapping rules. These need to be created on the Identity Provider appliance. We will actually create quite a few mapping rules at this time although the SAML federation will initially use only the first of these rules.

When using the appliance console to create Mapping Rules, cut-and-paste is used to load the JavaScript content of the rules. Before we get started, we need to open our first rule in a text editor so we can copy it.

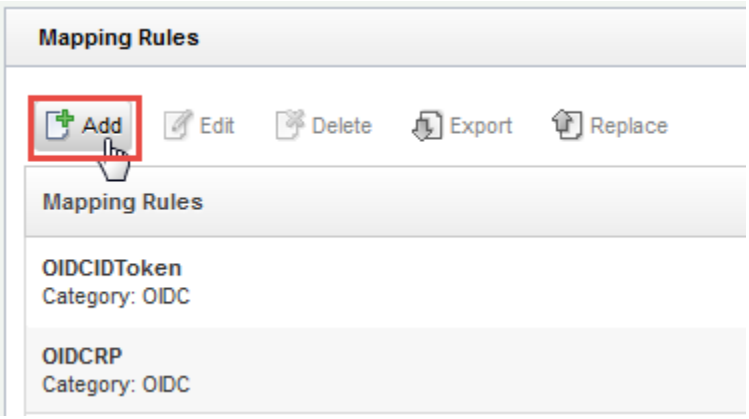
Go to the **…/providedfiles/mappingrules/sp** directory and open the **oidc_adv.js** file in a text editor.

Select all the text in the file and then copy it. On Windows you can use **Ctrl-a** to select all and **Ctrl-c** to copy.

Now we're ready to create a Mapping Rule on the appliance with this content.



In the LMI Administration console, navigate to **Secure Federation→Global Settings: Mapping Rules**.



Click **Add** to add a new mapping rule.

Paste the rule text into the *Content* box. On Windows you can use **Ctrl-c** to paste.

Enter **oidc_adv** as the rule *Name* and select **OIDC** as the *Category*.

Click **Save** to save the new Mapping Rule.

Repeat the process above for all of the files in the *…/providedfiles/mappingrules/sp* directory

Once all Mapping Rules are loaded, **deploy** the pending changes.

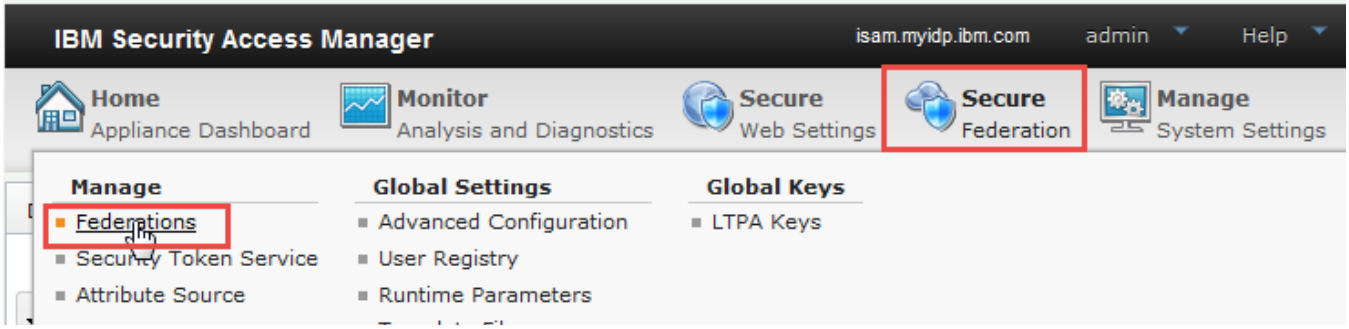
SCRIPT-END:
The script should display the following
INFO:FederationManager:Upload all mapping rules
INFO:FederationManager:Update a mapping rule
INFO:FederationManager:Successfully updated the Mapping Rule
INFO:FederationManager:Update a mapping rule
INFO:FederationManager:Successfully updated the Mapping Rule
INFO:FederationManager:Update a mapping rule
INFO:FederationManager:Successfully updated the Mapping Rule
INFO:FederationManager:Update a mapping rule
INFO:FederationManager:Successfully updated the Mapping Rule

13.2.3 Configuring OpenID Connect Relying Party Federation for hybrid flow

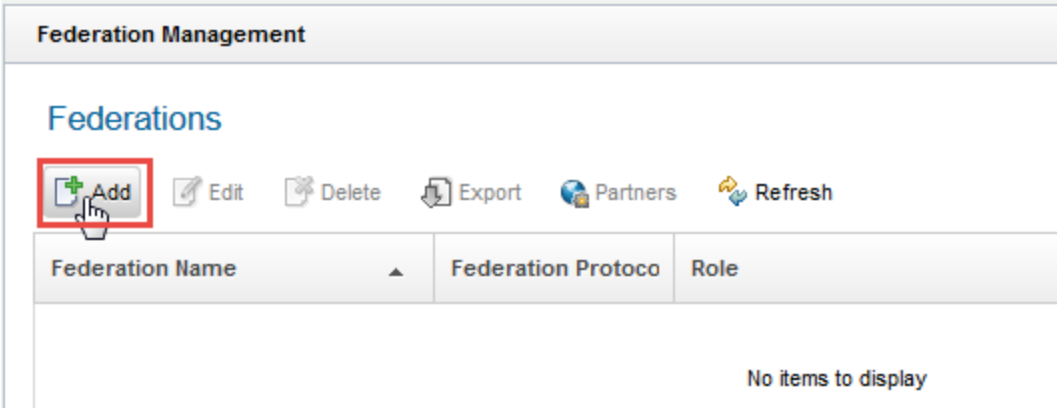
In this section we configure the relying party (SP) to a federation which supports hybrid flow.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

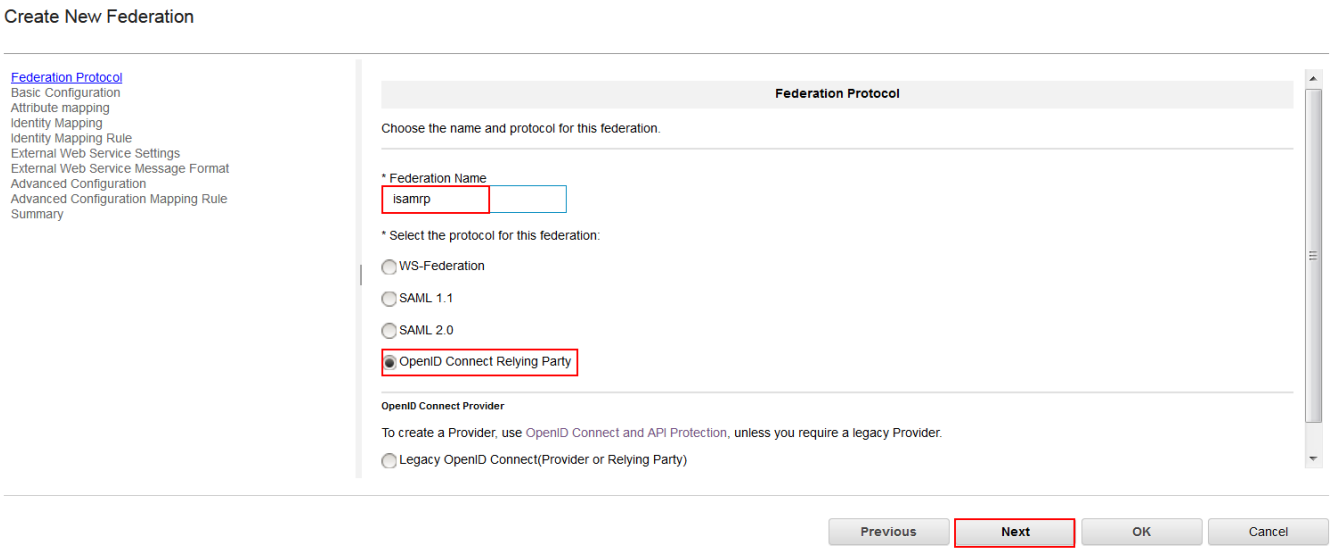
For the SP, run this script: **OIDCRPConfig.py -configure RPFederation**



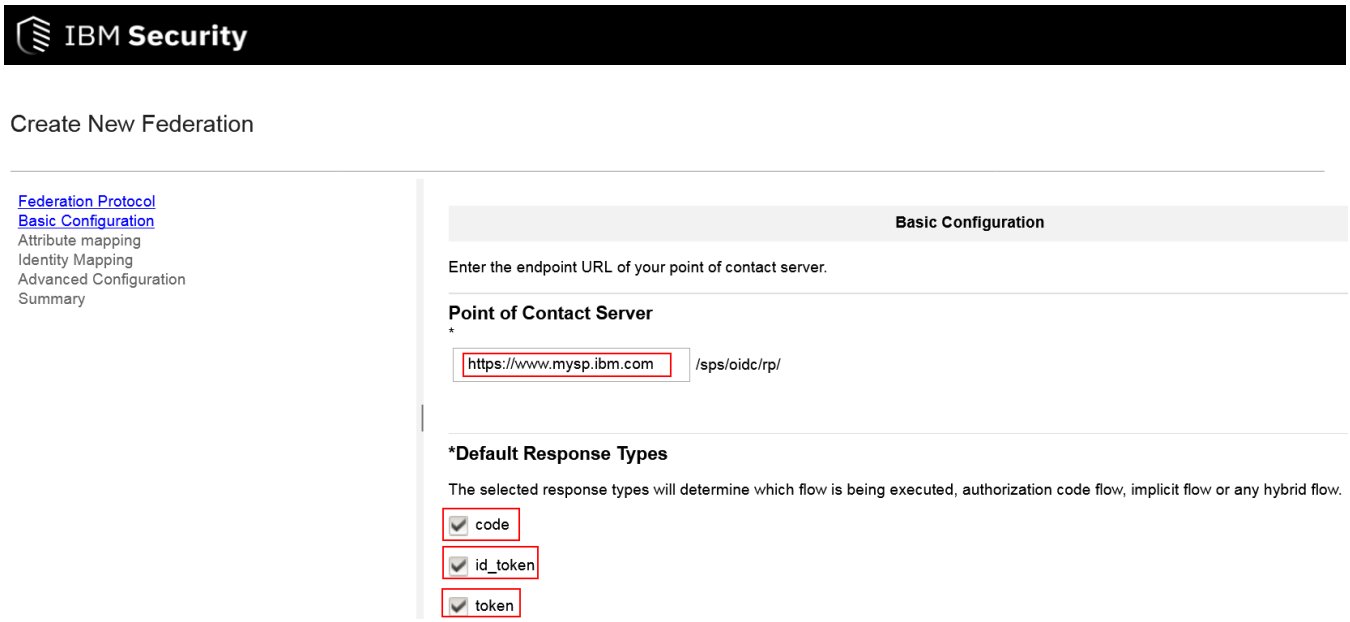
Using the administration console, navigate to **Secure Federation→Manage: Federations**.



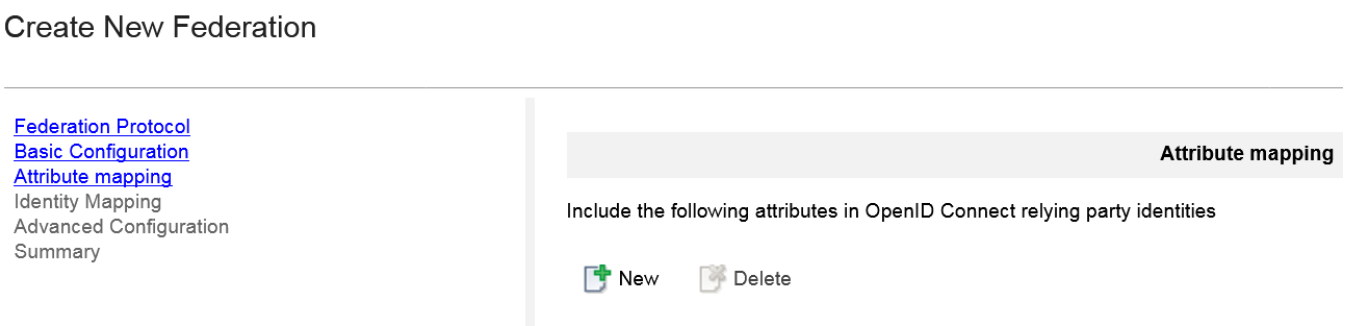
Click **Add** to create a new federation.



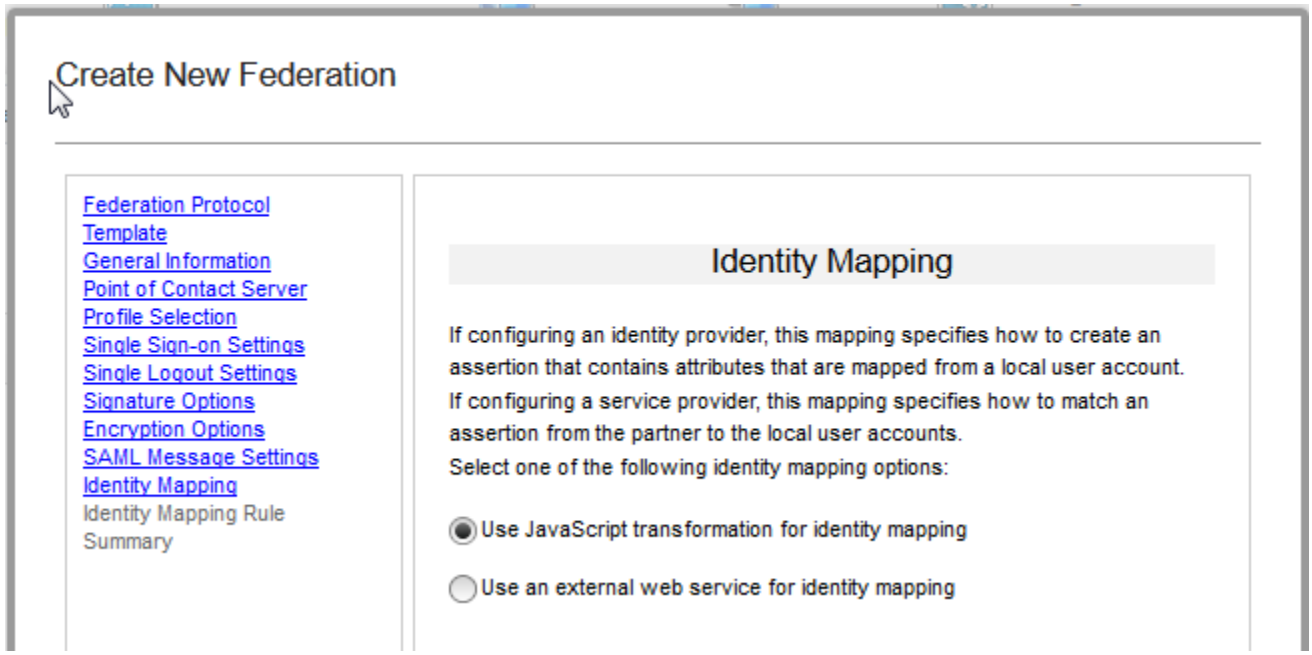
Create a new **OpenID Connect Relying Party** federation named **isamrp** as shown and click **Next**.



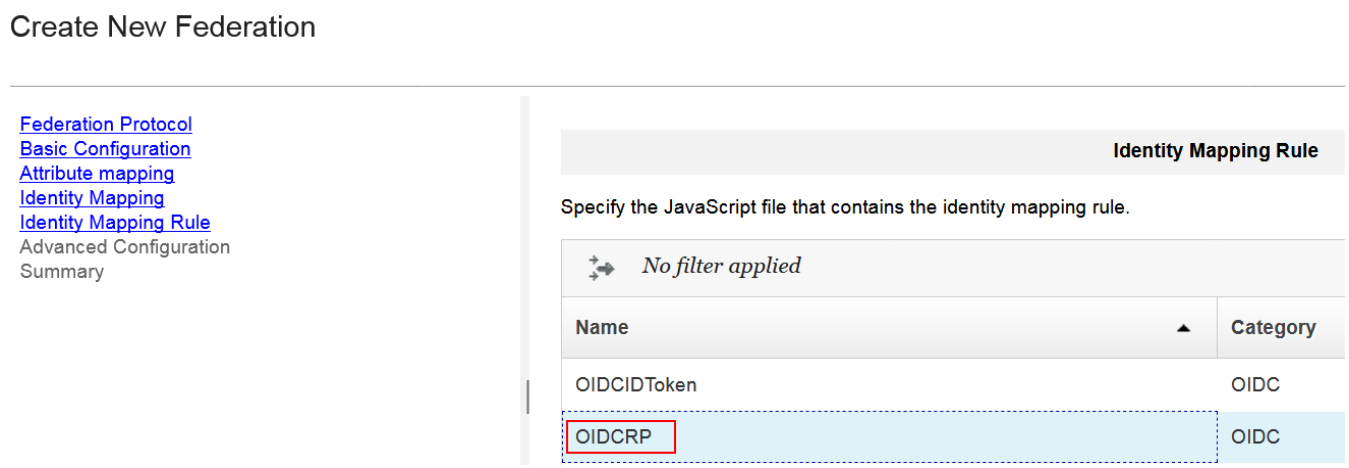
On the Point of Contact Server panel, enter **https://www.mysp.ibm.com/isam** and select all the Response Types checkbox **code, id_token** and **token** click **Next**.



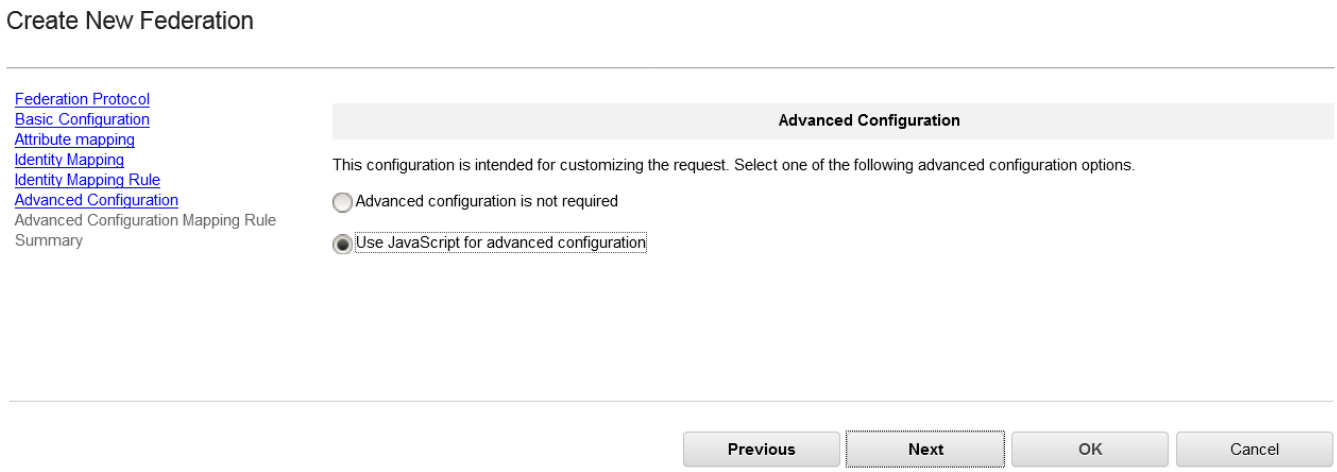
Skip the Attribute Mapping panel and click **Next**.



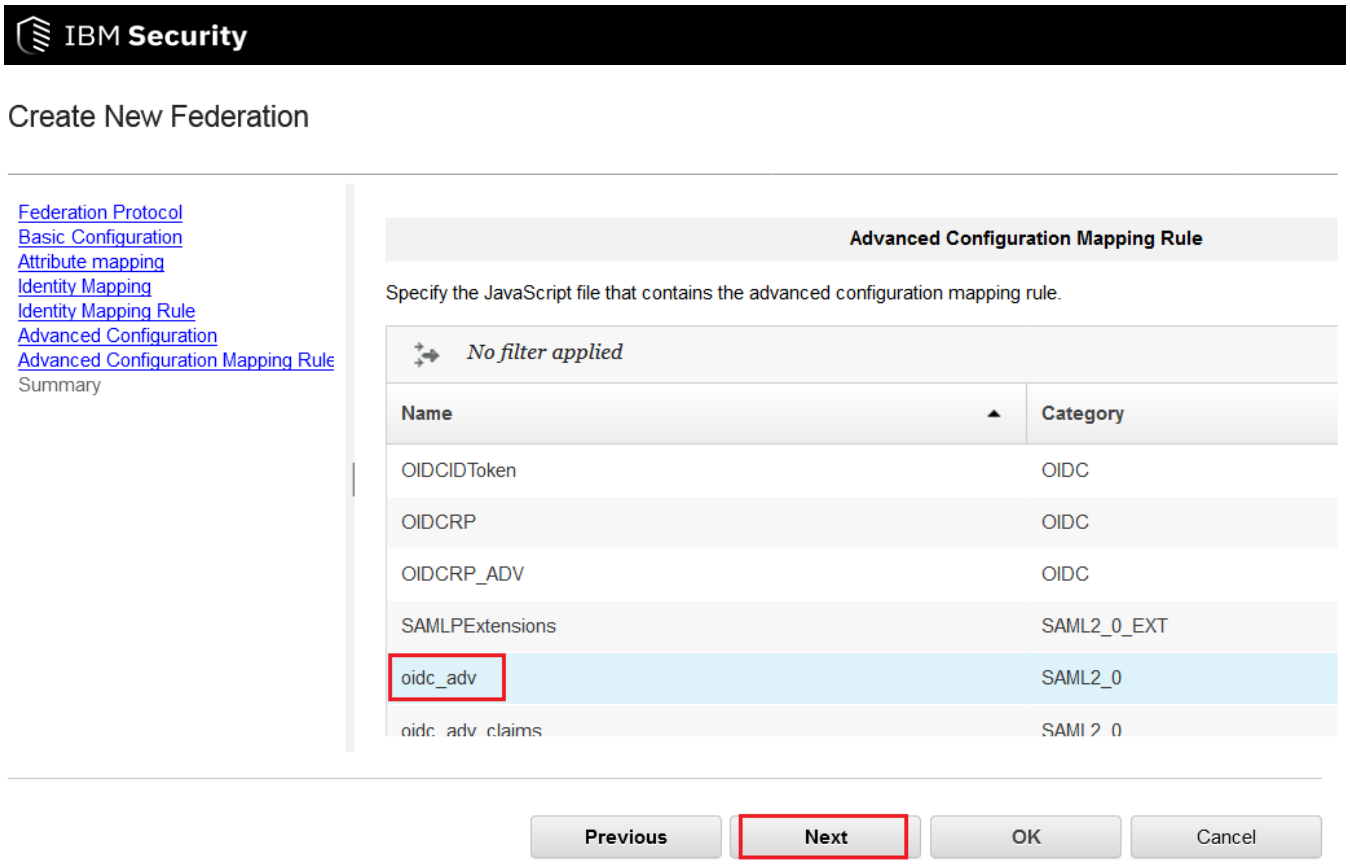
On the Identity Mapping panel, we will use the default of **Use Javascript transformation for identity mapping** so just click **Next**.



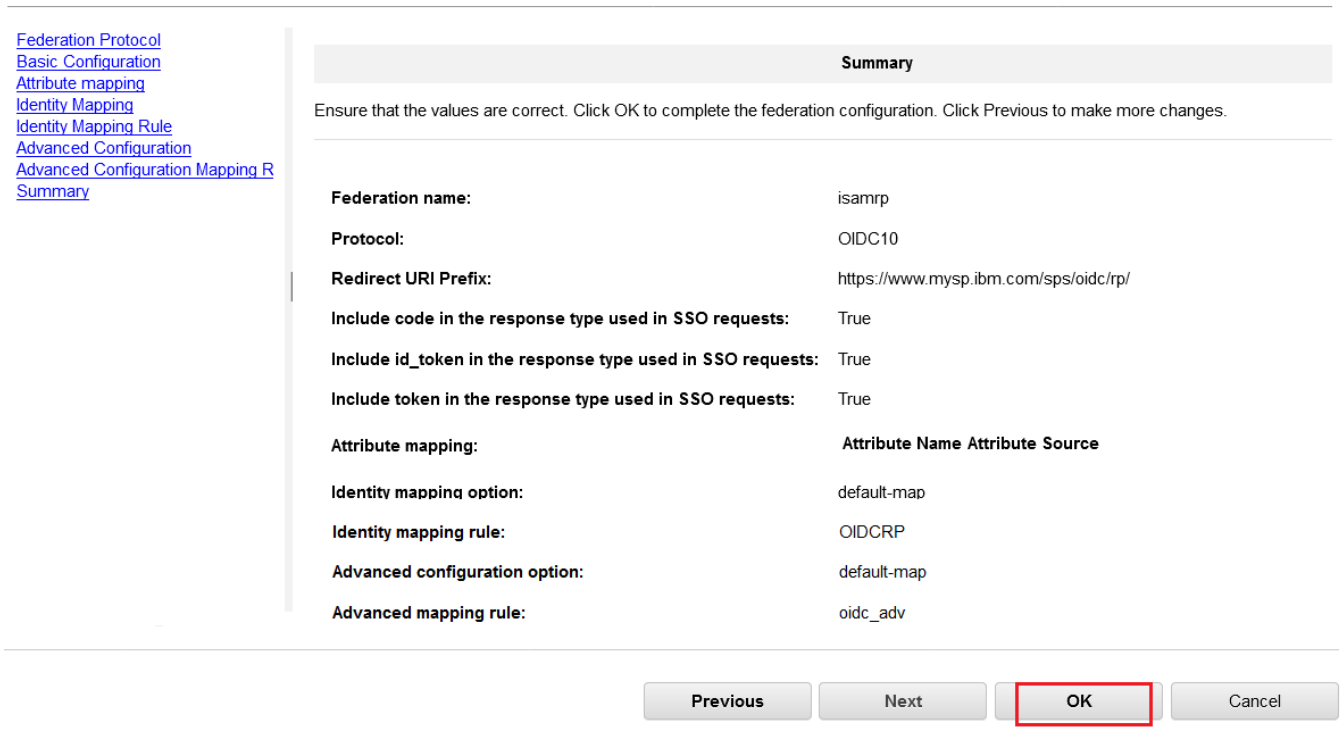
On the Identity Mapping Rule panel, select **OIDCRP** from the drop-down list and click **Next**.



On the Advanced Configuration panel, we will use the default of **Use Javascript transformation for advanced configuration** so just click **Next**.



On the Advanced Configuration Mapping Rule panel, select **oidc_adv** from the drop-down list and click **Next**.

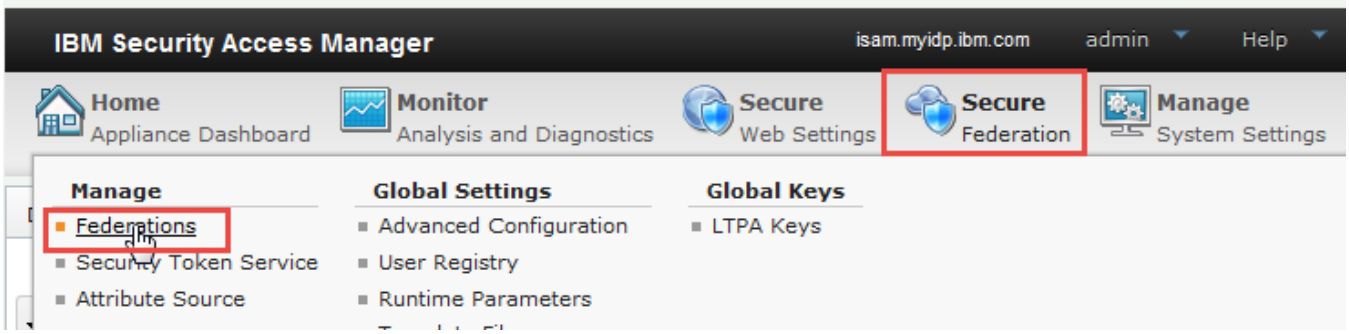


On the Summary panel, click **OK** to create the federation.

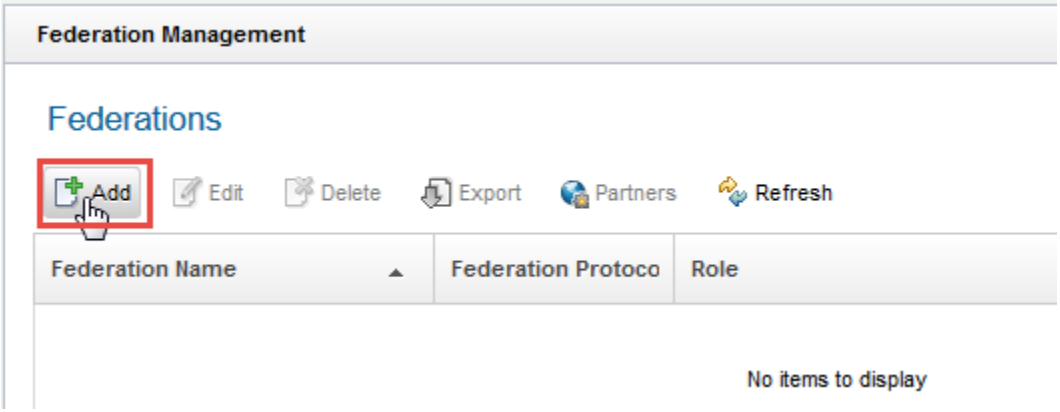
Follow on-screen instructions to **deploy** pending changes.

13.2.4 Configuring OpenID Connect Relying Party Federation for implicit flow

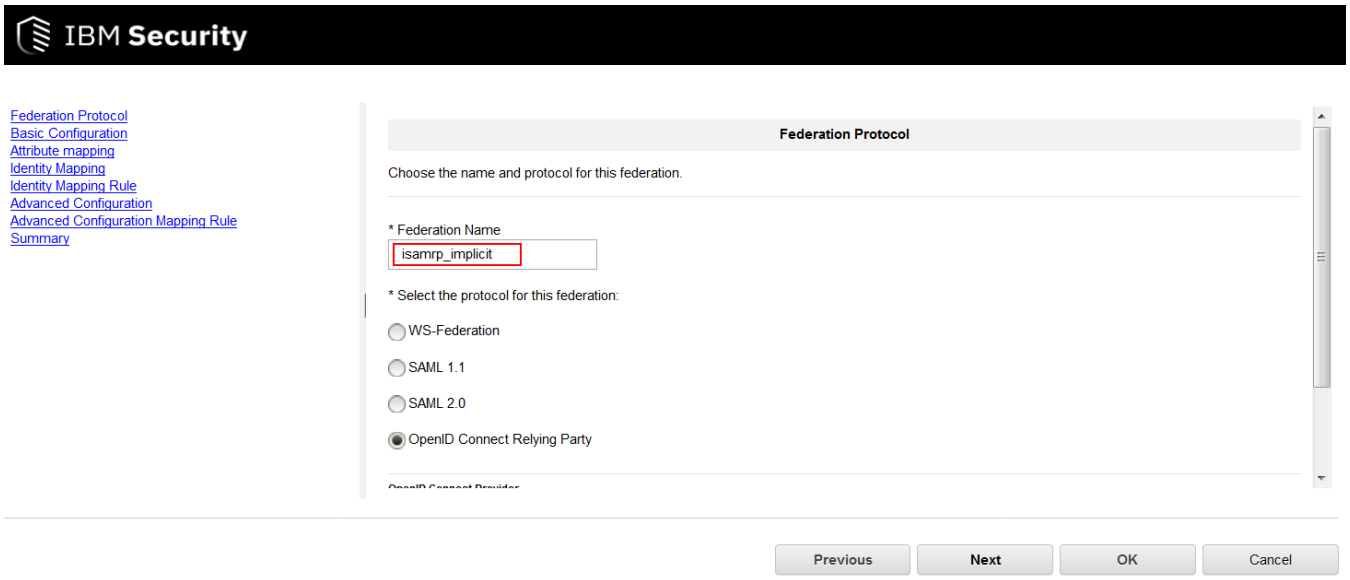
In this section we configure the relying party (SP) to a federation which supports implicit flow.



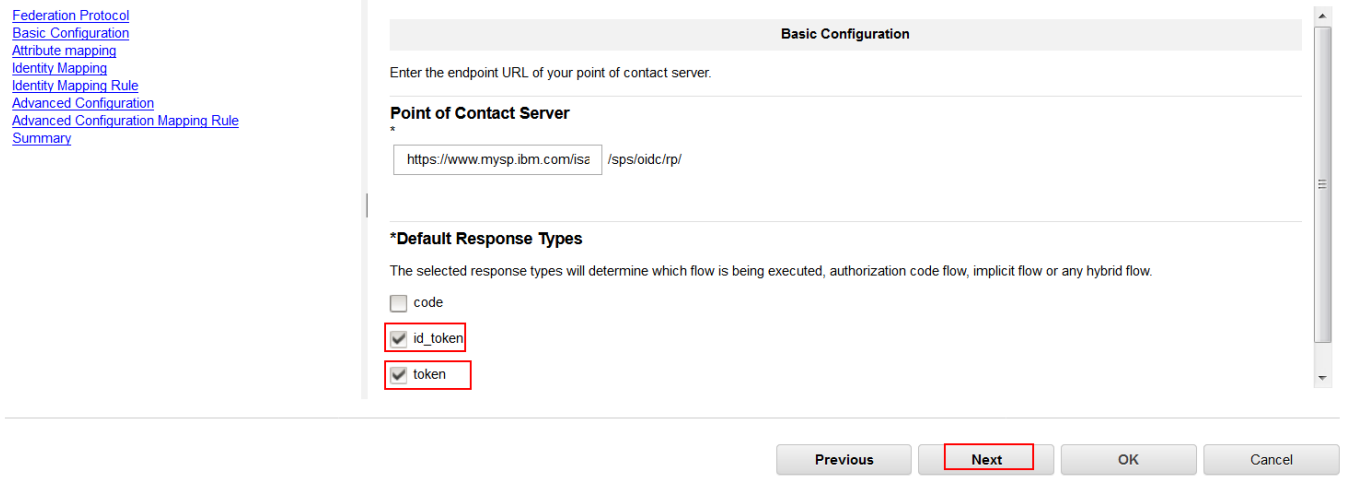
Using the administration console, navigate to **Secure Federation→Manage: Federations**.



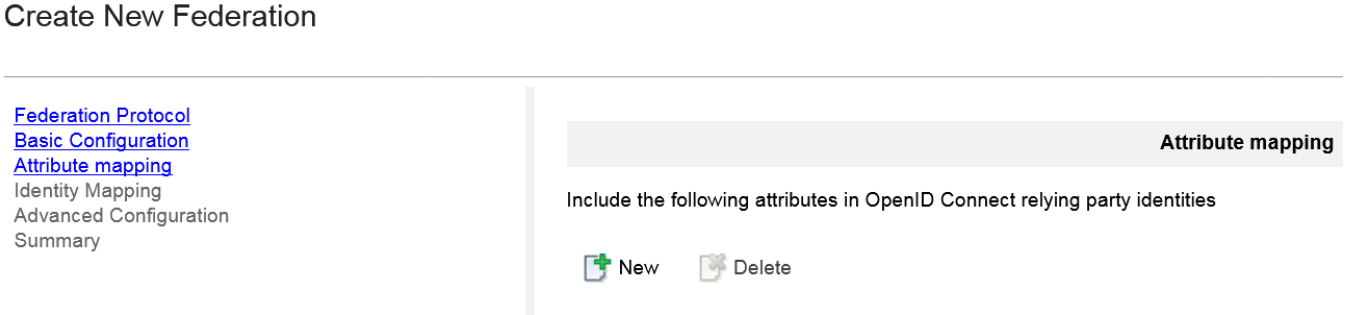
Click **Add** to create a new federation.



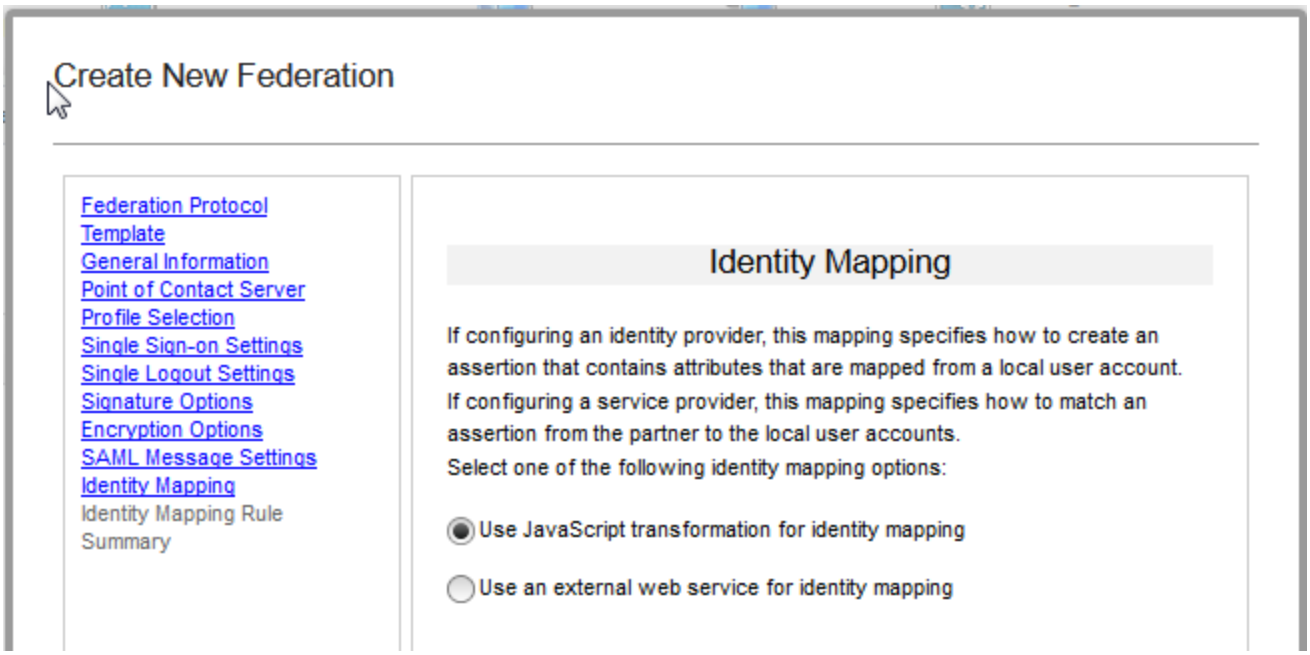
Create a new **OpenID Connect Relying Party** federation named **isamrp_implicit** as shown and click **Next**.



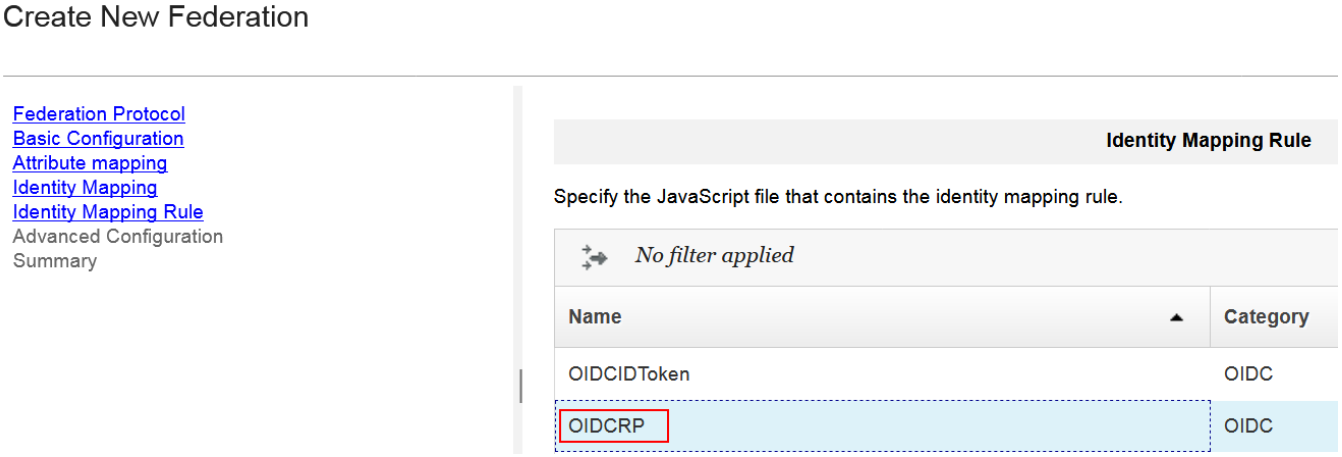
On the Point of Contact Server panel, enter **https://www.mysp.ibm.com/isam** and select all the Response Types checkbox **id_token** and **token** click **Next**.



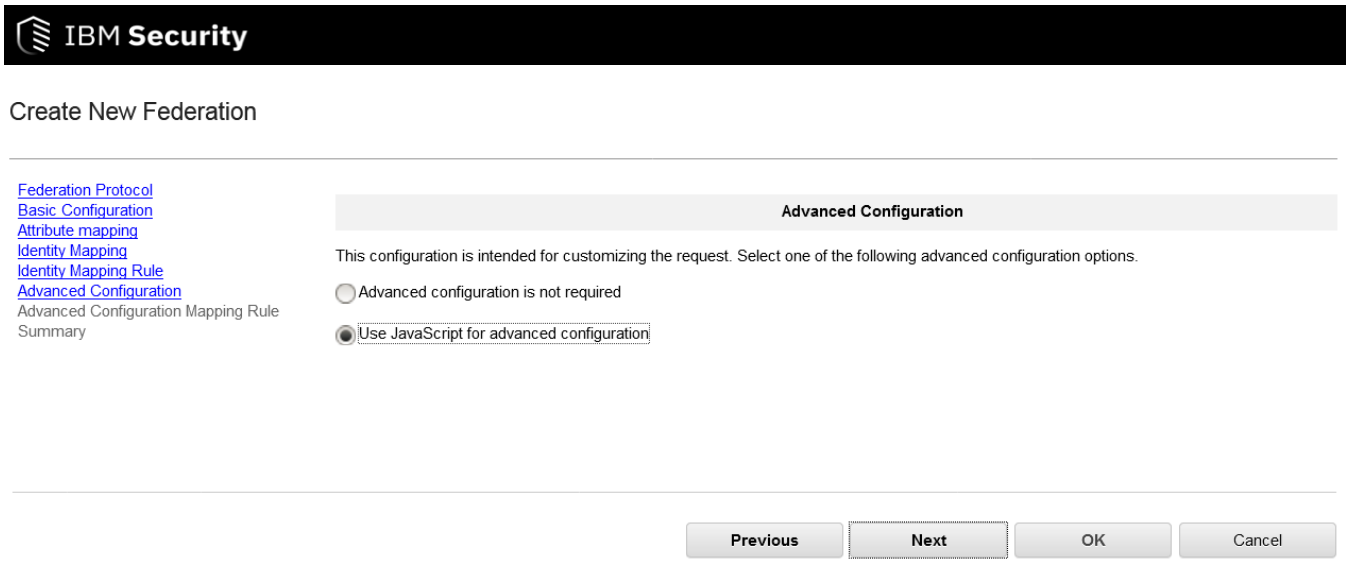
Skip the Attribute Mapping panel and click **Next**.



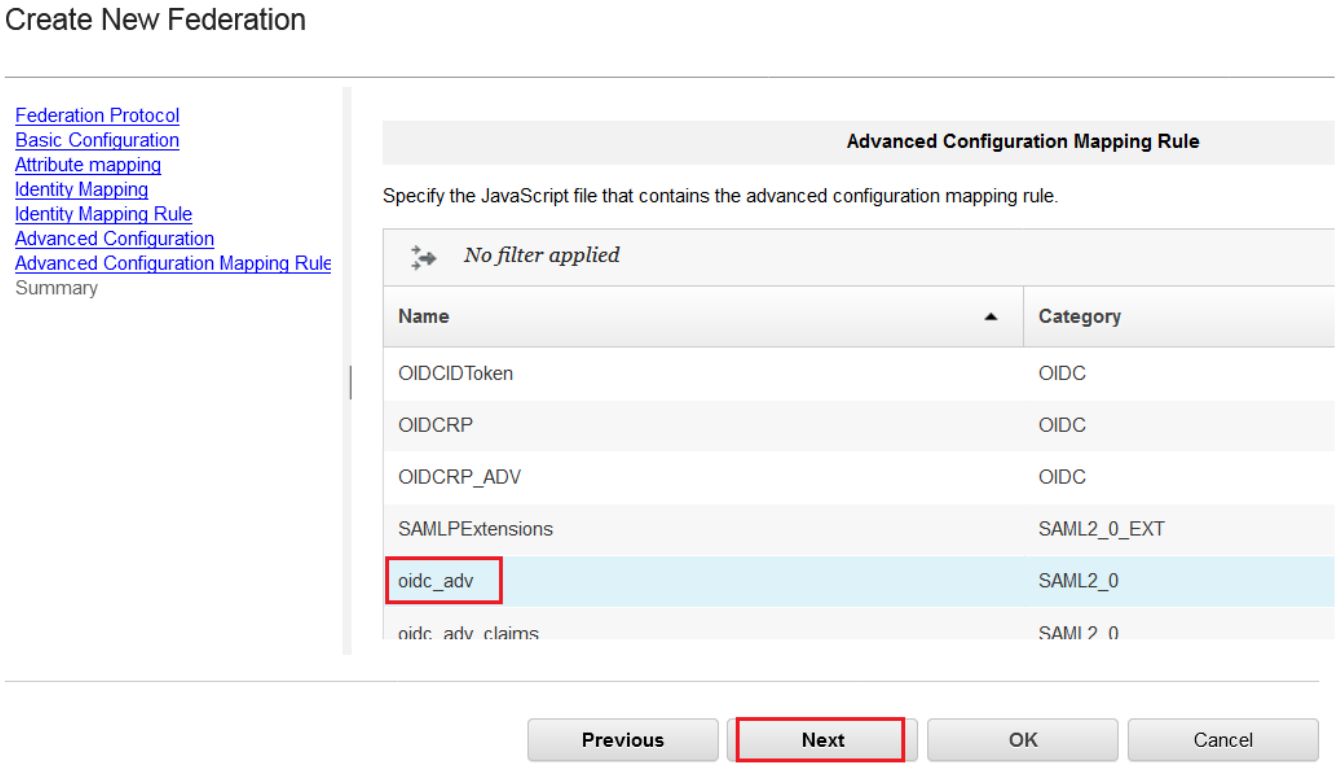
On the Identity Mapping panel, we will use the default of **Use Javascript transformation for identity mapping** so just click **Next**.



On the Identity Mapping Rule panel, select **OIDCRP** from the drop-down list and click **Next**.



On the Advanced Configuration panel, we will use the default of **Use Javascript transformation for advanced configuration** so just click **Next**.



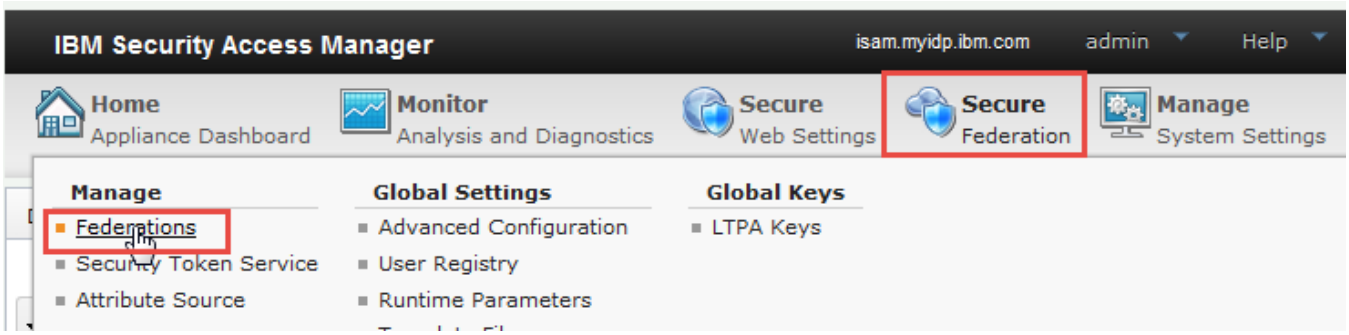
On the Advanced Configuration Mapping Rule panel, select **oidc_adv** from the drop-down list and click **Next**.

On the Summary panel, click **OK** to create the federation.

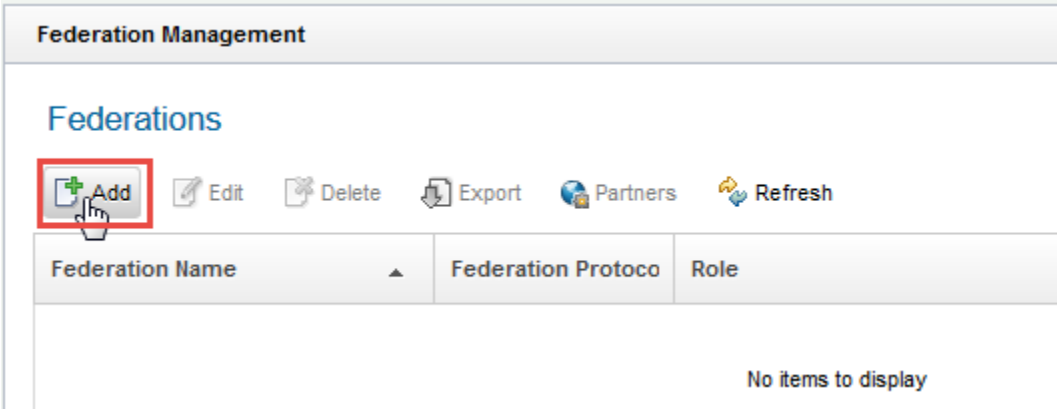
Follow on-screen instructions to **deploy** pending changes.

13.2.5 Configuring OpenID Connect Relying Party Federation for authorization_code flow

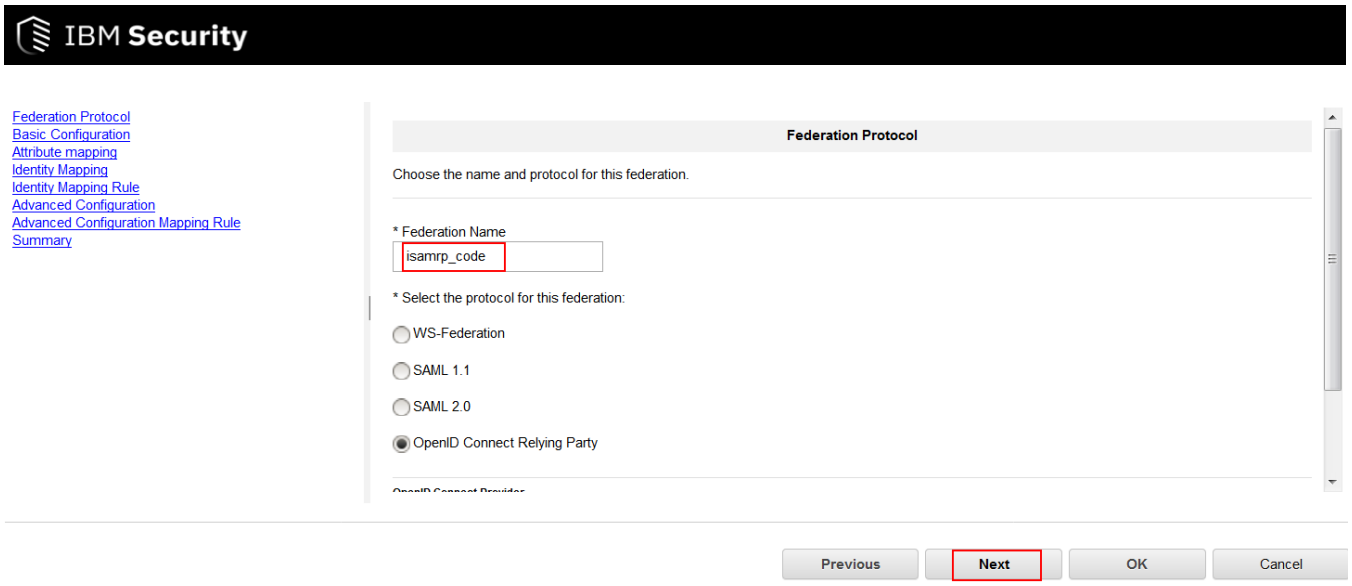
In this section we configure the relying party (SP) to a federation which supports authorization code flow.



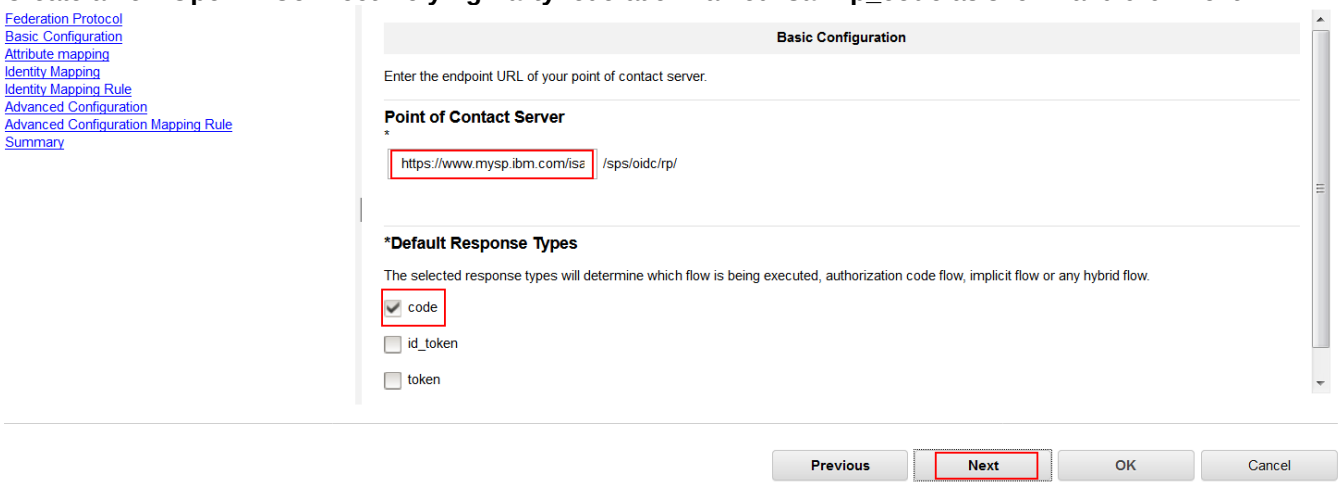
Using the administration console, navigate to **Secure Federation→Manage: Federations**.



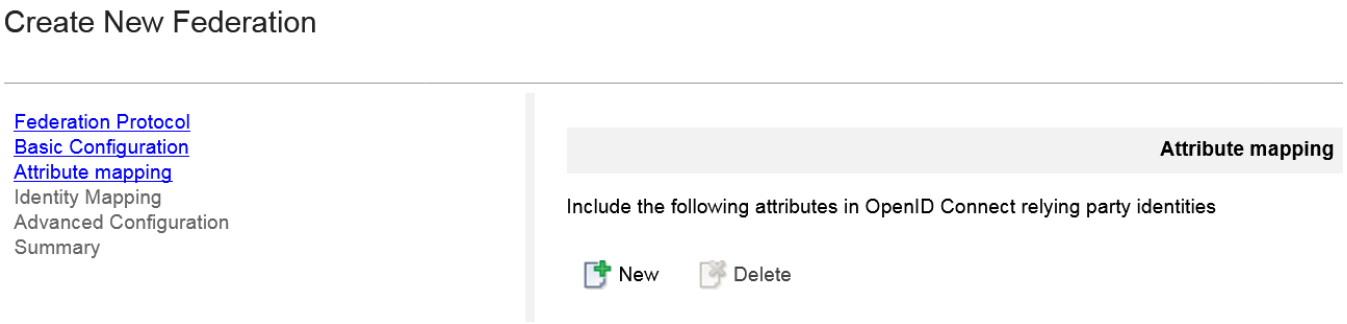
Click **Add** to create a new federation.



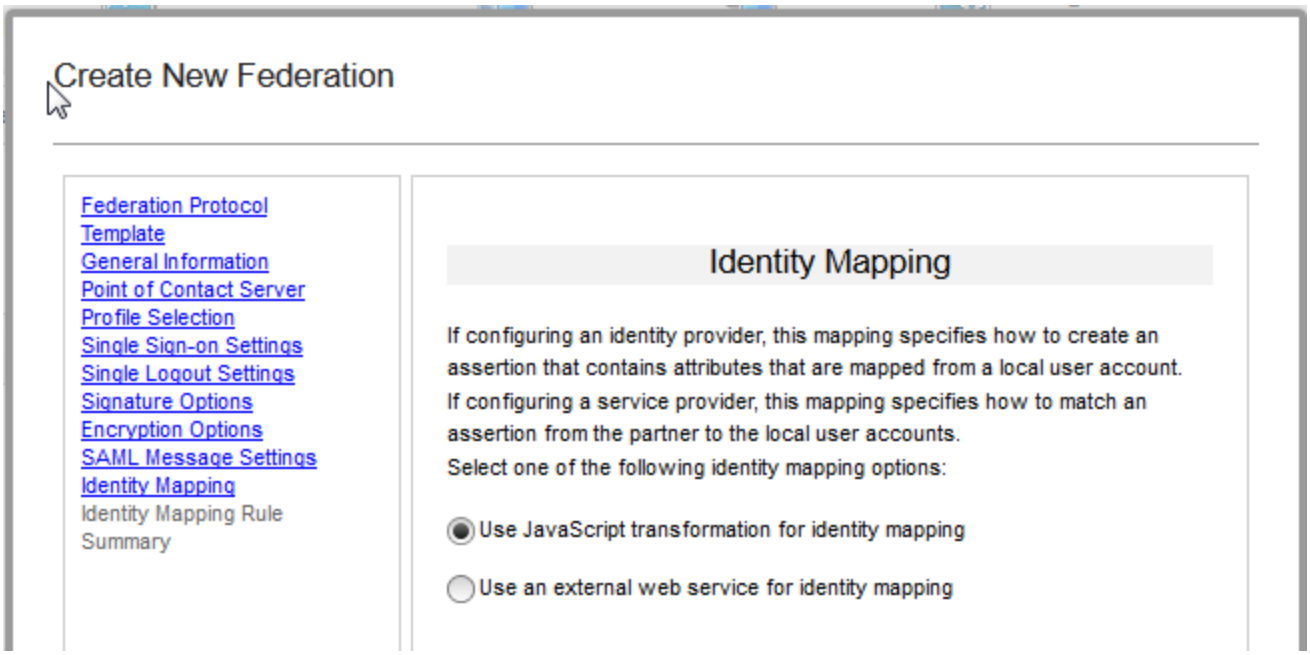
Create a new **OpenID Connect Relying Party** federation named **isamrp_code** as shown and click **Next**.



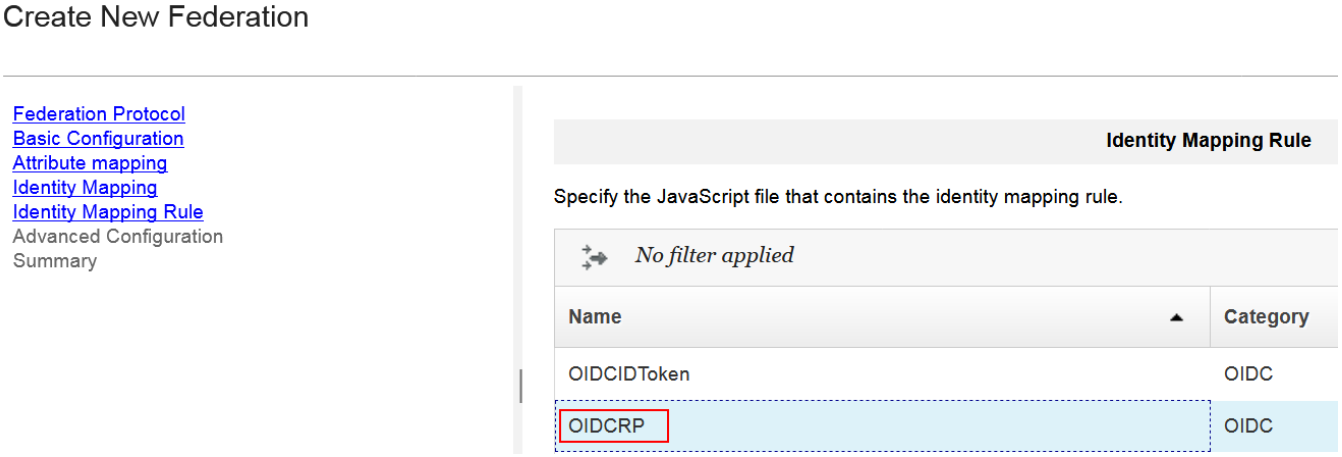
On the Point of Contact Server panel, enter **https://www.mysp.ibm.com/isam** and select all the Response Types checkbox **code** click **Next**.



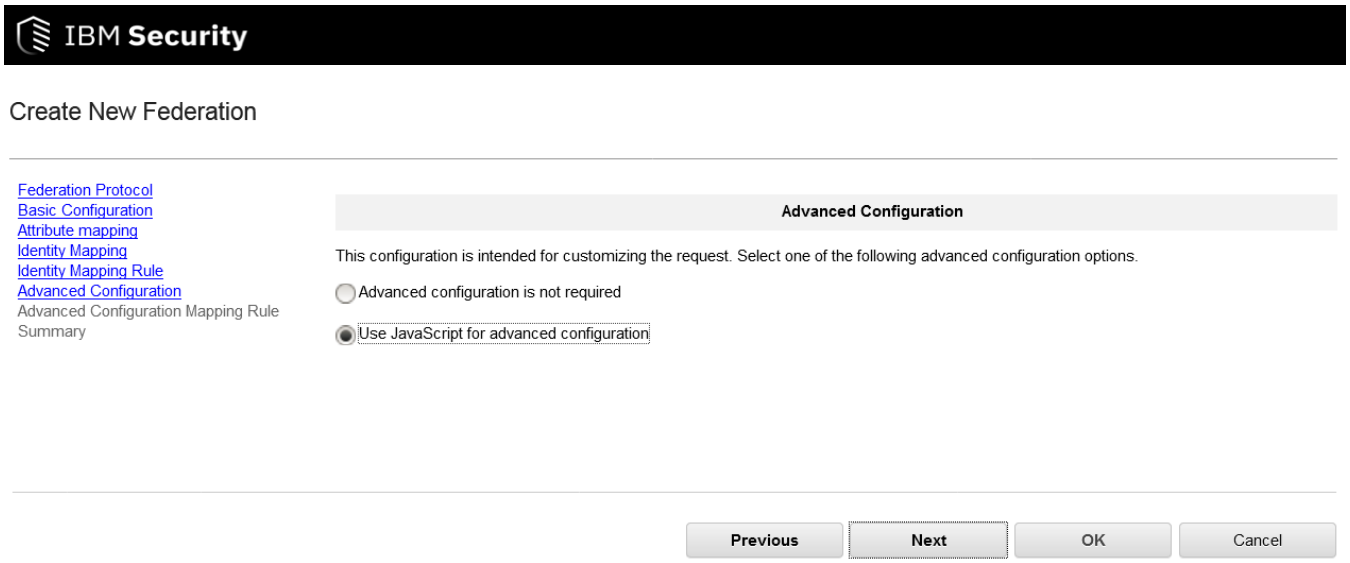
Skip the Attribute Mapping panel and click **Next**.



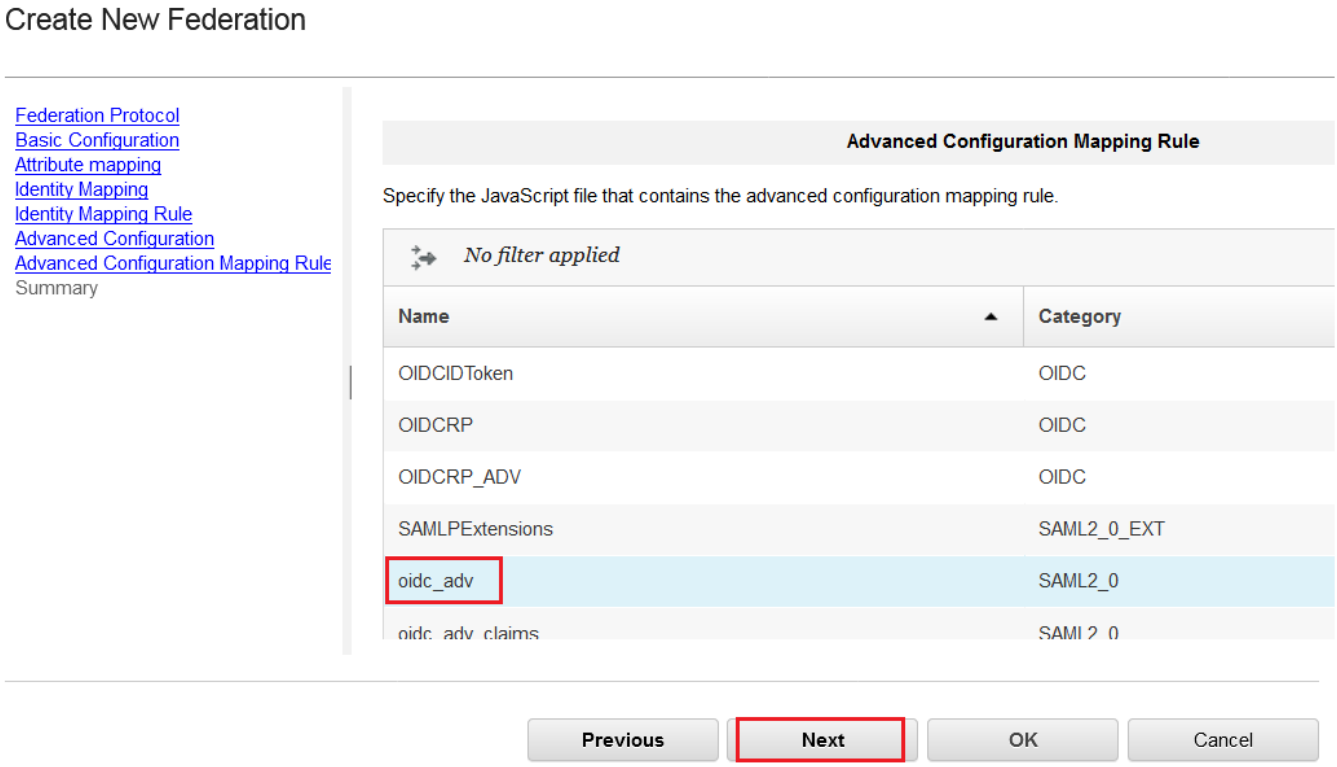
On the Identity Mapping panel, we will use the default of **Use Javascript transformation for identity mapping** so just click **Next**.



On the Identity Mapping Rule panel, select **OIDCRP** from the drop-down list and click **Next**.



On the Advanced Configuration panel, we will use the default of **Use Javascript transformation for advanced configuration** so just click **Next**.



On the Advanced Configuration Mapping Rule panel, select **oidc_adv** from the drop-down list and click **Next**.

On the Summary panel, click **OK** to create the federation.

Follow on-screen instructions to **deploy** pending changes.

SCRIPT-END:
The script should display the following:

INFO:FederationManager:Configuring the easuser password
INFO:FederationManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the OIDC RP Federation
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the OIDC RP Federation
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the OIDC RP

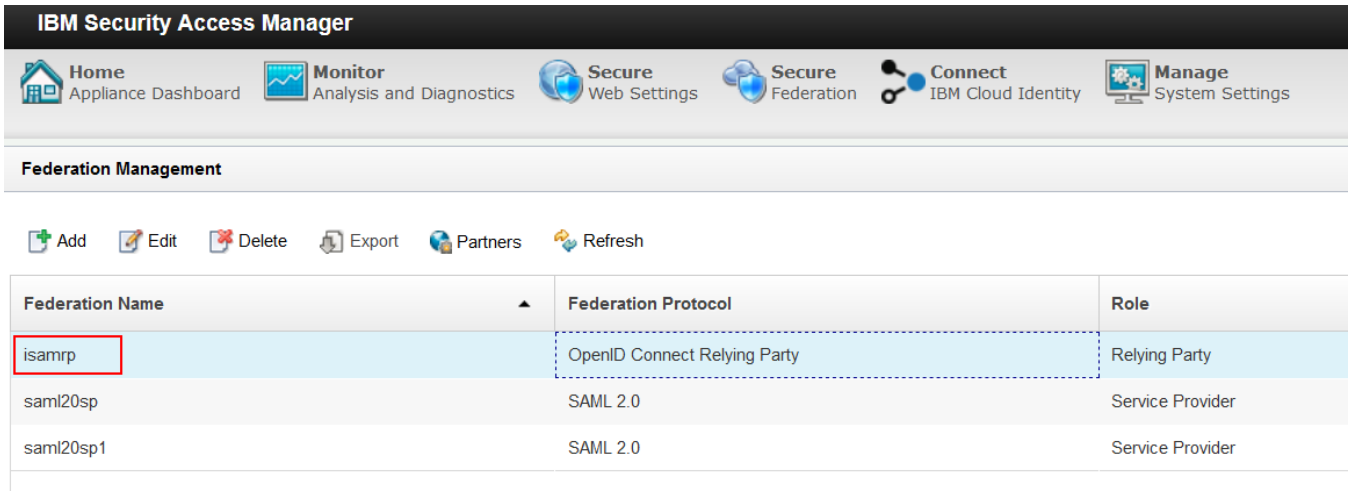
13.2.6 Configuring OpenID Connect Relying Party Partner for hybrid federation

In this section we configure a partner for the federation which supports hybrid flow.

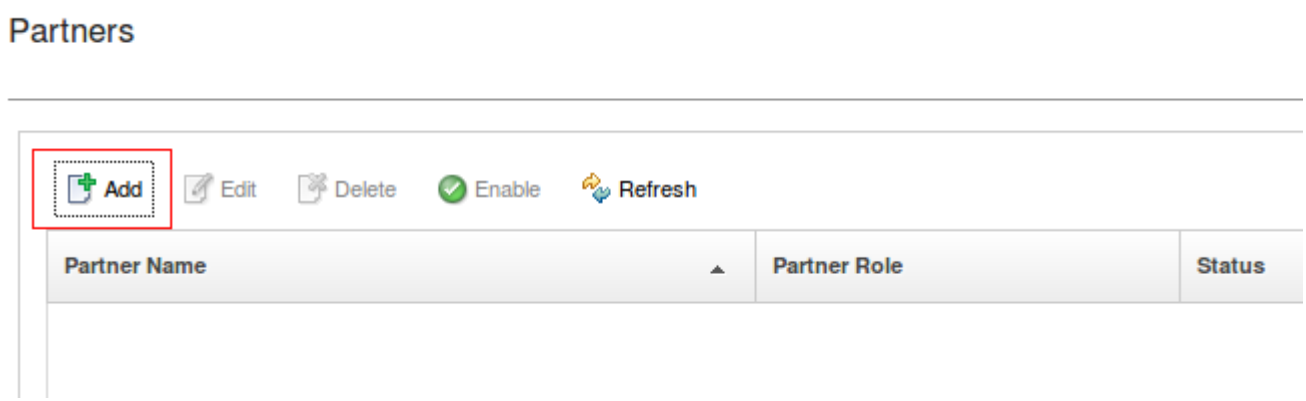
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

For the SP, run this script: **OIDCRPConfig.py -configure RPPartner**

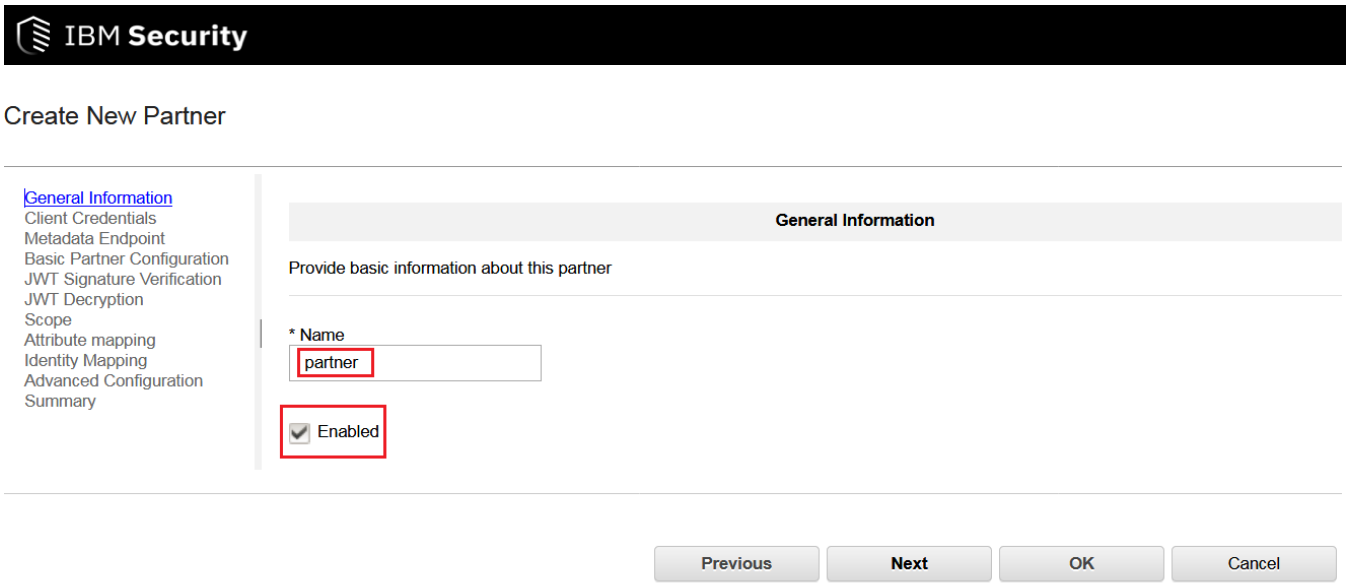
Under **Secure Federation** menu, click **on Manage: Federations**.



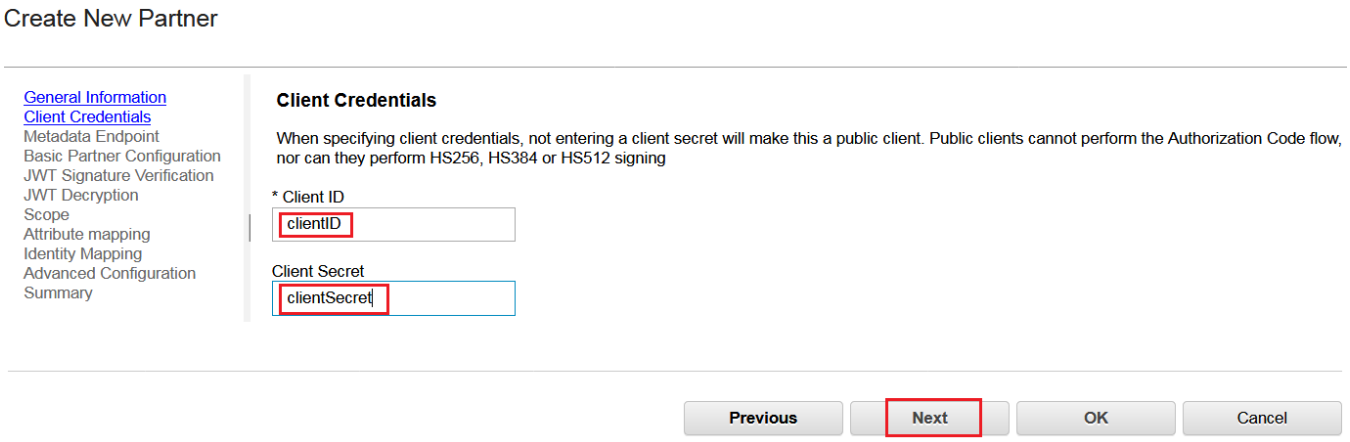
Select the **isamrp** federation and click **Partners**.



Click on **Add** to configure OIDC RP Partner.

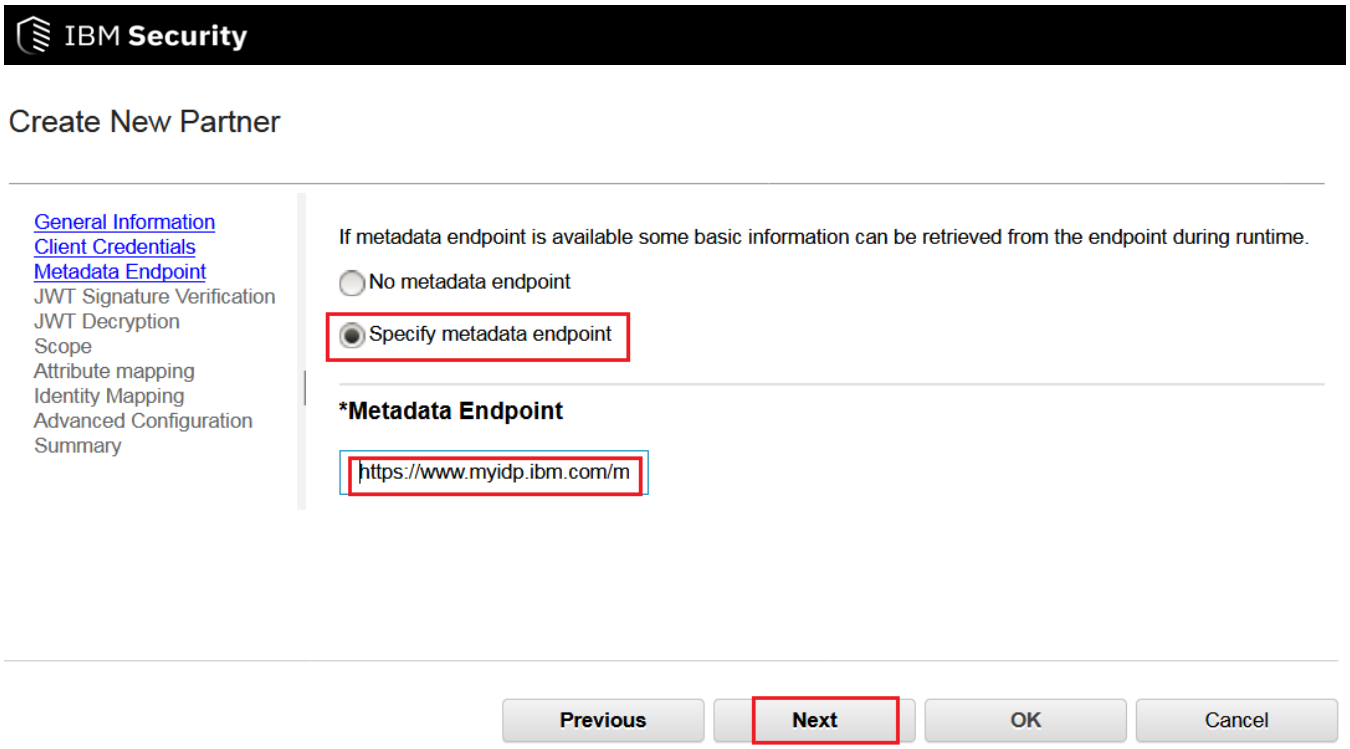


Enter **partner** as the name of the RP Partner and select the **Enabled** checkbox.



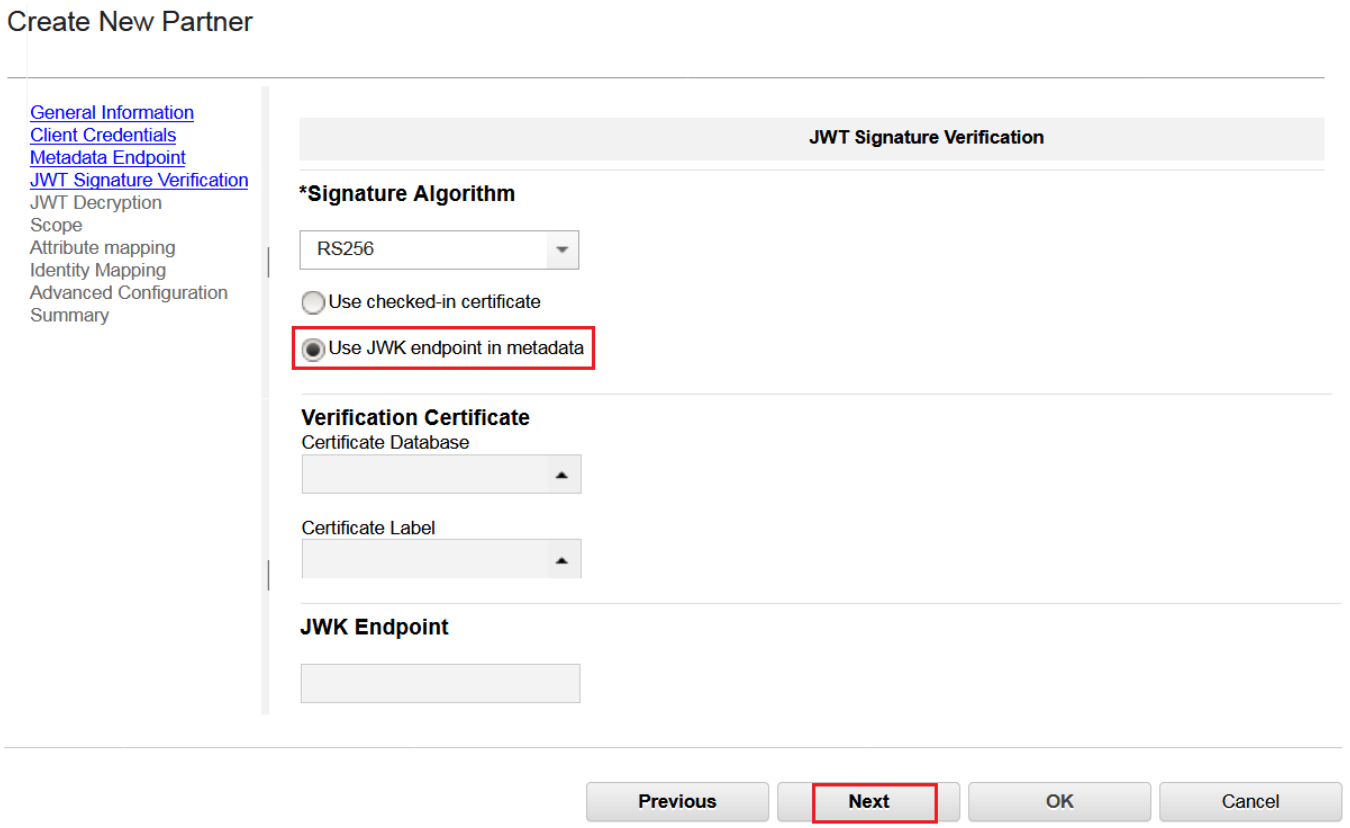
Enter **clientID** as Client ID and **clientSecret** as Client Secret.

Click on **Next**.



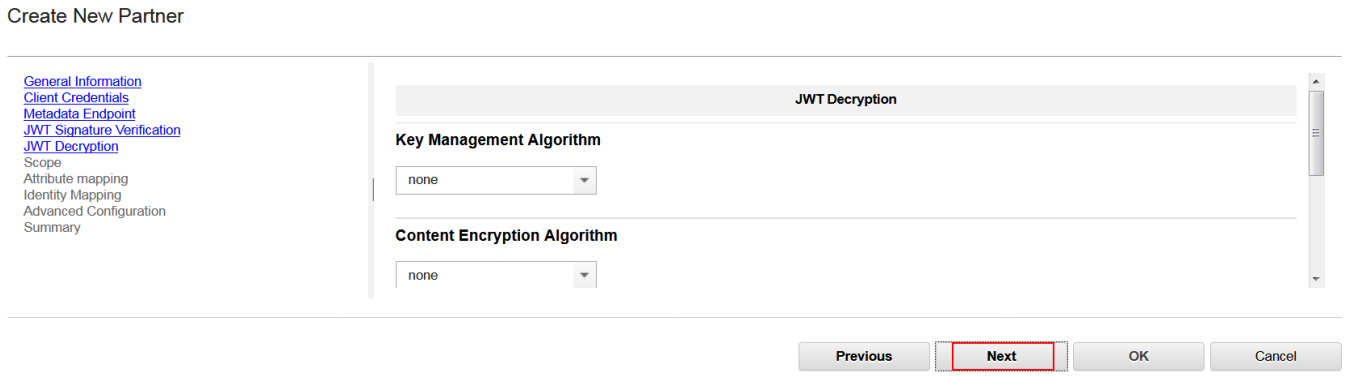
On the Metadata Endpoint panel, select **Specify metadata endpoint** radio button. Enter https://www.myidp.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCDefinition as the Metadata Endpoint.

Click on **Next.**

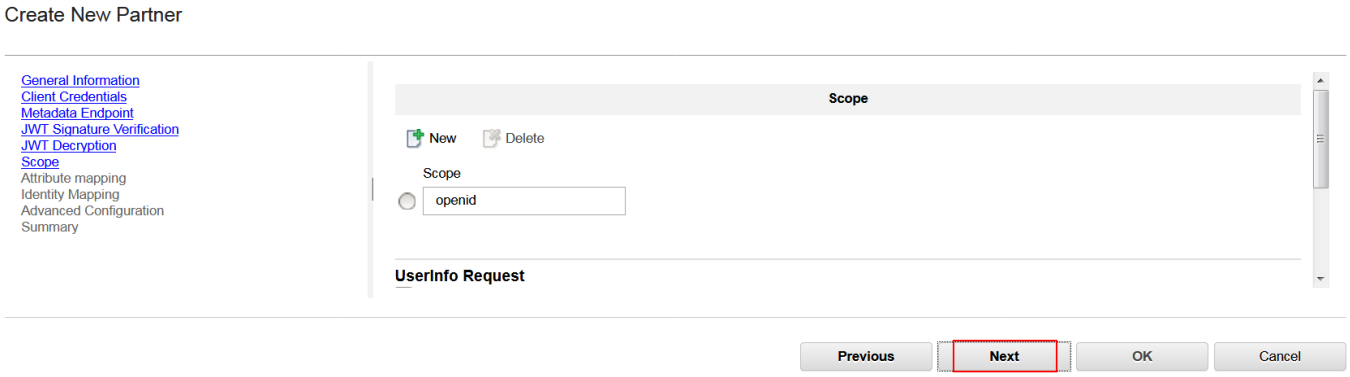


On the **JWT Signature Validation** panel, select **Use JWK endpoint in metadata.**

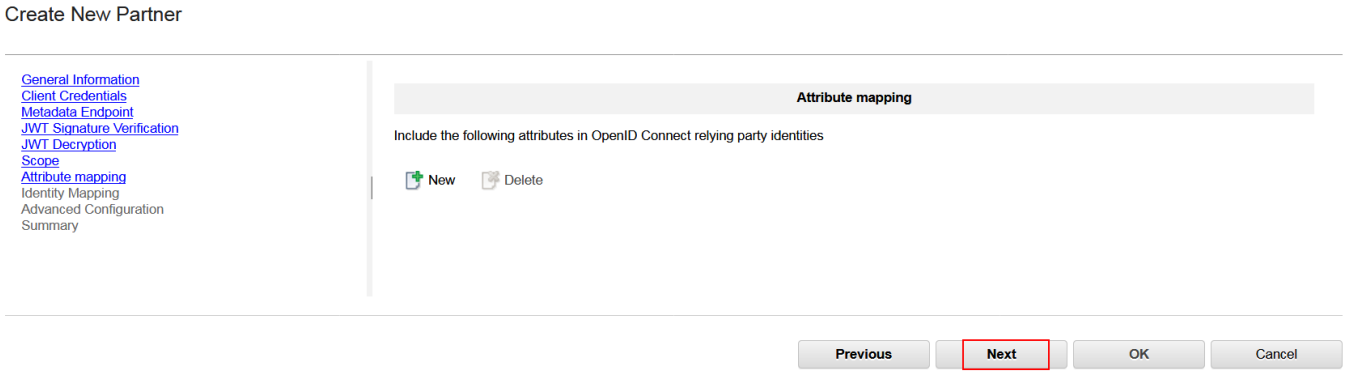
Click on **Next.**



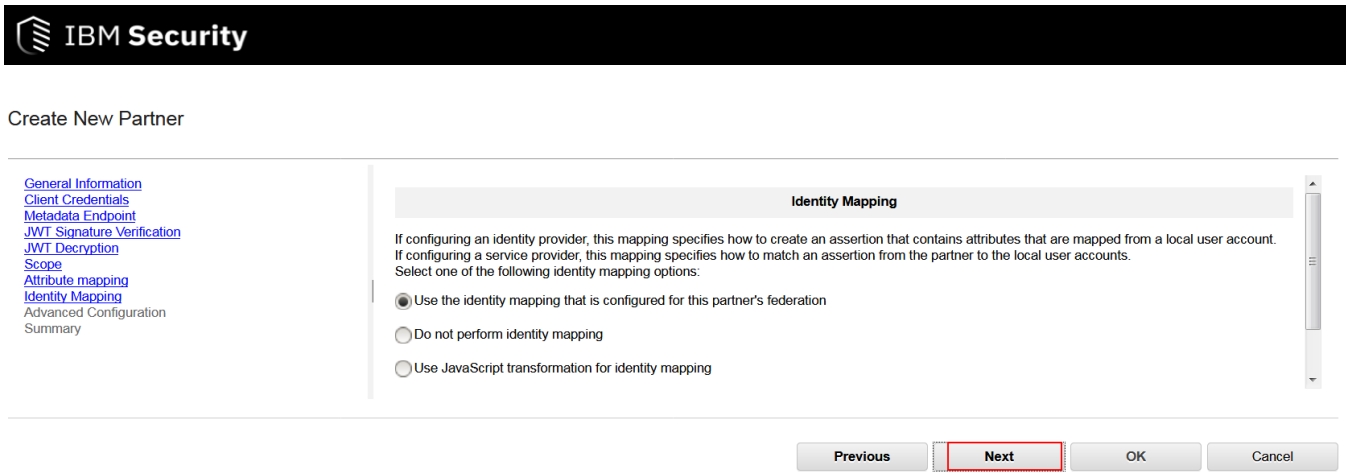
We don't need to change JWT Decryption so click **Next**.



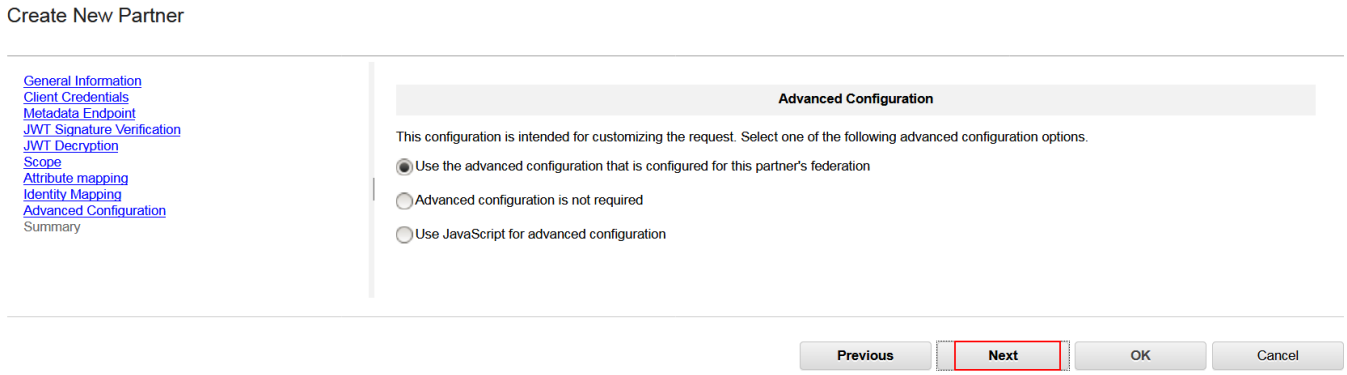
We don't need to change Scopes so click **Next**.



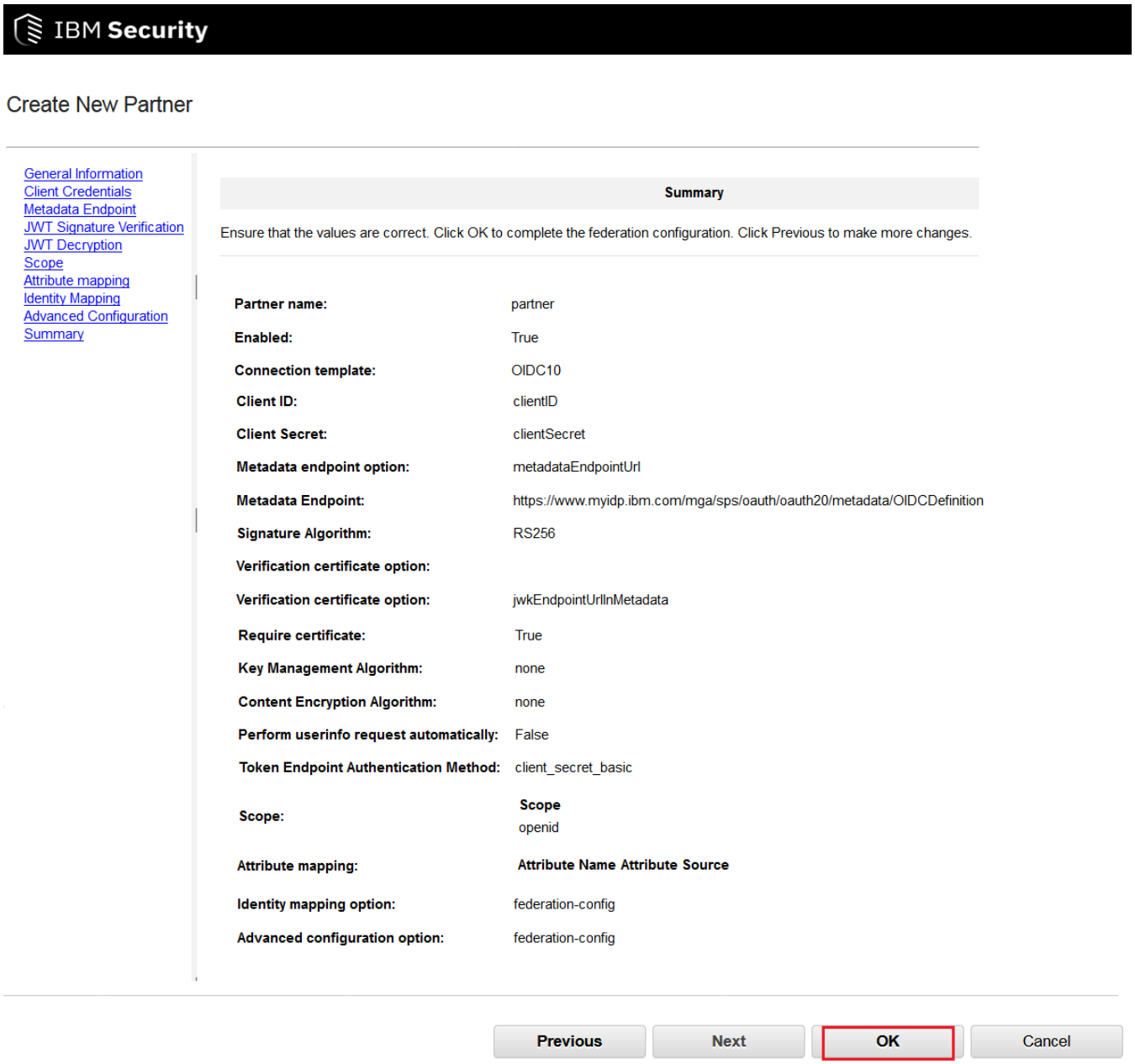
We don't need to change Attribute Mapping so click **Next**.



We don't need to change Identity Mapping so click **Next**.



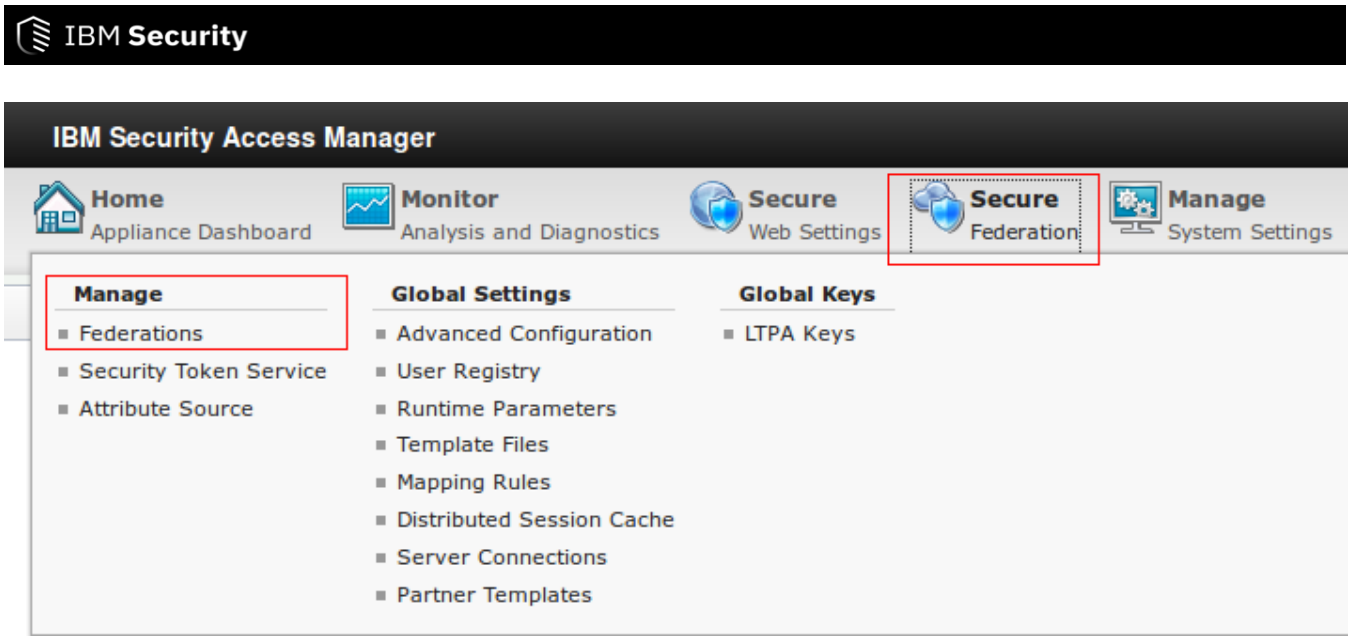
We don't need to change Advanced Configuration so click **Next**.



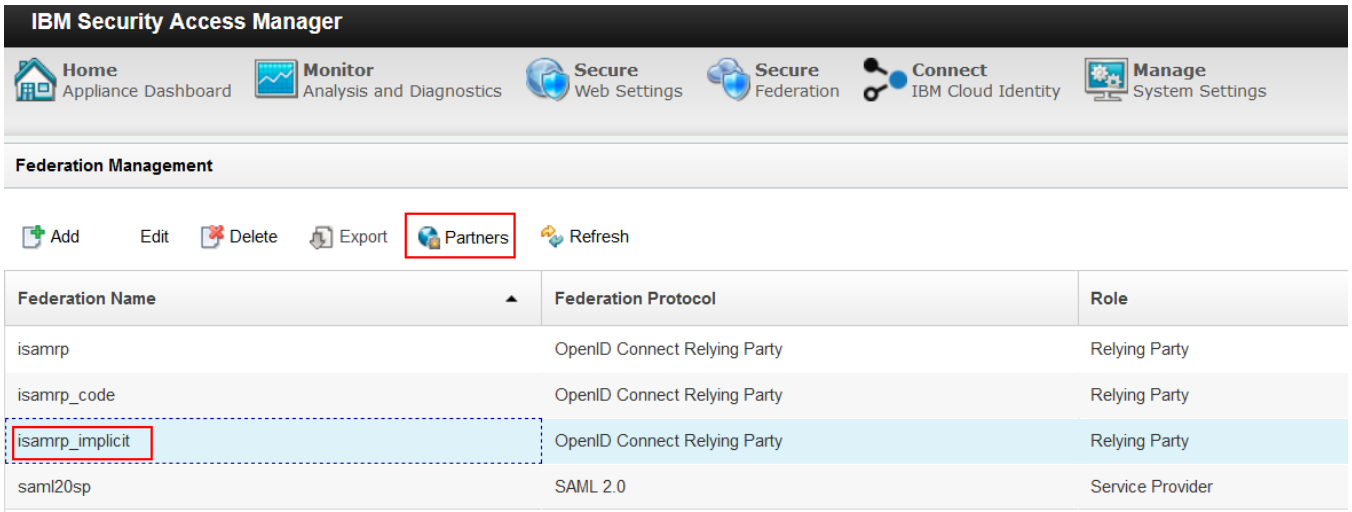
Click OK on the Summary panel.
Follow on-screen instructions to **deploy** pending changes.

13.2.7 Configuring OpenID Connect Relying Party Partner for implicit federation

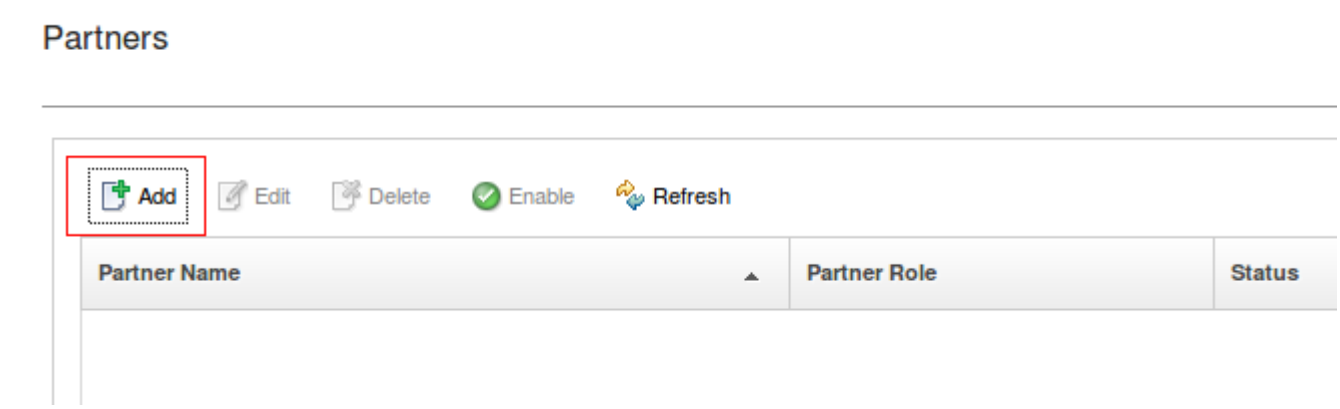
In this section we configure a partner for the federation which supports implicit flow.



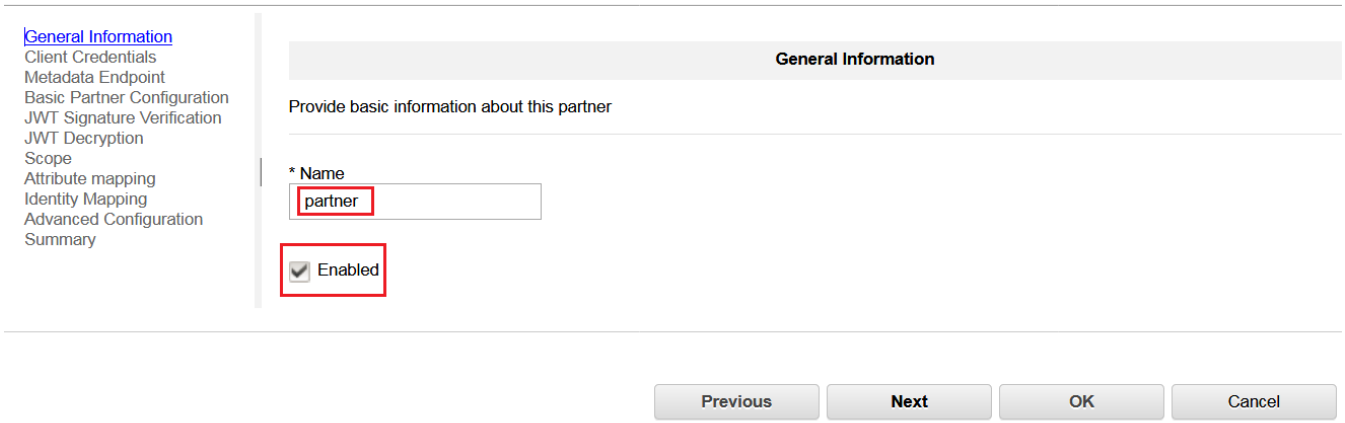
Under **Secure Federation** menu, click **on Manage: Federations**.



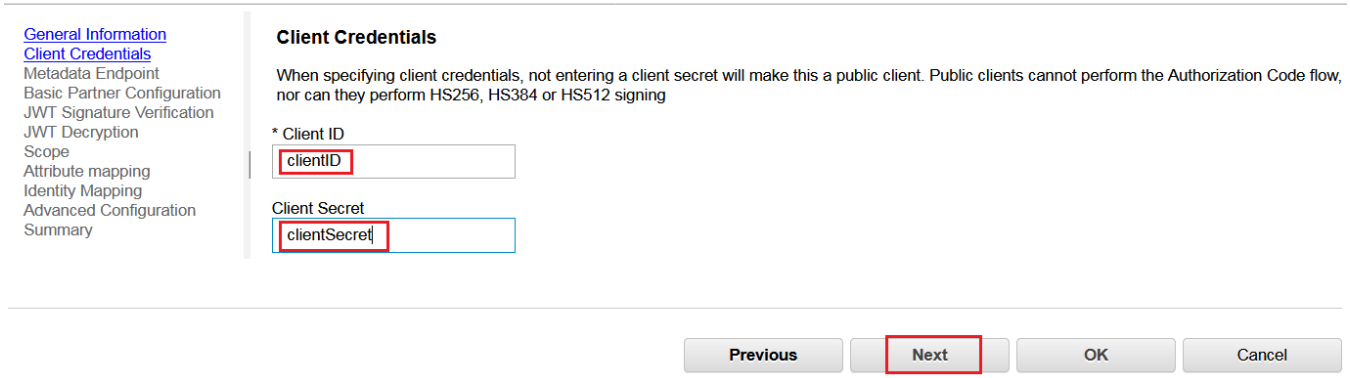
Select the **isamrp_implicit** federation and click **Partners**.



Click on **Add** to configure OIDC RP Partner.

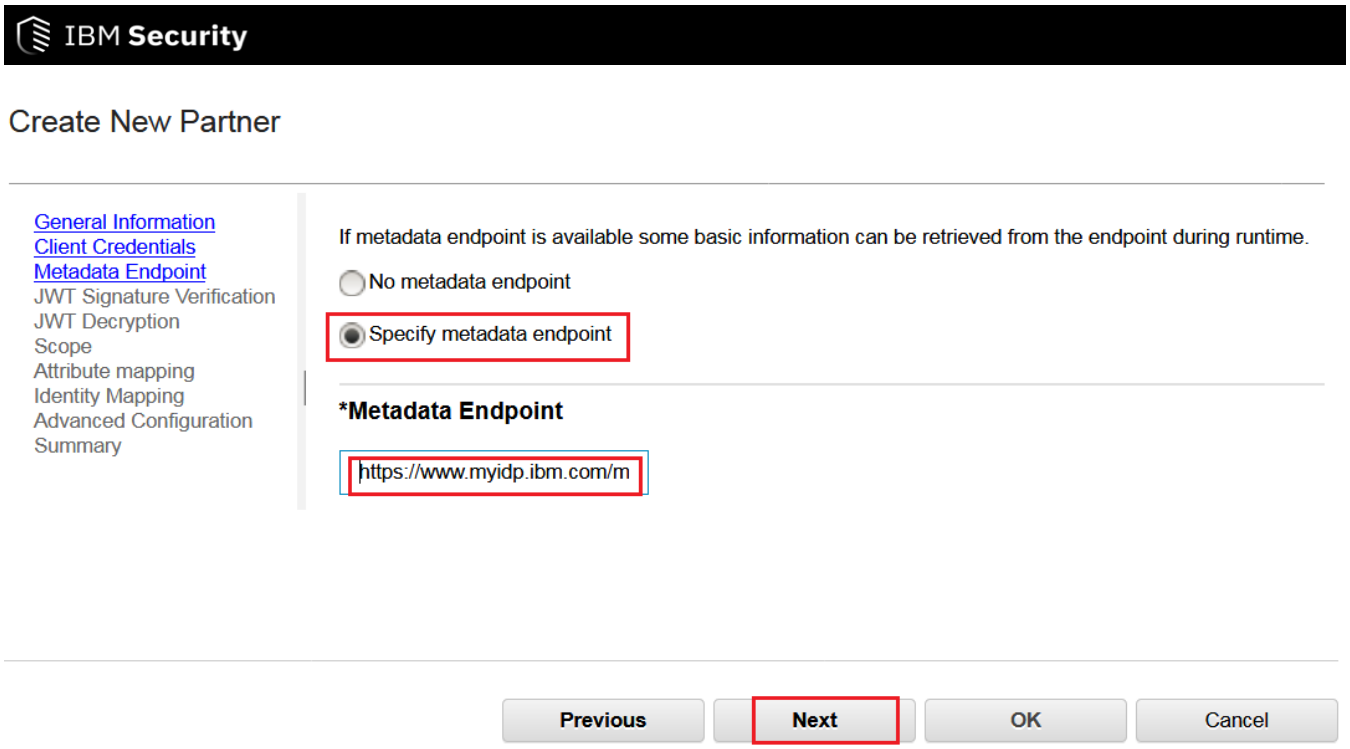


Enter **partner** as the name of the RP Partner and select the **Enabled** checkbox.



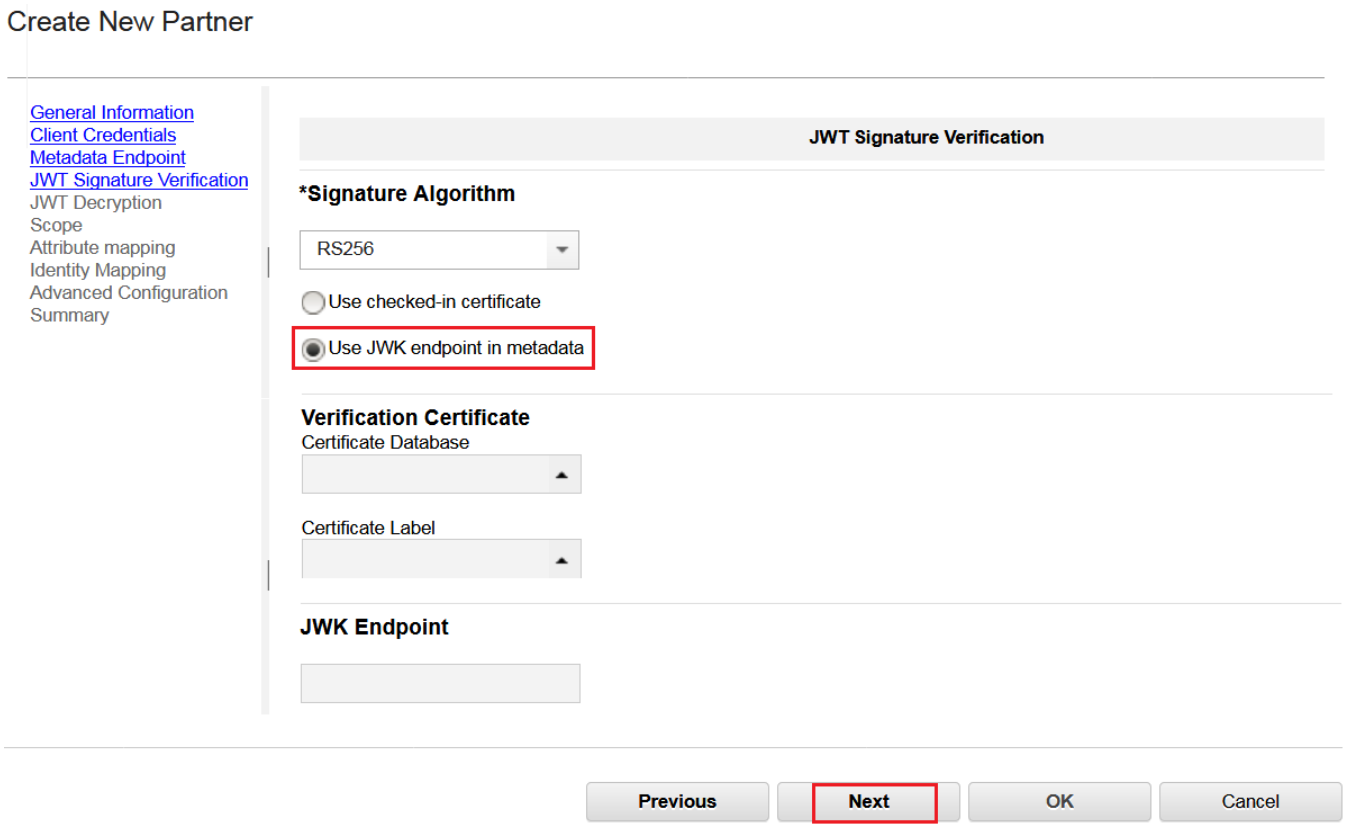
Enter **clientID** as Client ID and **clientSecret** as Client Secret.

Click on **Next**.



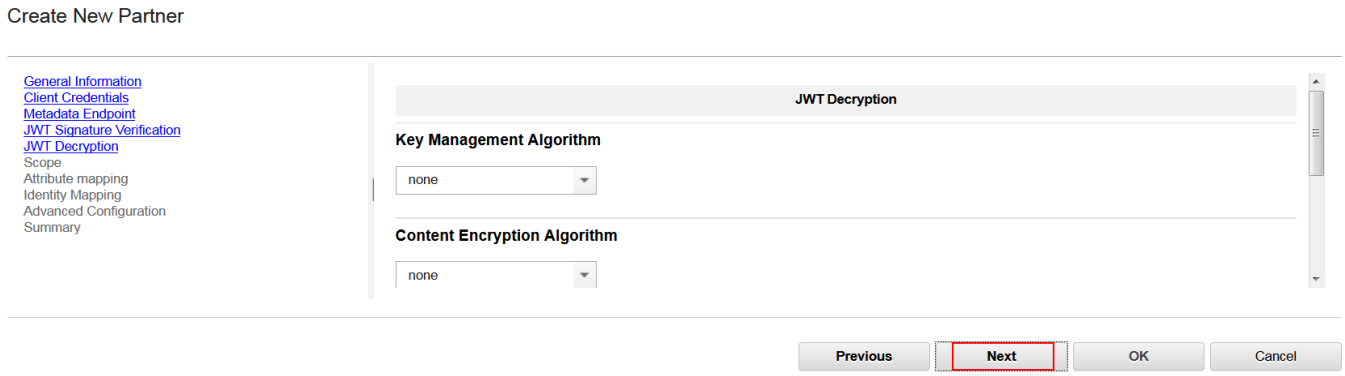
On the Metadata Endpoint panel, select **Specify metadata endpoint** radio button. Enter https://www.myidp.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCDefinition as the Metadata Endpoint.

Click on **Next.**

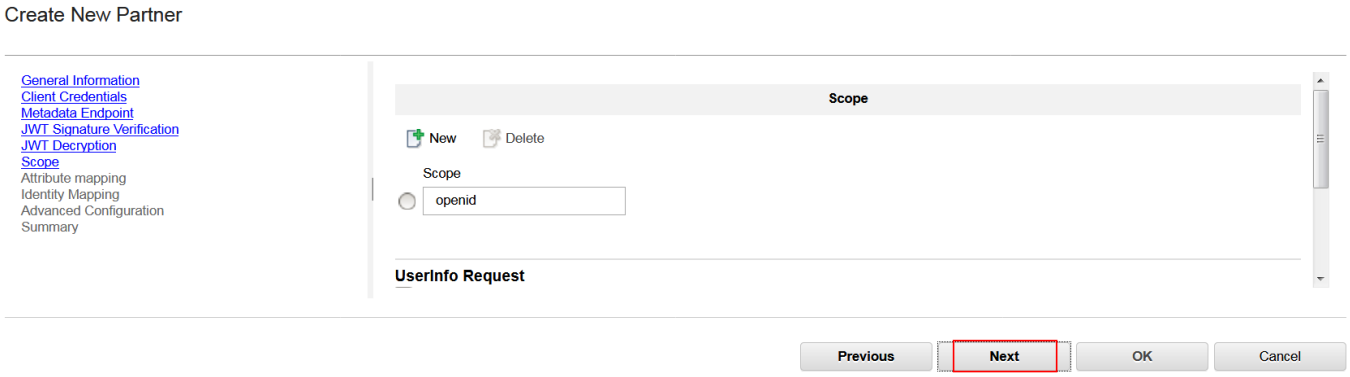


On the **JWT Signature Validation** panel, select **Use JWK endpoint in metadata.**

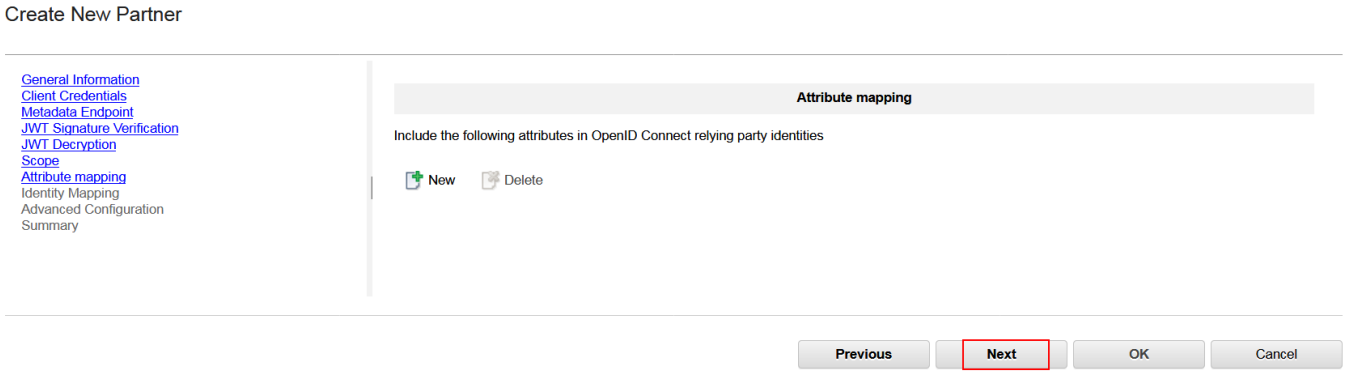
Click on **Next.**



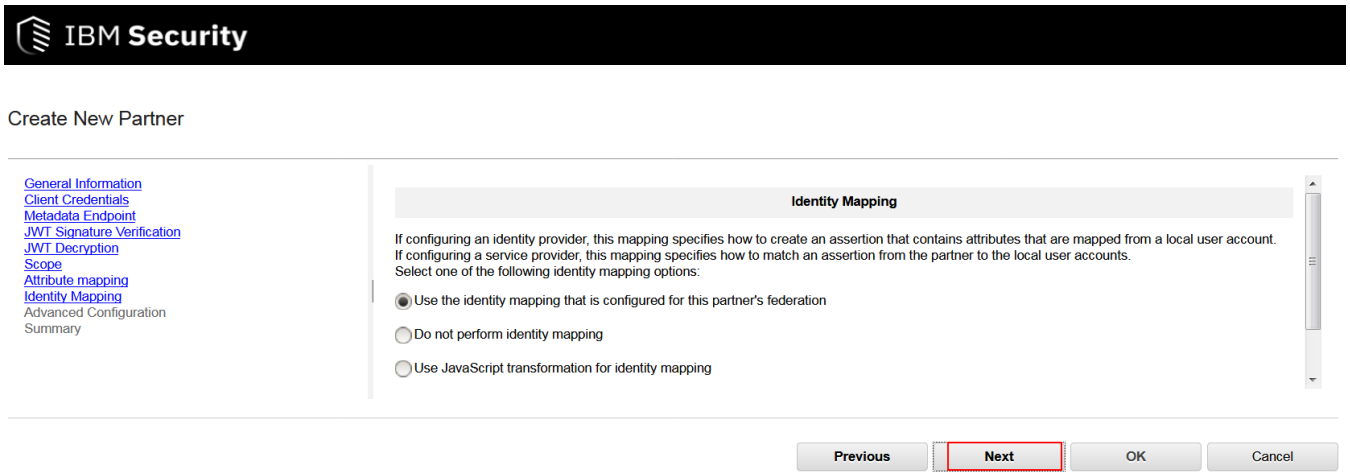
We don't need to change JWT Decryption so click **Next**.



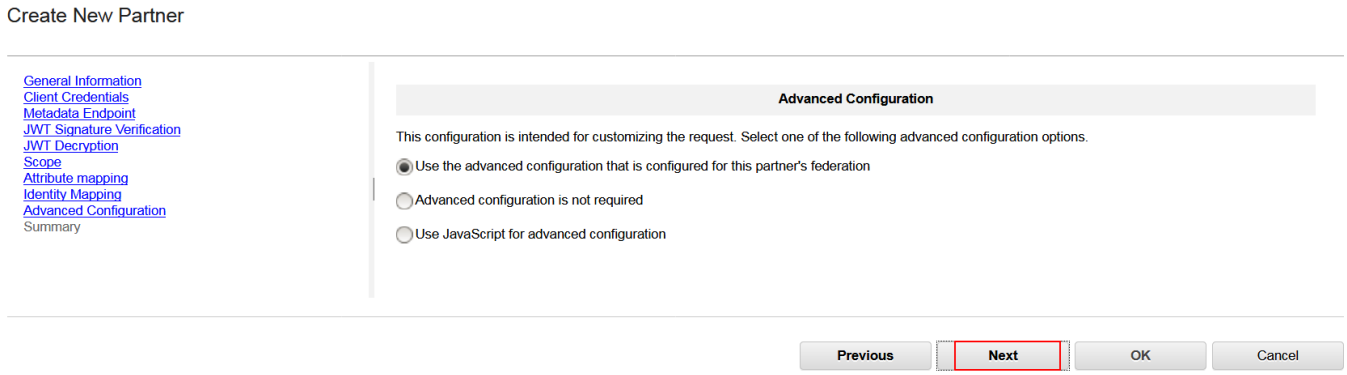
We don't need to change Scopes so click **Next**.



We don't need to change Attribute Mapping so click **Next**.



We don't need to change Identity Mapping so click **Next**.



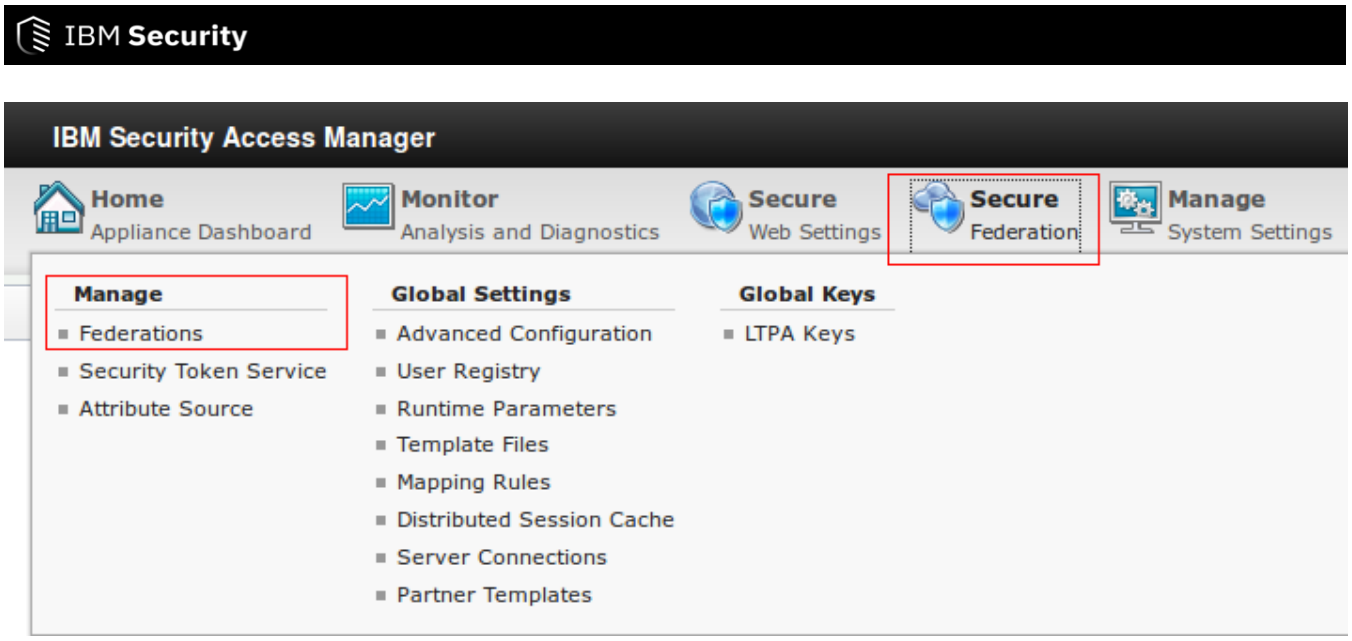
We don't need to change Advanced Configuration so click **Next**.

Click OK on the Summary panel.

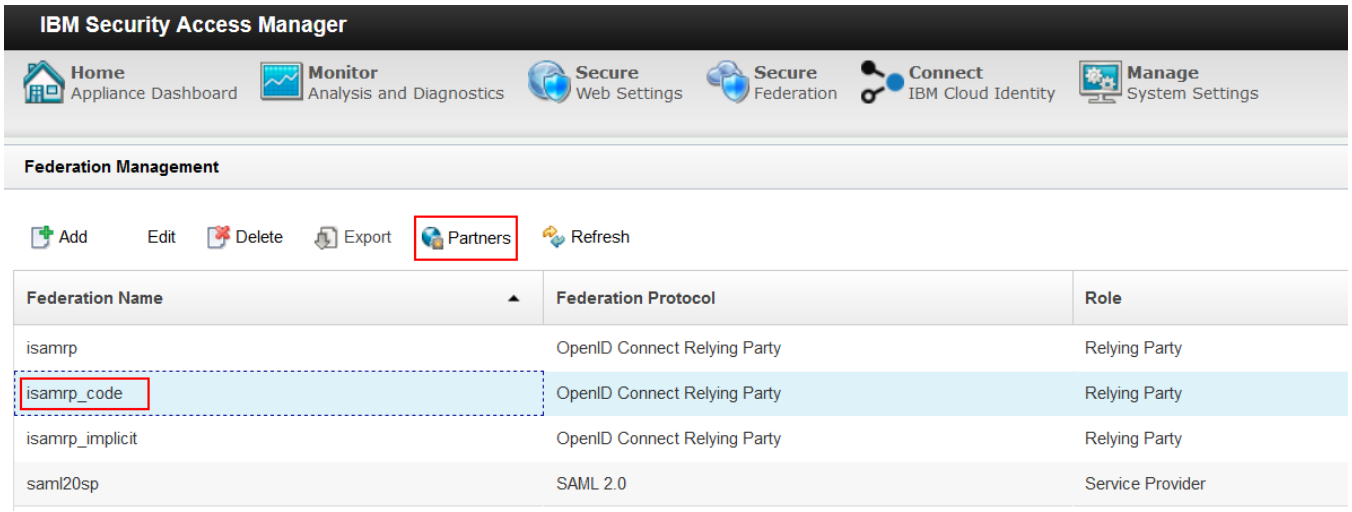
Follow on-screen instructions to **deploy** pending changes.

13.2.8 Configuring OpenID Connect Relying Party Partner for authorization_code federation

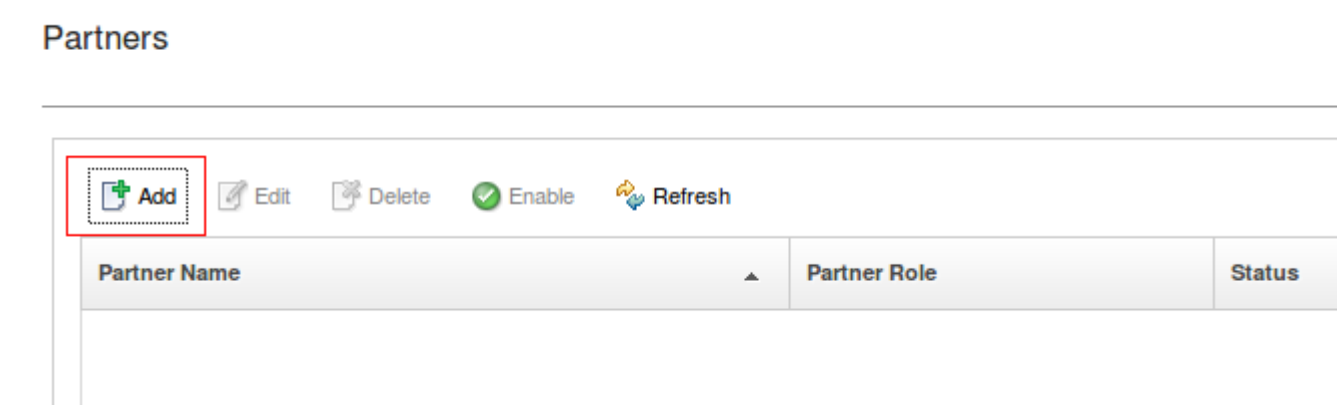
In this section we configure a partner for the federation which supports authorization code flow.



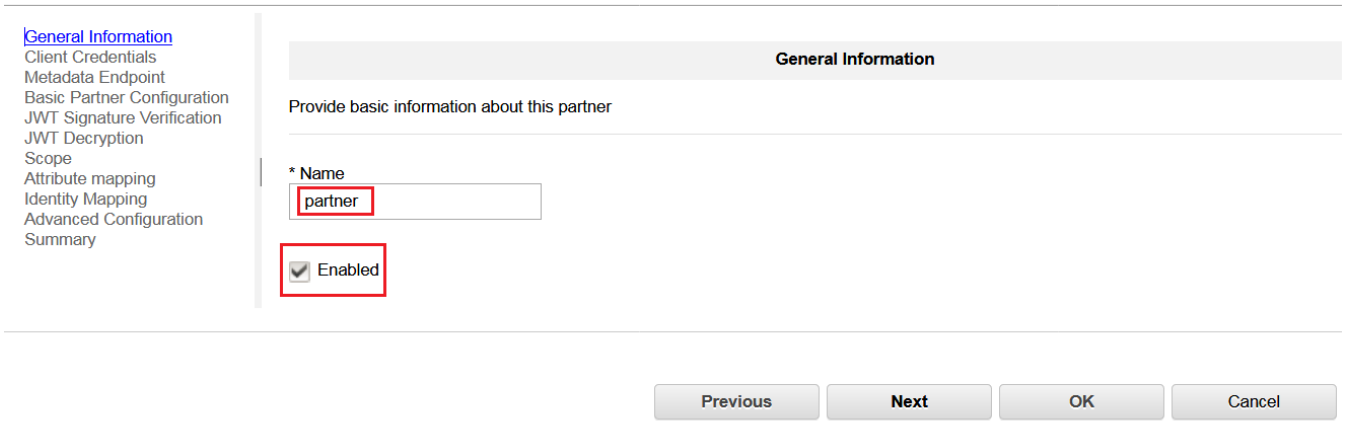
Under **Secure Federation** menu, click **on Manage: Federations**.



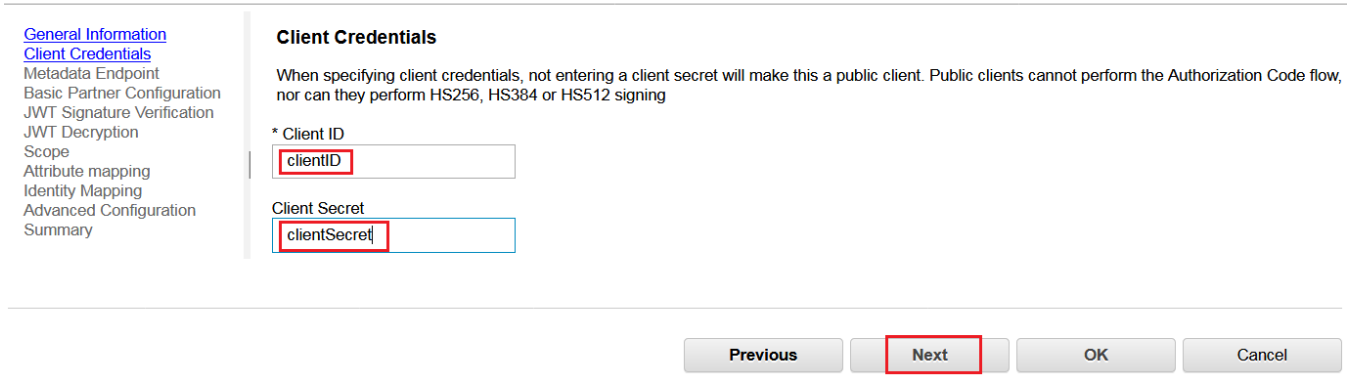
Select the **isamrp_code** federation and click **Partners**.



Click on **Add** to configure OIDC RP Partner.

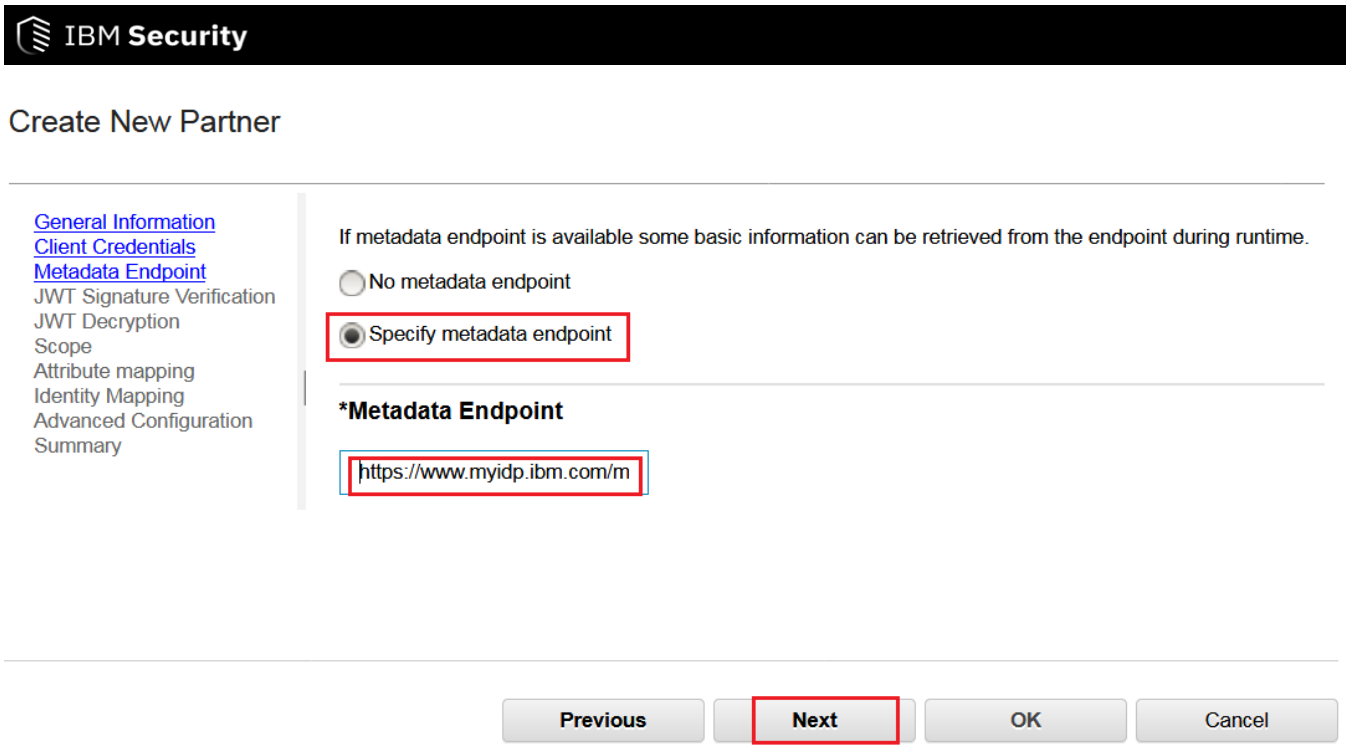


Enter **partner** as the name of the RP Partner and select the **Enabled** checkbox.



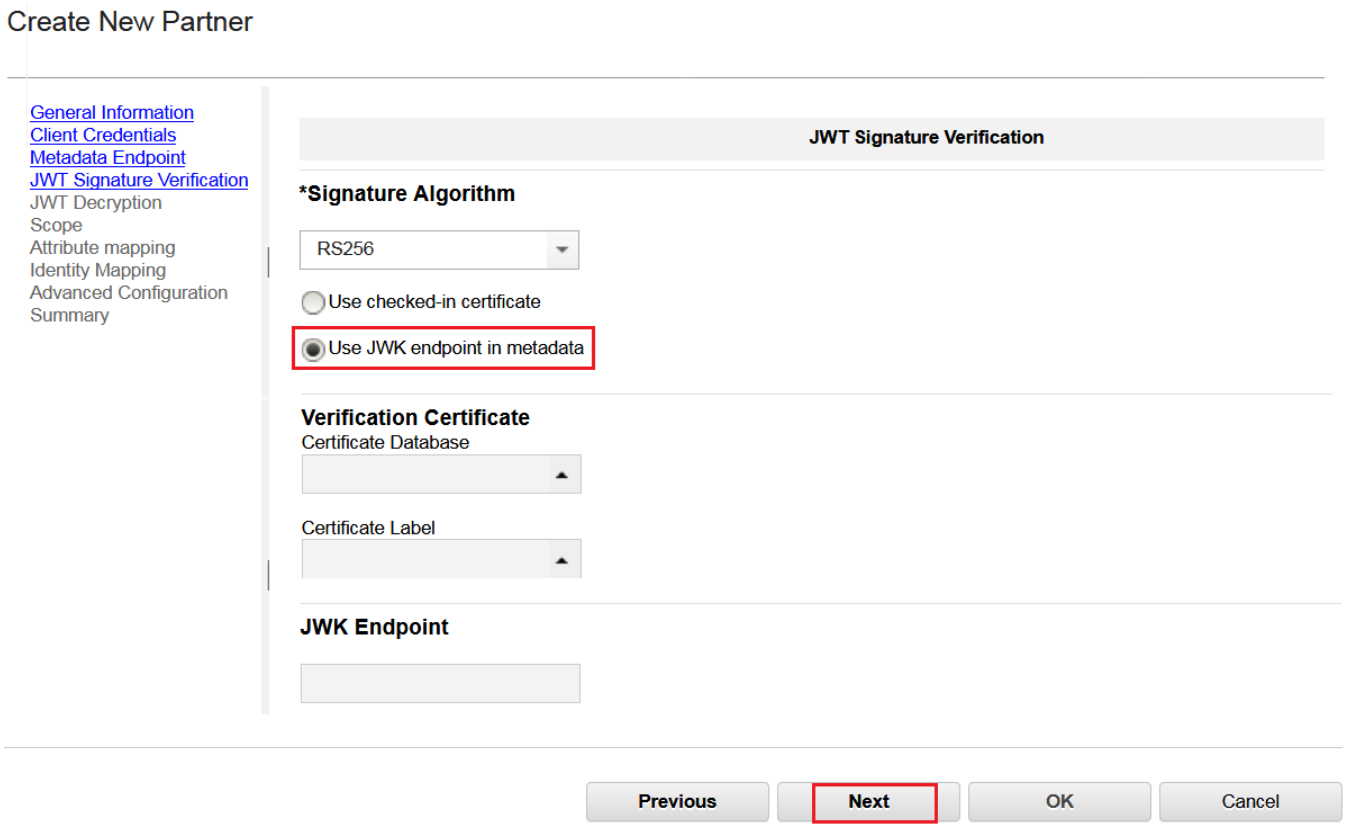
Enter **clientID** as Client ID and **clientSecret** as Client Secret.

Click on **Next**.



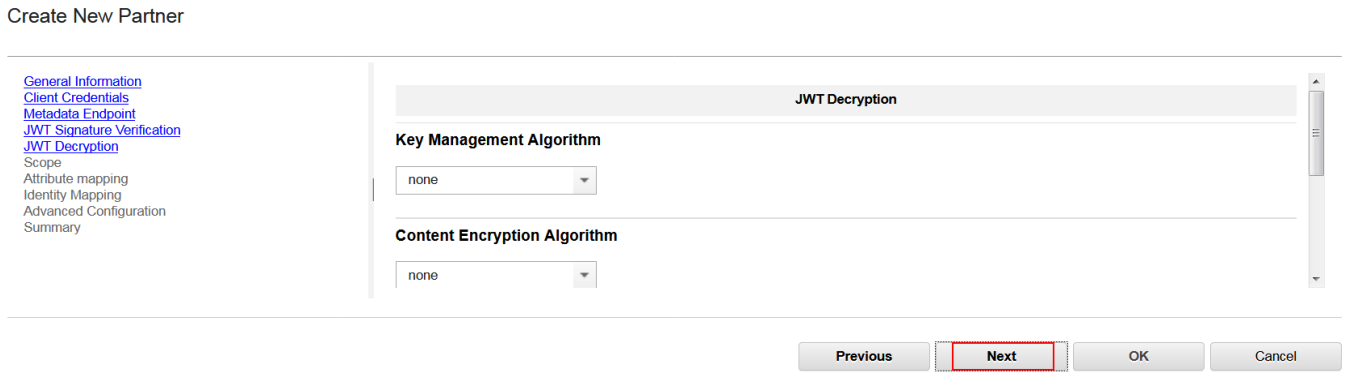
On the Metadata Endpoint panel, select **Specify metadata endpoint** radio button. Enter https://www.myidp.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCDefinition as the Metadata Endpoint.

Click on **Next.**

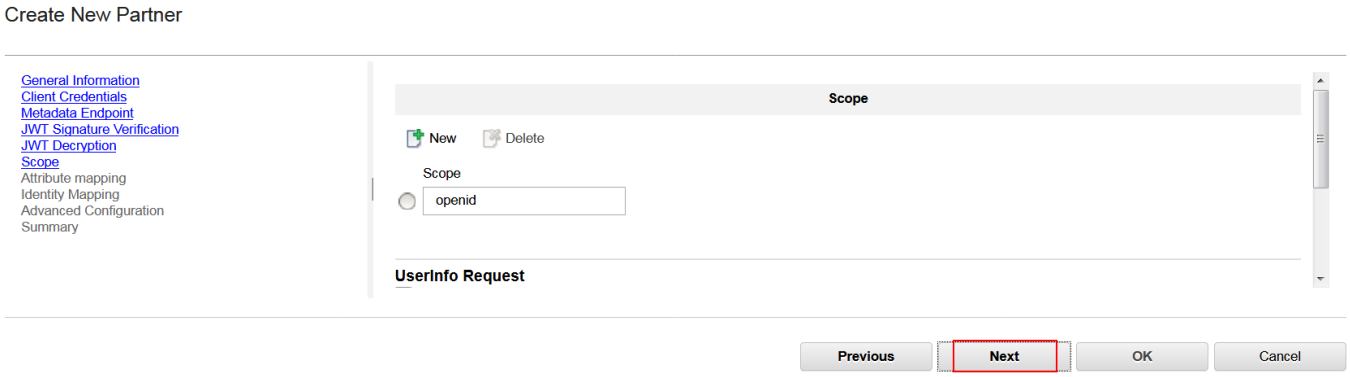


On the **JWT Signature Validation** panel, select **Use JWK endpoint in metadata.**

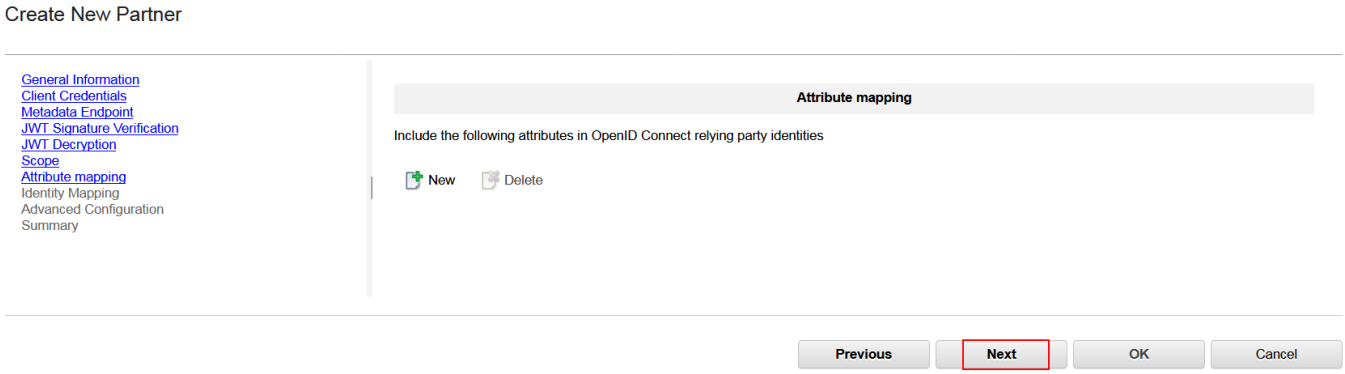
Click on **Next.**



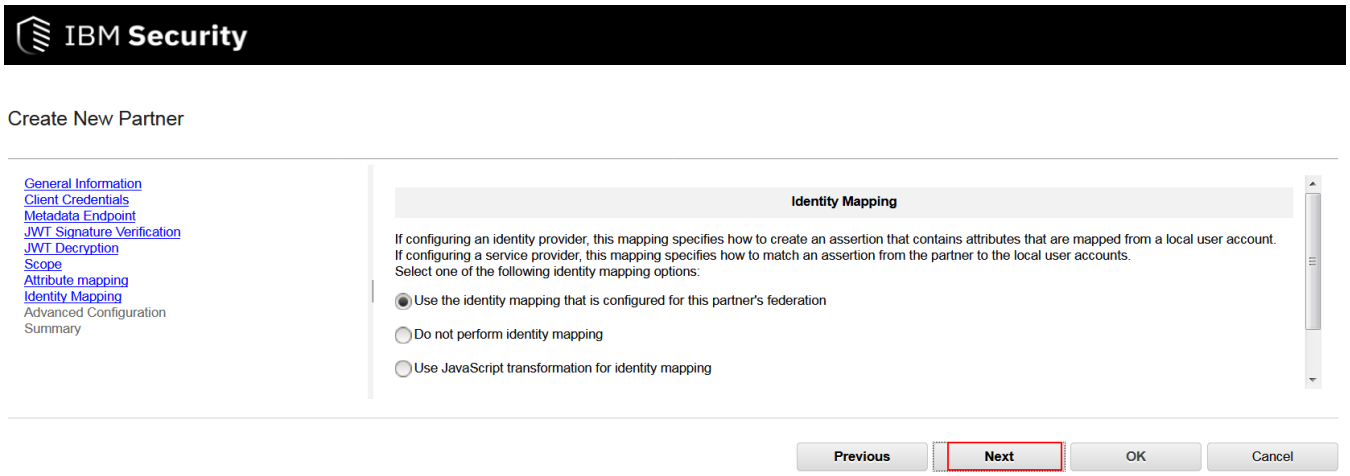
We don't need to change JWT Decryption so click **Next**.



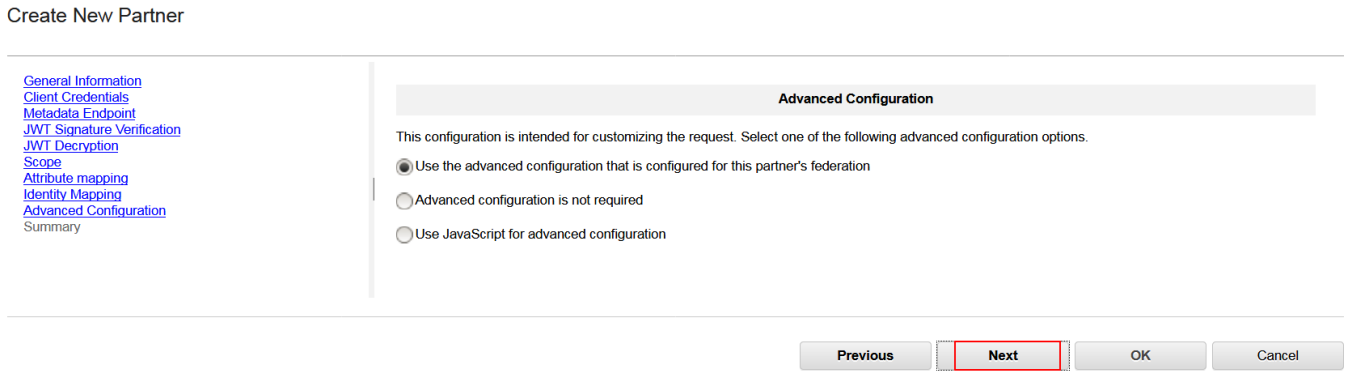
We don't need to change Scopes so click **Next**.



We don't need to change Attribute Mapping so click **Next**.



We don't need to change Identity Mapping so click **Next**.



We don't need to change Advanced Configuration so click **Next**.

Click OK on the Summary panel.

Follow on-screen instructions to **deploy** pending changes.

SCRIPT-END:
The script should display the following:

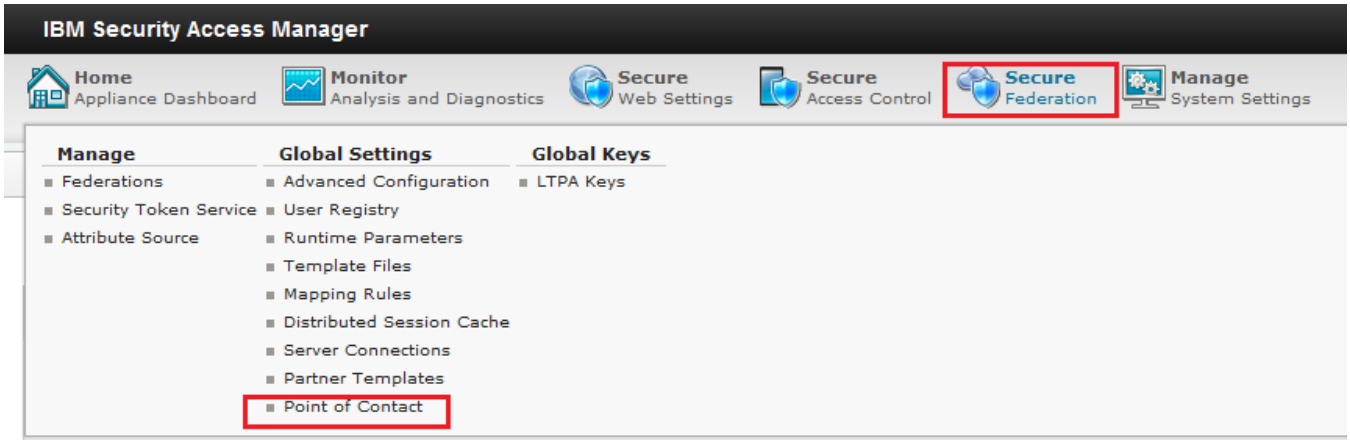
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner

13.2.9 Configuring Point of Contact profile

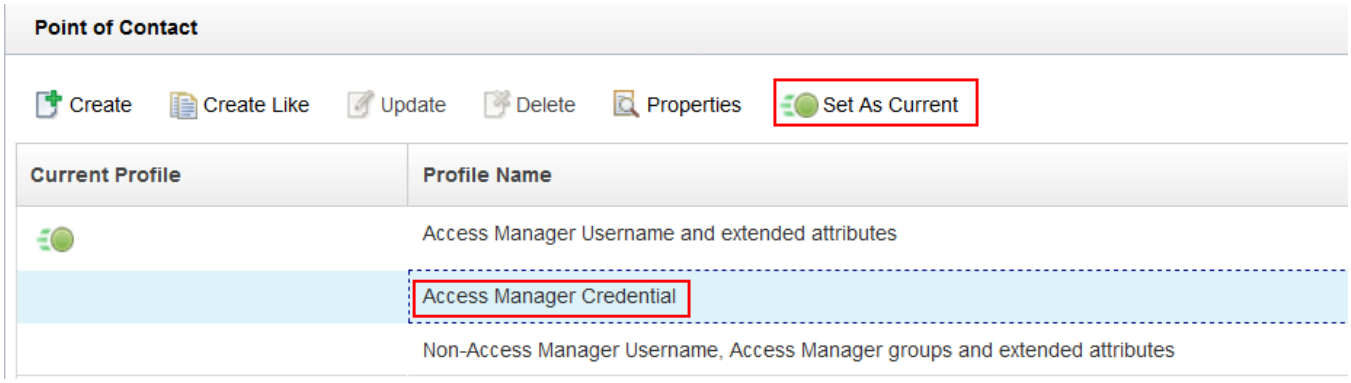
In this section we set the point of contact to “Access Manager Credential” profile.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

For the SP, run this script: **OIDCRPConfig.py -configure POC_For_OIDC**



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



Select the "**Access Manager Credential**" PoC Profile and Click "**Set As Current**" button to set this profile as Current Profile.

Deploy pending changes.

SCRIPT-END:
The script should display the following:

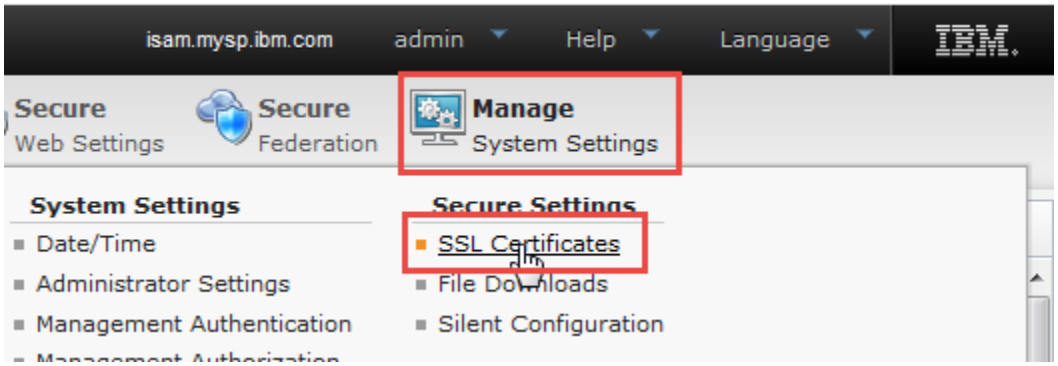
INFO:FederationManager:Configure POC profile
INFO:FederationManager:Successfully configured POC profile

13.2.10 Adding Signer Certificate

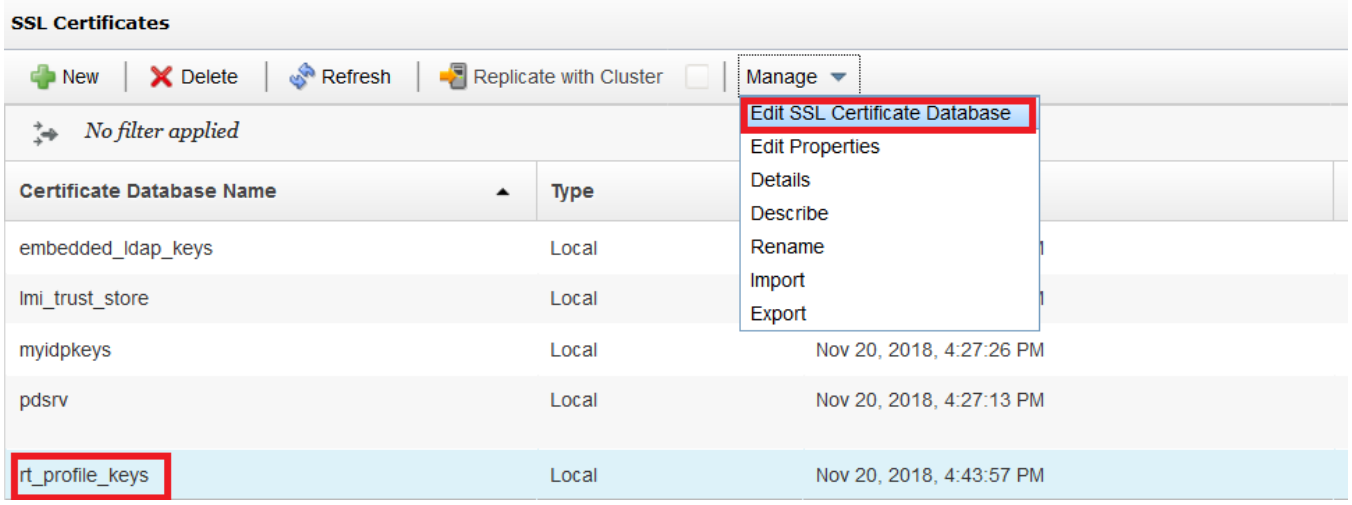
In this section we add the IdP webseal signer certificate to the federation runtime keystore, since the federation runtime will access the token endpoint to exchange a code for a token.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

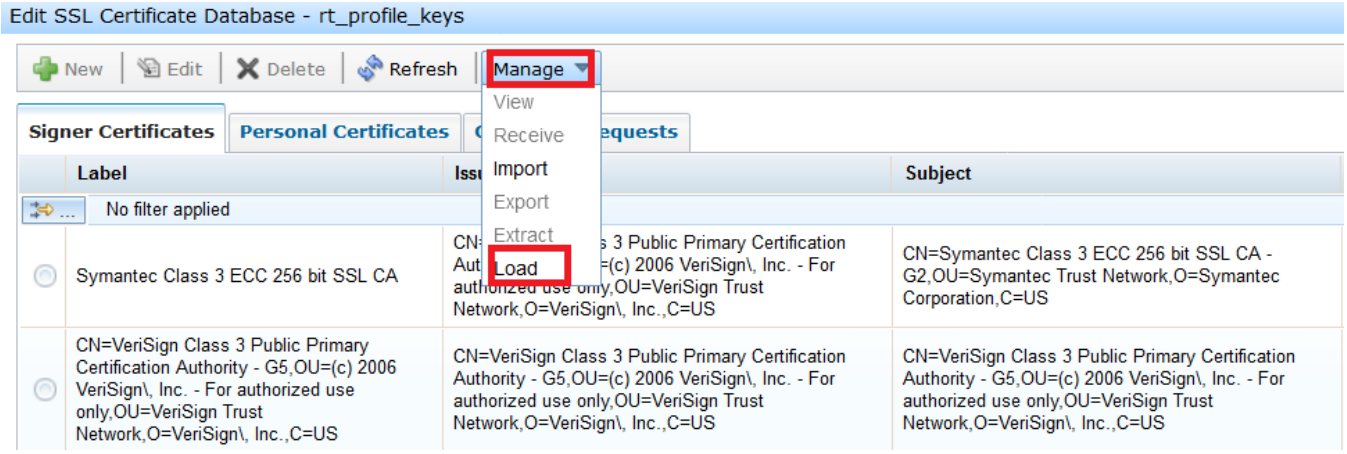
For the SP, run this script: **OIDCRPConfig.py -configure Add_Signer_Cert**



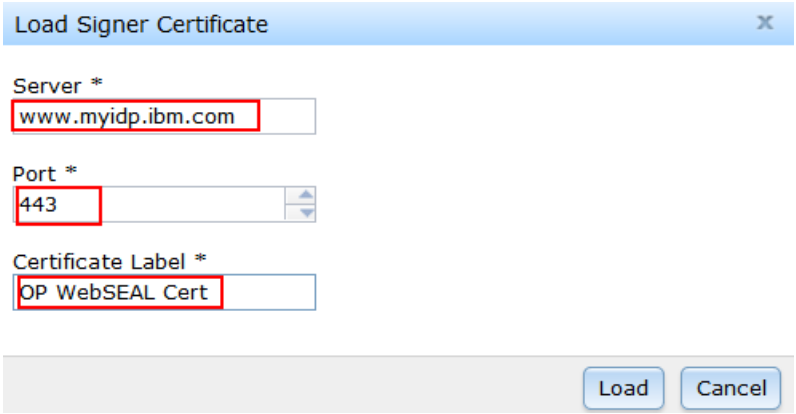
Navigate to **Manage System Settings > Secure Settings: SSL Certificates**



Select rt_profile_keys and Click on Manage -> Edit SSL Certificate Database.



Click on Manage -> Load



Enter www.myidp.ibm.com as the Server name, enter **443** as the Port and enter **OP WebSEAL cert** as the Certificate Label

Click Load and deploy pending changes.

SCRIPT-END:

The script should display the following:

INFO:WGAManager:Configuring Signer Certificates

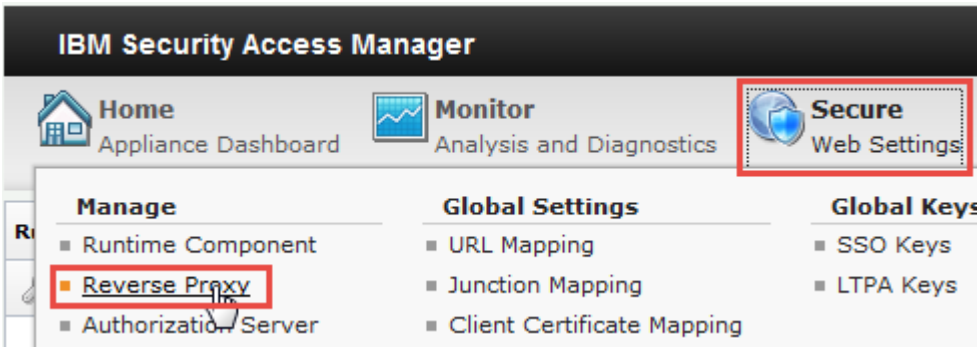
INFO:WGAManager:Successfully configured Signer Certificates

13.2.11 Configuring Reverse Proxy for OpenID Relying Party

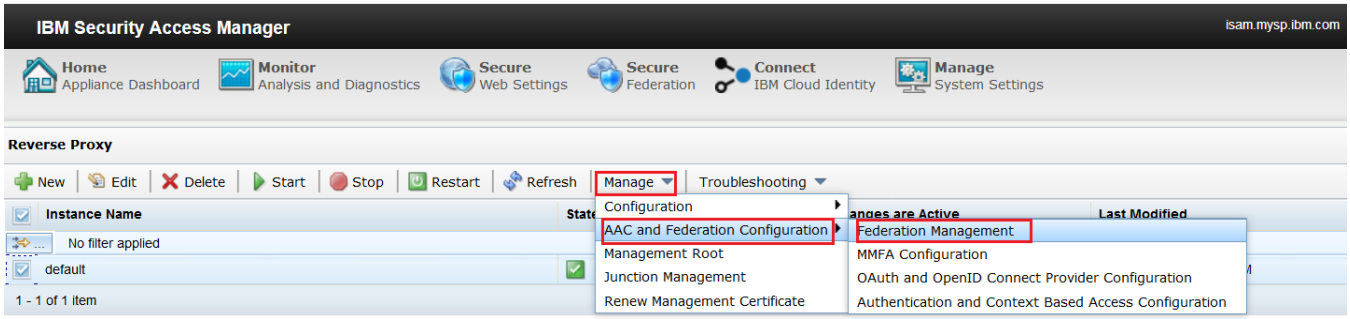
In this section we are configuring the reverse proxy instance for OpenID relying party federations.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

For the SP, run this script: **OIDCRPConfig.py -configure WebSEAL_Conf_OIDC_RP**

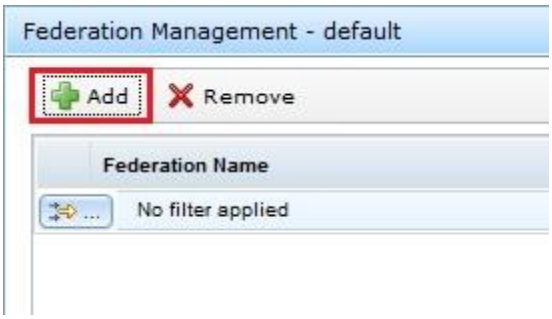


In the mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.

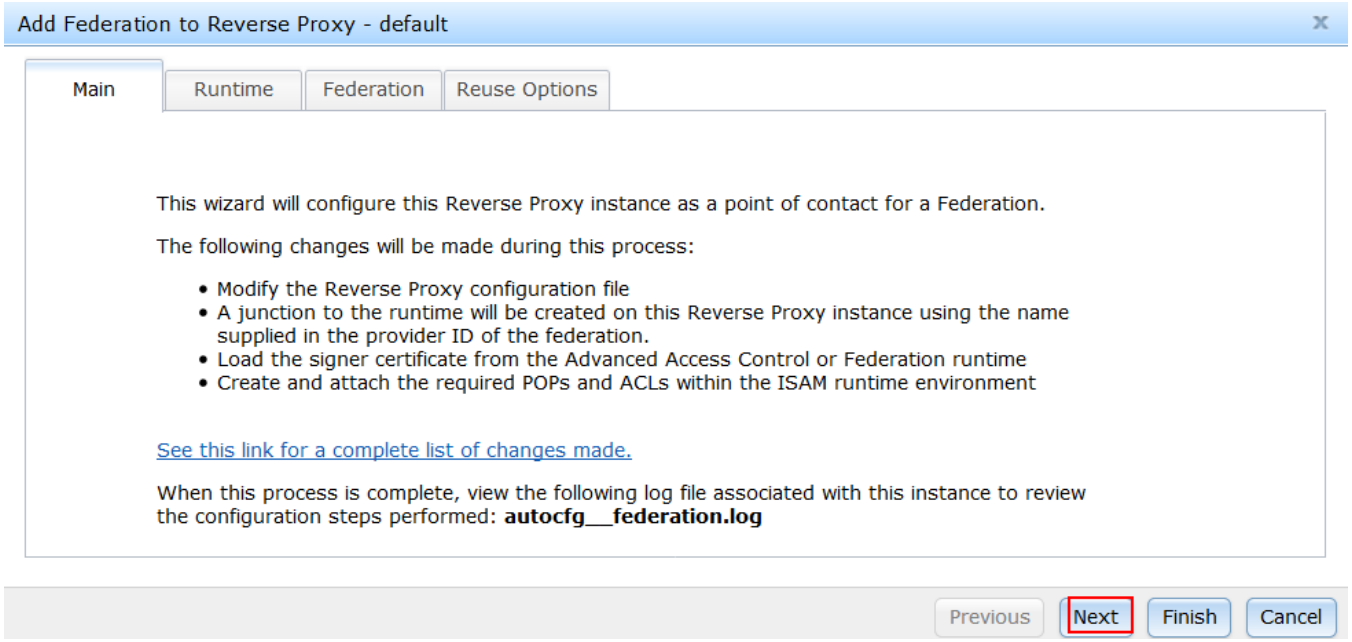


Select the Reverse Proxy instance, and click on **Manage -> AAC and Federation Configuration -> Federation Management.**

Select the Reverse Proxy instance, and click on Manage -> Federation Management.

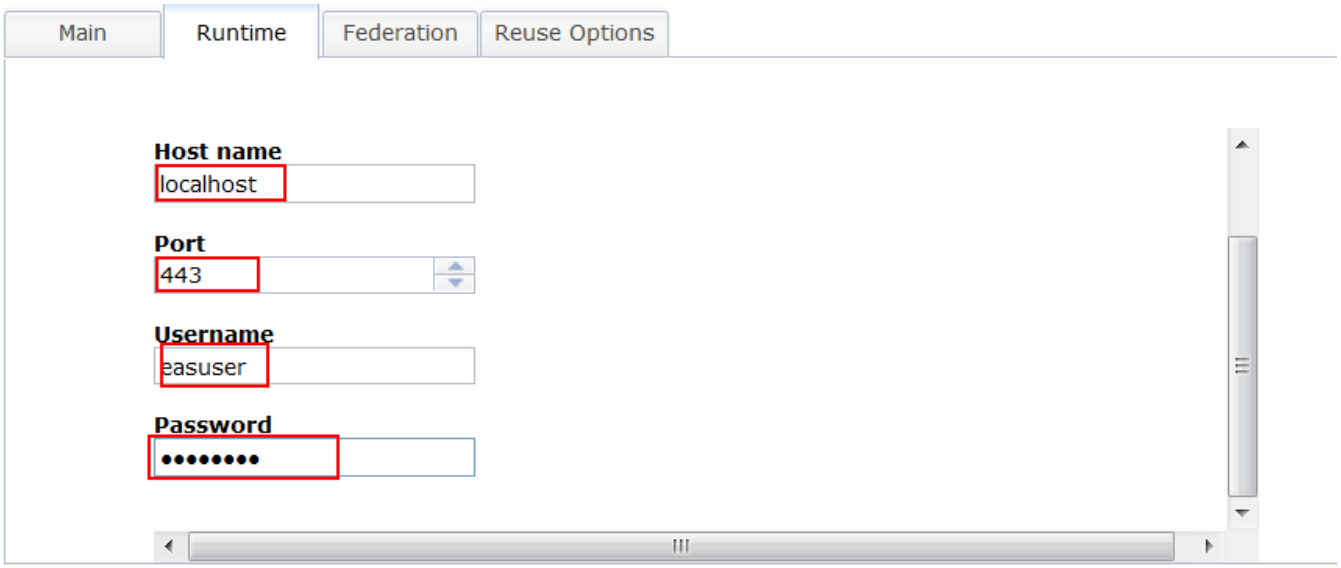


To add the RP federation, click on the **Add** button.

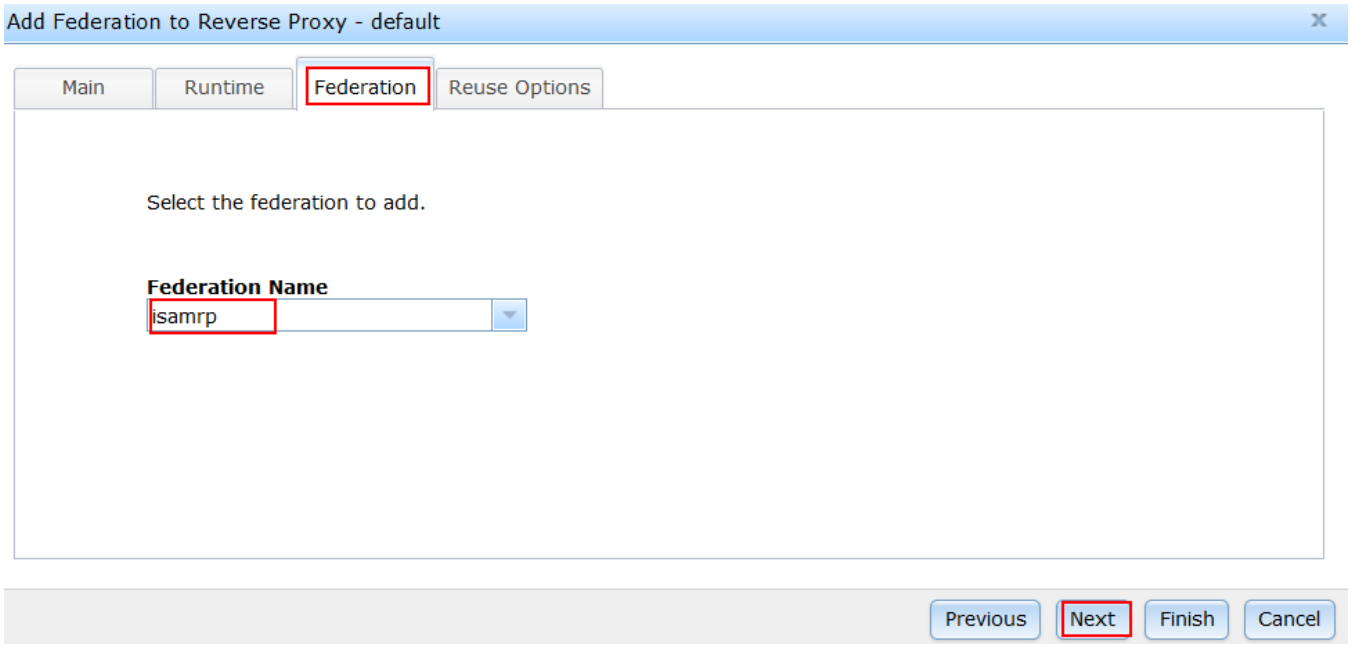


Click Next on the Main panel.

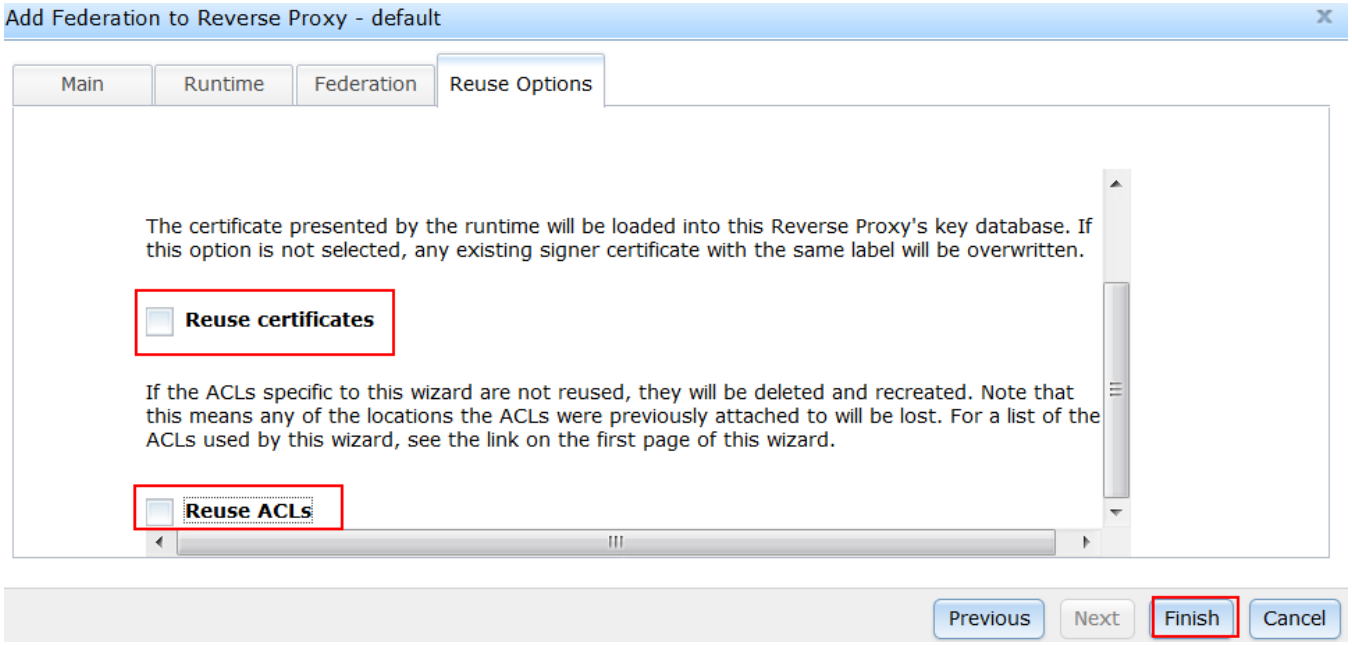
There are three panels which need to be filled out.



Inside the **Runtime** pane, user has to provide the details to authenticate with federation runtime. The details include the host, port, user name and password. All of them are required. When you move to the next pane, these details are used to connect to the Federation Runtime to retrieve a list of configured federations.



On the Federation Tab, Select the RP federation created in the previous section.



The next tab is the ACLs and Certificates panel. You can choose to reuse ACLs and Certificates if they exist or create new ones.

Repeat this step for the other federations **isamrp_code, isamrp_implicit.**

Once all the panels are done, click on **Finish** and then **Deploy** the Pending changes.

SCRIPT-END:

The script should display the following:

INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP
INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP
INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP

If the configure -All option was used the script end should look like this

The script should display the following:

INFO:FederationManager:Upload all mapping rules
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Create a mapping rule
INFO:FederationManager:Successfully created the Mapping Rule
INFO:FederationManager:Successfully updated the Mapping Rule
INFO:FederationManager:Successfully updated the Mapping Rule
INFO:FederationManager:Configuring the easuser password
INFO:FederationManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Successfully configured the OIDC RP Federation
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Successfully configured the OIDC RP Federation
INFO:FederationManager:Configuring the OIDC RP Federation
INFO:FederationManager:Successfully configured the OIDC RP Federation
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configure POC profile
INFO:FederationManager:Successfully configured POC profile
INFO:WGAManager:Configuring Signer Certificates
INFO:WGAManager:Successfully configured Signer Certificates
INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP
INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP
INFO:WGAManager:Configure WebSEAL for OIDC RP
INFO:WGAManager:Successfully configured WebSEAL for OIDC RP

13.3 Testing OIDC Single Sign-on Flow

13.3.1 Testing Hybrid Flow

We are now ready to test the OpenID Connect configuration.

> **Note:** It is recommended that after every single sign-on you restart your browser to remove all session cookies at both IdP and SP between each of the tests below.

A RP(SP) initiated OIDC flow can be triggered using

https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>

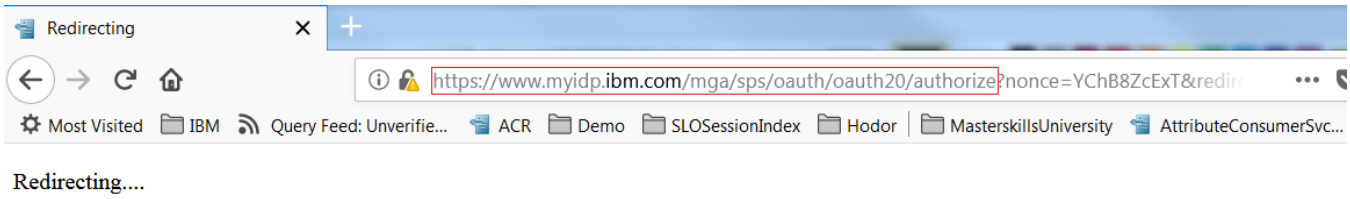
Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/partner?Target=/isam/mobile-demo/diag

Trigger the flow using a browser.

If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by hitting the authorize endpoint.

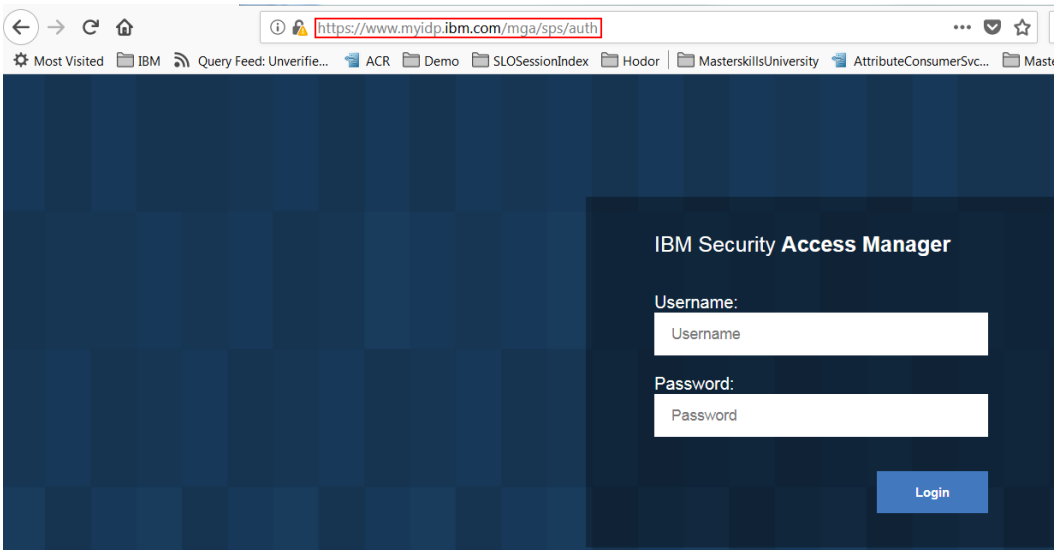
An example of the authorize URL

https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?nonce=HS8qF166ty&redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp%2Fredirect%2Fpartner&response_type=code+id_token+token&response_mode=form_post&state=5uxsESNh5e&scope=openid&client_id=clientID



The authorize URL then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.

At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDI0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran a hybrid flow, the OP will send code, access_token and id_token as a response.

Inspect the id_token using https://jwt.io/

Encoded PASTE A TOKEN HERE

eyJraWQiOiI0eHZRY2VGVU5OX1l2WERtb0ZRRWtfVGtoMTVEOHdTS2xvWTV6LVd6UWM4IiwiYWxnIjoiUlMyNTYifQ.eyJub25jZSI6IkhTOHFGMTY2dHkiLCJpYXQiOjE1NDMxOTkwOTUsImlzcyI6Imh0dHBzOi8vd3d3Lm15aWRwLmlibS5jb20iLCJhdF9oYXNoIjoiWFBDdVZzUmVLNmVuQjdzWVhqQWRXUSIsInN1YiI6InRlc3R1c2VyIiwiZXhwIjoxNTQzMjAyNjk1LCJjX2hhc2giOiJRTGI1eGxnMGR6TWJzdUx1RHRNUkRRIiwiYXVkIjoiY2xpZW50SUQifQ.OwC9HcZZ7jJJR84CpE_GzDwtlfHZHmycxx5o1XsG2o28hKKXiUhJdvULoe1nIaS7kgQSVyz7t1wLHlSSQnAD5GSdgHiGLzXj1Nqvn_jRF3HO_ViutSvKdm_0rFXudbSltF4ymSBS9scBQGJLg3WZwS7fBSGb3sFdMt70NQ2_WZk

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "4xvQceFUNN_YvXDmoFQEk_Tkh15D8wSKloY5z-WzQc8",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "nonce": "HS8qF166ty",
  "iat": 1543199095,
  "iss": "https://www.myidp.ibm.com",
  "at_hash": "XPCuVsReK6enB7sYXjAdWQ",
  "sub": "testuser",
  "exp": 1543202695,
  "c_hash": "QLb5xlg0dzMbsuLuDtMRDQ",
  "aud": "clientID"
}
```


To verify the access_token use the Userinfo endpoint, we could use a browser extension for a REST tool or use postman to make this request.

```
curl --request GET \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo \
  --header 'authorization: Bearer bytgVJZsycvBAP1JNJ62' \
  --cookie 'AMWEBJCT!%252Fmga!JSESSIONID=00007t9wM8_slegs0YGxELNKi5e%3A71b43363-316b-4790-baf
```

Userinfo endpoint - https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo
Header – Auhtorization: Bearer <Access Token>

Output

```
{
  "sub": "testuser"
}
```

13.3.2 Testing Authorization Code Flow

We are now ready to test the OpenID Connect Federation that we have configured.

> **Note:** It is recommended that you restart your browser to remove all session cookies at both IdP and SP between each of the tests below.

A RP(SP) initiated OIDC flow can be triggered using

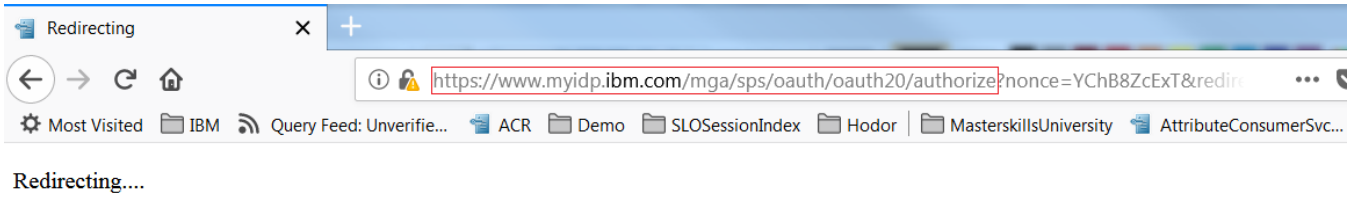
https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp_code/kickoff/partner?Target=/isam/mobile-demo/diag

Trigger the flow using a browser.

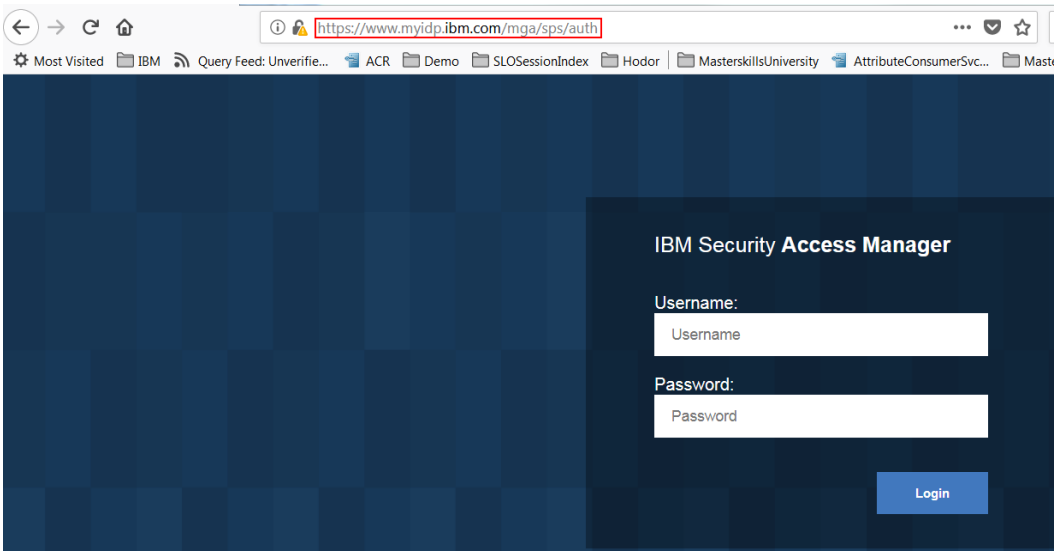
If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by hitting the authorize URL.

An example of the authorize URL
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp_code%2Fredirect%2Fpartner&response_type=code&state=i7zHd5vJCL&scope=openid&client_id=clientID



The authorize URL then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.

At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDl0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran a authorization_code flow, the OP will send code as a response in the query string. The ISAM runtime time, uses the code to exchange it for a access_token, refresh_token and id_token.

13.3.3 Testing Implicit Flow

We are now ready to test the OpenID Connect Federation that we have configured.

> **Note:** It is recommended that you restart your browser to remove all session cookies at both IdP and SP between each of the tests below.

A RP(SP) initiated OIDC flow can be triggered using

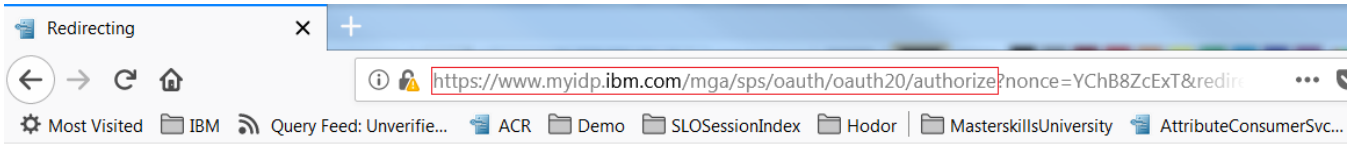
https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp_implicit/kickoff/partner?Target=/isam/mobile-demo/diag

Trigger the flow using a browser.

If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by hitting the authorize URL.

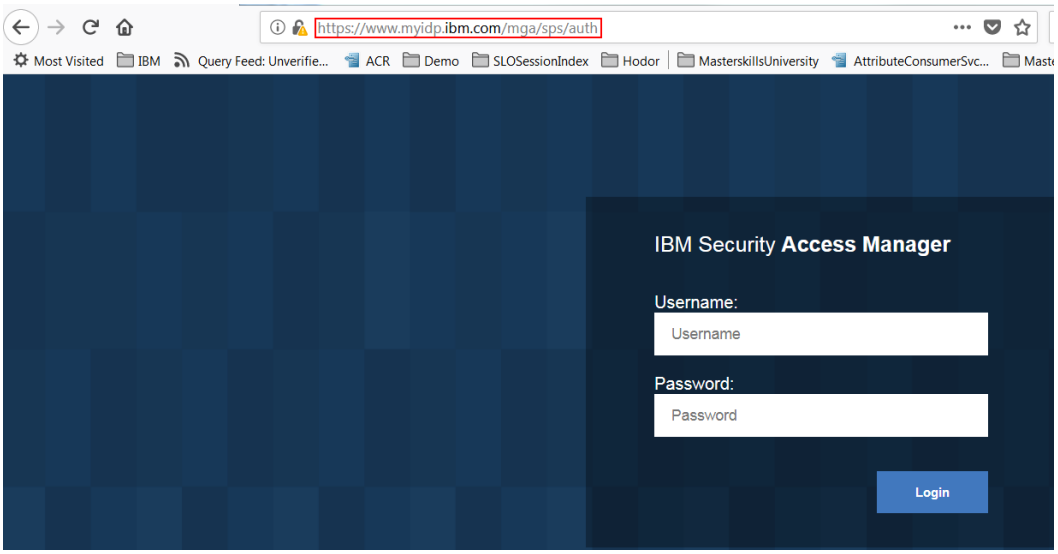
An example of the authorize URL
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?nonce=DXBmCxAi5v&redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp_implicit%2Fredirect%2Fpartner&response_type=id_token+token&response_mode=form_post&state=cVCH6ePkU0&scope=openid&client_id=clientID



Redirecting....

The authorize URL then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.

At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweln7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDl0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran an implicit flow, the OP will send access_token and id_token as a response.

Inspect the id_token using https://jwt.io/

Encoded PASTE A TOKEN HERE

eyJraWQiOiI0eHZRY2VGVU5OX1l2WERtb0ZRRWtfVGtoMTVEOHdTS2xvWTV6LVd6UWM4IiwiYWxnIjoiUlMyNTYifQ.eyJub25jZSI6IkhTOHFGMTY2dHkiLCJpYXQiOjE1NDMxOTkwOTUsImlzcyI6Imh0dHBzOi8vd3d3Lm15aWRwLmlibS5jb20iLCJhdF9oYXNoIjoiWFBDdVZzUmVLNmVuQjdzWVhqQWRXUSIsInN1YiI6InRlc3R1c2VyIiwiZXhwIjoxNTQzMjAyNjk1LCJjX2hhc2giOiJRTGI1eGxnMGR6TWJzdUx1RHRNUkRRIiwiYXVkIjoiY2xpZW50SUQifQ.OwC9HcZZ7jJJR84CpE_GzDwtlfHZHmycxx5o1XsG2o28hKKXiUhJdvULoe1nIaS7kgQSVyz7t1wLHlSSQnAD5GSdgHiGLzXj1Nqvn_jRF3HO_ViutSvKdm_0rFXudbSltF4ymSBS9scBQGJLg3WZwS7fBSGb3sFdMt70NQ2_WZk

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "4xvQceFUNN_YvXDmoFQEk_Tkh15D8wSKloY5z-WzQc8",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "nonce": "HS8qF166ty",
  "iat": 1543199095,
  "iss": "https://www.myidp.ibm.com",
  "at_hash": "XPCuVsReK6enB7sYXjAdWQ",
  "sub": "testuser",
  "exp": 1543202695,
  "c_hash": "QLb5xlg0dzMbsuLuDtMRDQ",
  "aud": "clientID"
}
```

To verify the access_token use the Userinfo endpoint, we could use a browser extension for a REST tool or use postman to make this request.

```
curl --request GET \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo \
  --header 'authorization: Bearer bytgVJZsycvBAP1JNJ62' \
  --cookie 'AMWEBJCT!%252Fmga!JSESSIONID=00007t9wM8_slegs0YGxELNKi5e%3A71b43363-316b-4790-baf
```

Userinfo endpoint - https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo
Header – Auhtorization: Bearer <Access Token>


```
{
 "sub": "testuser"
}
```

14 Configuring OpenID Connect Flows to request for id_token and Userinfo claims

In this section, we configure advanced mapping rule for the OIDC relying party to request for additional UserInfo and id_token claims.

The OIDC specification recommends the use of the UserInfo endpoint. The UserInfo endpoint is useful, for example, when a Relying Party cannot parse a JWT Token to obtain information about the authenticated user.

Without the use of customization, the /userinfo endpoint contains only the field sub.

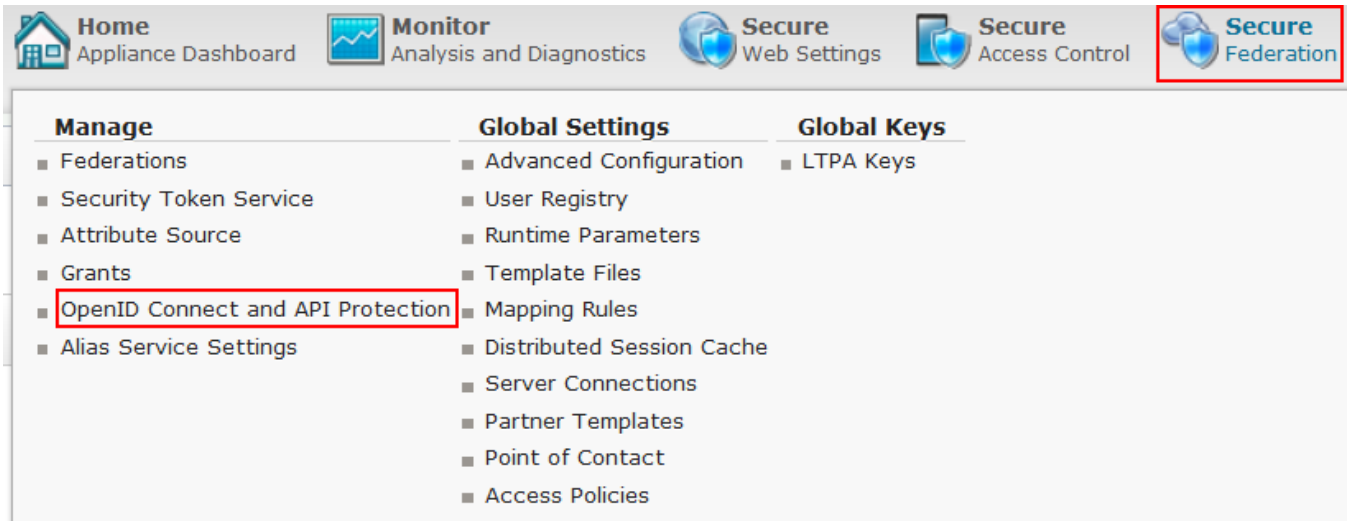
In this section, we will configure the OIDC OP to send return specific claims to userinfo and id_token.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the SP.

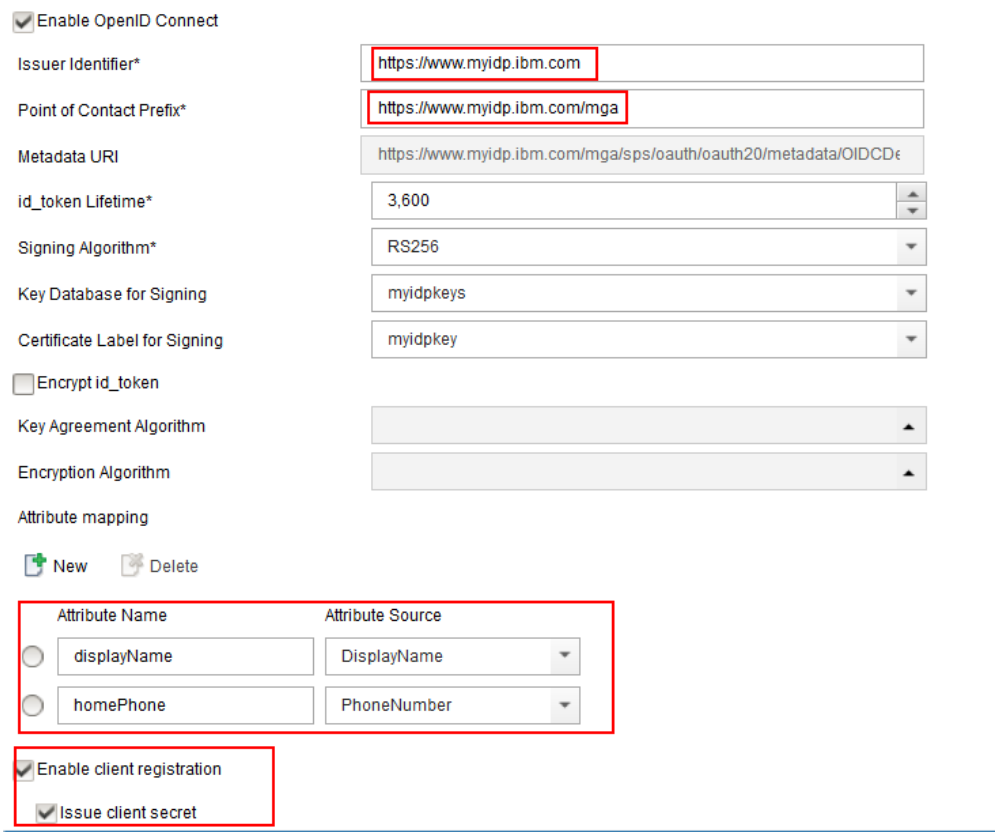
For the SP, run this script: **OIDCClaims.py -configure All**

14.1 Configuring OIDC OP Attribute Mapping

In this section, we check that the API Definition created in section 22.1.1 has the attribute mapping configured. Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**



Click **Edit** on OIDCDefinition.



Make sure that displayName amd homePhone Attribute Mapping are configured**.**

14.2 Configuring Advanced Mapping rule OIDC RP Federation

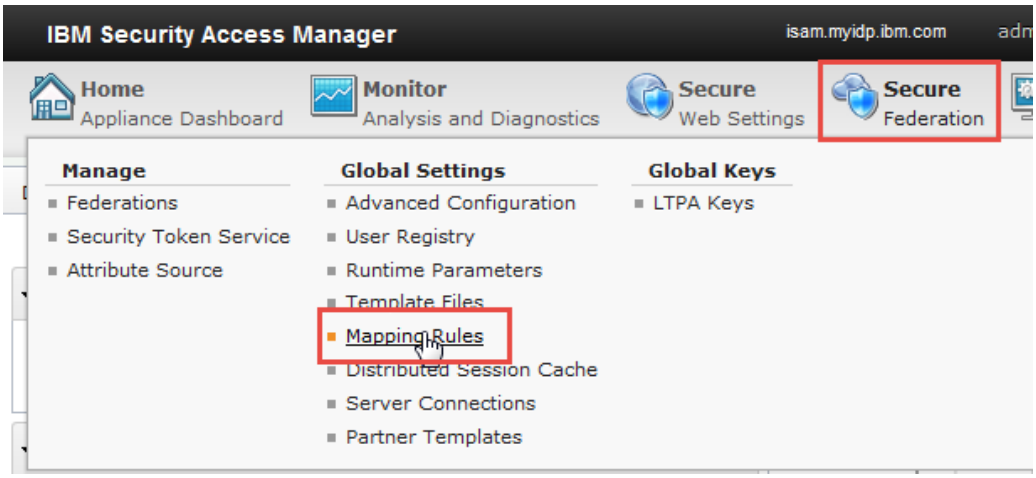
In this section, we configure an advanced configuration mapping rule at the OIDC RP federation to request for userinfo and id_token claims.

Go to the **…/providedfiles/Automation/sp_files/mapping_rules** directory and open the **oidc_adv_claims.js** file in a text editor

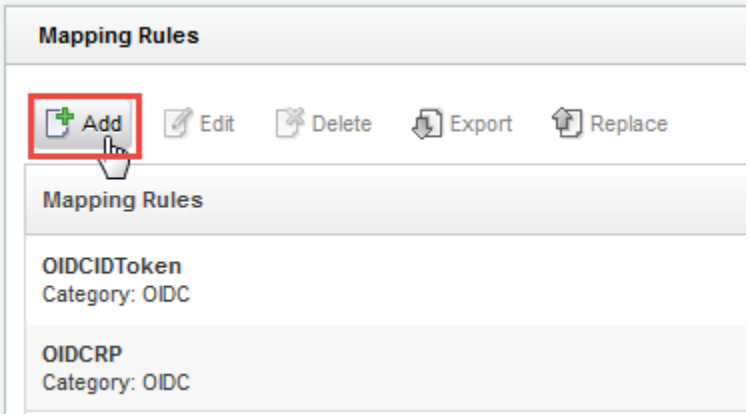
```
    IDMappingExtUtils.traceString(traceString);
  }

  /*
   * The following is an example of how to author and include an claims object in the request to /authorize.
   *
   * This claims parameter will request the email claim in the id_token as essential.
   */
  var add_claim_parameter = true;
  if(add_claim_parameter) {
    var claims = {
      "id_token": {
        "homePhone": {"essential":true}
      },
      "userinfo": {
        "displayName": {"essential":true}
      }
    };
    stsuu.addContextAttribute(new Attribute("claims", "urn:ibm:SAM:oidc:rp:authorize:req:param", JSON.stringify(claims)));
  }
}
```

In the above example, the authorize endpoint with contain a request for id_token claim homePhone, and userinfo claim displayName.

Now we're ready to create a Mapping Rule on the appliance with this content.



In the LMI Administration console, navigate to **Secure Federation→Global Settings: Mapping Rules**.



Click **Add** to add a new mapping rule.

Create Mapping Rule

```
   */
  var add_claim_parameter = true;
  if(add_claim_parameter) {
    var claims = {
      "id_token": {
        "homePhone": {"essential":true}
      },
      "userinfo": {
        "displayName": {"essential":true}
      }
    };
    stsuu.addContextAttribute(new Attribute("claims", "urn:ibm:SAM:oidc:rp:authorize:req:param", JSON.stringify(claim
  }
}

/*
 * An operation with the value "token" means we have received a request at our /redirect back from the OP
 * (typically with the authorization code, and are about to invoke a request to the token endpoint.
 * We can use this hook point for adding parameters to the /token and /userinfo request.
 */
if (operation == "token") {
```

Name: oidc_adv_claims

Category: OIDC

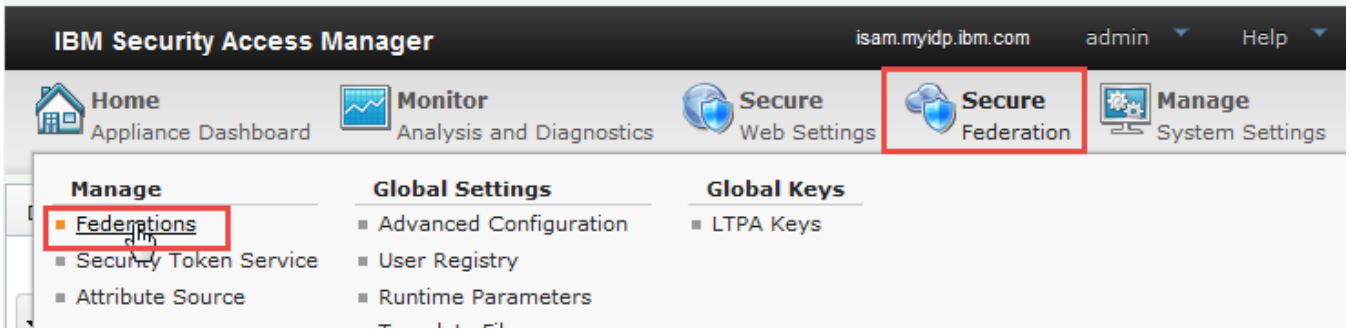
Save Close

Paste the rule text into the *Content* box. On Windows you can use **Ctrl-c** to paste.

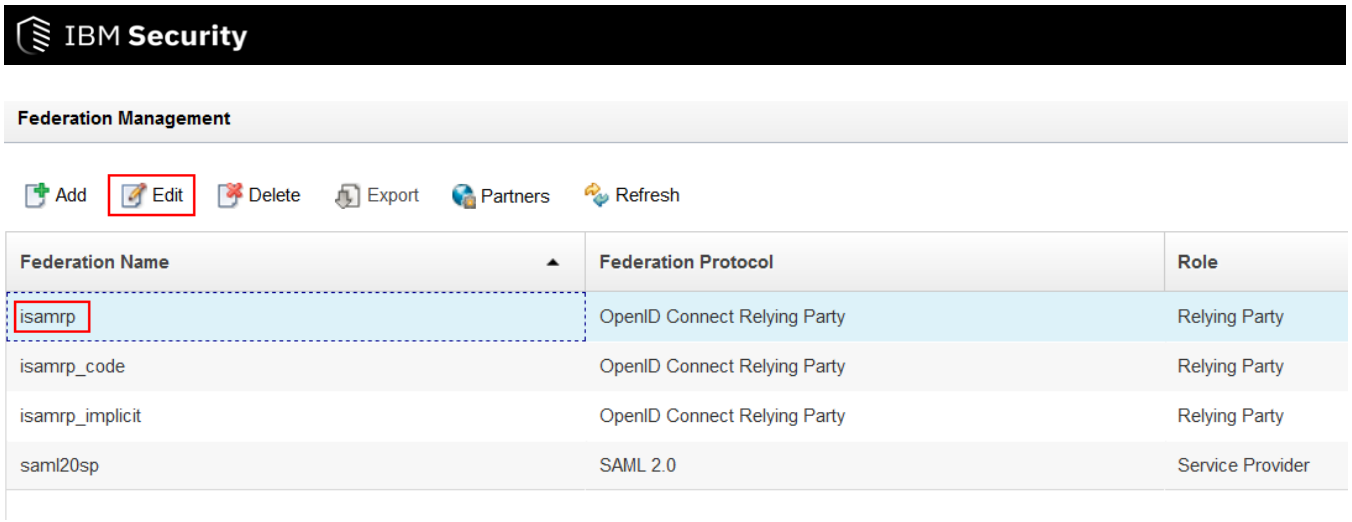
Enter **oidc_adv_claims** as the rule *Name* and select **OIDC** as the *Category*.

Click **Save** to save the new Mapping Rule.

We need to configure the federation to point to the advanced mapping rule we just created.

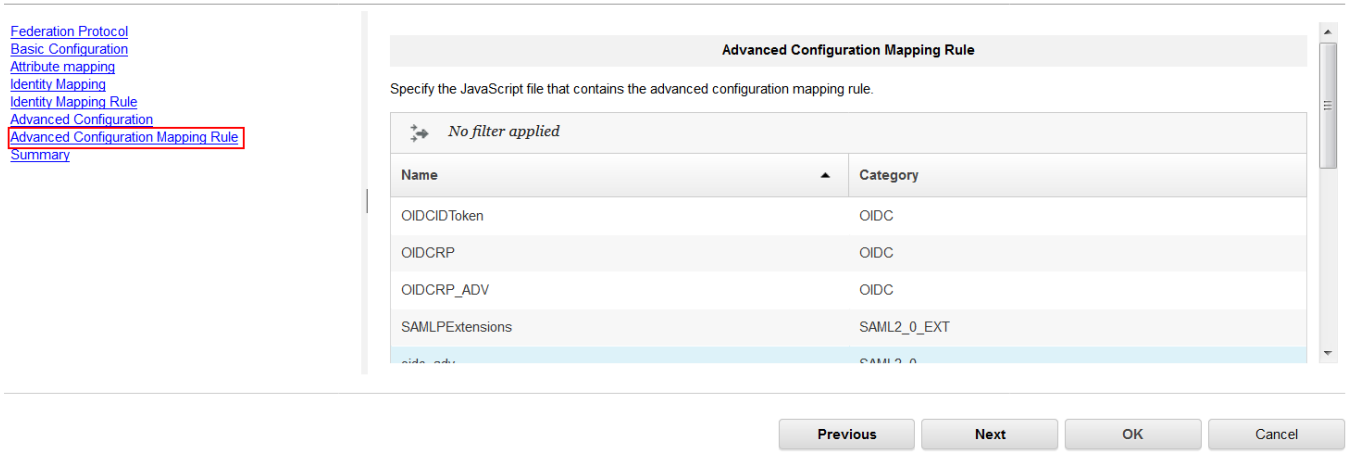


Using the administration console, navigate to **Secure Federation→Manage: Federations**.

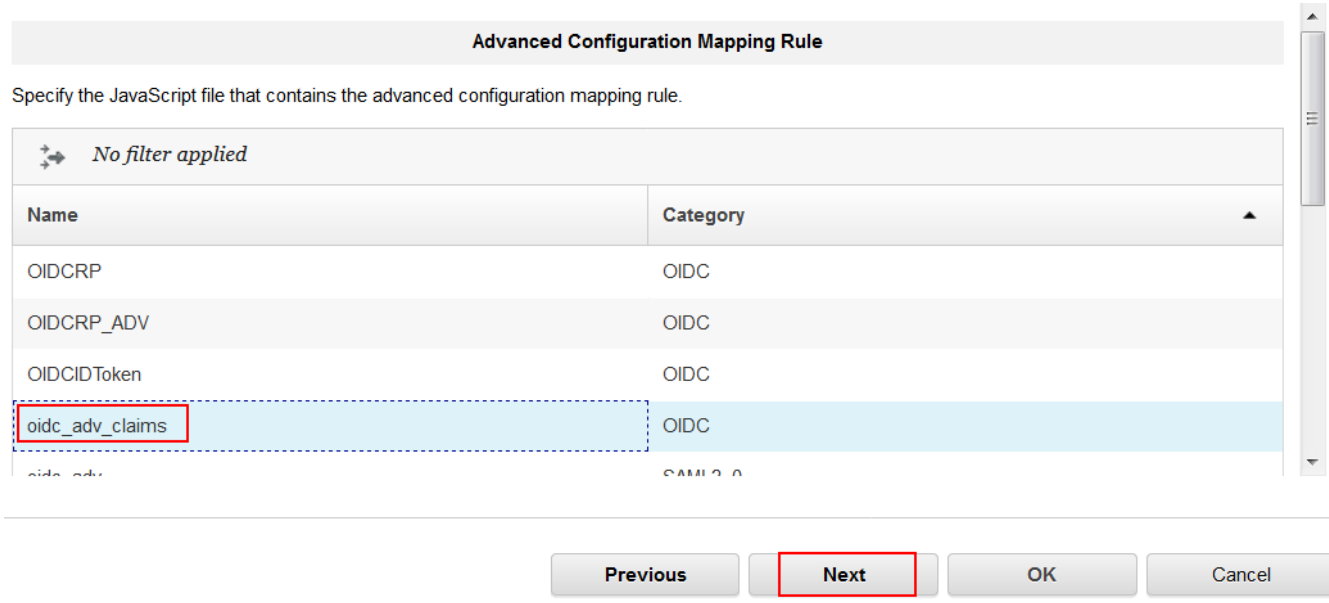


Select the **isamrp** federation and Click on the **Edit** button.

Update Federation



Navigate to Advanced Configuration Mapping Rule.



Select **oidc_adv_claims** as the Advanced Configuration Mapping Rule. Click Next.

On the **Summary** page click OK.

Deploy pending changes.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Updating OIDC RP Federation mapping rules
INFO:FederationManager:Successfully updated the OIDC RP Federation mapping rules
INFO:FederationManager:Configuring POC profile
INFO:FederationManager:Successfully configured POC profile

14.3 Testing the OpenID Connect flow

We are now ready to test the OpenID Connect configuration.

A RP(SP) initiated OIDC flow can be triggered using

https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/partner?Target=/isam/mobile-demo/diag

Trigger the flow using a browser.

If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by accessing the authorization endpoint.

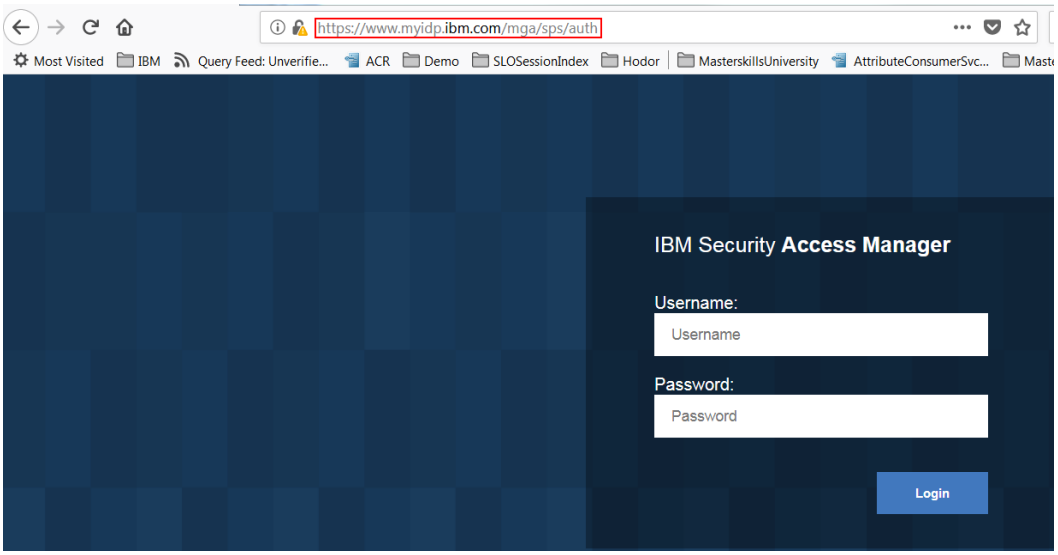
An example of the authorize URL
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?nonce=M7kbD9PnZc&redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp%2Fredirect%2Fpartner&response_mode=form_post&**claims=%7B%22id_token%22%3A%7B%22homePhone%22%3A%7B%22essential%22%3Atrue%7D%7D%2C%22userinfo%22%3A%7B%22displayName%22%3A%7B%22essential%22%3Atrue%7D%7D%7D**&scope=openid&response_type=code+id_token+token&state=gxt9W1Wpf4&client_id=clientID

The following claims are requested

{"id_token":{"homePhone":{"essential":true}},"userinfo":{"displayName":{"essential":true}}}

The authorize endpoint then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.

At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDl0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran a hybrid flow, the OP will send code, access_token and id_token as a response.

Inspect the id_token using https://jwt.io/, verify that the homePhone is returned.

IBM Security

Encoded PASTE A TOKEN HERE

eyJraWQiOiI0eHZRY2VGVU5OX1l2WERtb0ZRRWtf
VGtoMTVEOHdTS2xvWTV6LVd6UWM4IiwiYWxnIjoi
UlMyNTYifQ.eyJub25jZSI6Ijl1a2NsSG1PWWEiL
CJob21lUGhvbmUiOiI1NTUtMTIzNDUiLCJpYXQiO
jE1NDQxNzA4NDEsImlzcyI6Imh0dHBzOi8vd3d3L
m15aWRwLmlibS5jb20iLCJhdF9oYXNoIjoiOXRVZ
nZPc2IwdkM3NFhodTRaZERidyIsInN1YiI6InRlc
3R1c2VyIiwiZXhwIjoxNTQ0MTc0NDQxLCJjX2hhc
2giOiJ2MUswdDZIdU9NWlBoSlBzS0hSc1h3IiwiY
XVkIjoiY2xpZW50SUQifQ.EhqeN1fb5d-
gz0UFXCLbSD1rkwAg2d99xpdEASm4dNOgbpKDFD-
BC32aefROo_jVeaGSNGBOo43G207IdXBWF5Qnntr
6j-Yj-sClh9KtfcIbgfcA8fqpbaN_keE-
0Qcmk5z6vEpWCXHADzlsSggxFuP7ucrzW3Yojebq
RzOD6fI

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "4xvQceFUNN_YvXDmoFQEk_Tkh15D8wSKloY5z-WzQc8",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "nonce": "9ukclHmOYa",
  "homePhone": "555-12345",
  "iat": 1544170841,
  "iss": "https://www.myidp.ibm.com",
  "at_hash": "9tUfvOsb0vC74Xhu4ZdDbw",
  "sub": "testuser"
  "exp": 1544174441,
  "c_hash": "v1K0t6HuOMZPhJPsKHRsXw",
  "aud": "clientID"
}
```


To verify the access_token use the Userinfo endpoint, we could use a browser extension for a REST tool or use postman to make this request.

```
curl --request GET \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo \
  --header 'authorization: Bearer bytgVJZsycvBAP1JNJ62' \
  --cookie 'AMWEBJCT!%252Fmga!JSESSIONID=00007t9wM8_slegs0YGxELNKi5e%3A71b43363-316b-4790-baf
```

Userinfo endpoint - https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo
Header – Auhtorization: Bearer <Access Token>

Verify that the displayName is returned.

Response


```
{
  "sub": "testuser",
  "displayName": "Test User"
}
```

15 Configuring OpenID Connect Dynamic client registration

OpenID Connect (OIDC) publishes a specification that allows registration of a client to an OpenID Connect Provider.

This enables someone to onboard their application to an OpenID Connect provider through a standard well-formed API. See the specification https://openid.net/specs/openid-connect-registration-1_0.html.

The primary information that an application administrator is required to provide is the redirect URI that the application uses when requesting an identity.

Note: The script requires an access_token as input, please use a valid access_token generated as a part of testing the OIDC flow, in section 24.3.

SCRIPT-START:
Run this script: **DynamicClientRegistration.py -configure All -Bearer <Access Token>**

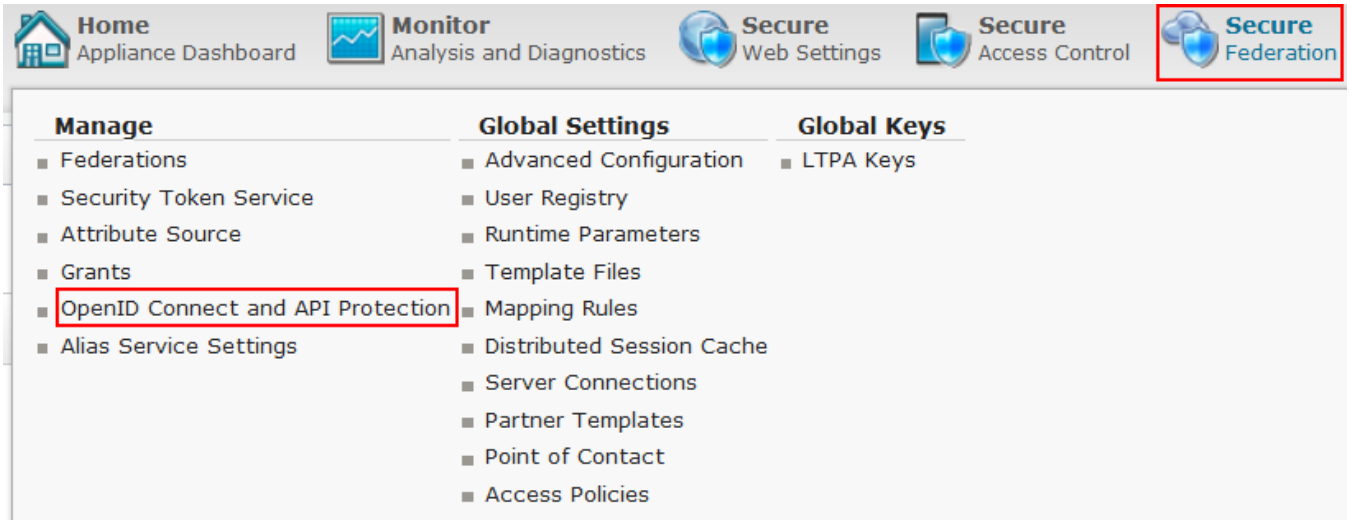
If you use this script, skip to the corresponding SCRIPT-END notice

15.1 Registering a client

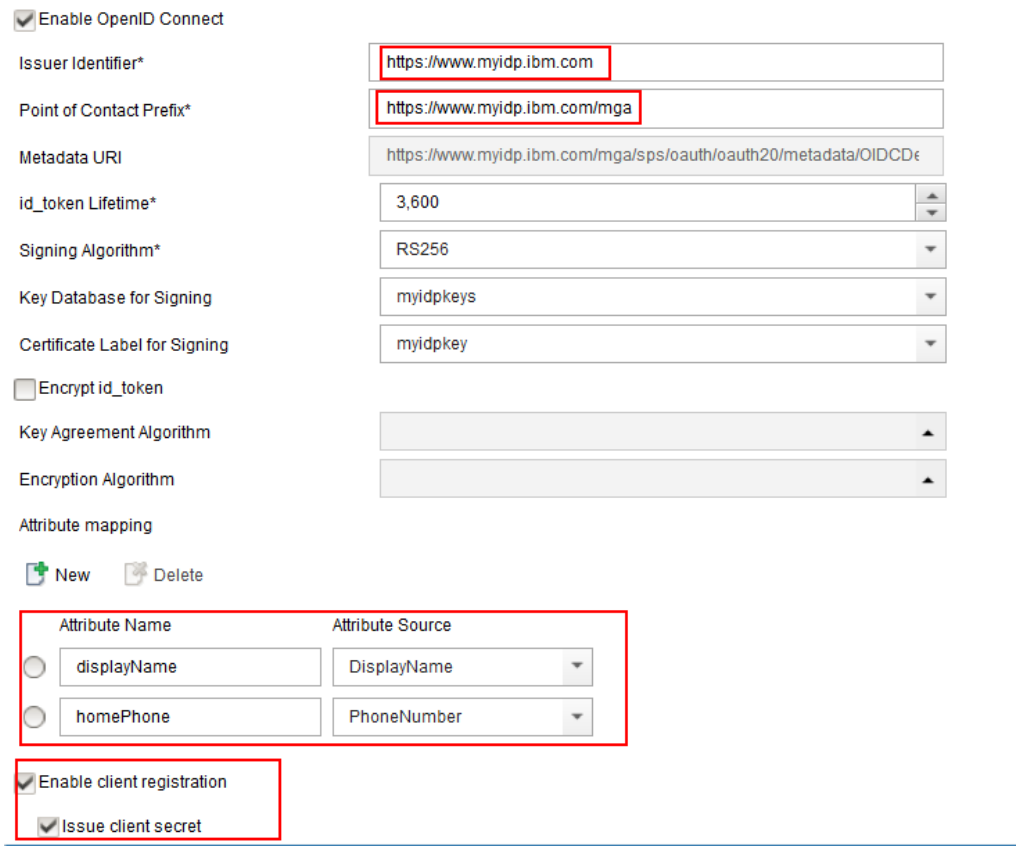
In this section, we enable client registration for the OIDCDefinition and we register a new dynamic client.

15.1.1 Enabling Client Registration

Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**



Click **Edit** to create a new API Definition



Make sure the checkbox **Enable client registration** and **Issue client secret** are enabled**.**

15.1.2 Client Registration

To register a client, issue a HTTP POST to the Client Registration Endpoint.

```
curl --request POST \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/register/OIDCDefinition \
  --header 'accept: application/json' \
  --header 'authorization: Bearer ajsCxAPqcIhYAtqCAAEX' \
  --header 'content-type: application/json' \
  --data '{"redirect_uris": ["https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/redirect/dynamic_partner"],"company_name":"IBM Applications"}'
```

The curl command above serves as an example

Enter **https://www.myidp.ibm.com/mga/sps/oauth/oauth20/register/OIDCDefinition** for the URL value.

Set **Accept** and **Content-Type** header to application/json. We need to provide an access token for the authorization header, retrieve the access token from a successful OIDC single sign-on flow as mentioned in section 22.3.

The payload sent to the registration endpoint needs to include a redirect_uri, which is set to **https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/redirect/dynamic_partner** and company name set to **IBM Applications**.

The response to the POST

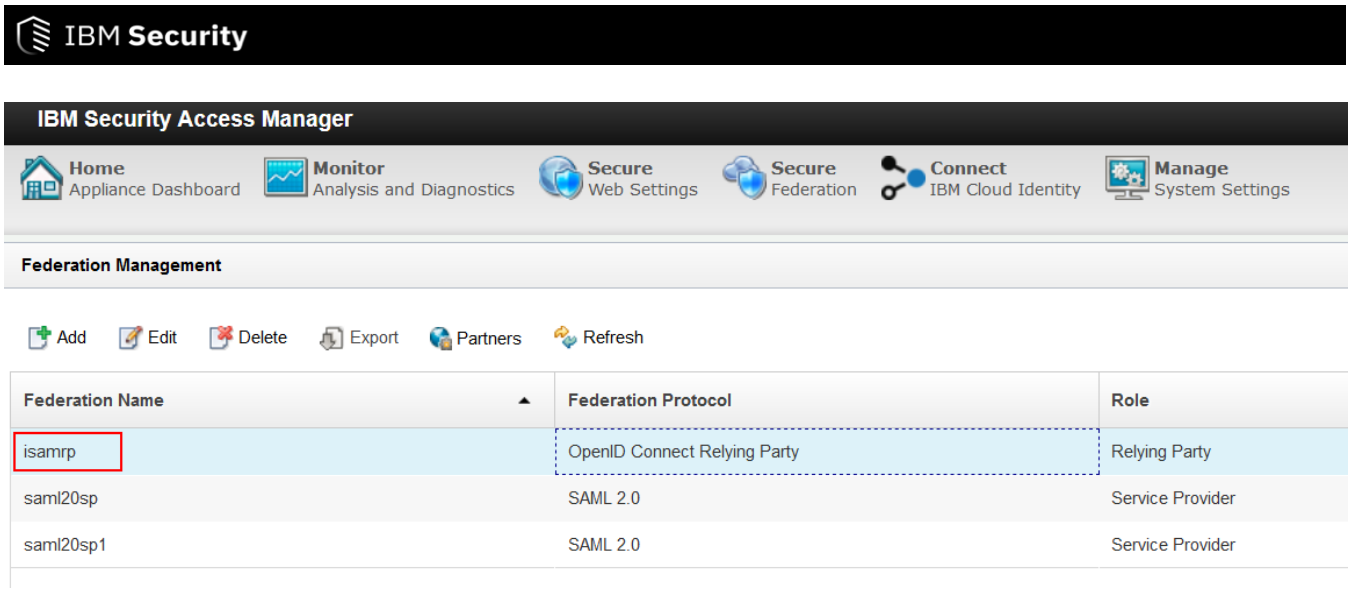

```
{
    "client_secret_expires_at": 0,
    "owner_username": "testuser",
    "company_name": "IBM Applications",
    "registration_client_uri": "https:\/\/www.myidp.ibm.com\/mga\/sps\/oauth\/oauth20\/register\/OIDCDefinition?client_id=8ijMHcbpp8NLeY0CTUjk",
    "client_secret": "GShxNRu3Y4kdP5GrXHNL",
    "client_id_issued_at": 1544063920,
    "redirect_uris": ["https:\/\/www.mysp.ibm.com\/isam\/sps\/oidc\/rp\/isamrp\/redirect\/dynamic_partner"],
    "registration_access_token": "YUvLxRZ8gwcSLZsdS1fl",
    "client_id": "8ijMHcbpp8NLeY0CTUjk"
}
```


We need to take note of the client_id and client_secret to create the corresponding OIDC relying party partner.

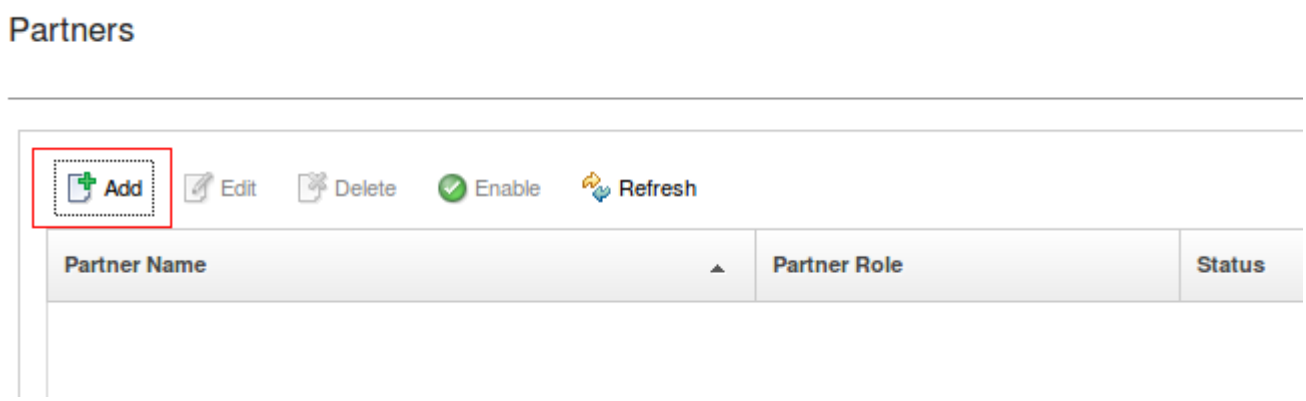
15.2 Configuring OIDC Relying Party Partner

Using the dynamic client information, we create a new relying party partner to an existing OIDC RP federation.

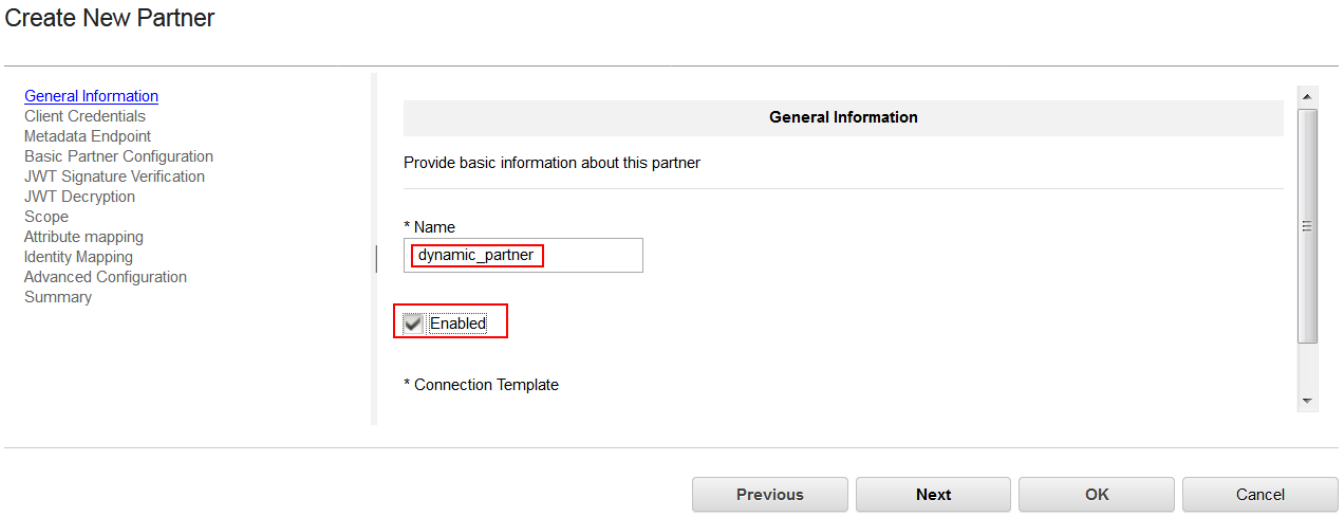
Under **Secure Federation** menu, click **on Manage: Federations**.



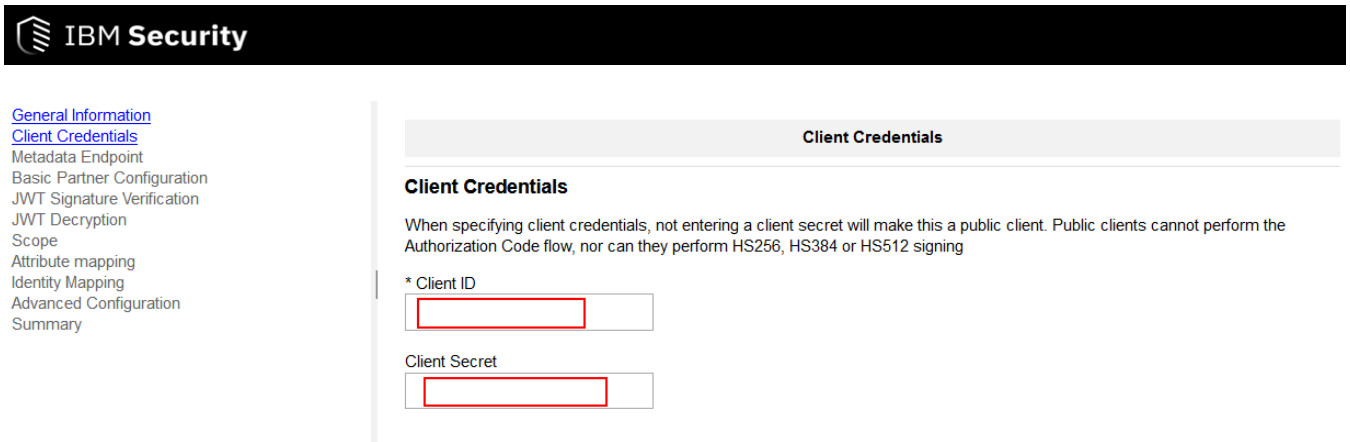
Select the **isamrp** federation and click **Partners**.



Click on **Add** to configure OIDC RP Partner.

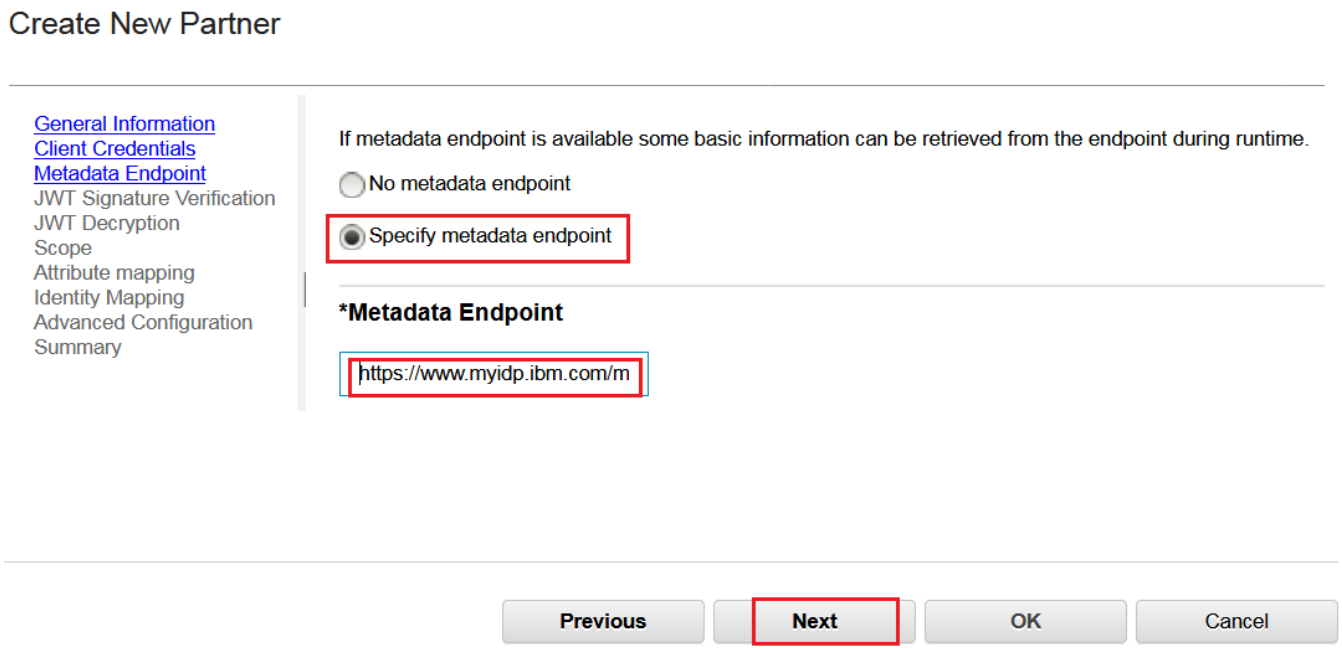


Enter **dynamic_partner** as the name of the RP Partner and select the **Enabled** checkbox.



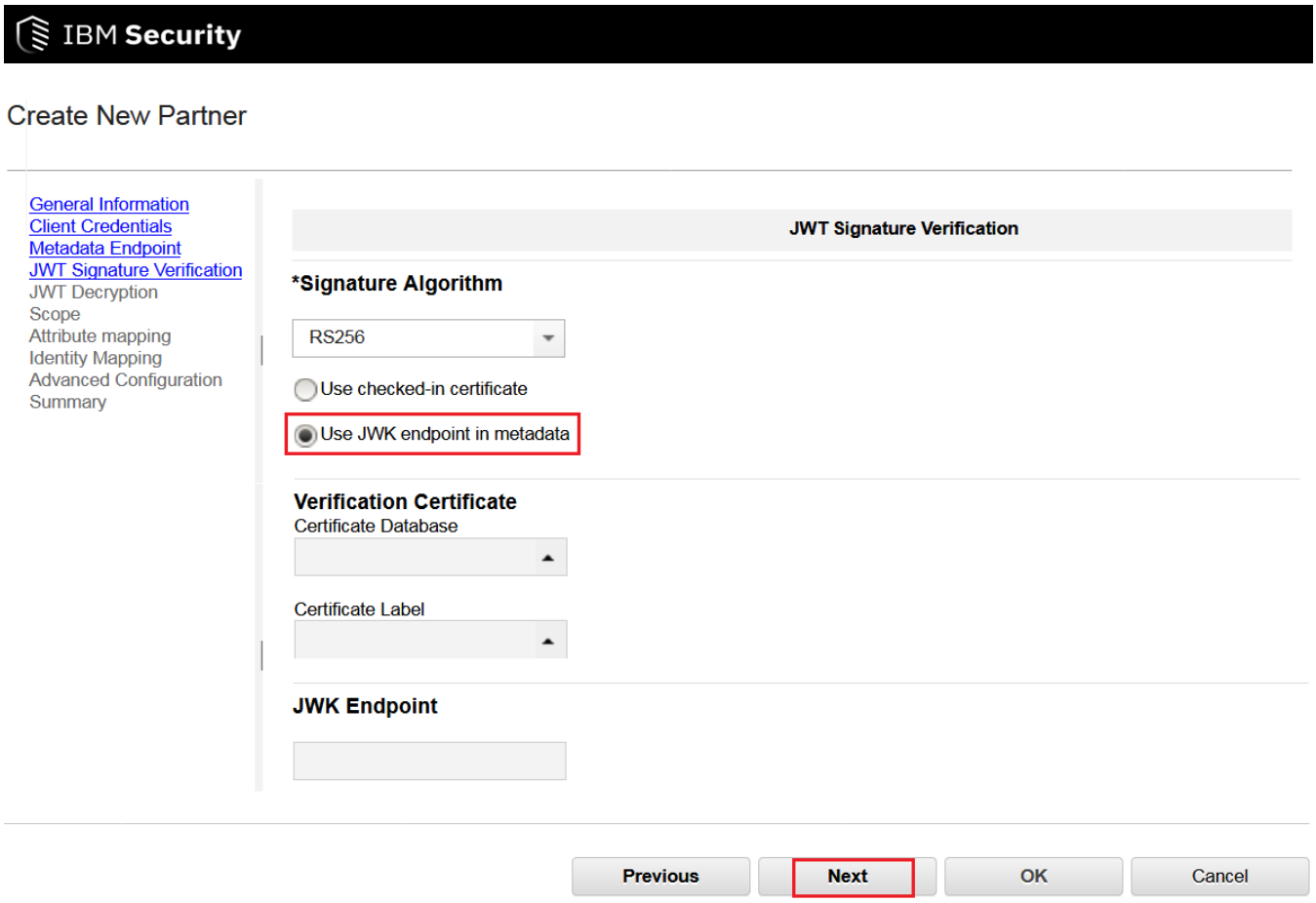
Enter Client ID and Client Secret based on the post response retrieved in section 28.1.2

Click on **Next**.



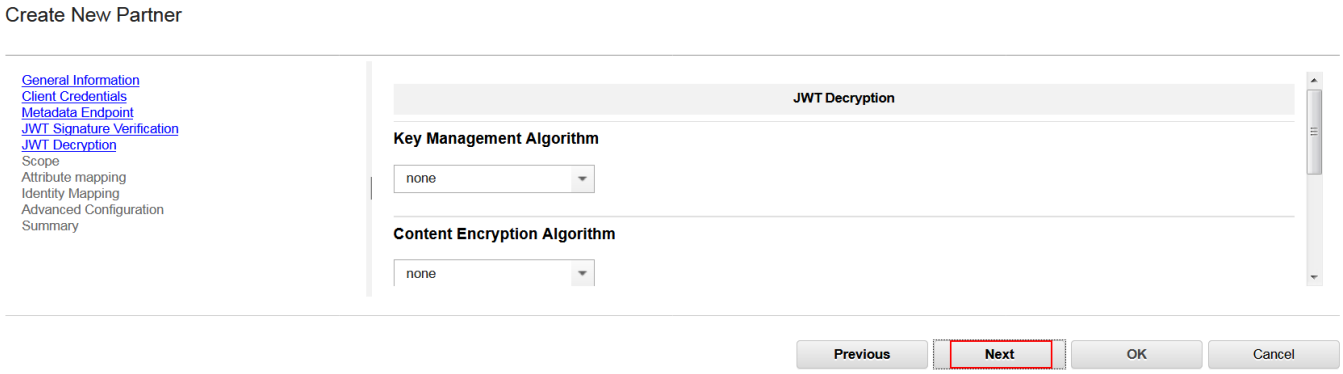
On the Metadata Endpoint panel, select **Specify metadata endpoint** radio button. Enter https://www.myidp.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCDefinition as the Metadata Endpoint.

Click on **Next.**



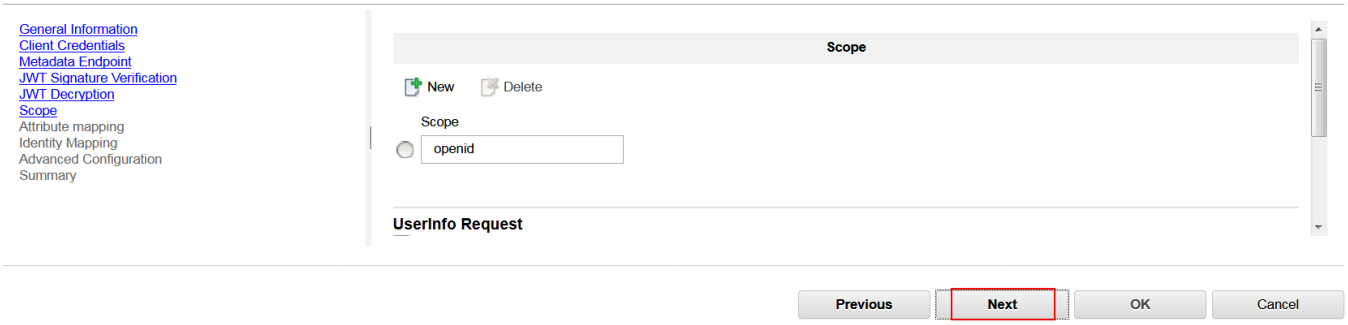
On the **JWT Signature Validation** panel, select **Use JWK endpoint in metadata.**

Click on **Next.**



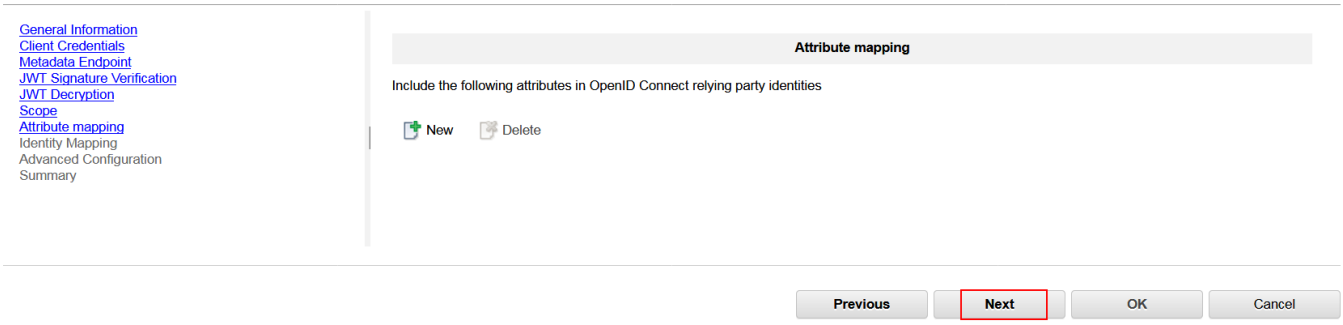
We don't need to change JWT Decryption so click **Next**.

Create New Partner



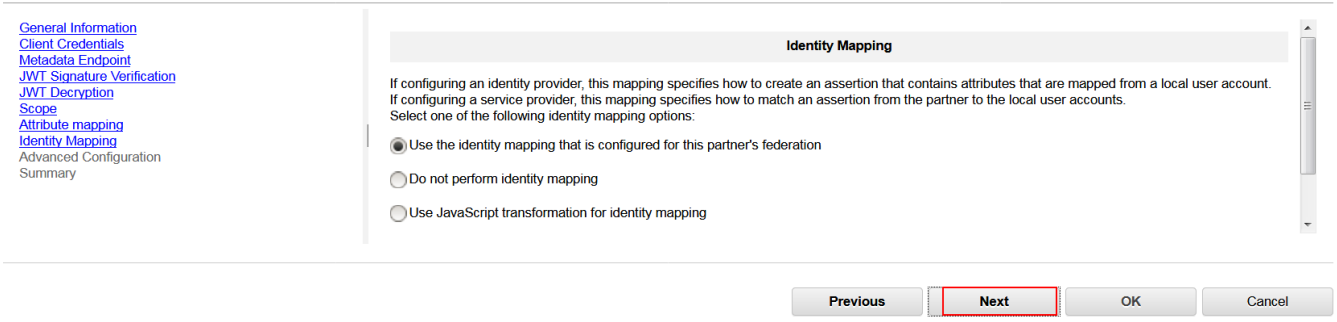
We don't need to change Scopes so click **Next**.

Create New Partner

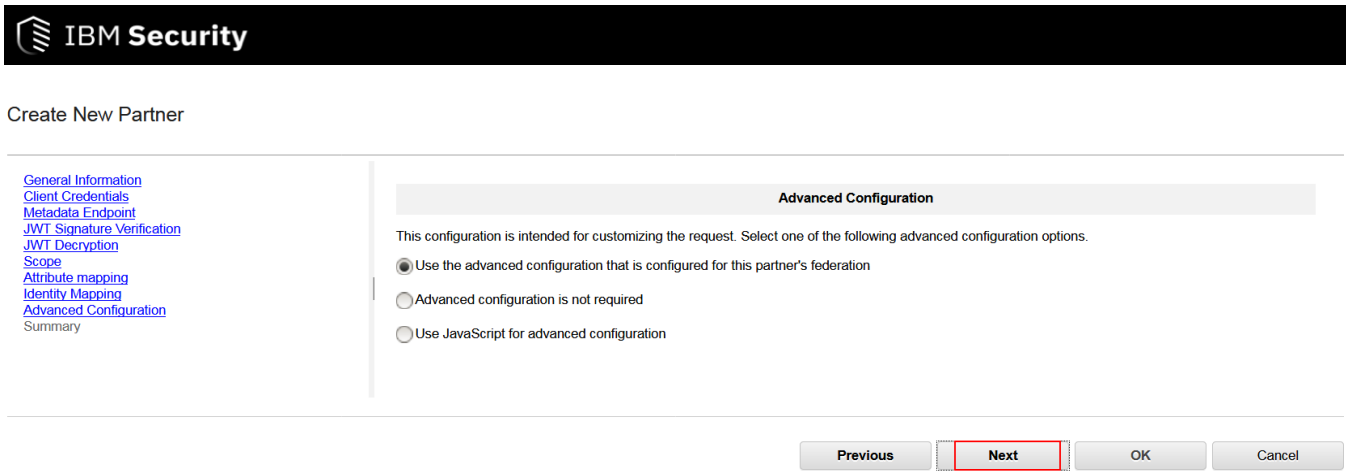


We don't need to change Attribute Mapping so click **Next**.

Create New Partner



We don't need to change Identity Mapping so click **Next**.



We don't need to change Advanced Configuration so click **Next**.

Click OK on the Summary panel.
Follow on-screen instructions to **deploy** pending changes.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Registering Client
INFO:FederationManager:Successfully registered client
INFO:FederationManager:Configuring the OIDC RP Partner
INFO:FederationManager:Successfully configured the OIDC RP Partner
INFO:FederationManager:Configuring POC profile
INFO:FederationManager:Successfully configured POC profile

15.3 Testing the OpenID connect Single Sign-On flow

We are now ready to test the OpenID Connect configuration.

A RP(SP) initiated OIDC flow can be triggered using

https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/dynamic_partner?Target=/isam/mobile-demo/diag

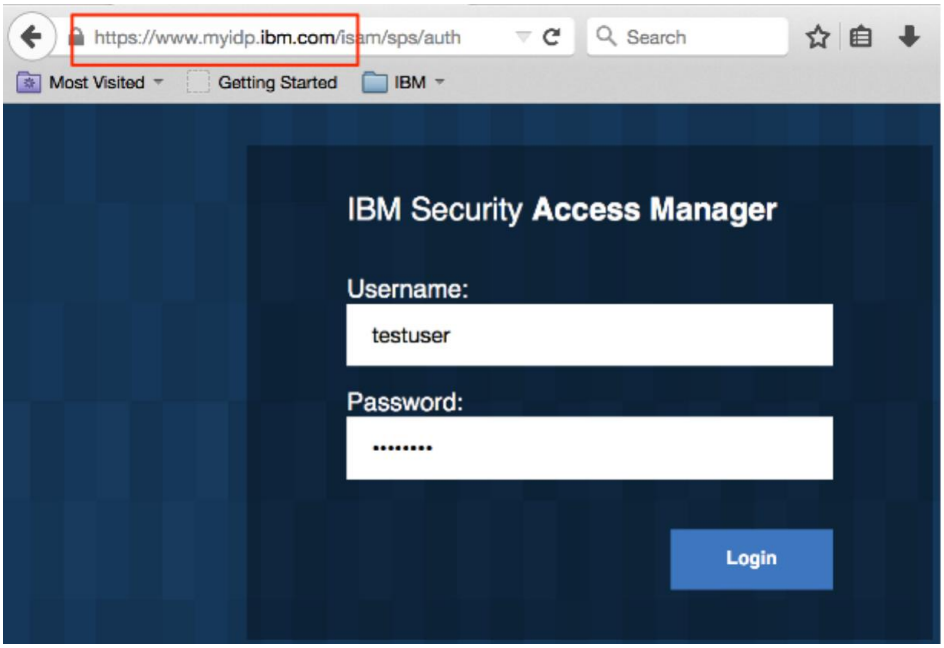
Trigger the flow using a browser.

If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by hitting the authorize URL.

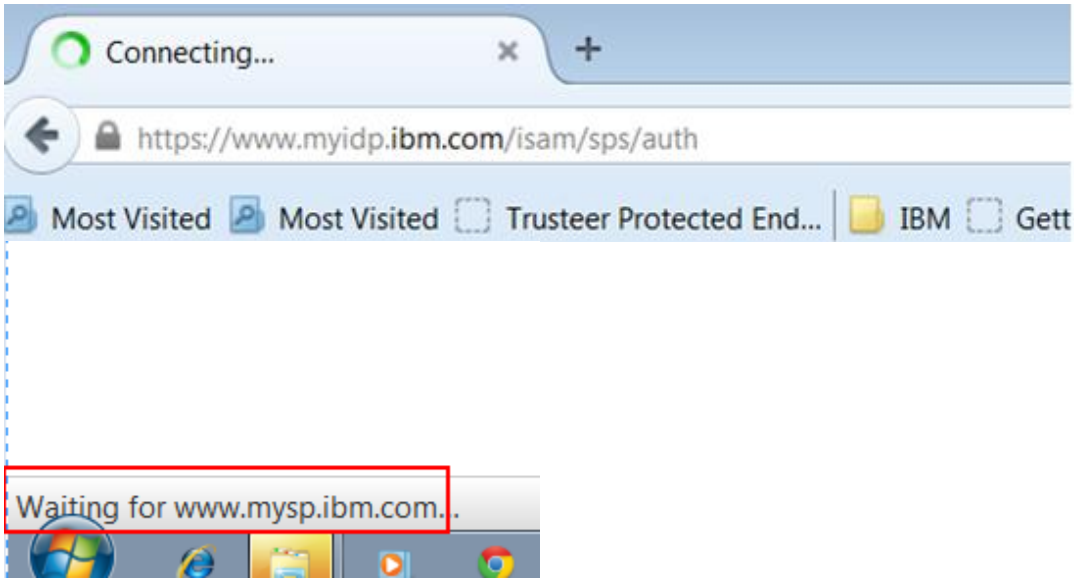
An example of the authorize URL
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?nonce=p1PQnPpTn8&redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp%2Fredirect%2Fdynamic_partner&response_type=code+id_token+token&response_mode=form_post&state=Zz9YmAsMZq&scope=openid&client_id=IptNGxJZCl6wVFvkAgrr

The authorize url then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.



At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDI0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran a hybrid flow, the OP will send code, access_token and id_token as a response.

Inspect the id_token using https://jwt.io/

Encoded PASTE A TOKEN HERE

eyJraWQiOiI0eHZRY2VGVU5OX1l2WERtb0ZRRWtfVGtoMTVEOHdTS2xvWTV6LVd6UWM4IiwiYWxnIjoiUlMyNTYifQ.eyJub25jZSI6InAxUFFuUHBUbjgiLCJpYXQiOjE1NDQwNjYwMjIsImlzcyI6Imh0dHBzOi8vd3d3Lm15aWRwLmlibS5jb20iLCJhdF9oYXNoIjoiMWZtbzg1dE9feGpsWVdvUmllcFNYZyIsInN1YiI6InRlc3R1c2VyIiwiZXhwIjoxNTQ0MDY5NjIyLCJjX2hhc2giOiI4RnVuY1c5QjZHU0xuSHl4SnV6aUVBIiwiYXVkIjoiSXB0Tkd4SlpDbDZ3VkZ2a0FncnIifQ.X1pIz5tVlS7CJwYgfJBYFT7hotJeKAeEyuCZFlNmMc5SeuOV3C_ztWbxa3Xybm-9lwT4smUxc2SodosdkwTYcLyXN6k4u_wOyTVLHd5IrXSu2FCEwEbBU4iqTAGJrBwcxA4D76QLaAgfKe--jdgyukztro2NqX62qAUMK6dyAWM

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "4xvQceFUNN_YvXDmoFQEk_Tkh15D8wSKloY5z-WzQc8",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "nonce": "p1PQnPpTn8",
  "iat": 1544066022,
  "iss": "https://www.myidp.ibm.com",
  "at_hash": "1fmo85tO_xjlYWoRiepSXg",
  "sub": "testuser",
  "exp": 1544069622,
  "c_hash": "8FuncW9B6GSLnHyxJuziEA",
  "aud": "IptNGxJZCl6wVFvkAgrr"
}
```


To verify the access_token use the Userinfo endpoint, we could use a browser extension for a REST tool or use postman to make this request.

```
curl --request GET \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo \
  --header 'authorization: Bearer CRaSsmobyYSobQTx0E5y' \
```

Userinfo endpoint - https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo
Header – Auhtorization: Bearer <Access Token>

16 Introduction to the Security Token Service (STS)

The security token service is a component of the federation runtime that accepts WS-Trust XML/SOAP requests for the validation and exchange of one security token type for another. It is used by the federation runtime, and can also be used by standalone WS-Trust clients, or by the ISAM reverse proxy for “TFIM-SSO” junctions.

A lot has been written about the security token service from Tivoli Federated Identity Manager and, for the most part, the STS in ISAM 9 offers the same runtime capabilities.

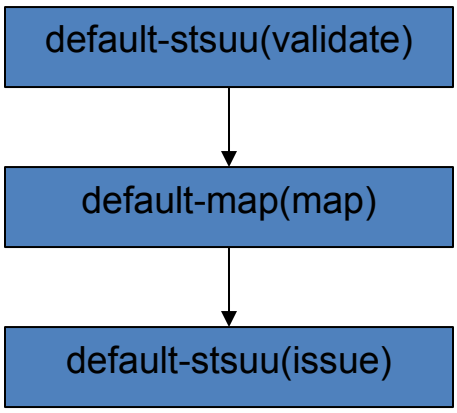
What is particularly different in ISAM 9 is the configuration model – configuration is available via REST API, or via a user interface in the appliance administration console (LMI or local management interface).

In this section we will create a simple chain in the STS to map one XML-based security token to another, with a simple javascript mapping rule performing some attribute manipulation. We will then invoke that STS chain using simple cUrl commands with XML files representing the request. Much of what we do here is similar to this technical article on the TFIM STS:

http://www.ibm.com/connections/blogs/sweeden/entry/using_curl_to_send_requests_to_the_tfim_security_token_service6

This chain is not particularly useful, however understanding how to configure and invoke it is very useful when we start to look at more advanced use cases of using the STS in later sections of this cookbook.

The trust chain we will configure uses a template format depicted below:



The STSUU, or STSUniversalUser, is a simple XML format token that contains collections of attributes. It is the common format used to hold identity data within the STS. An example STSUU is shown here:

```
<stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
  <stsuuser:Principal>
    <stsuuser:Attribute name="name">
      <stsuuser:Value>john</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:Principal>
  <stsuuser:AttributeList />
</stsuuser:STSUniversalUser>
```

The mapping module will be javascript code that will add an extra attribute to the STSUU.

```
importPackage(Packages.com.tivoli.am.fim.trustserver.sts);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.uuser);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);

// demo rule for mapping stsuu to stsuu
// just add an additional attribute
var testAttr = new Attribute(
  "testattr_from_auxilary_chain",
  "urn:mytype",
  "myvalue_from_auxilary_chain");
stsuu.addAttribute(testAttr);

// and clear out the RST attributes
stsuu.getRequestSecurityTokenAttributeContainer().clear();
```

The resulting STSUU after the mapping is applied will look like this:

```
<stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
  <stsuuser:Principal>
    <stsuuser:Attribute name="name">
      <stsuuser:Value>john</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:Principal>
  <stsuuser:AttributeList>
    <stsuuser:Attribute name="testattr_from_auxilary_chain" type="urn:mytype">
      <stsuuser:Value>myvalue_from_auxilary_chain</stsuuser:Value>
    </stsuuser:Attribute>
  </stsuuser:AttributeList>
</stsuuser:STSUniversalUser>
```

All of our configuration will be on the Identity Provider system although to some extent this is an arbitrary choice since there is an STS running on the Service Provider system too.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **STSTest.py –configure All**

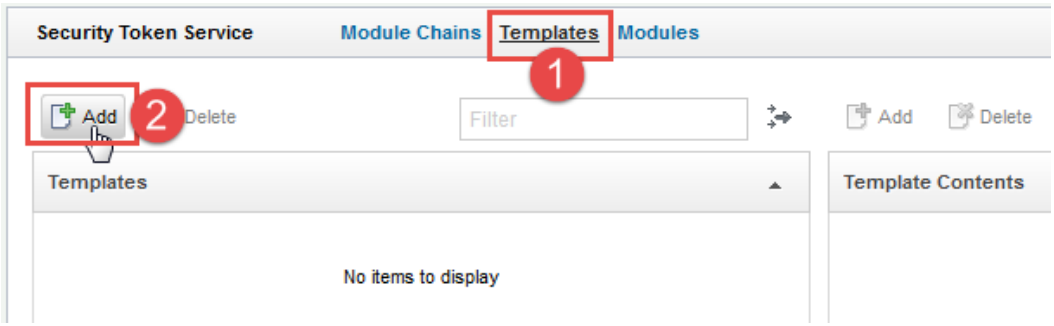
If you use this script, skip to the corresponding SCRIPT-END notice

16.1 Configuring the "STSUU to STSUU" Chain Template

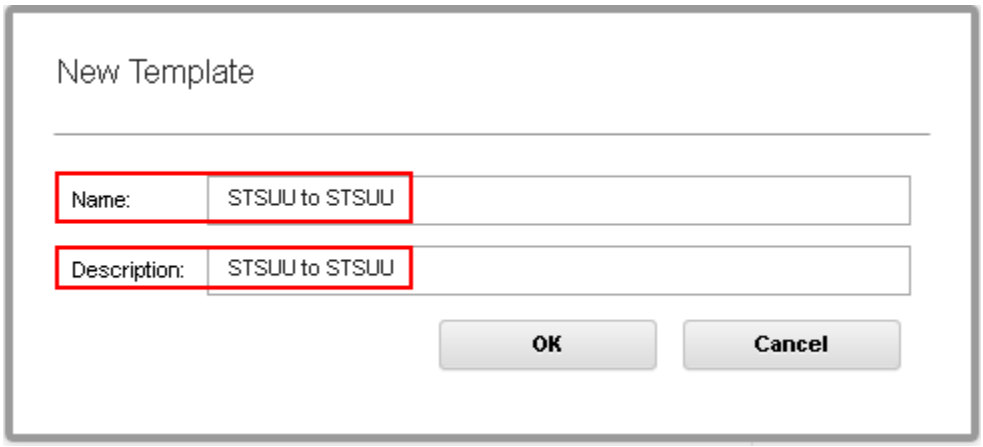
First we create an STS Chain Template. This defines an ordered list of Modules that will make up any chain built on this template.

Access the Identity Provider LMI console at https://isam.myidp.ibm.com and authenticate with **admin** and **Passw0rd**.

Navigate to **Secure Federation→Manage: Security Token Service.**



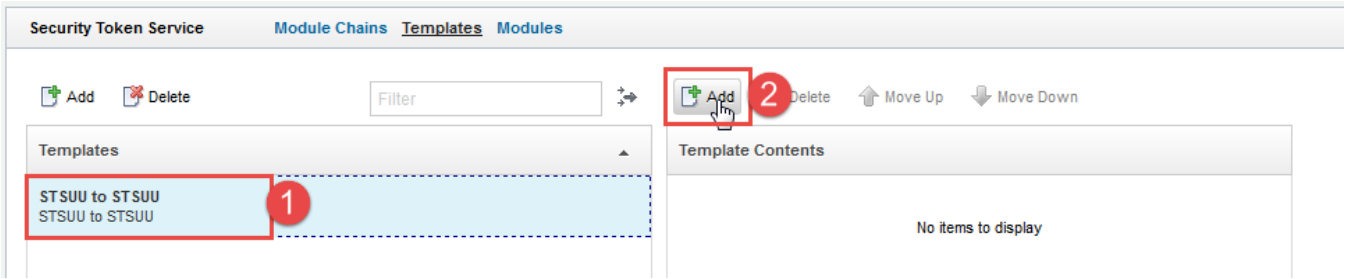
Click on the **Templates** menu and then click the **Add** button to create a new template.



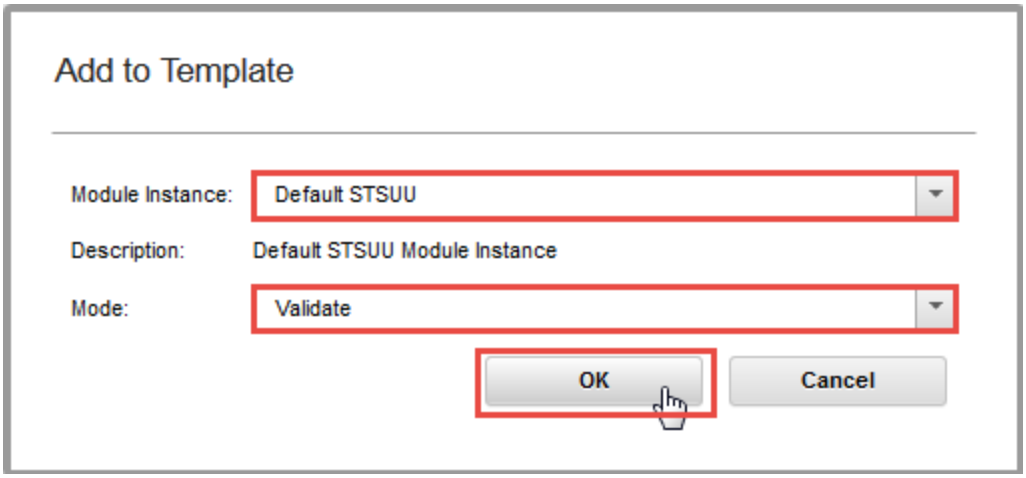
Provide a Name and a suitable Description for the template then click **OK**.

Deploy the pending changes.

Now the Template has been created we to need to populate it by adding modules to the Template Contents.

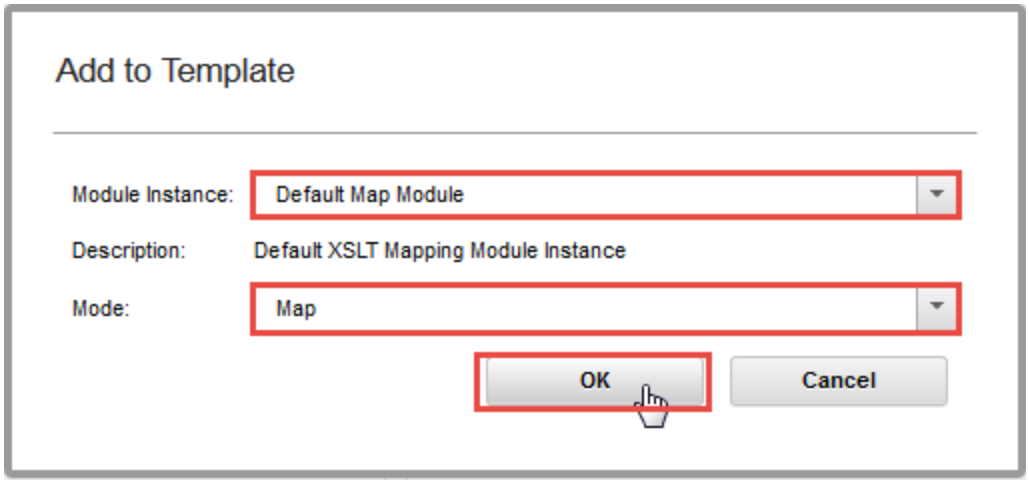


Select the new template and click **Add** on the right-hand panel to add a Module Instance to the template contents.



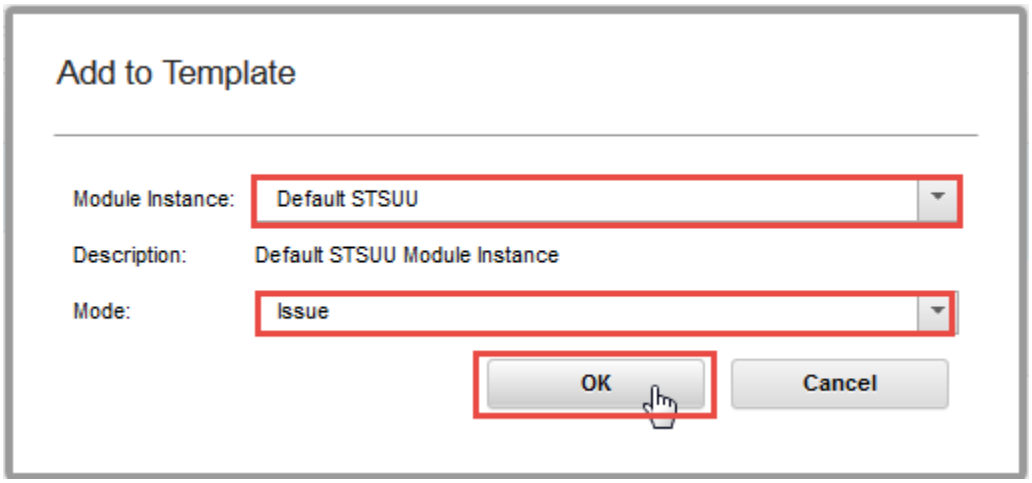
First we're going to add a **Default STSUU** Module Instance in **Validate** mode. This will process the incoming STSUU XML token and create an internal STSUU object for processing. Enter these values and click **OK**.

Click **Add** again to add a second Module Instance.



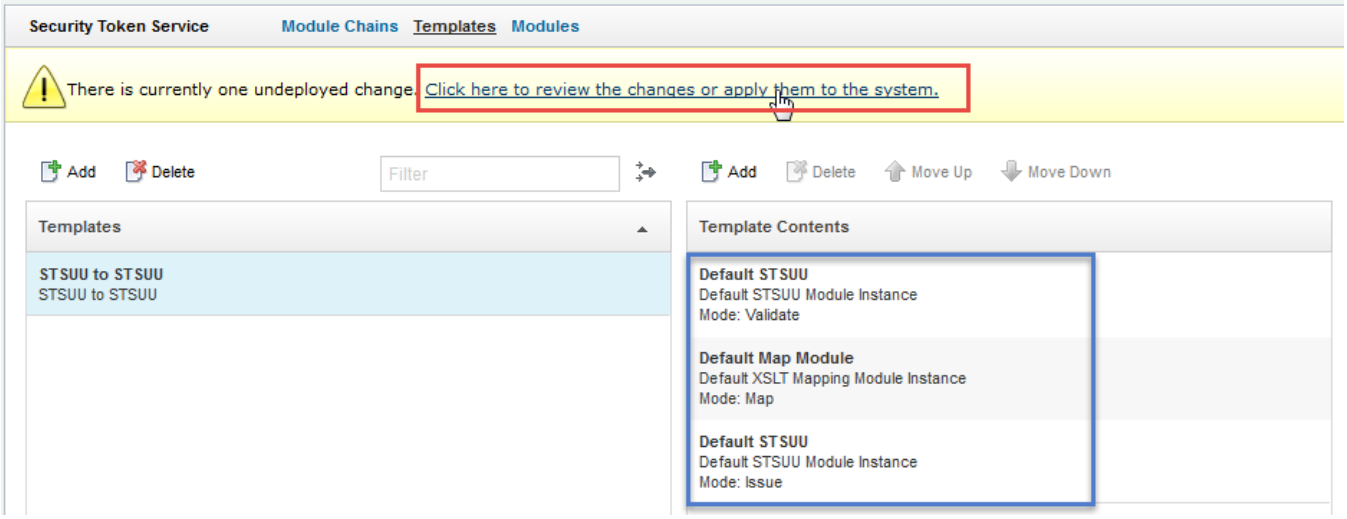
Now we add a **Default Map Module** in **Map** mode. This will run JavaScript code which can process the internal STSUU object created by the first module. Enter these values and click **OK**.

Click **Add** one more time.



The final module in the Template will be another **Default STSUU** module but, this time, in **Issue** mode. This module will create an STSUU XML token to be returned to the caller. Enter these values and click **OK**.

The template contents should look like this:

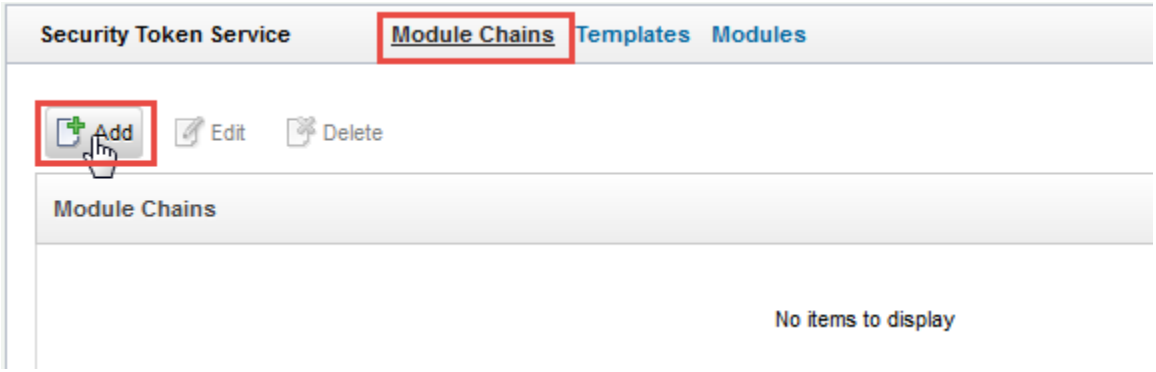


The template is complete. **Deploy** the changes.

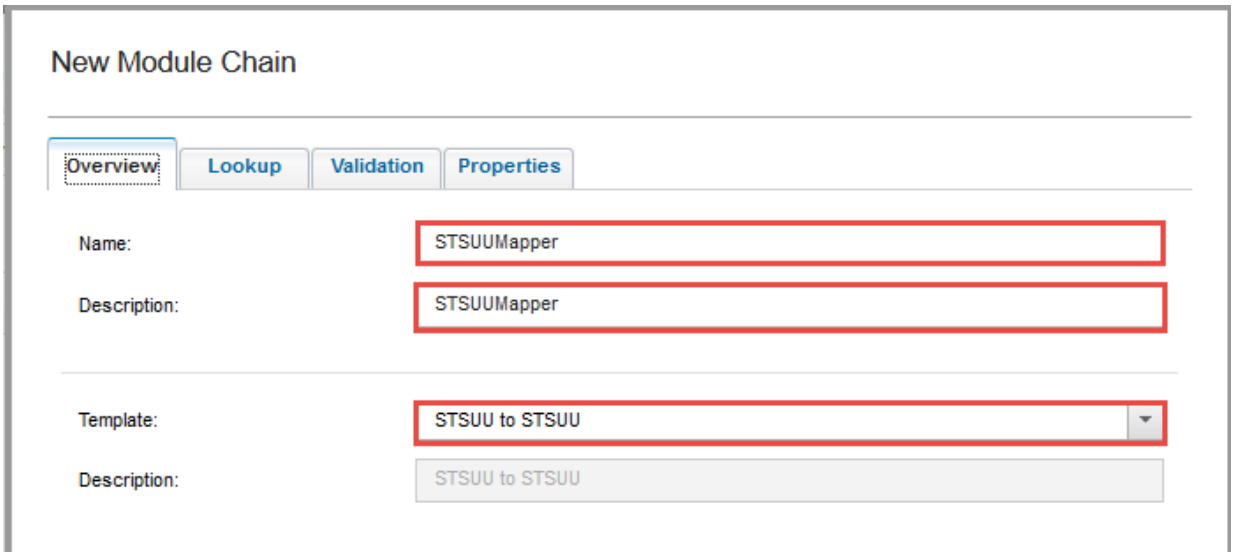
16.2 Configuring the "STS Test" Module Chain

We will now create a Module Chain from the new template.

As we're seen, the template determines the modules in the chain, the mode they will operate in, and the order in which they will run. The rest of the configuration is specified at the Module Chain level.

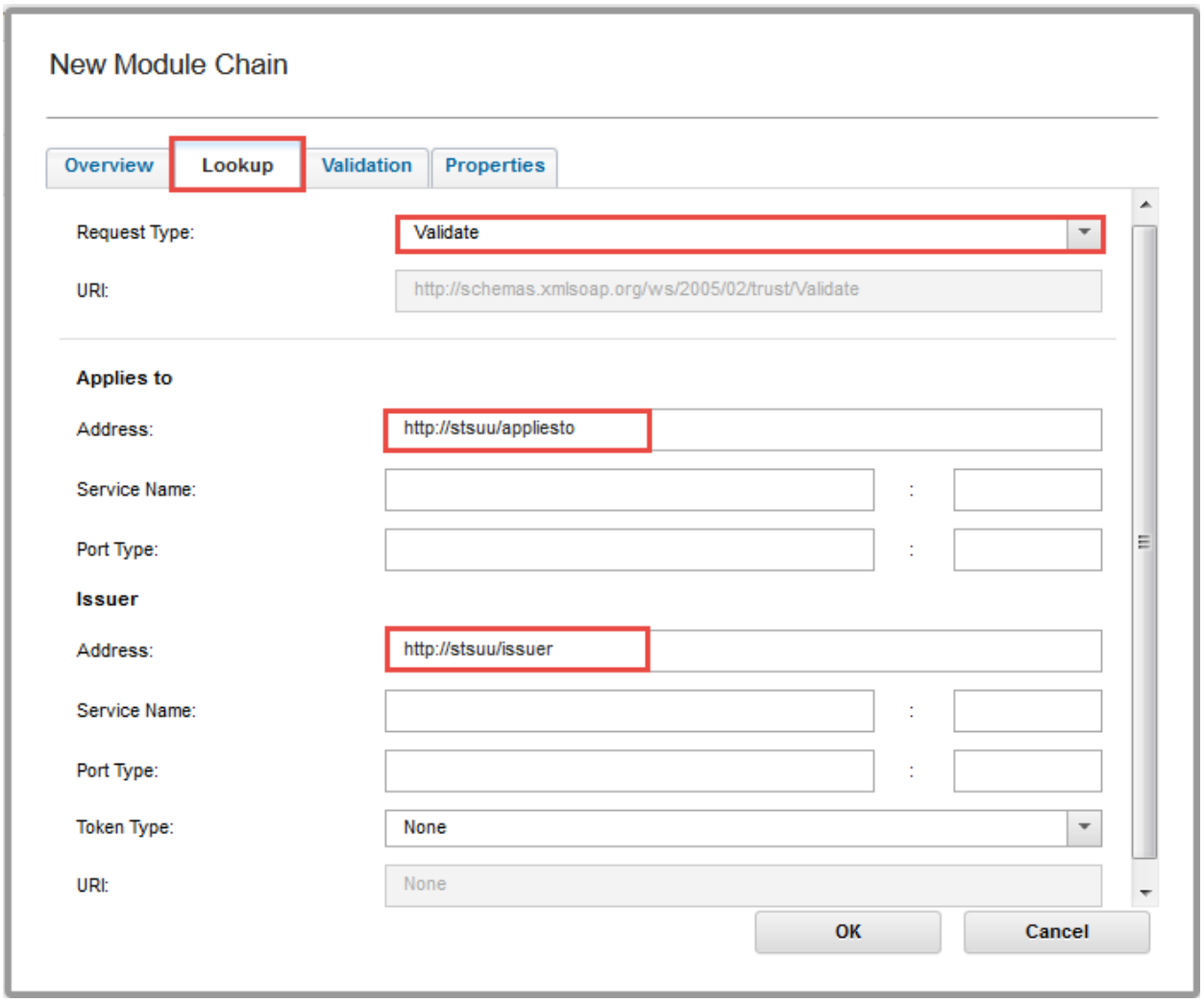


Still in the Security Token Service screen, click on the **Module Chains** tab and then click **Add** to add a new Module chain.



Enter **STSUUMapper** as the Name for the chain and provide a description. Select the **STSUU to STSUU** Template for the chain (this is probably already selected as it is the only template available).

Click on the **Lookup** tab.



When a WS-Trust request arrives at the STS, the information provided in the **Lookup** tab of each defined Module Chain is used to determine which one should process the request. Only if all specified Lookup fields match the incoming WS-Trust request is it considered a match.

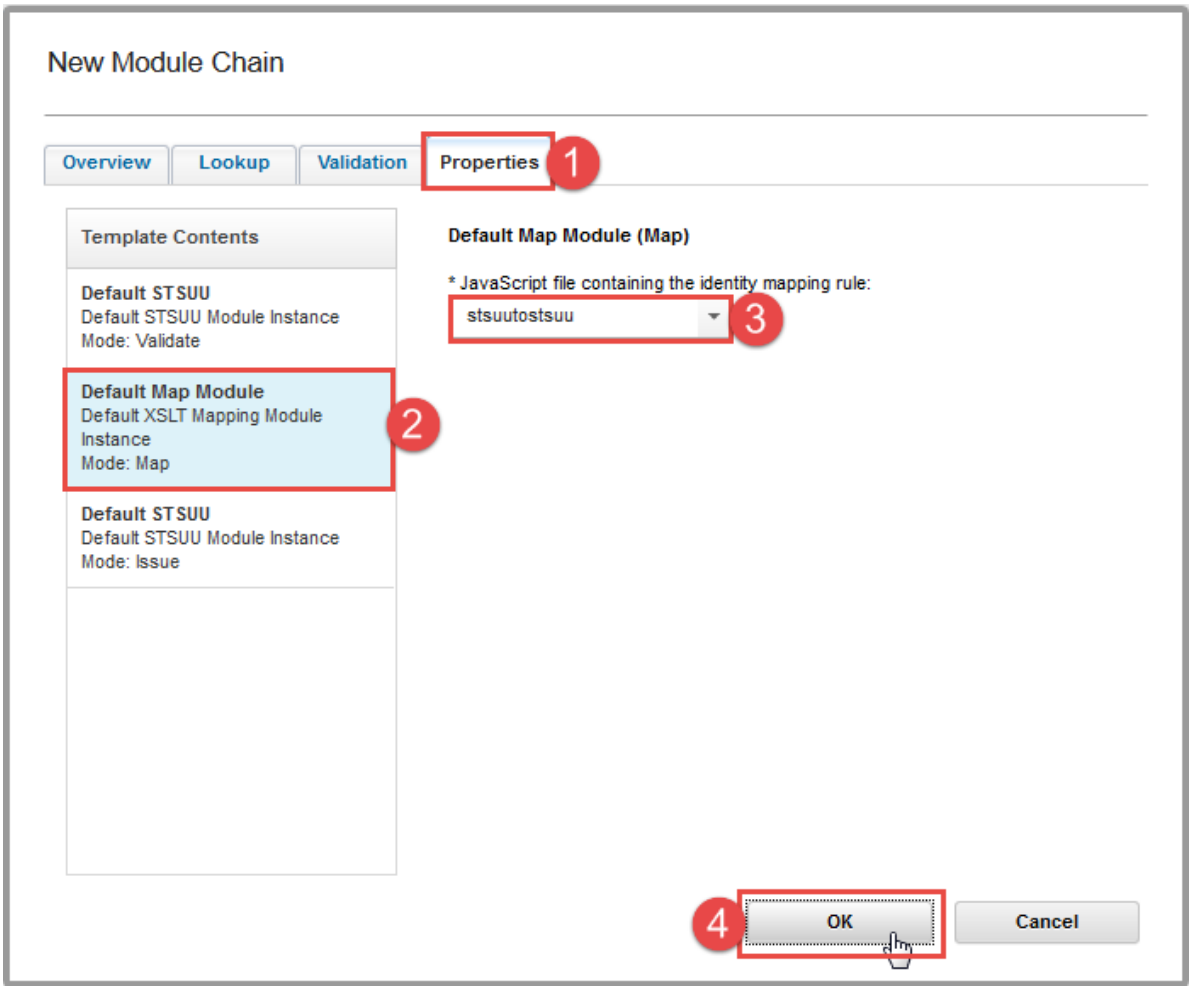
Select **Validate** from the *Request Type* drop-down list.

Enter **http://stsuu/appliesto** in the *Address* box under *Applies to*.
Enter **http://stsuu/issuer** in the *Address* box under *Issuer*.

Click on the **Validation** tab.

The validate tab provides configuration for validating the signatures on incoming WS-Trust requests and for signing outgoing WS-Trust responses. We will not be using these capabilities here.

Click on the **Properties** tab.



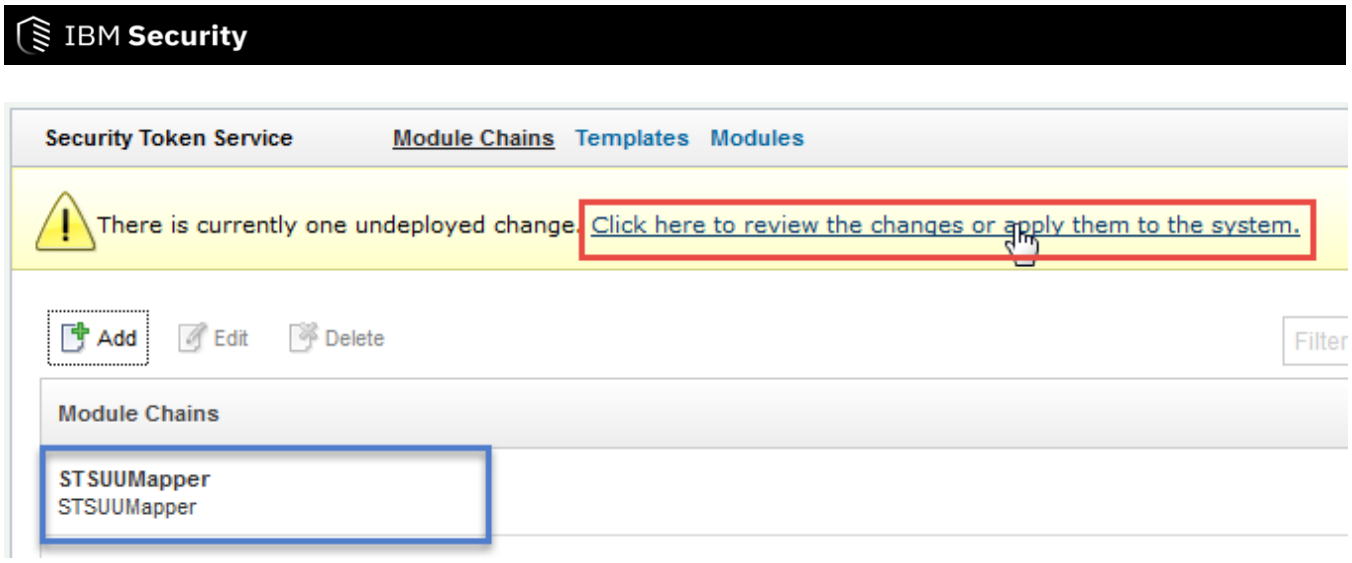
In the Properties tab, we configure the chain-specific properties for each module in the chain.

The STSUU module does not have any chain-specific properties so we don't need to worry about them.

Select the **Default Map Module** from the list of modules on the left-hand side. This opens the properties panel for that module.

The only thing that we need to specify for the mapping module is which JavaScript file should be used. Select **stsuutostsuu** from the drop-down list.

Module Chain configuration is now complete. Click **OK** to save the new Module Chain.



The new chain is shown in the list of Module Chains. **Deploy** pending changes.

> If you want to view the *stsuutostsuu* mapping rule (or any other mapping rule) these can be found in the GUI console under **Secure Federation→Global Settings: Mapping Rules**.

16.3 Allowing access to the STS via the ISAM Reverse Proxy

By default, the Federation Runtime listens only on the 127.0.0.1 loopback interface of the ISAM appliance, and is therefore not directly accessible from outside. However, the Federation Runtime (of which the STS is a part) is accessible via the Reverse Proxy through the **/isam** junction that was created during our federation configuration. We just need to set an Access Control policy to allow access.

In order to allow external clients (such as a cUrl script running on our host machine) to access the STS WS-Trust endpoint via the Reverse Proxy we will attach an "unauthenticated-allowed" to them.

> In a production system it is likely that you would limit access to the STS to authorized clients and then implement an authentication mechanism such as Basic Auth or Client Certificates. We allow unauthenticated access here for simplicity.

Open an SSH session to the IdP appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

Authenticate with **admin** and **Passw0rd**.

Navigate to **isam** and start the **admin** utility:

```
isam.myxx.ibm.com> isam
isam.myxx.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console:

```
pdadmin> login -a sec_master -p Passw0rd
```

Enter the following commands to create and attach an unauthenticated ACL:

```
acl create sts-unauth
acl modify sts-unauth set group iv-admin TcmdbsvaBRrxl
acl modify sts-unauth set group webseal-servers Tgmdbsrxl
acl modify sts-unauth set user sec_master TcmdbsvaBRrxl
acl modify sts-unauth set any-other Tr
acl modify sts-unauth set unauthenticated Tr
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/TrustServer/SecurityTokenService sts-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/TrustServerWST13/services/RequestSecurityToken sts-
unauth
```

This will allow cUrl to reach both the WS-Trust 1.2 and WS-Trust 1.3 endpoints of the federation runtime without having to provide ISAM reverse proxy authentication credentials.

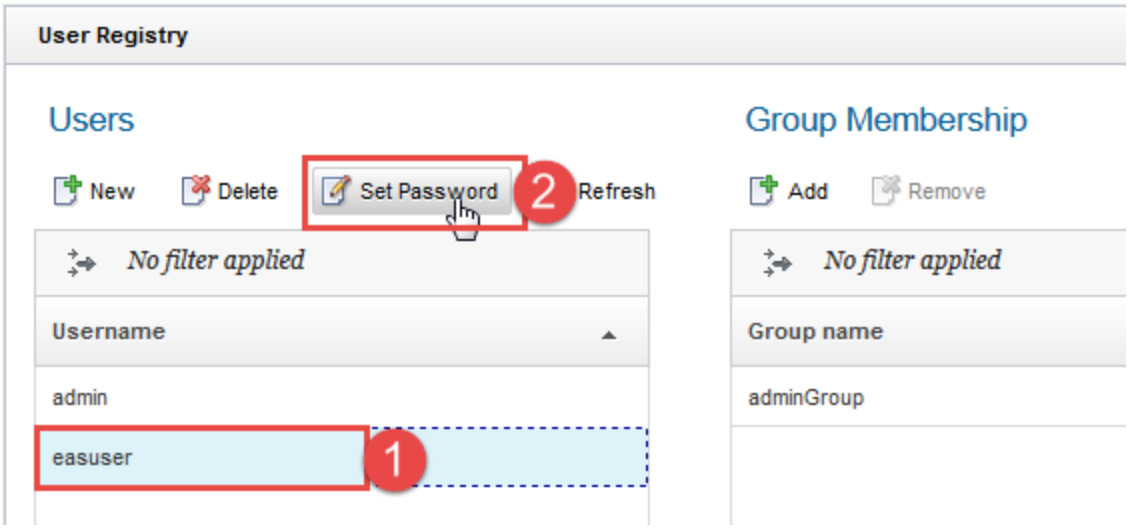
Type **exit** twice to end the session.

16.4 Updating the easuser password

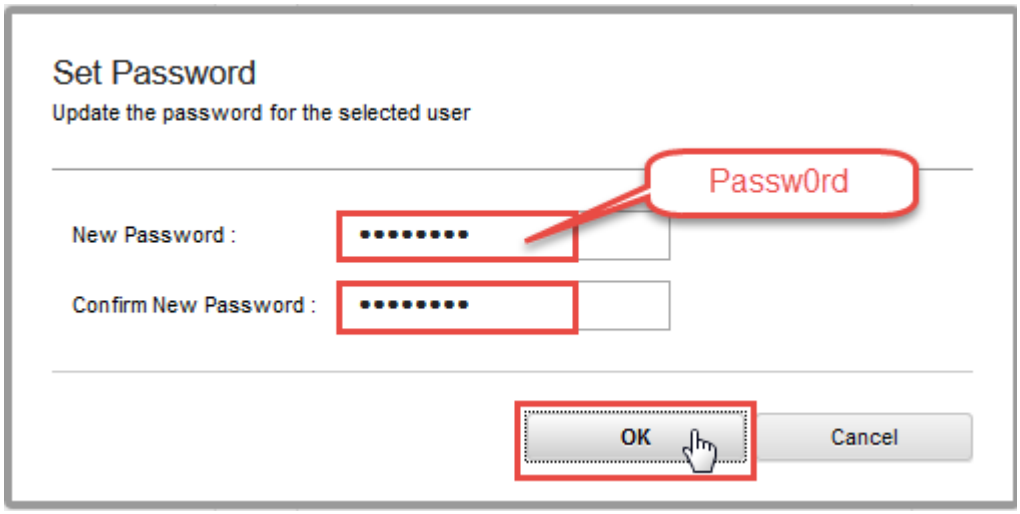
The federation runtime has its own authentication requirements for access to the STS endpoints, and this is provided by the federation runtime user registry. This built-in registry (which is independent to the LDAP registry used by the Reverse Proxy) includes a preconfigured user called “easuser” which has a default password of “passw0rd”. The easuser is typically used in ISAM reverse proxy configuration to allow it to be a client of the STS, and we will see this in action later in the document.

For now, we will change the easuser password to “Passw0rd”, for consistency with other passwords used throughout this cookbook, and so that you can see where and how this is done.

In the LMI, navigate to **Secure Federation -> User Registry**.



Select **easuser** and click **Set Password**



Enter **Passw0rd** in both entry boxes and click **OK**.

Deploy pending changes.

SCRIPT-END:
The script should display the following:
INFO:STSTest:Configuring the test STS chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:BaseManager:Configuring the easuser password
INFO:BaseManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the STS Module Chain Template
INFO:FederationManager:Successfully configured the STS Module Chain Template
INFO:FederationManager:Configuring the STS Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the STS Module Chain Mapping
INFO:STSTest:Successfully configured the test STS chain

You can test access to the STS now, with the following cUrl command:

```
curl -kv -u "easuser:Passw0rd" -H "Accept: application/xml"
https://www.myidp.ibm.com/isam/TrustServer/SecurityTokenService
```

You should see a 200 OK response, with text indicating that you have accessed the web service:

```
…
<h2>/SecurityTokenService</h2>
<h3>Hello! This is a CXF Web Service!</h3>
…
```

16.5 Invoking the STS Test chain with cUrl

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint. Both WS-Trust 1.2 and WS-Trust 1.3 formats are supported, and there are separate endpoints for each service version.

The *…/providedfiles/ststest* directory includes example SOAP messages, and both UNIX shell scipts and Windows batch scripts to invoke cUrl with the correct parameters for both WS-Trust 1.2 and 1.3. These are **rst12.sh** and **rst13.sh** respectively (or **rst12.bat** and **rst13.bat** for Windows)

These scripts require **cUrl** and **xmllint** commands to be installed on the system and available in the path.

The rst12.sh example is reproduced here:

```
$ ./rst12.sh
…
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"/>
  <soap:Body>
    <wst:RequestSecurityTokenResponse xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="uuidb5111432-014f-1475-a3e5-ad7b70207dca">
      <wsp:AppliesTo xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference>
          <wsa:Address>http://appliesto/stsuu</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:RequestedSecurityToken>
        <stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
          <stsuuser:Principal>
            <stsuuser:Attribute name="name" type="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
              <stsuuser:Value>john</stsuuser:Value>
            </stsuuser:Attribute>
          </stsuuser:Principal>
          <stsuuser:AttributeList>
            <stsuuser:Attribute name="testattr_from_auxilary_chain" type="urn:mytype">
              <stsuuser:Value>myvalue_from_auxilary_chain</stsuuser:Value>
            </stsuuser:Attribute>
          </stsuuser:AttributeList>
          <stsuuser:RequestSecurityToken/>
          <stsuuser:ContextAttributes/>
          <stsuuser:AdditionalAttributeStatement/>
        </stsuuser:STSUniversalUser>
      </wst:RequestedSecurityToken>
      <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
      <wst:Status>
        <wst:Code>http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid</wst:Code>
      </wst:Status>
    </wst:RequestSecurityTokenResponse>
  </soap:Body>
</soap:Envelope>
```

Notice the extra attribute included in the STSUU.

This concludes the basic STS chain example. Next we will look at more advanced uses of the STS in the context of federations and other ISAM Reverse proxy functions.

17 Advanced Federation Mapping Rules

The SAML federation we have used thus far in the cookbook implements a simple JavaScript mapping rule at the Identity Provider to decide which attributes from the ISAM credential make it into the SAML Assertion.

Often more complex identity mapping may be required at an Identity Provider, such as sourcing additional attributes from LDAP, or from an external web service. Similarly at a Service Provider, more advanced mapping rules may wish to call out to an HR provisioning system.

In this section we will demonstrate two different ways a federation can utilize external callouts to third party services as part of identity mapping. The first technique will make use of a utility class that can be called directly from a JavaScript mapping module to perform HTTP(s) client operations in a very generic manner. The second technique will use a purpose-built mapping module that is able callout to a 3rd party web service in several defined formats.

Much of the demand for these types of capabilities comes from two key factors:

- The ISAM appliance does not permit you to upload your own custom STS modules as TFIM did
- Many customers have similar requirements related to being able to call out to their own web services, as part of a service-oriented architecture.

Finally we will combine the ability to do external callout with the use of an advanced STS chain which includes a new capability in ISAM 9 – the built-in LDAP attribute lookup module.

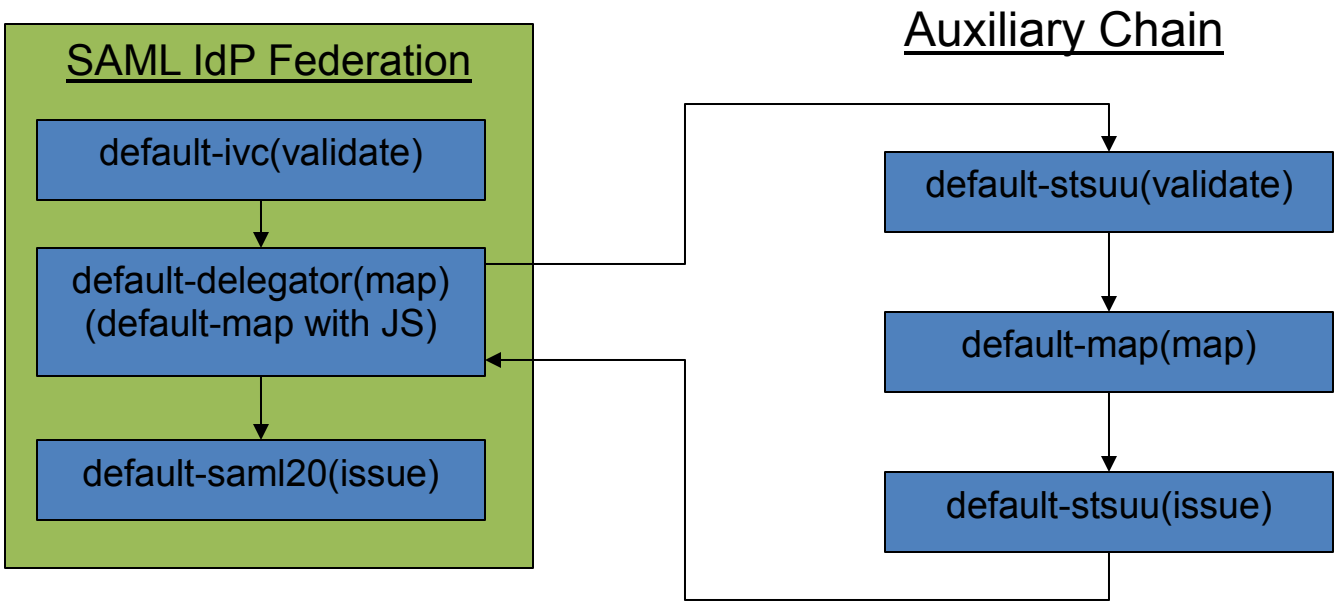
All of these capabilities will be demonstrated on the IdP image and, as a pre-requisite to this section, it is expected that you have a working SAML SSO federated relationship established through completing the earlier sections of this cookbook.

17.1 Using HttpClient from Javascript mapping rules

The HttpClient is a utility class that can be invoked from your Javascript mapping rule. As its name suggests, it is a generic HTTP(s) client that can perform HTTP methods to a URL endpoint, and can also deal with basic-auth and client certificate authentication requirements

The HttpClient has several different methods for HTTP operations, however in this scenario we will be using HTTP POST, and the external service we will be calling will be the federation runtime STS – in particular we will invoke the demonstration STS Test chain that was configured in the previous section. Note that we are only using the STS as the endpoint because it's something we already have available in the demonstration image – you could use the HttpClient to call out to any endpoint, for any HTTP GET/POST operation.

In essence, we will be modifying the JavaScript mapping rule of the federation to do this:



The resulting SAML assertion sent to the SP will contain the "testattr_from_auxilary_chain" attribute. Again, this is not particularly useful other than to demonstrate the use of the HttpClient code from a Javascript map module.

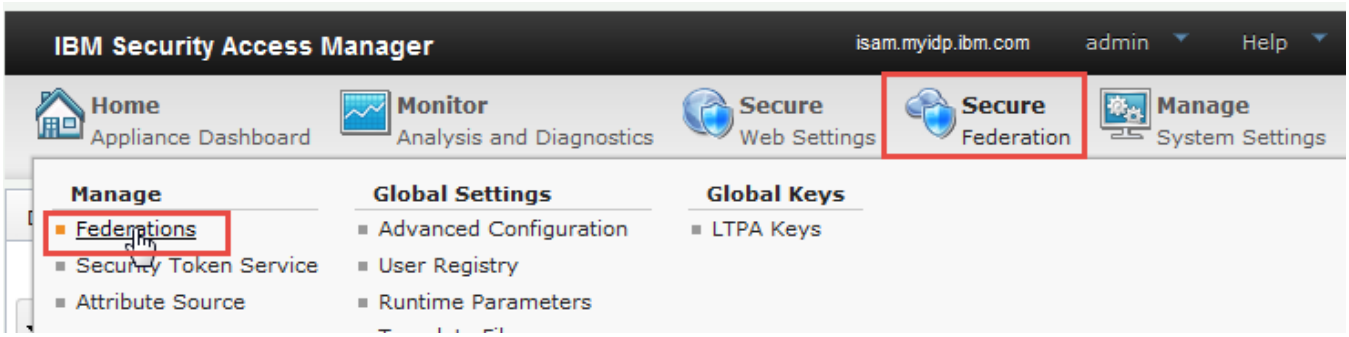
A copy of the HttpClient-enabled Javascript mapping rule can be found in the *…/providedfiles/mappingrules/idp* directory, called **ip_saml20_httpclient_wstrust.js**.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

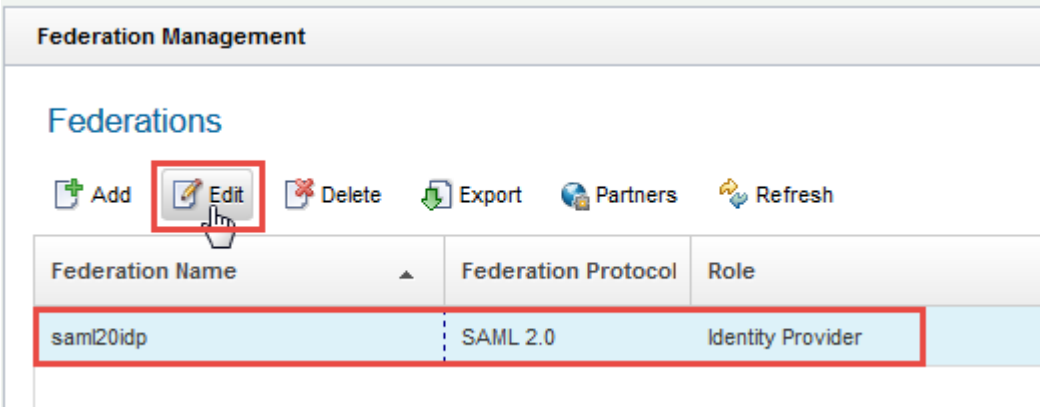
Run this script: **UploadIPMappingRule.py –configure HttpClientMappingRule**

If you use this script, skip to the corresponding SCRIPT-END notice

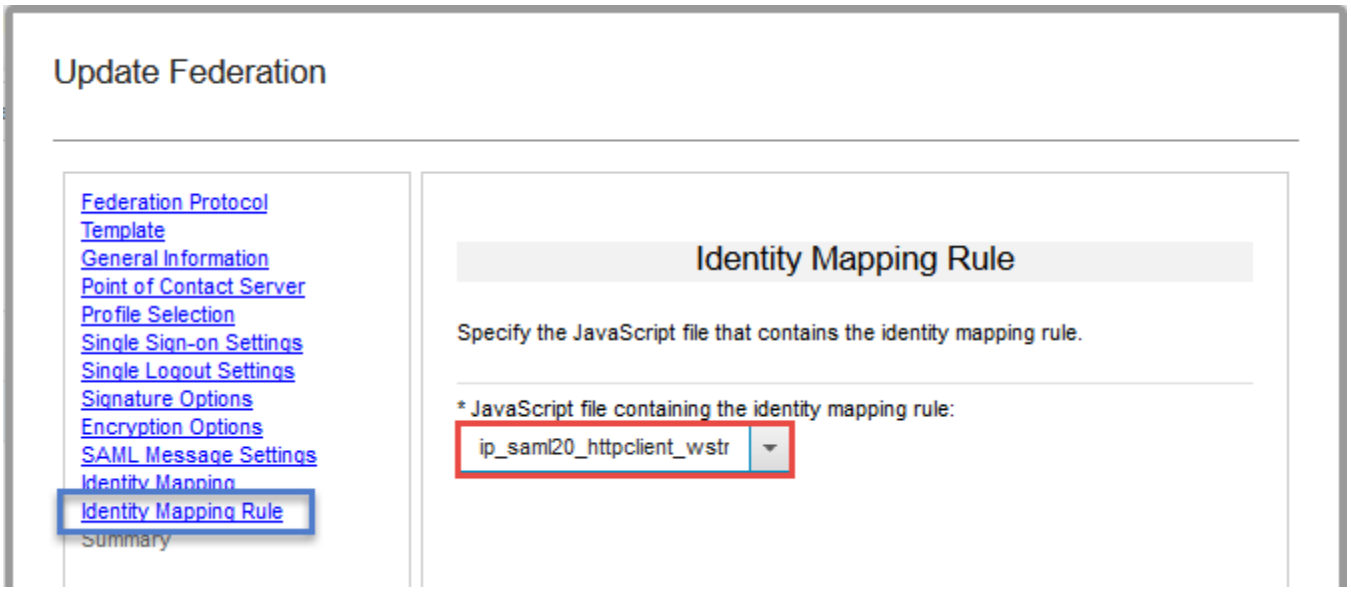
Access the Identity Provider LMI console at https://isam.myidp.ibm.com and authenticate with **admin** and **Passw0rd**.



Navigate to **Secure Federation→Manage: Federations**.



Since we have created only one federation which is **saml20idp**, select it and click **Edit**.



Click **Next** to move through the wizard until you get to the *Identity Mapping Rule* page.

Select **ip_saml20_httpclient_wstrust** as the *JavaScript file* from the drop-down list.

Click **Next** to show the summary page and then **OK** to complete the wizard.

Deploy the pending changes.

SCRIPT-END:
The script should display the following:
INFO:UploadIPMappingRule:Configuring the HttpClientMappingRule
INFO:FederationManager:Modifying IdP to change mapping rule
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully modified the Federation using PUT
INFO:UploadIPMappingRule:Successfully configured the HttpClientMappingRule

Now that the new mapping rule is in place, perform a SAML SSO. Use this trigger URL:

https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Fsaml20sp%2Fsaml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

Login to the IdP with **testuser** and **Passw0rd**.

Review the credential information on the SP diagnostics page and notice that the credential at the SP now contains the *testattr_from_auxilary_chain* attribute:

Access Manager Credential:
User: **testuser**

Name	Value(s)
lastName[0]	User
AuthenticationInstant[0]	2015-11-13T14:37:46Z
AZN_CRED_NETWORK_ADDRESS_BIN[0]	0xc0a82a01
AUTHENTICATION_LEVEL[0]	1
AZN_CRED_AUTH_METHOD[0]	ext-auth-interface
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_VkX1wQAAAAIAAAAwXX8AALieBVxdfwAAV3EwZTdvSnhlaHR5Y09CbDNaSUl2bjljTnB1Y0c5QzVQanBHcWxmLWZRUS1BZk9U:default
testattr_from_auxilary_chain[0]	myvalue_from_auxilary_chain
AuthenticationMethod[0]	urn:oasis:names:tc:SAML:1.0:am:password

This attribute was added at the Identity Provider via the mechanism we just set up, transferred to the SP in a SAML token, and then populated into the SP ISAM Credential.

17.2 Using the external http callout mapping module

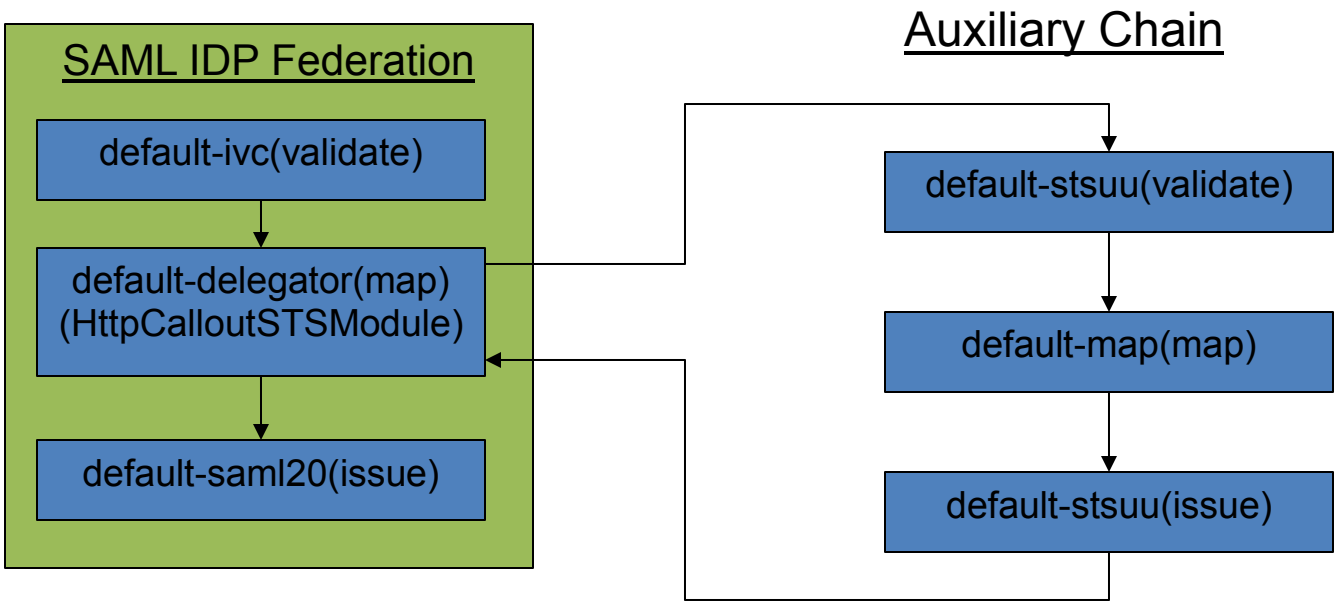
The HttpClient described in the previous section is a low-level interface for making external callouts, and requires a deal of JavaScript code to use it effectively. In many cases, particularly when calling out to an auxiliary STS chain, it would be better to have a purpose-built mapping module for that purpose.

In ISAM 9 we deliver that module – largely based on the popularity of the STSMap module developed as part of this TFIM article:

http://www.ibm.com/connections/blogs/sweeden/entry/compex_federation_identity_and_attribute_mapping_for_tivoli_federated_idenity_manager1

ISAM 9 includes a mapping module that is capable of calling out to external services in one of two formats – XML, and WS-Trust (1.2). In this section we will focus on the WS-Trust capability, and in particular will use the module as a direct replacement for the Javascript/HttpClient mapping module that was configured in the previous section.

In this section we will effectively configure this pattern:

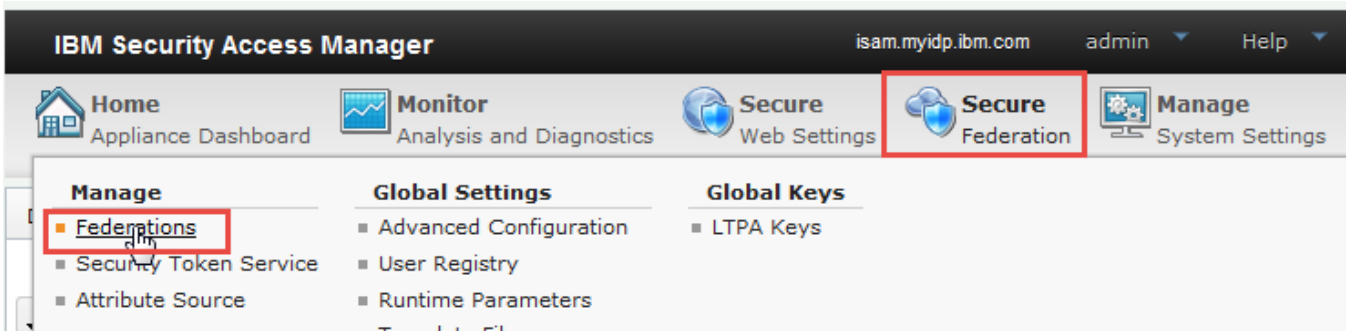


SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

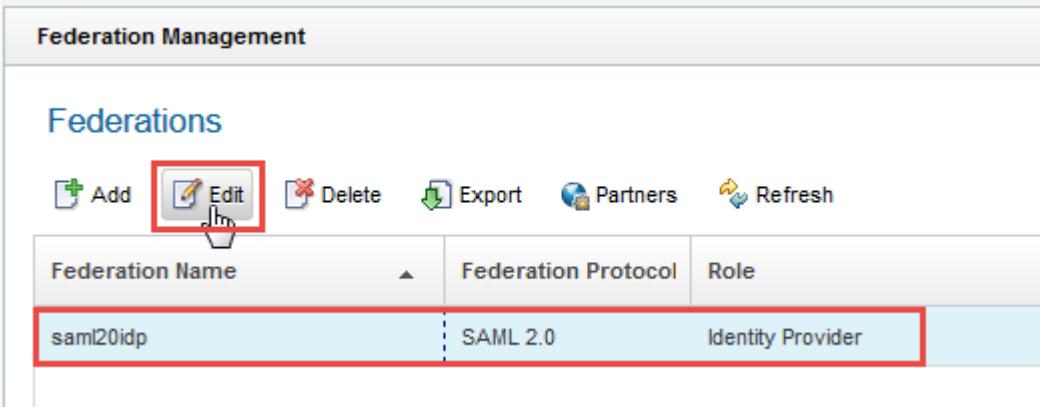
Run this script: **UploadIPMappingRule.py –configure ExternalHttpCallout**

If you use this script, skip to the corresponding SCRIPT-END notice

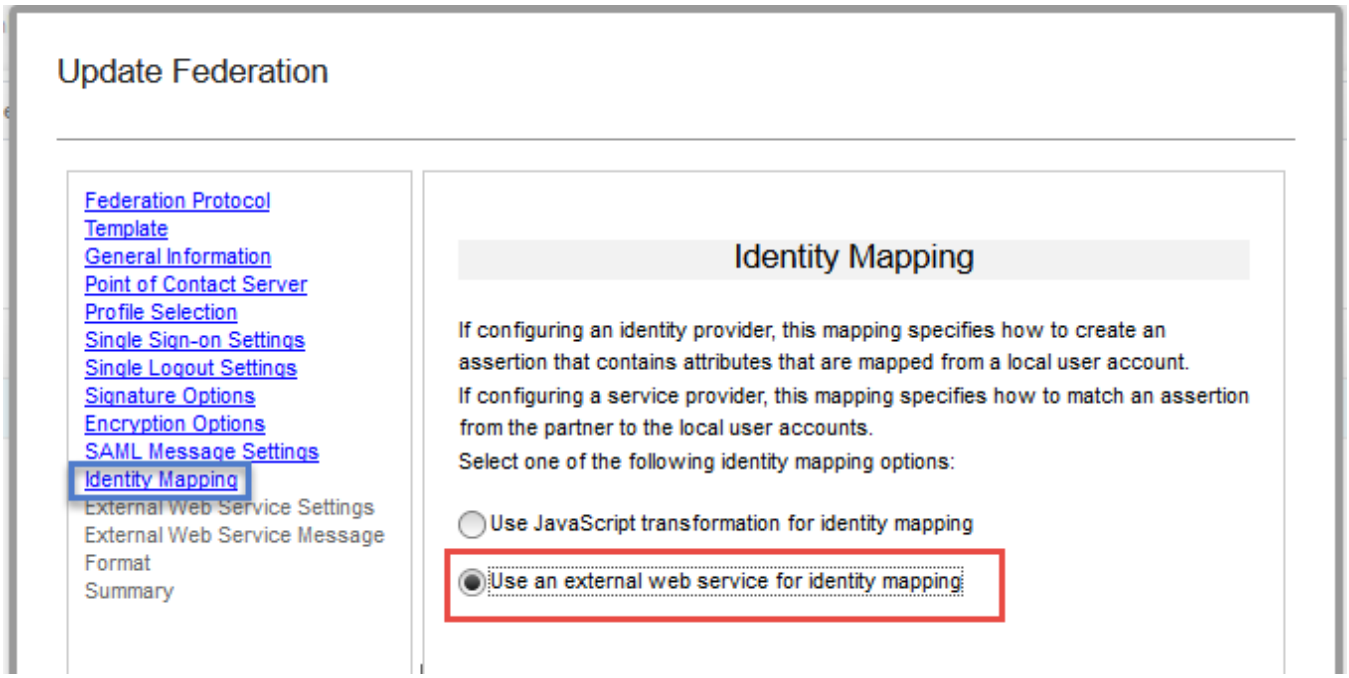
Access the Identity Provider LMI console at https://isam.myidp.ibm.com and authenticate with **admin** and **Passw0rd**.



Navigate to **Secure Federation→Manage: Federations**.

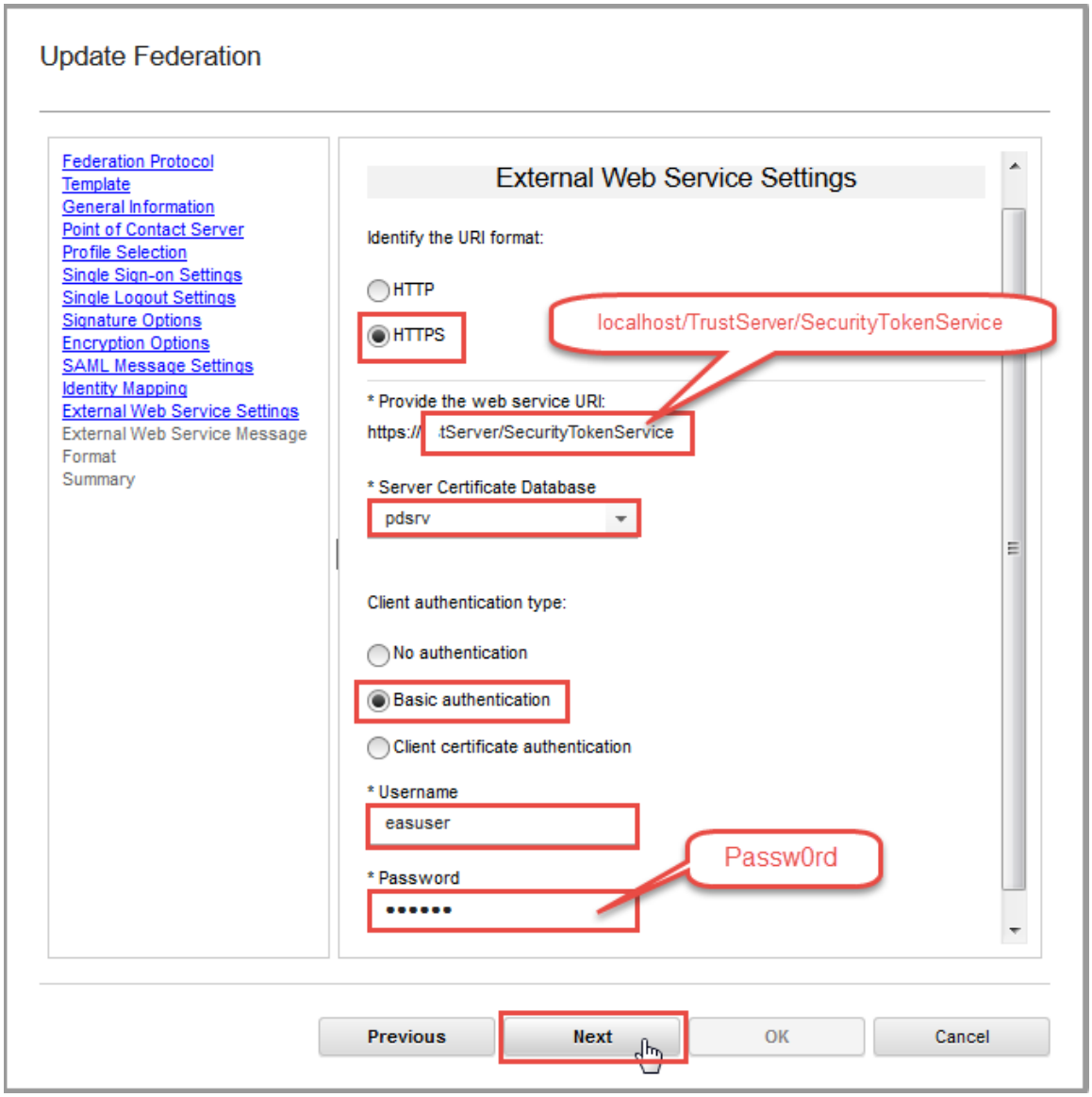


Since we have created only one federation which is **saml20idp**, select it and click **Edit**.



Click **Next** to move through the wizard until you get to the *Identity Mapping* page.

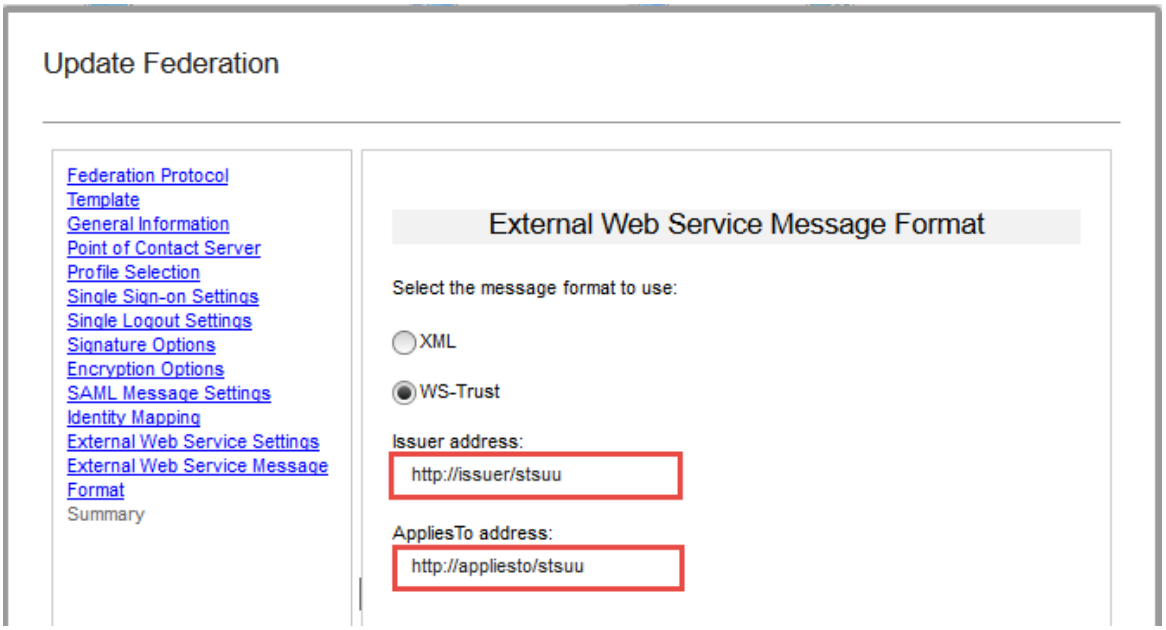
Select the **Use an external web service for identity mapping** radio-button. Notice that the wizard steps change to reflect the new information required. Click **Next**.



Edit the External Web Service Settings, as shown above.

- Connection type: **HTTPS**
- URL: **localhost/TrustServer/SecurityTokenService**
- Server Certificate Database: **pdsrv**
- Client Authentication: **Basic Authentication**
- Username: **easuser**
- Password: **Passw0rd**

Then click **Next**.



Edit the External Web Service Message as shown above.

Set the *message format* to **WS-Trust**
Set the *Issuer address* to **http://issuer/stsuu** and *AppliesTo address* to **http://appliesto/stsuu**

> The *Issuer address* and *AppliesTo address* values here match those we specified when we created our auxiliary chain.
>
> Note that the external web service mapping module always uses the WS-Trust 1.2 "Validate" request type (http://schemas.xmlsoap.org/ws/2005/02/trust/Validate) when making WS-Trust calls so you must use this (as we did) when creating a chain to be called by this module.

Click **Next** to show the summary page and then **OK** to complete the wizard.

Deploy the pending changes.

SCRIPT-END:
The script should display the following:
INFO:UploadIPMappingRule:Configuring the ExternalHttpCallout
INFO:FederationManager:Modifying IdP Federation JSON to enable ExternalHttpCallout
INFO:FederationManager:Successfully modified the Federation using PUT
INFO:UploadIPMappingRule:Successfully configured the ExternalHttpCallout

Perform SSO again, and once more you should see the auxiliary attribute appear at the service provider: Use this trigger URL:

https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Fsaml20sp%2Fsaml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

Login to the IdP with **testuser** and **Passw0rd**.

Access Manager Credential:
User: **testuser**

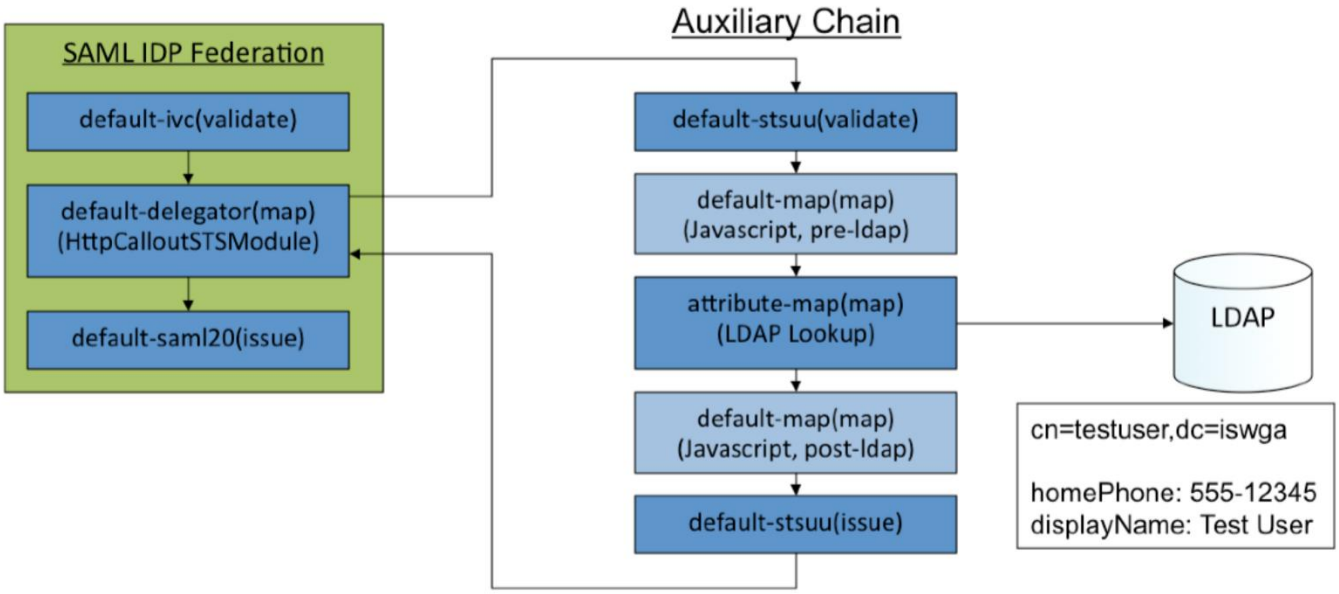
Name	Value(s)
lastName[0]	User
AuthenticationInstant[0]	2015-11-13T15:02:48Z
AZN_CRED_NETWORK_ADDRESS_BIN[0]	0xc0a82a01
AUTHENTICATION_LEVEL[0]	1
AZN_CRED_AUTH_METHOD[0]	ext-auth-interface
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_VkX7nwAAAAIAAAAwXX8AAMglBVhdfwAAV3EwZTdvSnhlaHR5Y09CbDNaSUI2bjljTnB1Y0c5QzVQanBHcWxmLWZRUS1BZk9U:default
testattr_from_auxilary_chain[0]	myvalue_from_auxilary_chain
AuthenticationMethod[0]	urn:oasis:names:tc:SAML:1.0:am:password

If the SSO succeeds and the attribute is NOT in the SP's credential, this is a sure sign that the external HTTP callout failed, and you should inspect the runtime log at the IdP to determine what went wrong.

17.3 Using an auxiliary STS chain for LDAP attribute lookup

The final example of advanced federation mapping rules builds on the previous use cases. This time we are going to replace the (simple) STS Test Chain with a more elaborate chain that performs LDAP attribute lookups to retrieve additional attributes to be included in the final SAML assertion.

Essentially we will be building this pattern:



The attribute-map STS module allows us to perform LDAP attribute lookups from external LDAP directories. Note that these do not have to be the ISAM user registry (we will use the on-appliance LDAP in these examples since it is available in our demo environment).

Configuration of the attribute map permits variable substitution from STSUU attributes for the base DN and search filter used in the LDAP search. To showcase this we will use a Javascript map module prior to the LDAP attribute map to influence the LDAP search that will take place. Our example JavaScript will set hardcoded values but it could also process values that appear in the current STSUU (which comes from the ISAM session at the IDP).

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **STSLDAPAttributeMapping.py –configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

17.3.1 Pre-LDAP javascript mapping module

In this example the "pre-ldap" javascript map is very trivial, and just inserts a canned attribute for BASE_DN which will be used for the LDAP search. Of course yours could be more elaborate:

```
importPackage(Packages.com.tivoli.am.fim.trustserver.sts);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.uuser);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);

//
// we can inspect the stsuu and make and decisions we want here before populating STSUU
// that will be used as input to the LDAP Attribute mapping rule.
//

// for this demo, just set the BASE_DN to the DN we want to search for - if one isn't already set
var existingbaseDN = stsuu.getAttributeValueByName("BASE_DN");
if (existingbaseDN != null && existingbaseDN.length() > 0) {
        IDMappingExtUtils.traceString("The ip_pre_ldap.js found an existing BASE_DN: " + existingbaseDN);
} else {
        IDMappingExtUtils.traceString("The ip_pre_ldap.js mapping rule is setting the BASE_DN");
        var baseDNAttr = new Attribute("BASE_DN", null, "cn=testuser,dc=iswga");
        stsuu.addAttribute(baseDNAttr);
}
```

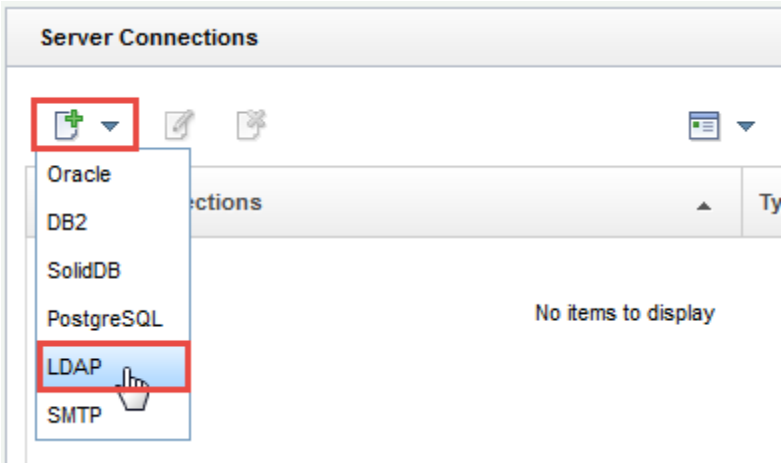
This rule is available in the *…/providedfiles/mappingrules/idp* directory as **ip_pre_ldap.js**.

17.3.2 Configuring the LDAP Attribute Map

The attribute-map requires the configuration of a Server Connection, and Attribute Sources, which are then referenced by the configuration of the AttributeMap STS module.

17.3.2.1 Configuring a Server Connection

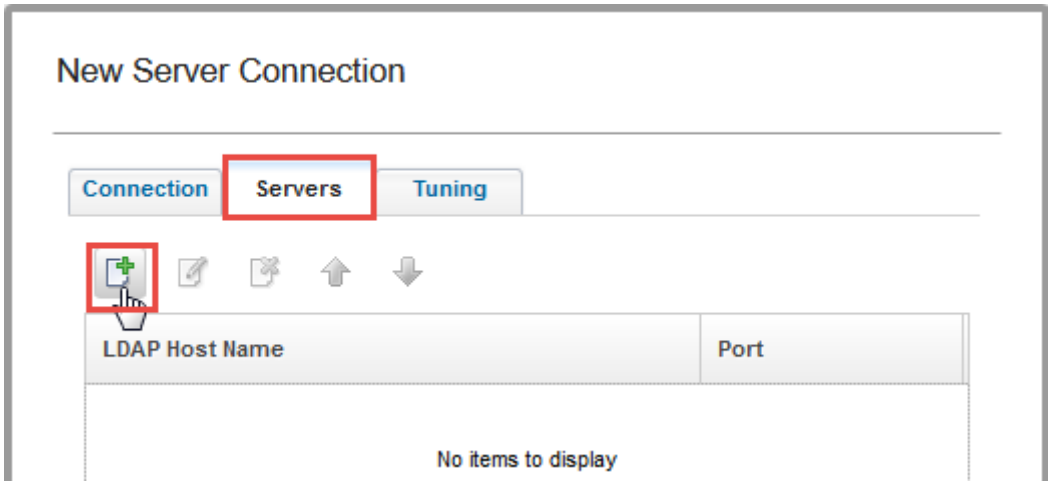
In the LMI, navigate to **Secure Federation -> Global Settings: Server Connections**.



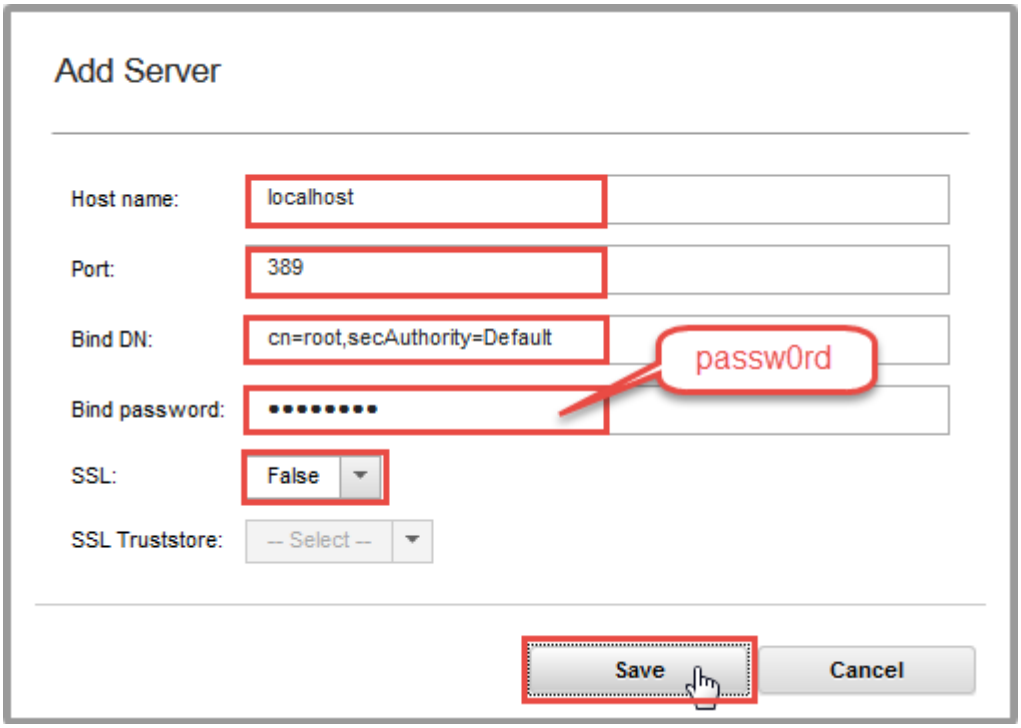
Click the **Add** button and then select **LDAP** from the drop-down list.

> These additional server connections are used by other parts of Access Manager. Only LDAP Server Connections can be used by the STS AttributeMap Module at this time.

On the **Connections** tab, set *Name* to **localldap** and set a *Description*.



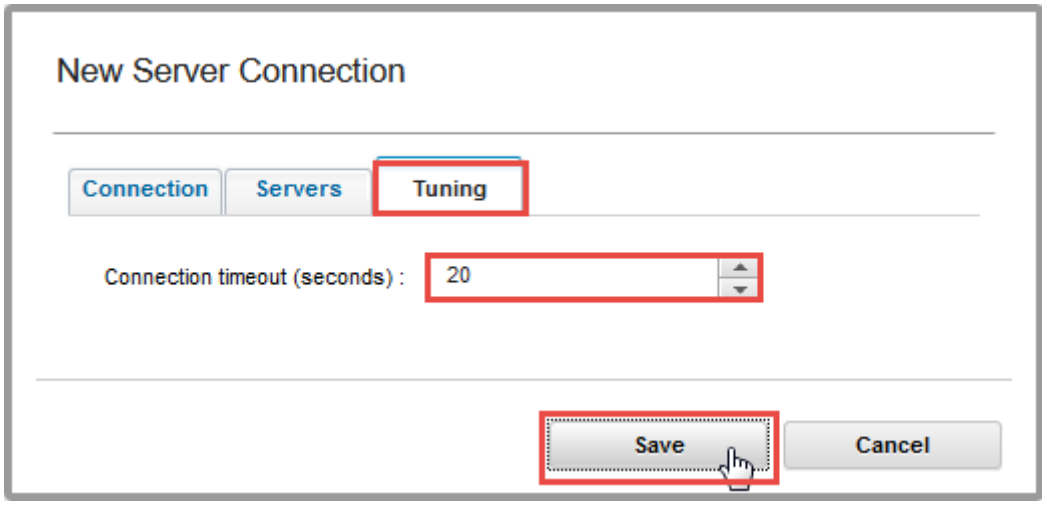
On the **Servers** tab, click **New** button to define a new LDAP server.



Enter the following server details:

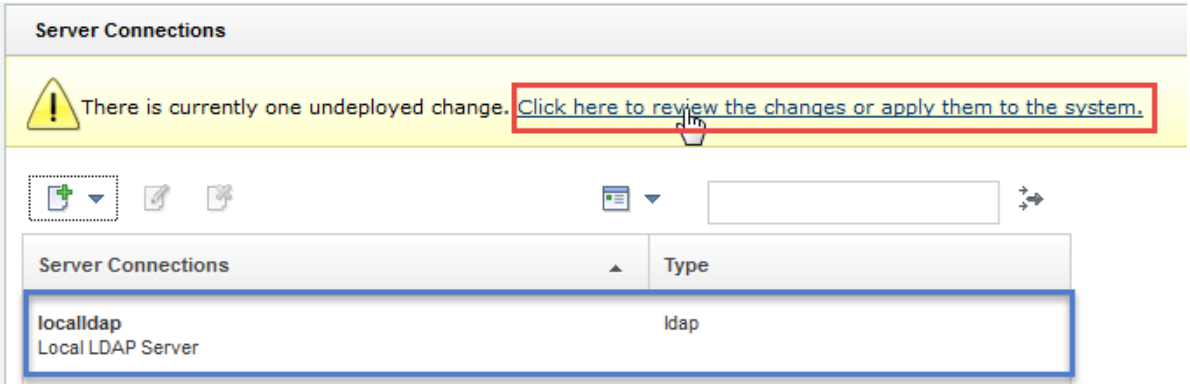
- Host Name: **localhost**
- Port: **389**
- Bind DN: **cn=root,secAuthority=Default**
- Bind password: **passw0rd**
- SSL: **False**

Click **Save**.



In the **Tuning** tab, set *the Connection timeout* to **20** seconds:

Click **Save**. The new server connection is shown in the list:



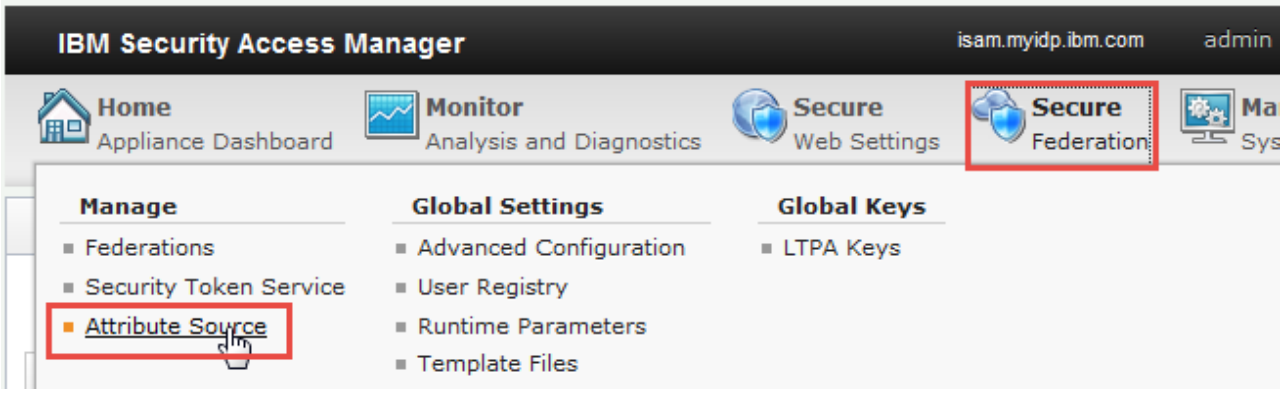
Deploy changes.

This completes configuration of the LDAP server connection. Later it will be referenced by the attribute sources.

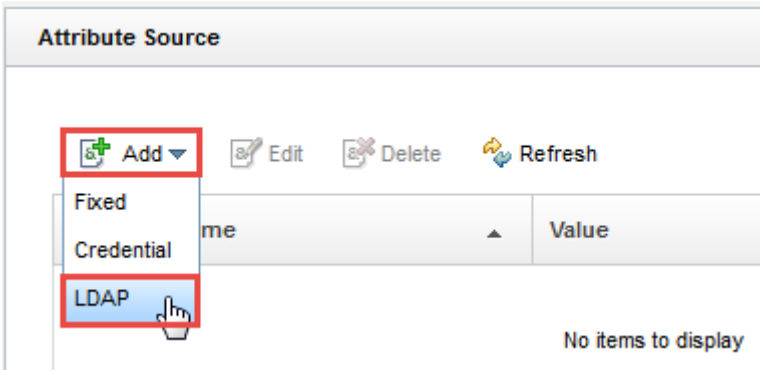
17.3.2.2 Configuring Attribute Sources

Attribute sources define where a particular attribute comes from, along with any configuration required to obtain that attribute. Attribute sources are referenced from the AttributeMap STS module.

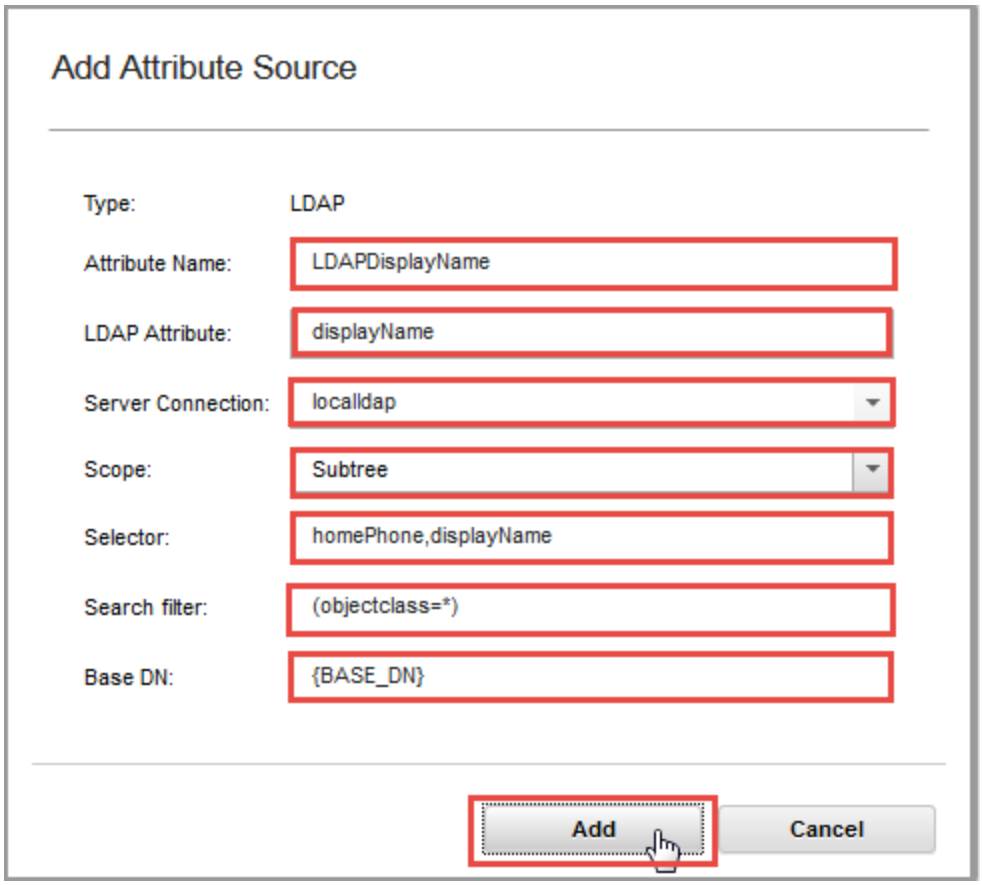
In this cookbook we will create two attribute sources – both are attributes read from the local LDAP server. The first represents the “displayName” attribute, and the second a “phone” attribute.



Navigate to **Secure Federation -> Manage: Attribute Source**.



Click on the **Add** button and select **LDAP** from the drop-down list.



Complete the following properties:

Property	Value
Attribute Name	`LDAPDisplayName`
LDAP Attribute	`displayName`
Server Connection	`localldap`
Scope	`Base`
Selector	`homePhone,displayName`
Search filter	`(objectclass=*)`
Base DN	`{BASE_DN}`

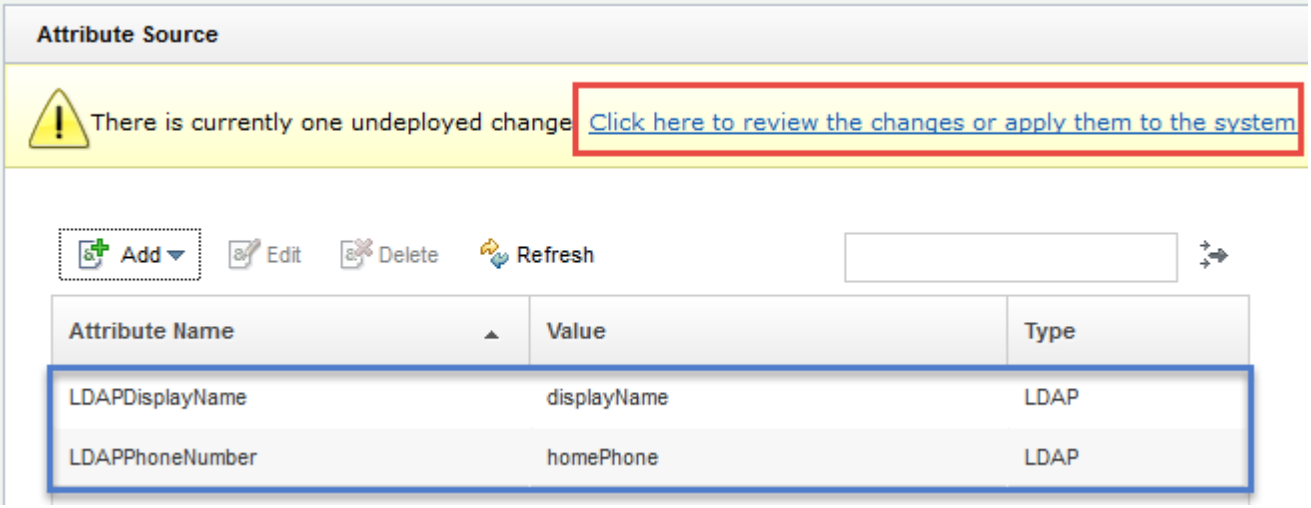
and then click **Add**.

Repeat the previous steps to create another attribute source for LDAPPhoneNumber:

Property	Value
Attribute Name	`LDAPPhoneNumber`
LDAP Attribute	`homePhone`
Server Connection	`localldap`
Scope	`Base`
Selector	`homePhone,displayName`
Search filter	`(objectclass=*)`
Base DN	`{BASE_DN}`

> Pay particular attention to the fact that Server Connection, Scope, Selector, Search Filter and Base DN are the same values, and that the “Selector” is the union of all the attributes we want retrieved. When these conditions exist, the LDAP search will be optimised to only run once and all attribute values will be retrieved from the same LDAP search results. This is important as otherwise a separate LDAP search would be run for each attribute.
>
> Also note that the Base DN is {BASE_DN}. This represents a macro substitution from an attribute in the STSUU Attribute List. In our example that will be populated by the *ip_pre_ldap.js* mapping rule. Only the Base DN and Search filter support macro substitution.

After adding both LDAP attributes they appear in the table as follows:



Deploy Changes.

This completes the configuration of LDAP attribute sources. Later these will be referenced in the AttributeMap module configuration.

17.3.3 Post-LDAP javascript mapping module

In this example the “post-ldap” JavaScript map decides which attribute of the STSUU are retained for use in constructing the SAML assertion. In our case we decide to keep some attributes from the credential that were populated when the testuser logged in, and the displayName and phone attributes which have been populated by the LDAP attribute mapping module.

```
importPackage(Packages.com.tivoli.am.fim.trustserver.sts);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.uuser);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);

//
// filter out STSUU attributes that we don't want in the SAML assertion after the LDAP
// search has run. A good example of this is the BASE_DN attribute, plus other attributes
// that were in the ISAM Credential at the SAML IDP.
//

//
// The simplest way to do this is to decide which attributes we want to keep, and discard the rest.
//
var keepAttrs = [ "emailAddress", "firstName", "lastName", "phone", "displayName" ];

var foundAttrs = {};
for (var i = 0; i < keepAttrs.length; i++) {
        var attr = stsuu.getAttributeContainer().getAttributeByName(keepAttrs[i]);
        if (attr != null) {
                foundAttrs[keepAttrs[i]] = attr;
        }
```

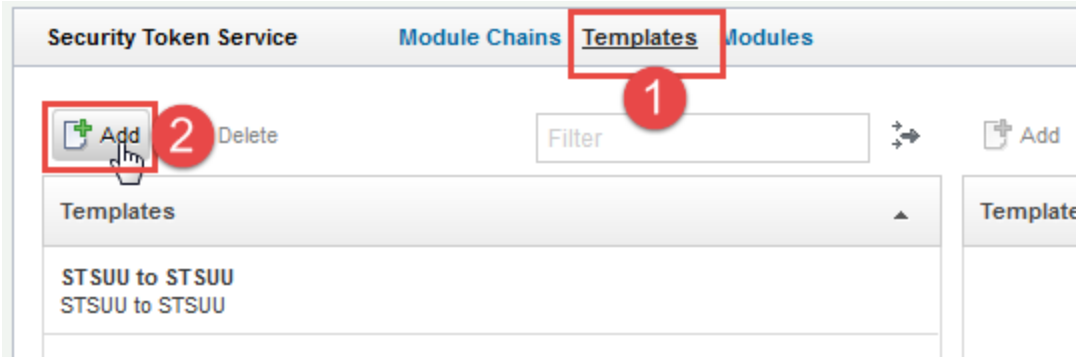
```
}


// empty attrs, then add back what we want
stsuu.clearAttributeList();
var keys = Object.keys(foundAttrs);
if (keys != null && keys.length > 0) {
        for (var i = 0; i < keys.length; i++) {
                stsuu.addAttribute(foundAttrs[keys[i]]);
        }
}
```

This rule is available in the *…/providedfiles/mappingrules/idp* directory as **ip_post_ldap.js**.

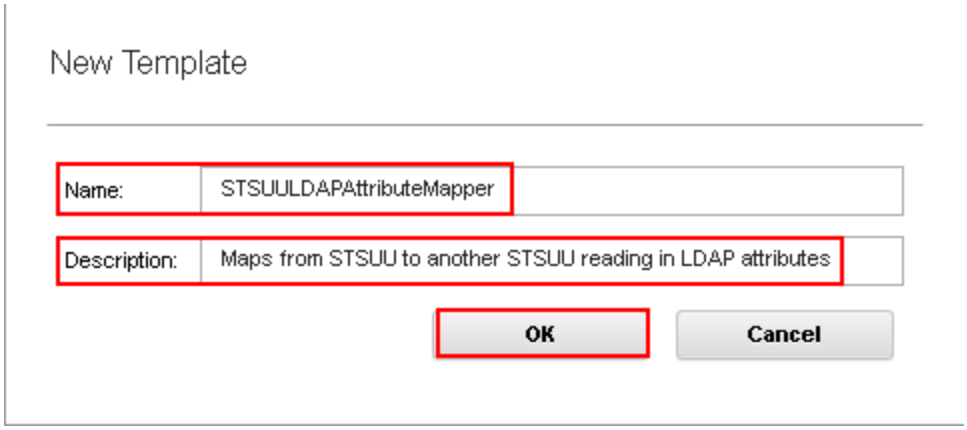
17.3.4 Configuring the STS chain for LDAP Attribute Map

Before we configure the Module Chain we need an STS template.

Navigate to **Secure Federation→Manage: Security Token Service.**



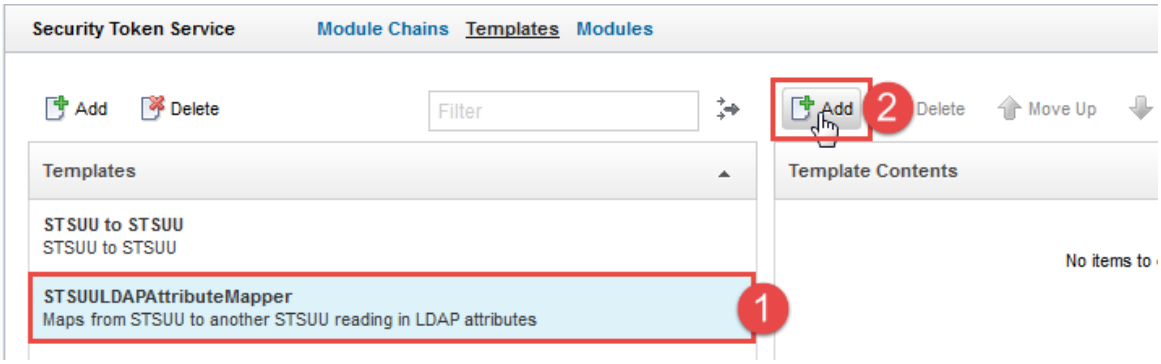
Click on the **Templates** menu and then click the **Add** button to create a new template.



Provide a Name and a suitable Description for the template then click **OK**.

Deploy the pending changes.

Now the Template has been created we to need to populate it by adding modules to the Template Contents.

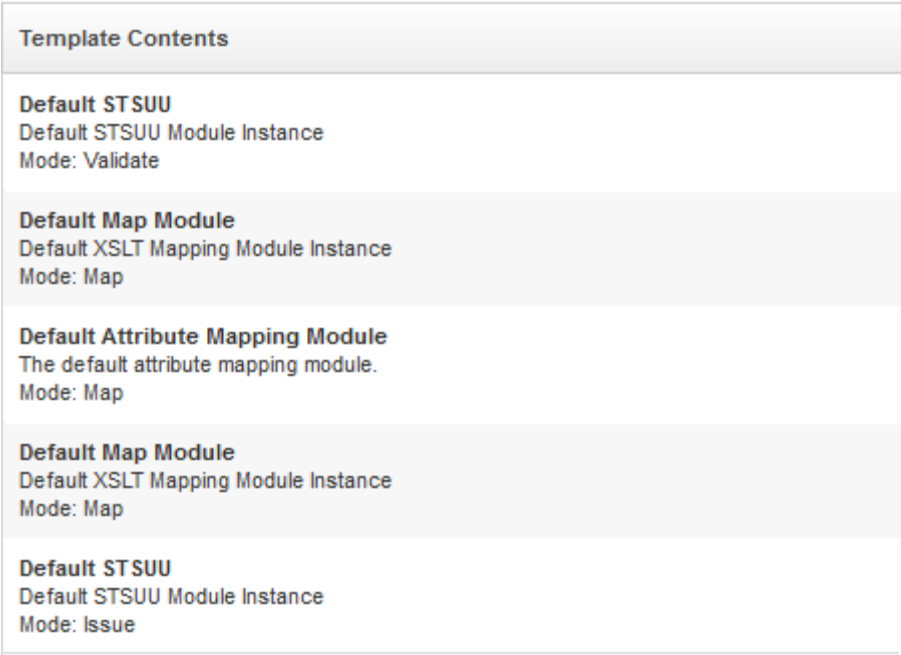


Select the new Template and add modules to it by clicking the **Add** button on the right-hand panel.

Add modules to create a template with the following modules (and modes):

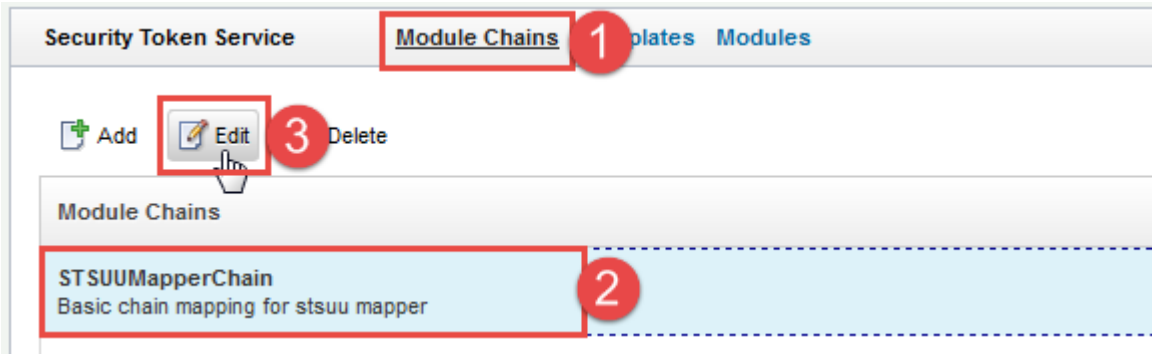
Module	Mode
Default STSUU	Validate
Default Map Module	Map
Default Attribute Mapping Module	Map
Default Map Module	Map
Defaut STSUU	Issue

Once complete, the chain template should look like this:

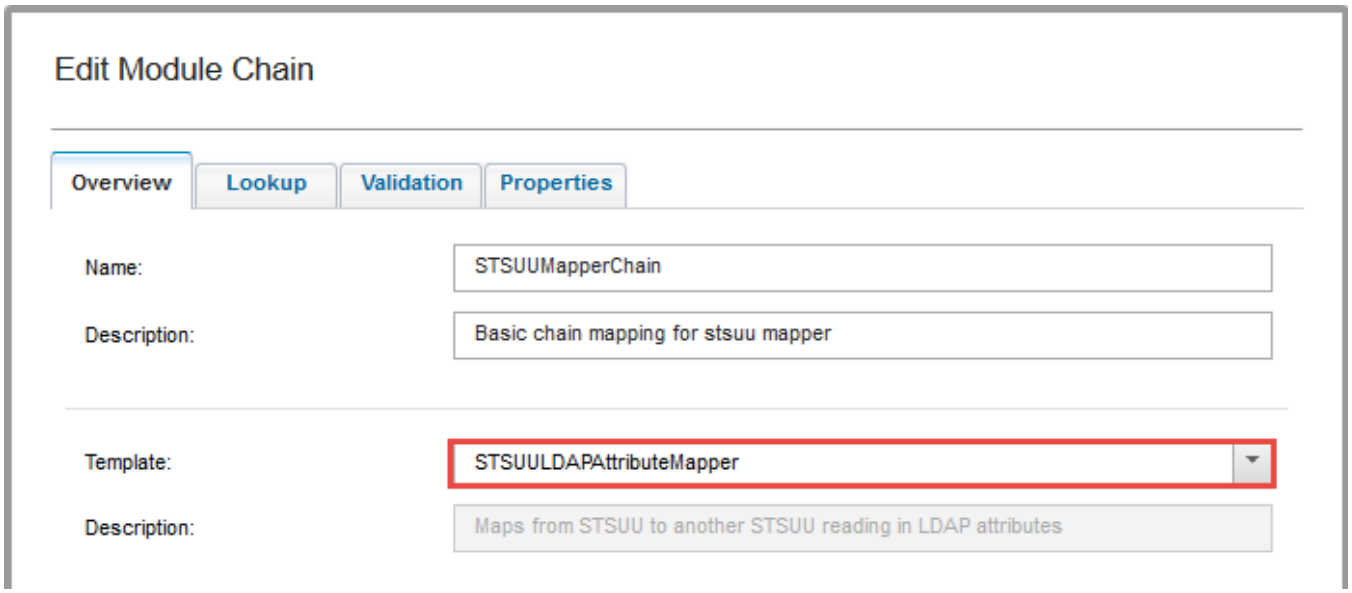


Deploy the pending changes.

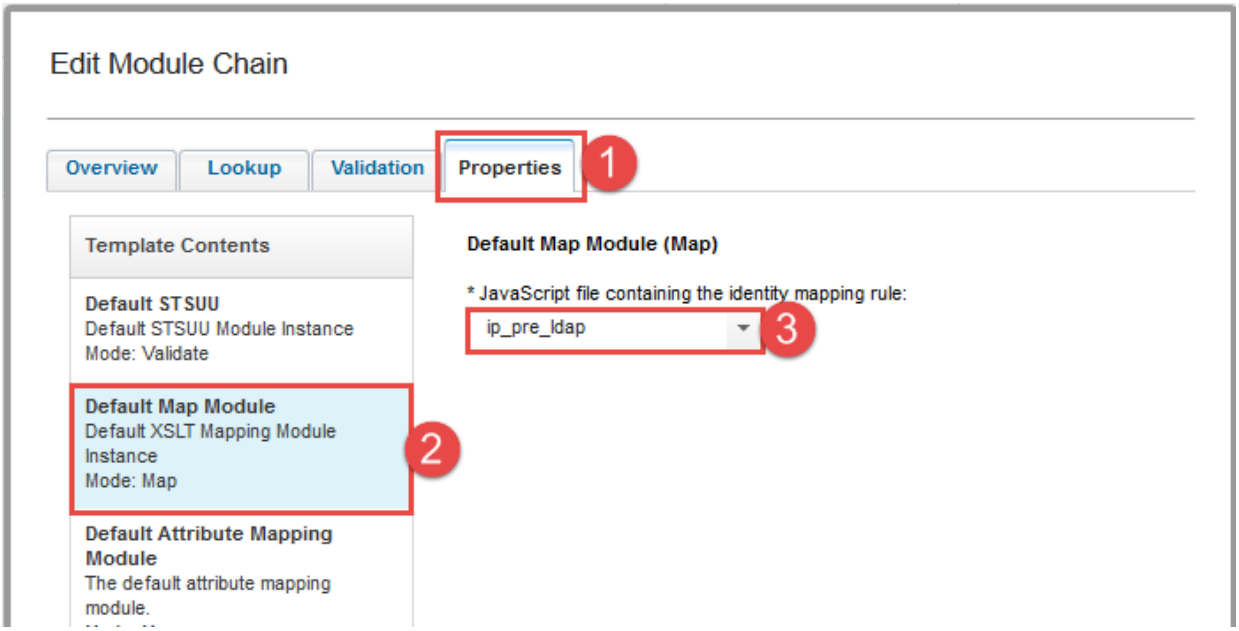
Rather than create a new Module Chain using this template, we will simply reconfigure the STS Test Chain we created previously touse this new template. This chain is already being called by our SAML Identity Provider.



Click on the **Module Chains** link, select the **STSUUMapperChain**, and then click **Edit**.

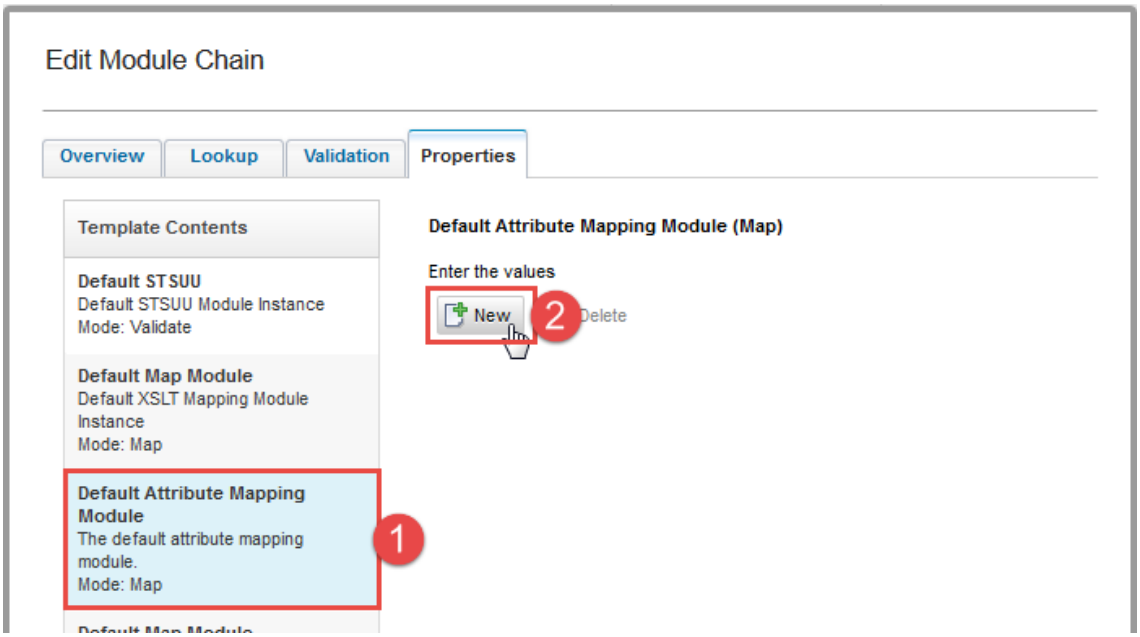


Select **STSUULDAPAttributeMapper** from the drop-down list for **Template**.



Select the **Properties** tab. This is where we configure the properties specific to each module in the chain.

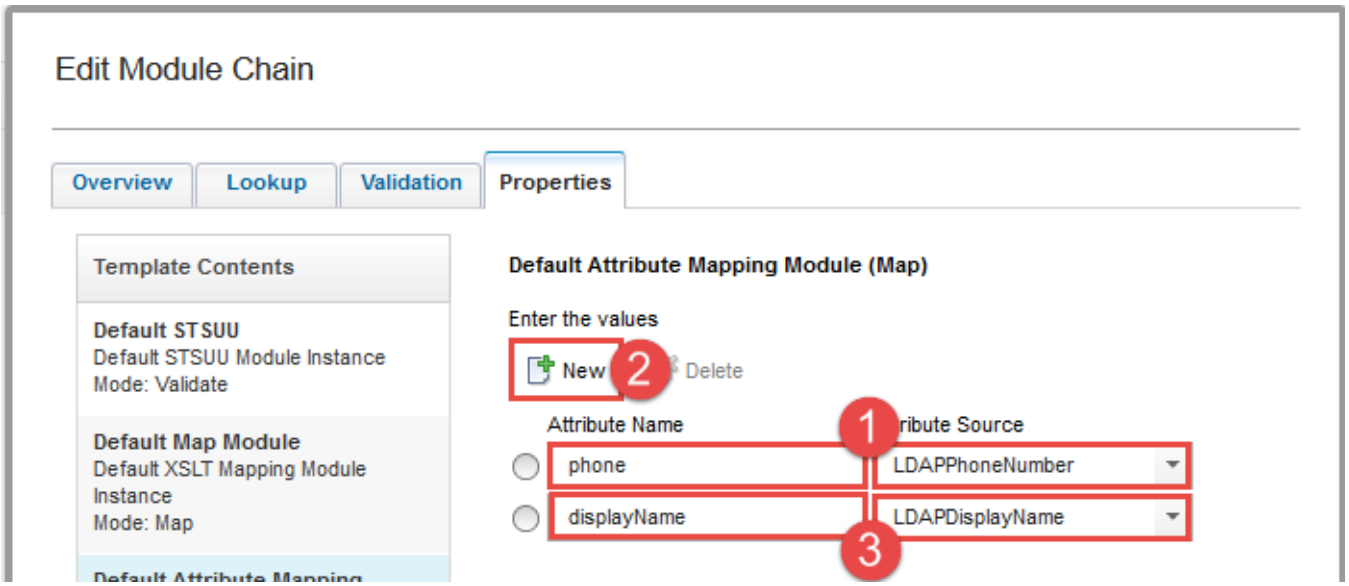
Select the first **Default Map Module**. Set the *JavaScript file* to **ip_pre_ldap**.



Select the **Default Attribute Mapping Module**. This is where we can add attributes to the STSUU from configured attribute sources.

Click **New** to add a new attribute source.

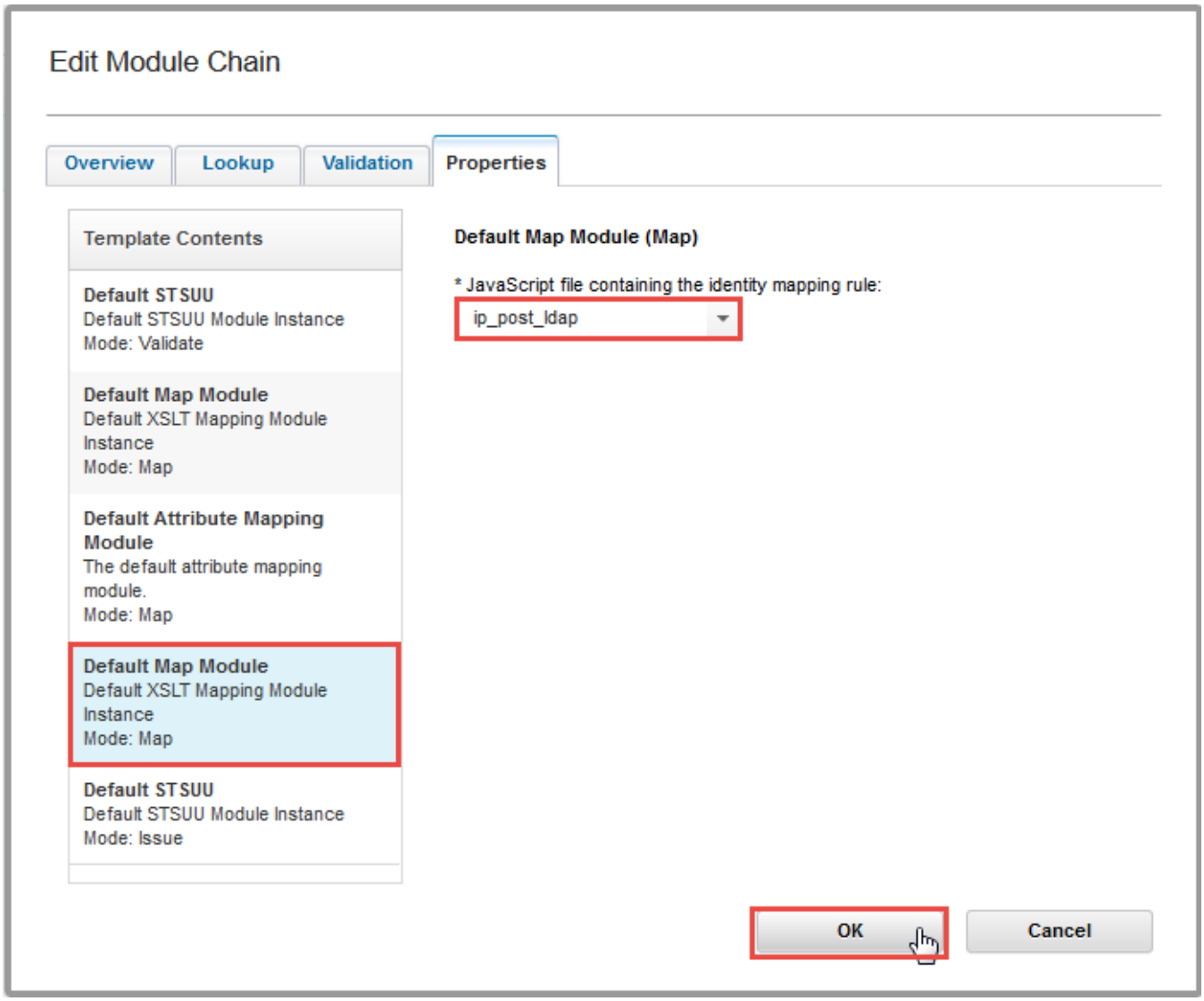
For the Default Attribute Map Module, we need to add the Attribute Sources that we configured earlier in this section.



Enter **phone** as the *Attribute Name* and select **LDAPPhoneNumber** as the *Attribute Source*.

Click **New** to add a second attribute.

Enter **displayName** as the *Attribute Name* and select **LDAPDisplayName** as the *Attribute Source*.



Select the second **Default Map Module**. Select **ip_post_ldap** as the *JavaScript file*.

Click **OK** to save the new Module Chain configuration,.

Deploy changes.

SCRIPT-END:
The script should display the following:
INFO:STSLDAPAttributeMapping:Configuring the LDAP Attribute chain
INFO:BaseManager:Configuring the server connection
INFO:BaseManager:Successfully configured the server connection
INFO:BaseManager:Configuring Attribute sources
INFO:BaseManager:Successfully configured attribute sources
INFO:FederationManager:Successfully configured the STS Module Chain Template
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:STSLDAPAttributeMapping:Successfully configured the LDAP Attribute chain

We have changed the template and module configurations of the *STSMapperChain* Module Chain but, since the *appliesto* and the *issuer* addresses associated with this Module Chain have not changed, our SAML Identity Provider will still call it as part of its identity mapping during SSO.

Perform SAML SSO again, and this time you should see the *displayName* and *phone* attributes appear at the service provider. Use this trigger URL:

https://www.myidp.ibm.com/isam/sps/saml20idp/saml20/logininitial?RequestBinding=HTTPPost&PartnerId=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Fsaml20sp%2Fsaml20&NameIdFormat=Email&Target=https://www.mysp.ibm.com/isam/mobile-demo/diag/

Login to the IdP with **testuser** and **Passw0rd**.

displayName[0]	Test User

…

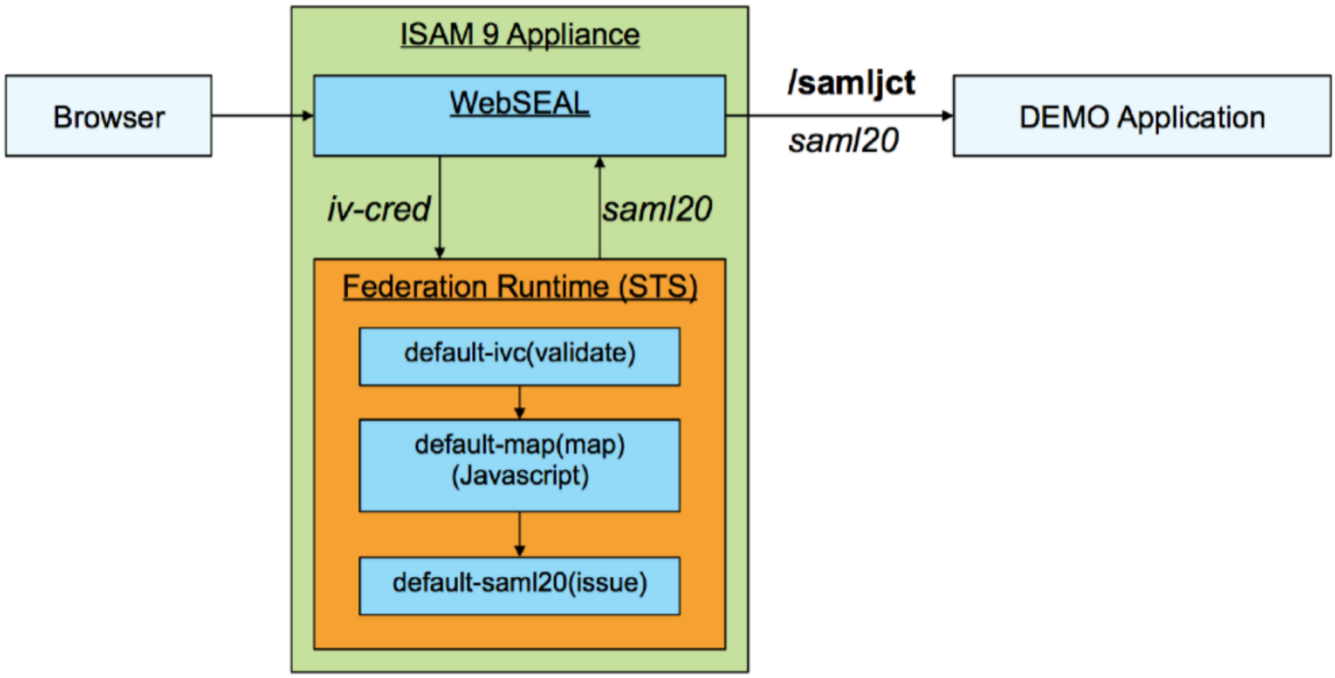
phone[0]	555-12345

This concludes the demonstration of advanced attribute mapping in federations. In this section you have learned about the HttpClient, the HTTPCallout STSModule, and how you can use them with WS-Trust to call a second STS chain. You have also learned how to configure the LDAP Attribute Mapping module in an STS chain to perform LDAP attribute lookups.

18 STS Tokens on Reverse Proxy Junctions

In this section we will demonstrate the use of the federation Security Token Service to implement STS chains that can be used by the ISAM reverse proxy (WebSEAL) for a capability known as TFIM-SSO junctions. This name comes from the former federation product which provided the STS capability. The reverse proxy configuration has not changed in any way in ISAM 9 – all that has changed is that the STS may now be provided by the built-in federation runtime.

The scenario we will implement in this cookbook will be a "SAML junction", where the current user's credential will be exchanged for a SAML assertion at the STS, and the SAML assertion will be sent across a junction as an HTTP header. The scenario is described in this diagram:



SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **SAMLJunction.py -configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

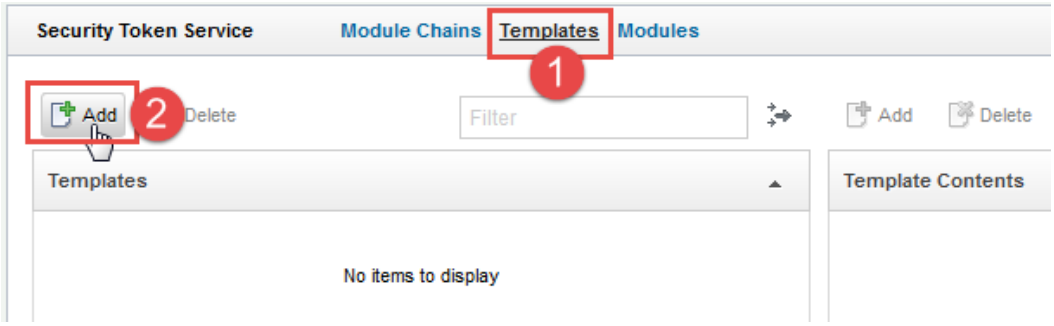
It is assumed that the password for the easuser on the Identity Provider appliance has already been changed to Passw0rd as described previously in this document.

18.1 Create the ISAM Credential to SAML 2.0 STS Chain Template

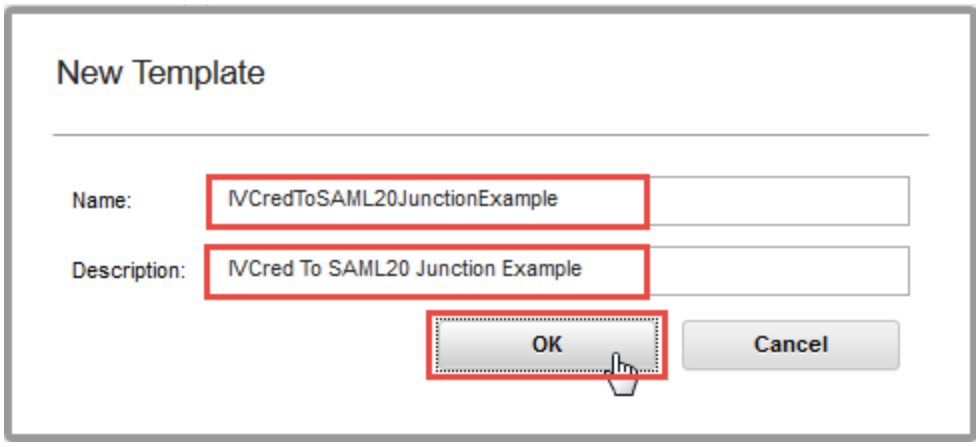
First we create an STS Chain Template. This defines an ordered list of Modules that will make up any chain built on this template.

Access the Identity Provider LMI console at https://isam.myidp.ibm.com and authenticate with **admin** and **Passw0rd**.

Navigate to **Secure Federation→Manage: Security Token Service.**



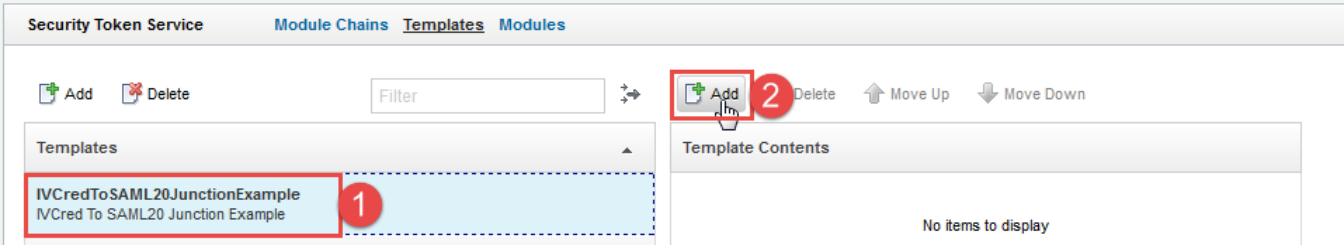
Click on the **Templates** menu and then click the **Add** button to create a new template.



Provide a Name and a suitable Description for the template then click **OK**.

Deploy the pending changes.

Now the Template has been created we to need to populate it by adding modules to the Template Contents.

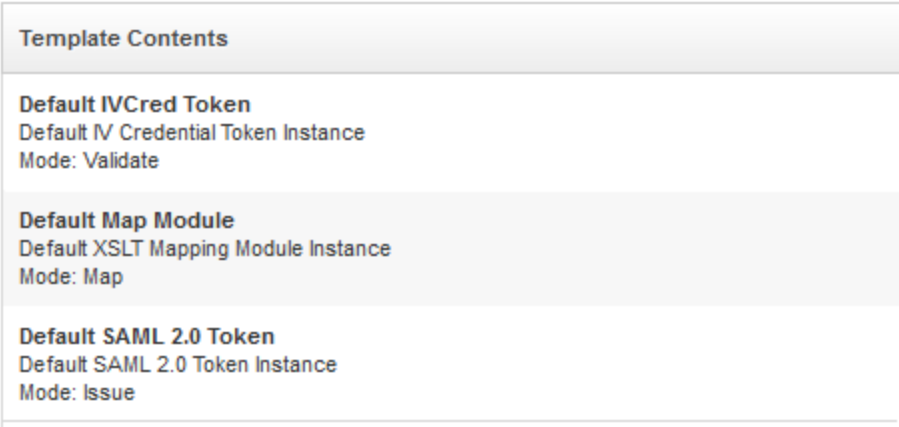


Select the new Template and add modules to it by clicking the **Add** button on the right-hand panel.

Add modules to create a template with the following modules (and modes):

Module	Mode
Default IVCred Token	Validate
Default Map Module	Map
Default SAML 2.0 Token	Issue

Once complete, the chain template should look like this:

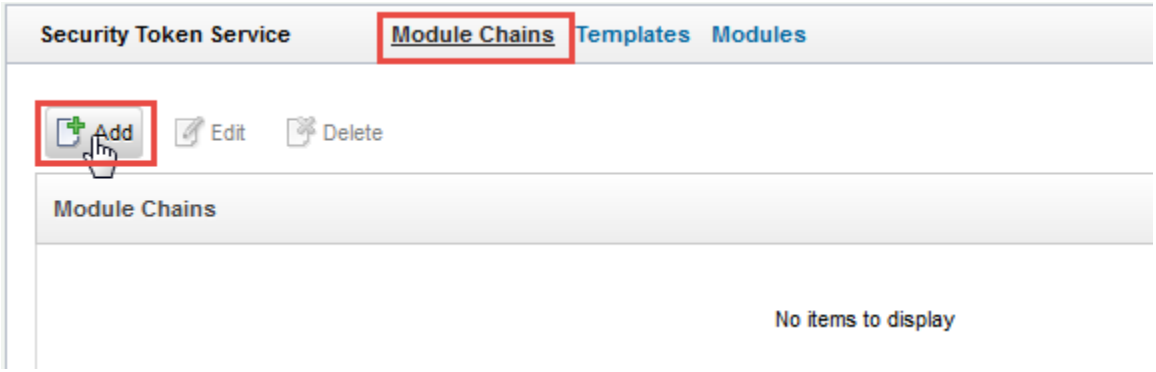


Deploy the pending changes.

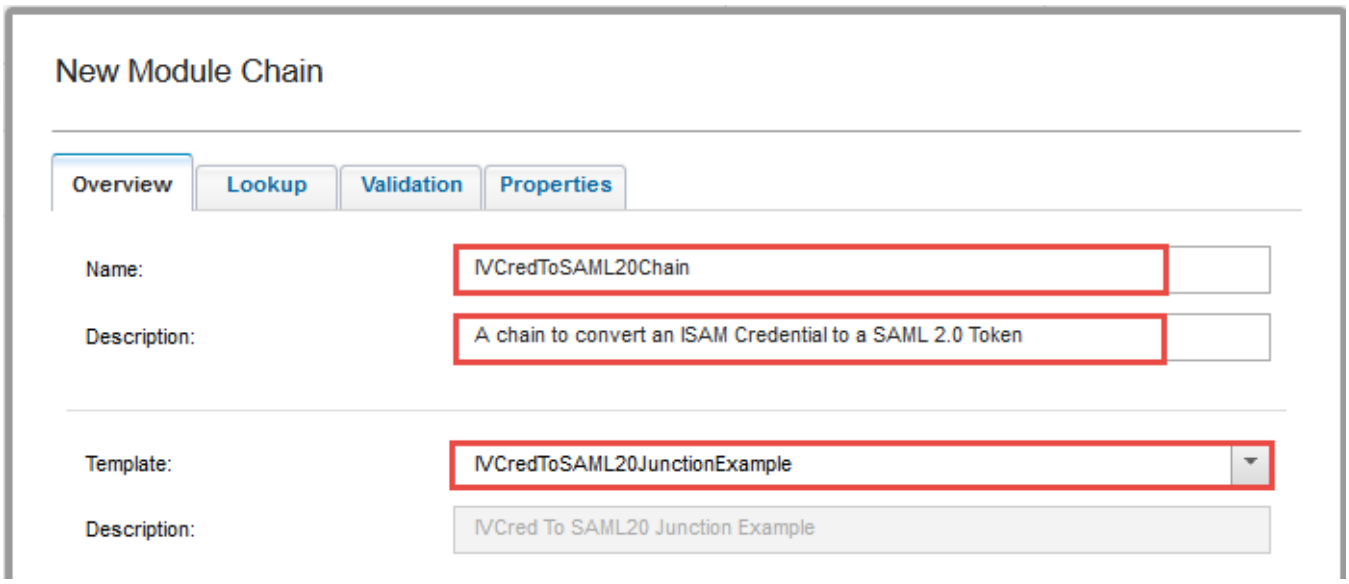
18.2 Create the ISAM Credential to SAML 2.0 STS Module Chain

We will now create a Module Chain from the new template.

As we're seen, the template determines the modules in the chain, the mode they will operate in, and the order in which they will run. The rest of the configuration is specified at the Module Chain level.

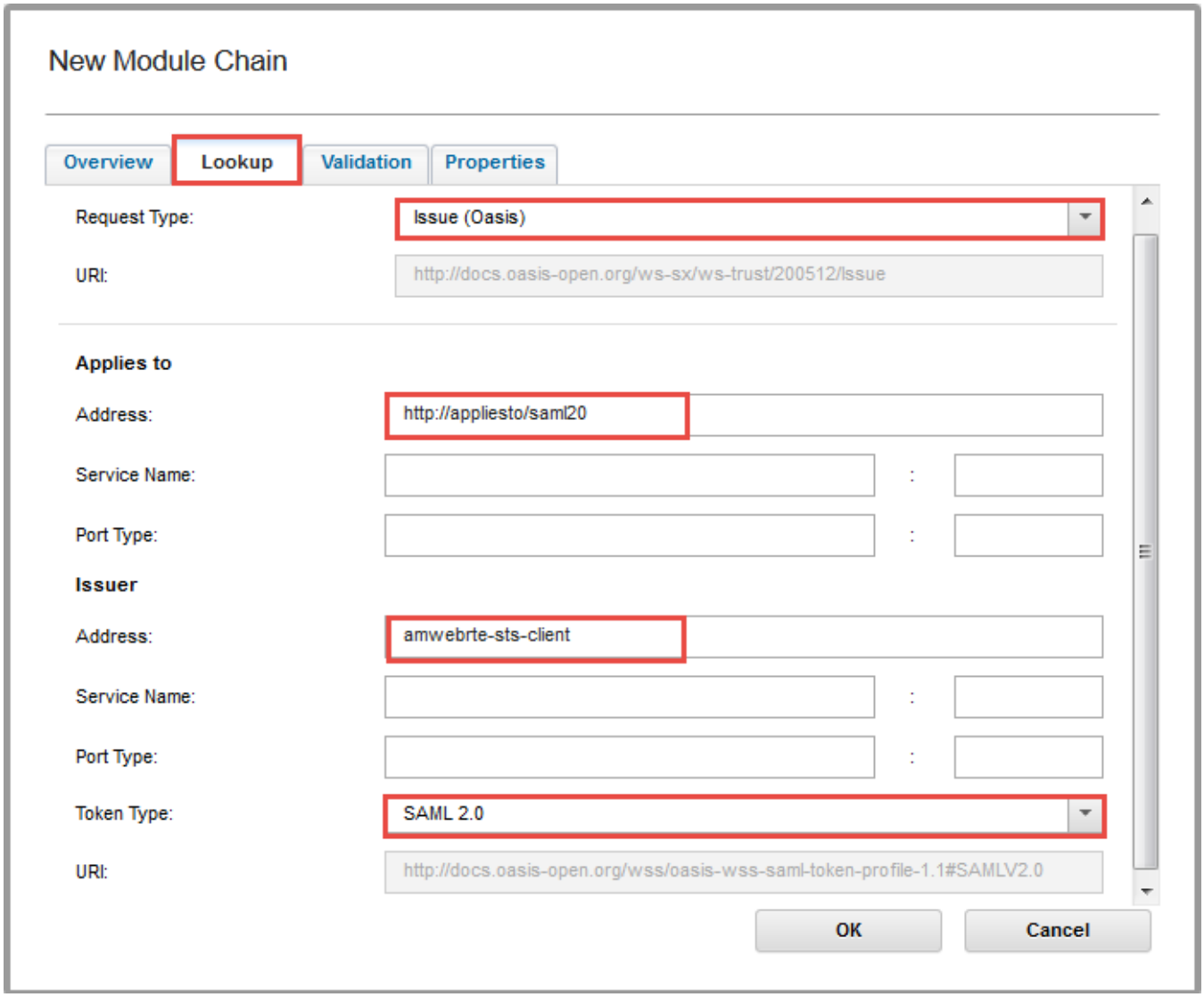


Still in the Security Token Service screen, click on the **Module Chains** tab and then click **Add** to add a new Module chain.



Enter **IVCredToSAML20Chain** as the Name for the chain and provide a description. Select the **IVCredToSAML20JunctionExample** Template for the chain.

Click on the **Lookup** tab.



When a WS-Trust request arrives at the STS, the information provided in the **Lookup** tab of each defined Module Chain is used to determine which one should process the request. Only if all specified Lookup fields match the incoming WS-Trust request is it considered a match.

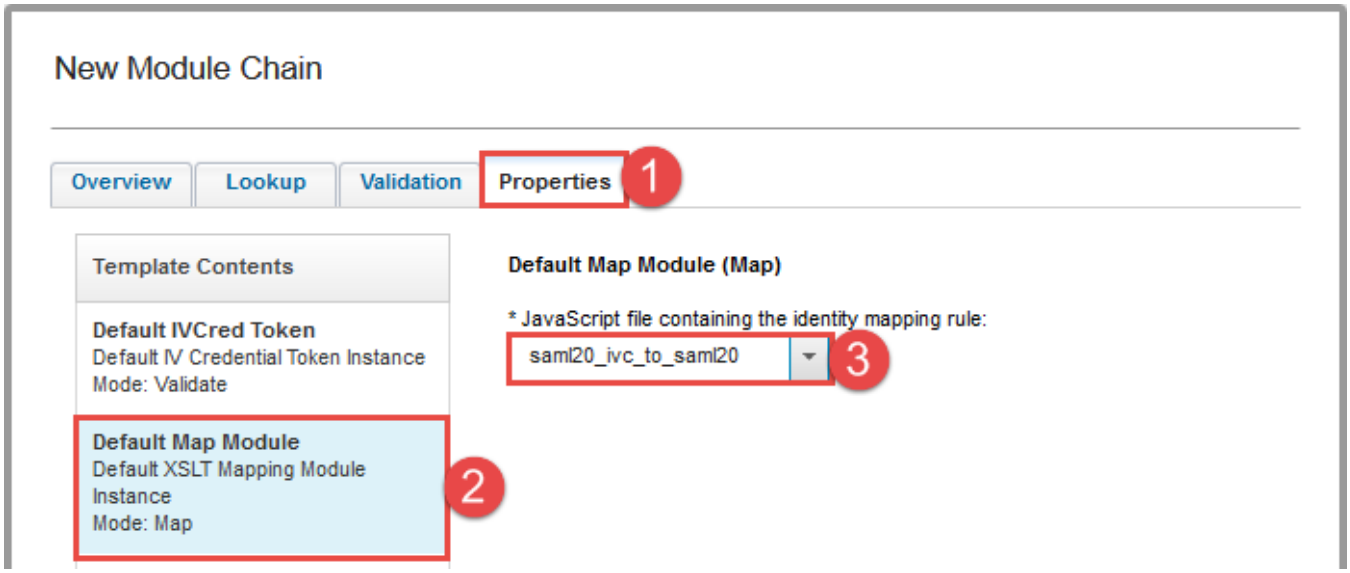
Select **Issue (Oasis)** from the *Request Type* drop-down list.

Enter **http://appliesto/saml20** in the *Address* box under *Applies to*.
Enter **amwebrte-sts-client** in the *Address* box under *Issuer*.
Select **SAML 2.0** from the *Token Type* drop-down list.

Click on the **Properties** tab.

In the Properties tab, we configure the chain-specific properties for each module in the chain.

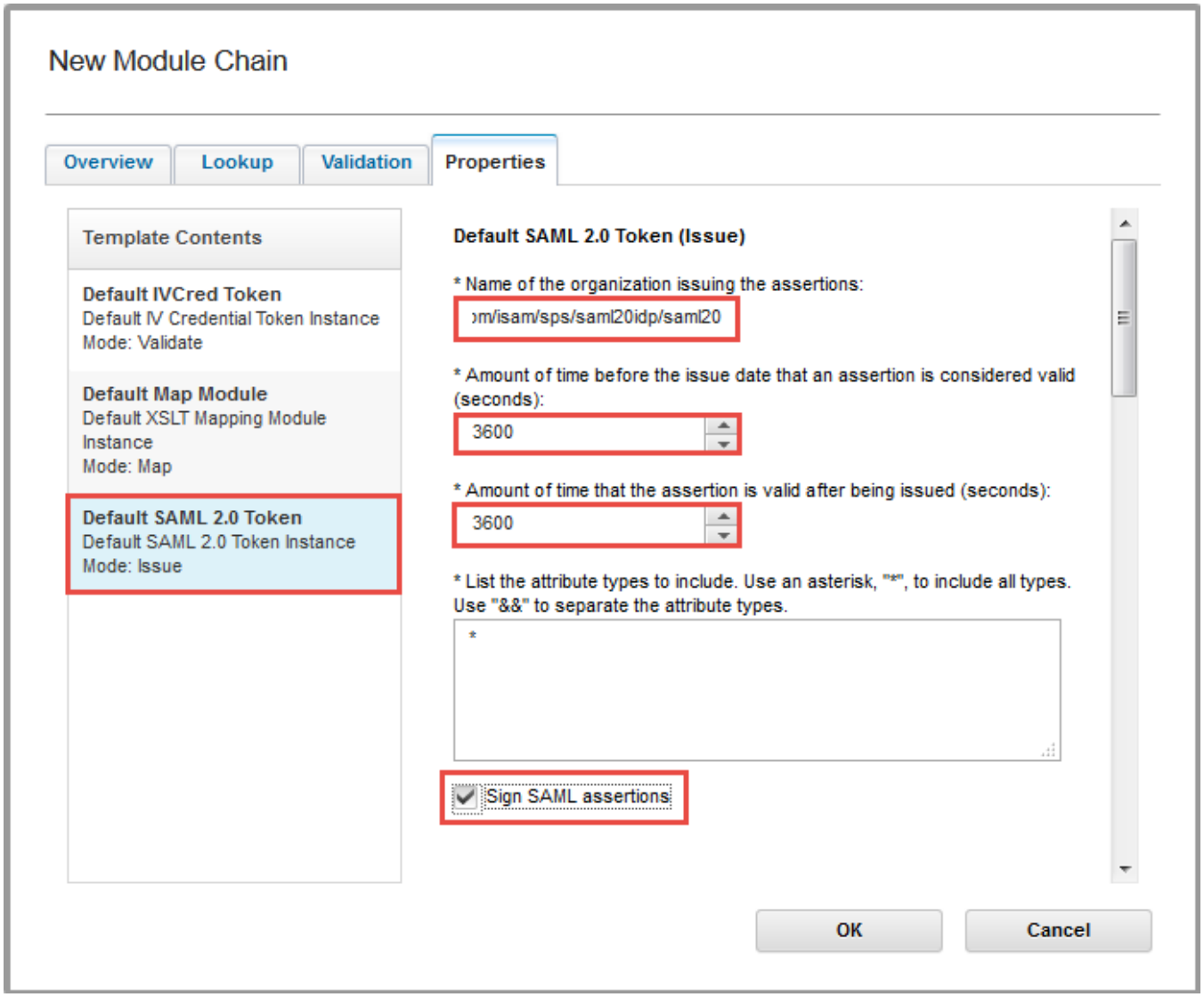
The properties tab initially shows the properties for the IVCred module. We don't want to enable signature validation (which is the only) option so there's nothing to change here.



Select the **Default Map Module** from the list of modules on the left-hand side. This opens the properties panel for that module.

The only thing that we need to specify for the mapping module is which JavaScript file should be used. Select **saml20_ivc_to_saml20** from the drop-down list.

Finally we need to configure the SAML 2.0 module to issue a SAML 2.0 token.

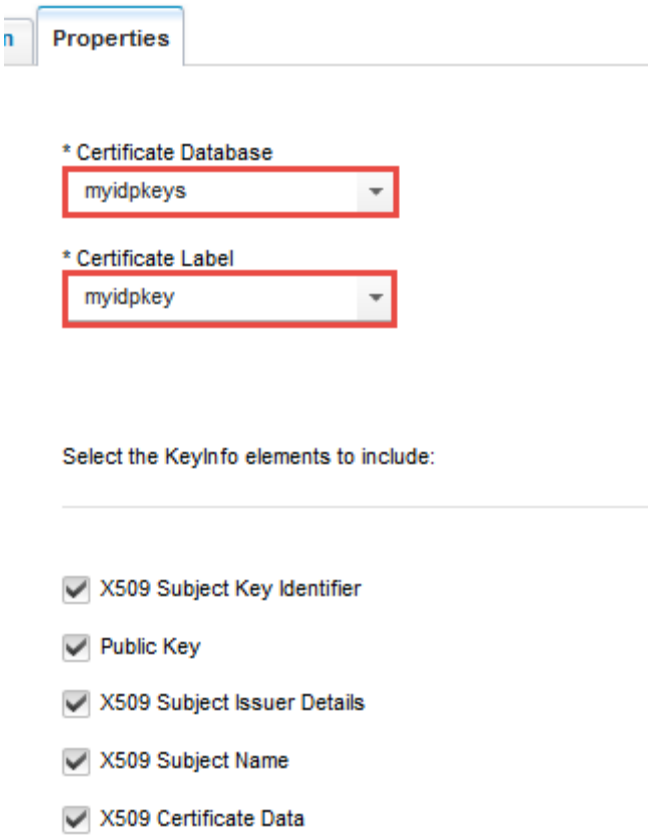


Select the **Default SAML 2.0 Token** module from the module list.

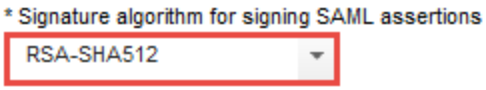
On the first page:

Set the *Issuer* to **https://www.myidp.ibm.com/isam/sps/saml20idp/saml20**
Set *amount of time before issue date that assertion is valid* to **3600** seconds
Set *amount of time after issue date that assertion is valid* to **3600** seconds
Check checkbox to enable signing of assertions

Scroll down the properties pane.

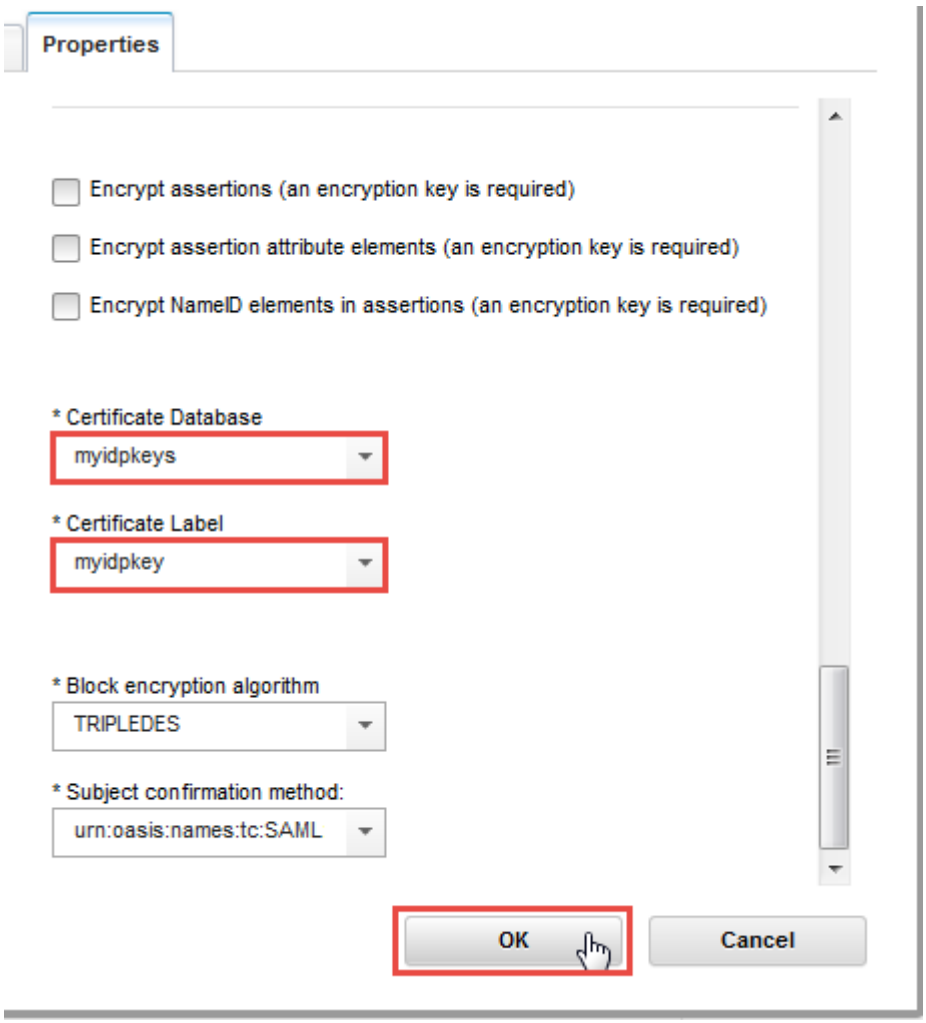


Select **myidpkeys** as the *Certificate Database* and **myidpkey** as the *Certificate label*.



Select **RSA-SHA512** as the signature algorithm.

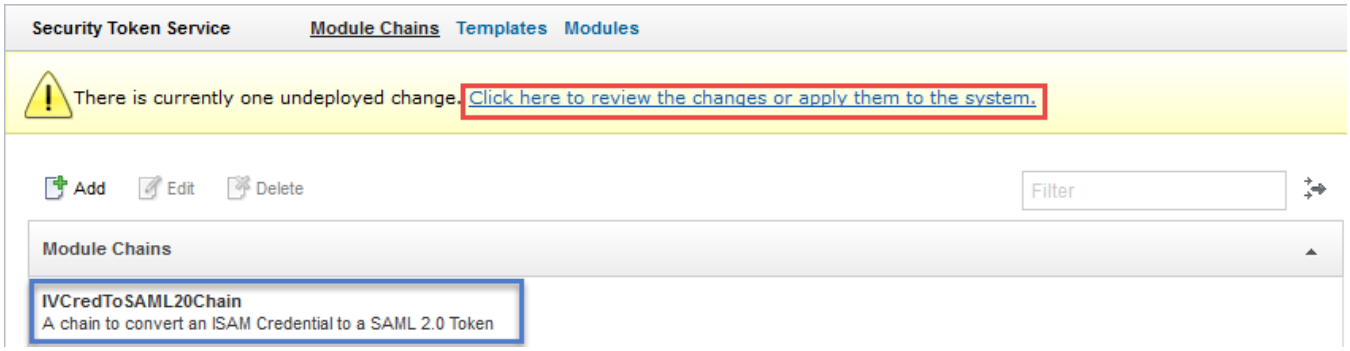
Scroll down to the bottom of the properties pane



Even though we are not going to configure encryption, we are forced to select a database and key label for encryption operations. This is a known issue with the UI.

Select **myidpkeys** as the *Certificate Database* and **myidpkey** as the *Certificate label*.

Module Chain configuration is now complete. Click **OK** to save the new Module Chain.



The new chain is shown in the list of Module Chains. **Deploy** pending changes.

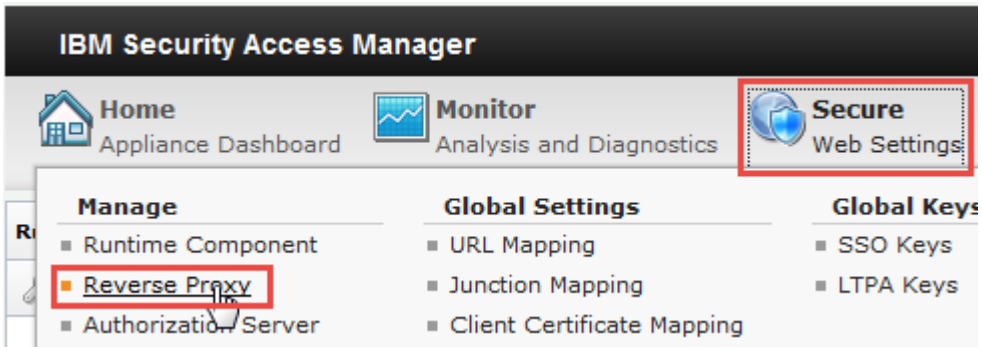
If you want to view the *saml20_ivc_to_saml20* mapping rule (or any other mapping rule) these can be found in the GUI console under **Secure Federation→Global Settings: Mapping Rules**.

18.3 Update the reverse proxy configuration file

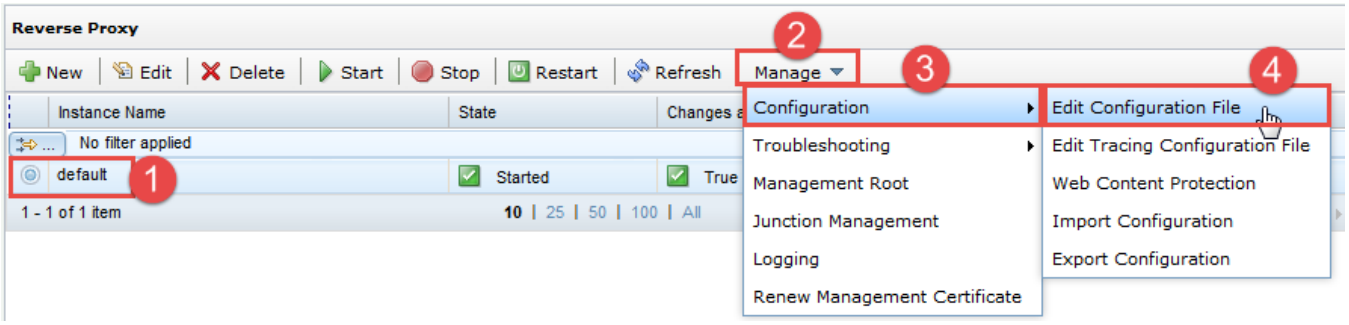
In this section we will modify the configuration file of the reverse proxy to allow it to communicate with the federation runtime STS, and to instruct it how we want the SAML assertion transmitted across the junction.

All of this information is configured per existing product documentation. Here is the product documentation reference:

http://www.ibm.com/support/knowledgecenter/SSPREK_9.0.0/com.ibm.isam.doc/wrp_config/concept/con_sso_usg_tfim.html



In the administration console of the IdP image, navigate to **Secure Web Settings -> Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

To find a location in this file, use the browsers search function. On Firefox this is activated using **Ctrl-f**.

Add the following stanza to the end of the configuration file to define an STS cluster. This instructs the reverse proxy how to contact the WS-Trust 1.3 endpoint of the federation runtime:

```
[tfim-cluster:samljct]
server = 9,https://localhost:443/TrustServerWST13/services/RequestSecurityToken
ssl-keyfile = pdsrv.kdb
ssl-keyfile-stash = pdsrv.sth
handle-pool-size = 10
handle-idle-timeout = 240
timeout = 240
basic-auth-user = easuser
basic-auth-passwd = Passw0rd
```

Also add the following stanza which instructs the reverse proxy how to use the STS and send tokens across the "/samljct" junction. Note that the *tfim-cluster-name* references the cluster stanza above.

```
[tfimsso:/samljct]
token-type = http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
applies-to = http://appliesto/saml20
renewal-window = 15
preserve-xml-token = false
tfim-cluster-name = samljct
token-transmit-type = header
token-transmit-name = SAMLAssertion
always-send-tokens = true
one-time-token = true
token-collection-size = 1
```

> In a production scenario you would never really use *one-time-token=true* along with *token-collection-size=1* because this means a separate call to the STS for every request across the junction, which is highly inefficient. For a demo, and to test the STS, this is ok.

Save the configuration file changes, deploy the updates, and restart the reverse proxy.

18.4 Create the /samljct Junction

Creation of the junction can be done graphically via the UI in a manner similar to that shown for the federation */isam* junction in section **Error! Reference source not found.**. In practice junctions are rarely created this way. Of course our recommendation is to automate the configuration of the junction, however in this section we will show you a pdadmin command that can be use on the command line.

Open an SSH session to the IdP appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

Authenticate with **admin** and **Passw0rd**.

Navigate to **isam** and start the **admin** utility:

```
isam.myidp.ibm.com> isam
isam.myidp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Enter the following command to create a junction called */samljct* which is configured for TFIM SSO (*-Y*):

```
server task default-webseald-isam.myidp.ibm.com create -t ssl -h localhost -p 443 -Y /samljct
```

Type **exit** twice to end the session.

Other junction flags may also be specified however only the minimal set required for this demonstration are shown above.

SCRIPT-END:

The script should display the following:

INFO:FederationManager:Configuring the STS Module Chain Template
INFO:FederationManager:Successfully configured the STS Module Chain Template
INFO:FederationManager:Configuring the STS Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the STS Module Chain Mapping
INFO:WGAManager:Configuring WebSEAL.conf file for SAML Junction at IdP
INFO:WGAManager:Successfully configured WebSEAL.conf file for SAML IdP
INFO:WGAManager:Configuring junction
INFO:WGAManager:Successfully configured junction
INFO:SAML Junction:End SAML junction creation and configuration

18.5 Enable the demonstration application

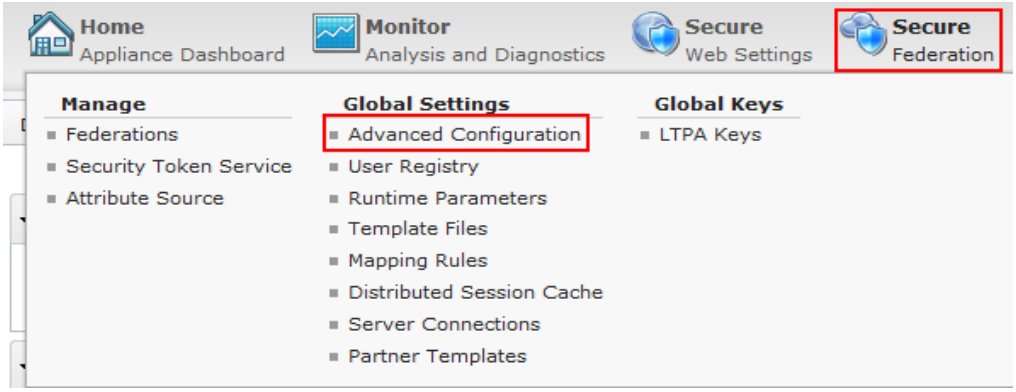
SCRIPT-START:

A script is available for this section as an alternative to following the manual steps.

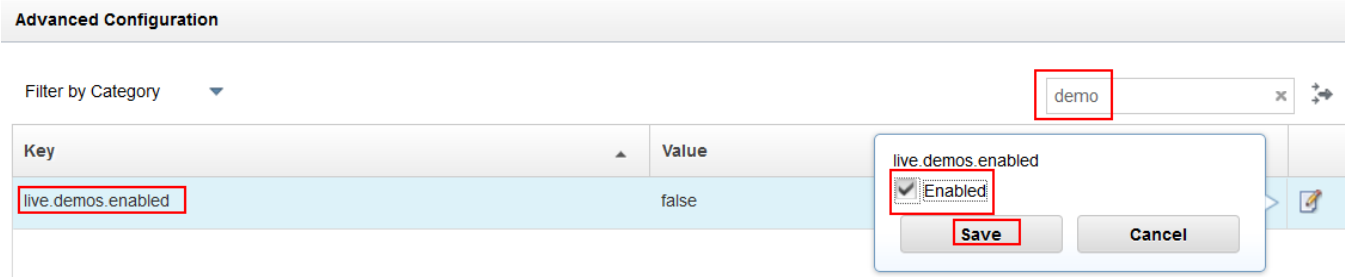
Run this script: **SAMLIPConfig.py -action Enable_Demo_Application**

If you use this script, skip to the corresponding SCRIPT-END notice

We need to enable the Live Demo application on the Identity Provider as a target for our SAML Junction so that we'll be able to see the SAML token sent in an HTTP header.



Navigate to **Secure Federation > Manage > Advanced Configuration**

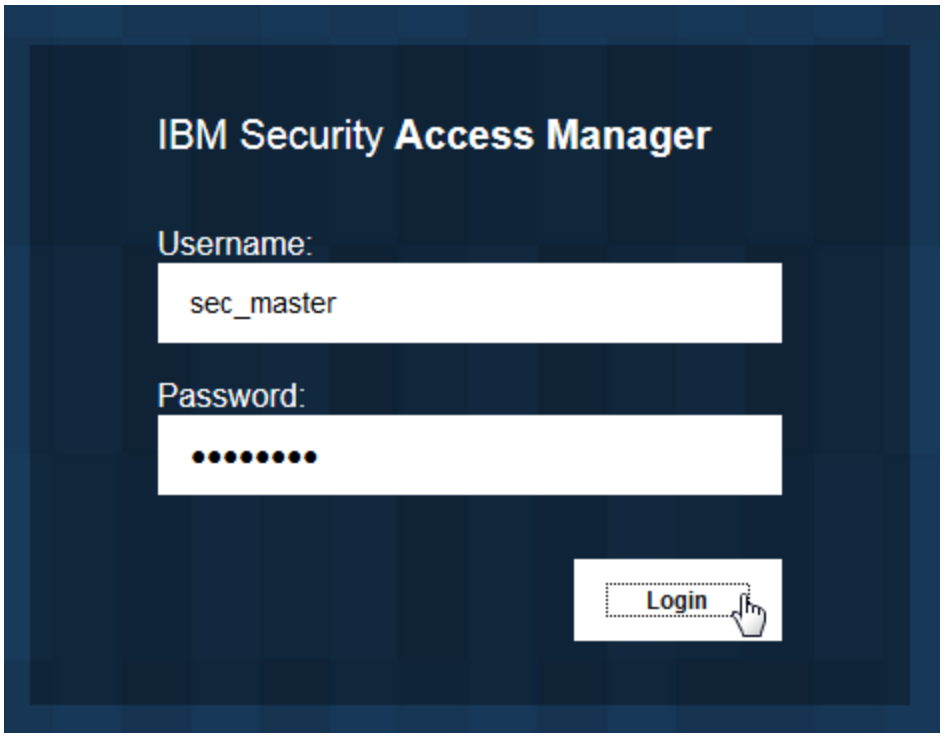


In the filter box seach for *demo*. Enable the **live.demos.enabled** key as shown above. Click **Save**.

A warning will be displayed at the top of the window. Click the link to activate the configuration change you have just made. A pop-up dialog is displayed showing the pending changes. **Deploy** the changes.

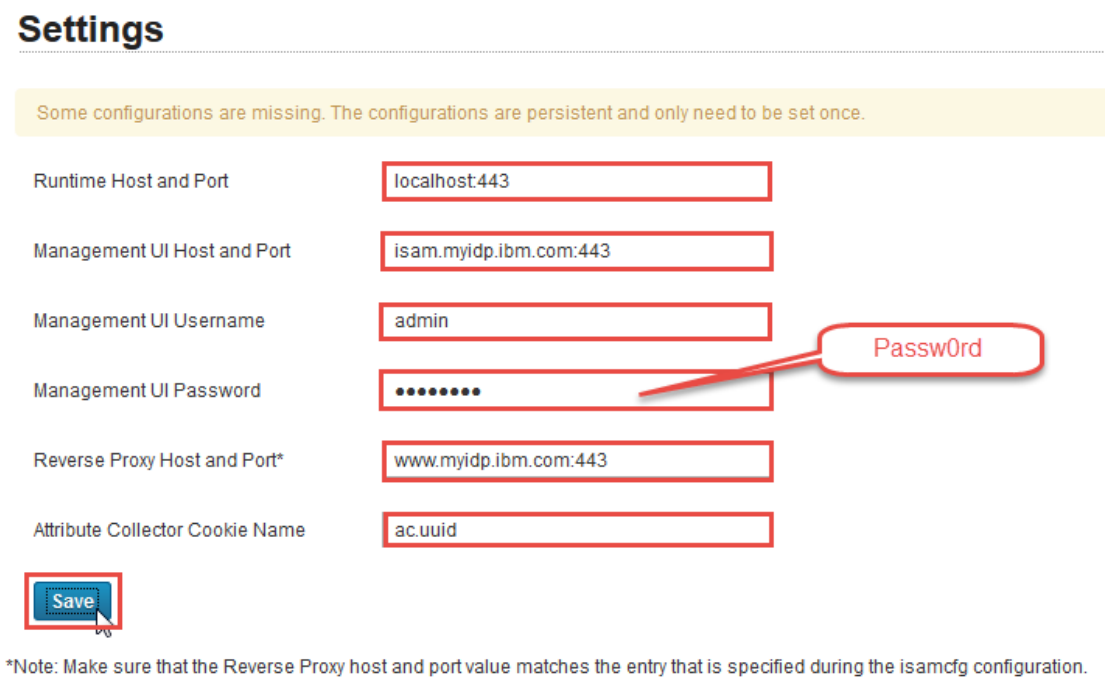
Once deployment is complete, navigate to:

https://www.myidp.ibm.com/isam/mobile-demo



Login with username **sec_master** and password **Passw0rd**

You will see a settings screen. This screen will be shown only for the first time during demo application configuration.



Enter the details that are shown in the form above and click **Save**. A success message as shown.

18.6 Authorize Access to Mobile Demo Application

The demo application is located on the /isam junction which, by default, only allows access to specified resources. We need to modify the Access Manager authorization policy to grant authenticated users access to the demo application (at /isam/mobile-demo).

Open an SSH session to the IdP appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

Authenticate with **admin** and **Passw0rd**.

Navigate to **isam** and start the **admin** utility:

```
isam.myidp.ibm.com> isam
isam.myidp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Enter the following command to attach the *default-webseal* ACL to the demo application. This ACL grants access to all authenticated users:

```
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/mobile-demo default-webseal
```

Type **exit** twice to end the session.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Setting demo application settings
INFO:FederationManager:Successfully set demo application settings

18.7 Test the /samljct Junction

You can test that the /samljct junction is configured and working properly by accessing the following URL with your browser:

https://www.myidp.ibm.com/samljct/mobile-demo/diag/

Login with “testuser” and “Passw0rd” at the login page.

Notice in the HTTP Headers section of the diagnostics page that the samlassertion header is populated:

HTTP Headers:

x-trusteer-rapport:	ak=358BA11FD748513C8B0F93B2D2BE55F8D9F1156A0544ED42D797E82B416C48B5; av=a3; rs=0.00171; i=0
iv_server_name:	default-webseald-isam.myidp.ibm.com
dnt:	1
samlassertion:	<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org /2001/XMLSchema-instance" ID="Assertion-uuid11af6a98-0151-1843-94b4-ad968d59f038" IssueInstant="2015-11-16T19:04:47Z" Version="2.0"><saml:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">https://www.myidp.ibm.com/isam/sps /saml20idp/saml20</saml:Issuer><ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="uuid11af6a9f-0151-1b7c-a862-ad968d59f038"><ds:SignedInfo><ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod> <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512">

This concludes setup and test of the SAML junction.

19 Advanced External Authentication Interface Configuration for Service Providers

This section includes exercises that demonstrate the different ways in which the federation runtime, in the role of a service provider authenticating to the reverse proxy, can send authentication data via the external authentication interface (EAI).

When acting as a service provider, the federation runtime is responsible for “logging in” to the WebSEAL reverse proxy. The federation runtime performs this authentication using EAI response HTTP headers. The EAI mechanism of authenticating to the reverse proxy has been available for many releases. What is important to know with ISAM 9 is how to influence the set of HTTP headers that the federation runtime will return, and the effect that will have on the ISAM session credential that is created in the reverse proxy.

19.1 Patterns of EAI Authentication

There are three labelled “patterns” of authentication that will be discussed and shown in these exercises:

- “USERNAME ”
- “PAC”
- “EXTUSER”

A brief description of each follows, then we will get into some exercises showing how the patterns vary, and the system requirements for each.

19.1.1 USERNAME method of EAI Authentication

The Username authentication mechanism is the default for ISAM 9. If no specific configuration changes are made to the system, this is the authentication mechanism that will apply. The USERNAME method of EAI authentication results in the federation runtime sending back a simple text username header, plus other headers for “extended attributes” to be added into the credential.

When this method is used, the ISAM reverse proxy will consult the ISAM user registry and the user MUST be in the registry. If the user is not in the registry, an error will occur. If the user is in the registry, an ISAM credential will be created that contains the user’s UUID, and all groups that user is a member of in the user registry. Any additional attributes will then be added into the credential to build the final credential for the session.

This is particularly useful in scenarios where only known, pre-registered users will be performing SSO into the service provider and user or group-based access control is required.

19.1.2 PAC method of EAI Authentication

When the PAC mechanism is used, the federation runtime (Specifically the IVCred STS Module) constructs a completely formed ASN-1 encoded ISAM credential and send that back as the “am-eai-pac” HTTP header during authentication.

It is important to note that the ISAM user registry is never accessed during this process and typically the PAC will not contain any user or group UUID information from the ISAM registry. This means that the resulting credential in the reverse proxy will not contain user or group information that can apply to ISAM ACL’s, and ACL-based access control in the ISAM reverse proxy is limited to “any-other” – that is any authenticated user. The PAC approach to authentication is discussed in some detail in this old article on TFIM, and this approach is often now called the “WebSEAL without a user registry” pattern of authentication:
http://www.ibm.com/developerworks/tivoli/library/t-tamwebsealur/

In ISAM, unlike TFIM, you cannot configure the IVCred STS module to use “pdacld” to build ISAM credentials, so if the federation runtime is returning the PAC authentication header, that PAC will never contain user or group UUID’s unless they have been programmatically added to the STSUU by a mapping rule before the IVCred module runs.

This method is particularly useful for scenarios where you don’t have a requirement for group-based access control at the service provider, and/or when you don’t know the users that will be arriving ahead of time (e.g. social authentication).

19.1.3 EXTUSER method of EAI Authentication

The EXTUSER method of EAI authentication is somewhat a hybrid of the first two approaches and makes use of relatively new EAI capability in the ISAM reverse proxy that was introduced in version 8.x. This method allows a username to be returned in a HTTP header which does NOT have to be a member of the ISAM registry, plus a list of groups that DO have to be in the ISAM registry, plus a set of extended attributes to be added into the credential.

When this method is used, the ISAM reverse proxy will create an empty credential with the non-verified username, then consult the ISAM registry for each of the group names that have been returned and build the group UUID’s into the credential, then add in all extended attributes.

This is particularly useful for using groups for role-based access control in ISAM, without having to pre-register all the usernames that will be used. By way of example, you might use attributes in an incoming SAML assertion to decide which “dynamic groups” to assign to the user credential for this session.

19.2 Configuration of EAI Authentication Headers

The configuration of EAI headers is really a handshake between the ISAM reverse proxy (header consumer) and the federation runtime (header producer). It is important that the values match on both components.

19.2.1 EAI Authentication Headers in ISAM Reverse Proxy

The ISAM reverse proxy contains configuration of what the header names are for various EAI header functions. This is already well documented in the knowledge center:

https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.1/com.ibm.isam.doc/wrp_config/concept/con_ext_authe_intfc.html

For the exercises in this cookbook, we recommend the following settings in your ISAM reverse proxy configuration:

EAI Authentication Configuration Property	Recommended Value
eai-pac-header	am-eai-pac
eai-user-id-header	am-eai-user-id
eai-xattrs-header	am-eai-xattrs
eai-ext-user-id-header	am-eai-ext-user-id
eai-ext-user-groups-header	am-eai-ext-user-groups

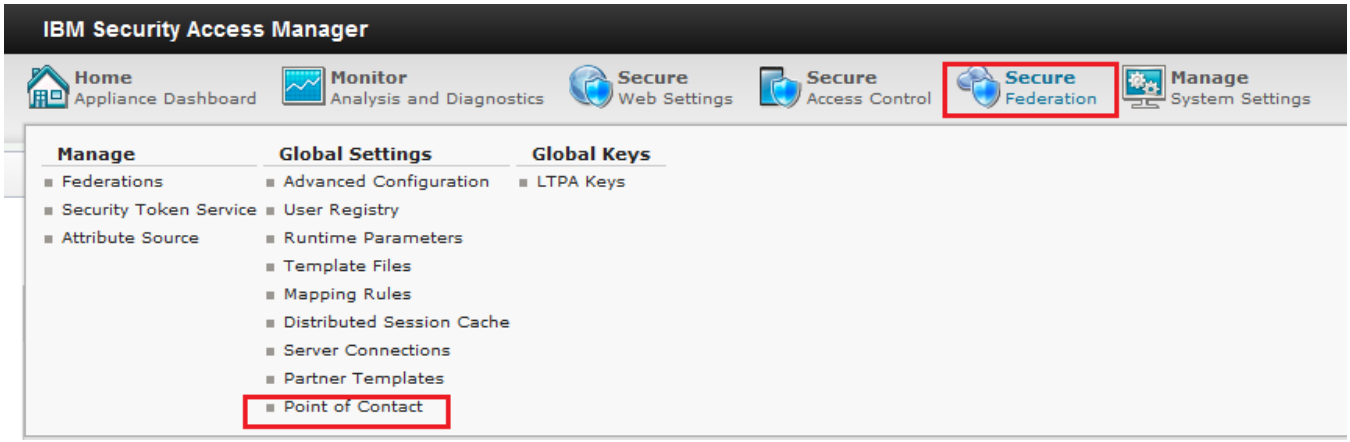
These are the default values in ISAM 9.0 and above.

19.2.2 EAI Authentication Headers in Federation Runtime

The federation runtime can be configured to send back (or not send back) HTTP headers for the PAC, username, groups, and extended attributes. These are controlled by Point of Contact "Profiles" that can be selected from the

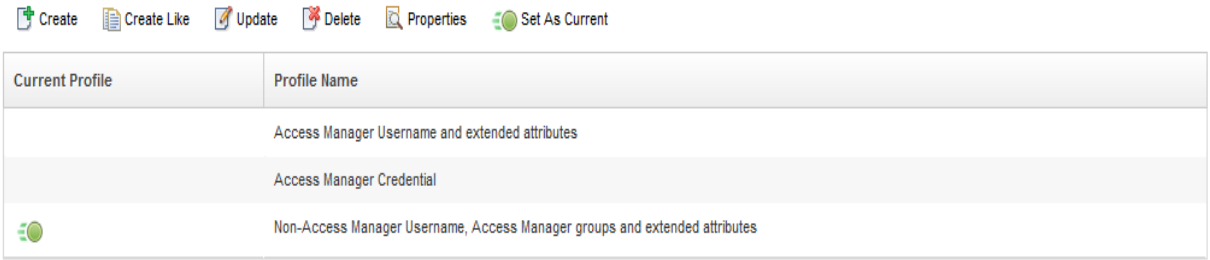
web console (or using the REST management interface). Default profiles exist for the Username, PAC, and External User patterns.

19.2.3 Point of Contact (PoC) Profiles Management



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

This UI comes up with the three pre-loaded default PoC Profiles which cannot be updated or deleted.



1. Access Manager Username and extended attributes (Refers to USERNAME EAI)
2. Access Manager Credential. (Refers to PAC EAI)
3. Non-Access Manager Username, Access Manager groups and extended attributes. (Refers to EXTUSER EAI)

19.3 Scenario: "USERNAME" authentication

In this scenario we will enable username authentication headers. This scenario requires that the "testuser" user account exists at the SAML SP.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the SP image only.

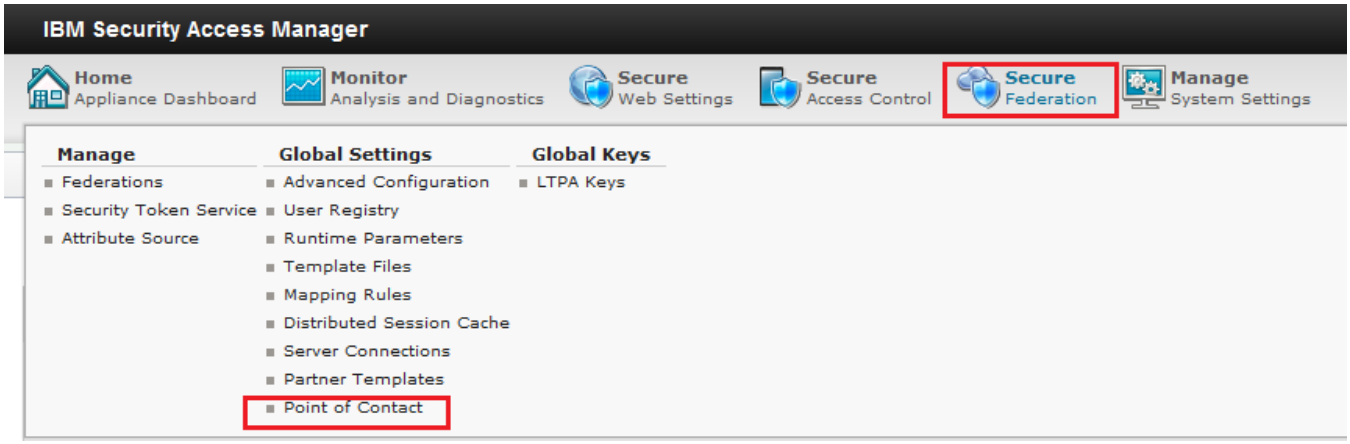
This is only for the SP, run this script: **SAMLSPConfig.py -configure PoC_Use_USERNAME**

If you use this script, skip to the corresponding SCRIPT-END notice

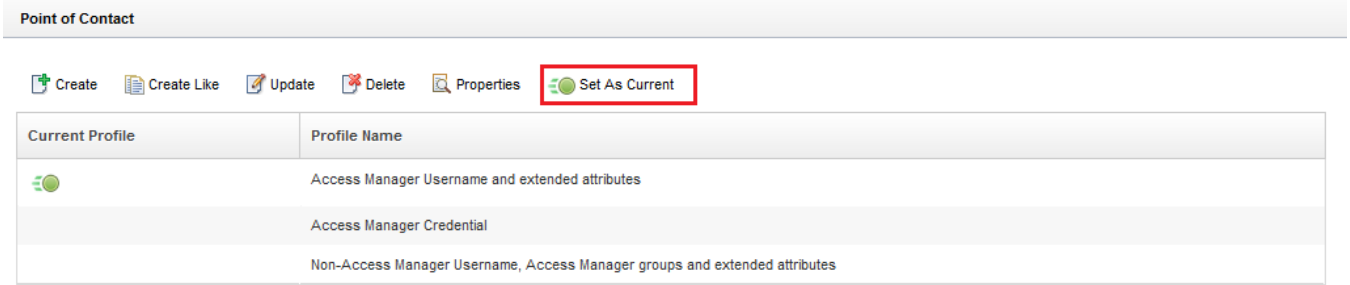
The following manual steps are only valid for SAM 9.0.1.0 onwards. For SAM 9.0.0.0 and 9.0.0.1, see section 0 - .

Go to the LMI Admin console of the SP using URL: https://isam.mysp.ibm.com

Authenticate with **admin** and **Passw0rd**.



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



Select the "Access Manager Username and extended attributes" PoC Profile and Click "**Set As Current**" button to set this profile as Current Profile.

Deploy pending change.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Set Advanced Configuration Parameters to use USERNAME
INFO:FederationManager:Successfully set Advanced Configuration Parameters to use USERNAME

Perform SAML SSO, and you should observe successful SSO:

Access Manager Credential:
User: **testuser**

tagvalue_session_index[0]	c10150e2-71bb-11e5-a34b-000c29adc316
AZN_CRED_IP_FAMILY[0]	AF_INET
AZN_CRED_PRINCIPAL_UUID[0]	ec6ebc08-71b9-11e5-9541-000c29adc316
AZN_CRED_QOP_INFO[0]	SSK: TLSV12: 0A
AZN_CRED_AUTHZN_ID[0]	testuser
AudienceRestrictionCondition.Audience[0]	https://www.mysp.ibm.com/isam/sps/saml20sp/saml20
AZN_CRED_PRINCIPAL_DOMAIN[0]	Default
AZN_CRED_REGISTRY_ID[0]	cn=testuser,dc=iswga
am_eai_xattr_session_lifetime[0]	1444752285
AZN_CRED_PRINCIPAL_NAME[0]	testuser
tagvalue_login_user_name[0]	testuser
tagvalue_max_concurrent_web_sessions[0]	unset
testattr[0]	myvalue

You should peruse all the attributes in the ISAM credential to understand how this credential differs from that created using the other methods.

19.4 Scenario: "PAC" authentication

In this scenario no user account is required at the SP, and the federation runtime will build an "asserted user" credential for the session.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the SP image only.

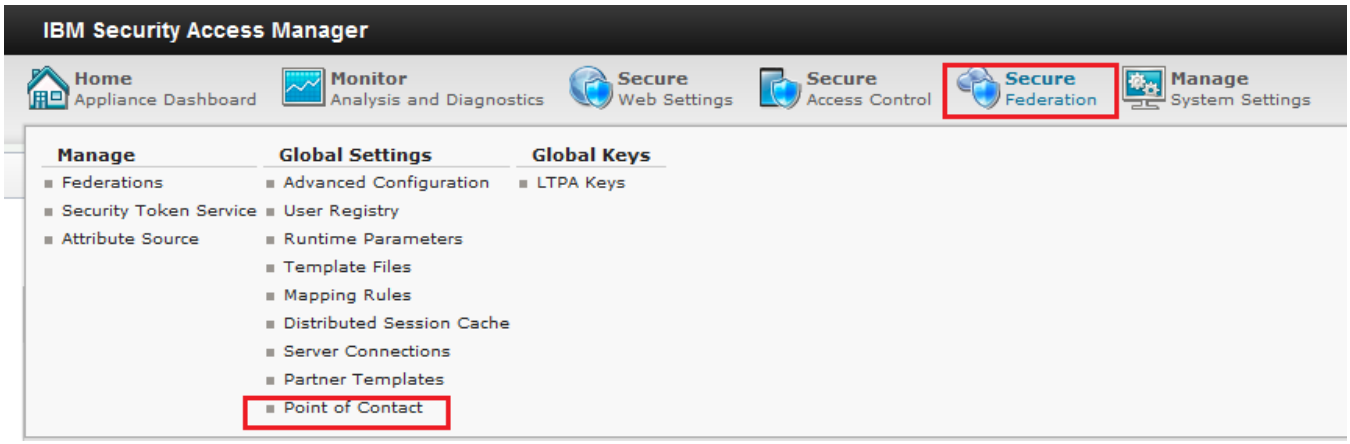
This is only for the SP, run this script: **SAMLSPConfig.py –configure PoC_Use_PAC**

If you use this script, skip to the corresponding SCRIPT-END notice

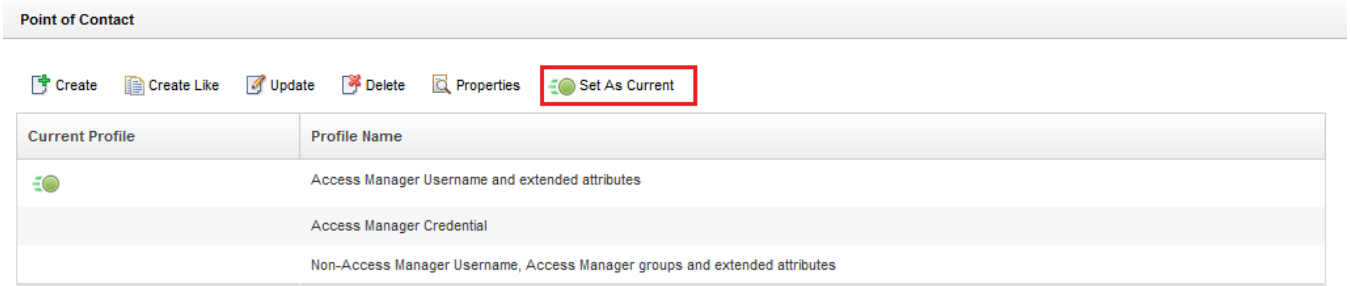
The following manual steps are only valid for SAM 9.0.1.0 onwards. For SAM 9.0.0.0 and 9.0.0.1, see section 0 - .

Go to the LMI Admin console of the SP using URL: https://isam.mysp.ibm.com

Authenticate with **admin** and **Passw0rd**.



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



Select the "Access Manager Credential" PoC Profile and Click "**Set As Current**" button to set this profile as Current Profile.

Deploy pending changes.

SCRIPT-END:

The script should display the following:

INFO:FederationManager:Set Advanced Configuration Parameters to use PAC
INFO:FederationManager:Successfully set Advanced Configuration Parameters to use PAC

Of course you can optionally also delete any related ISAM users from the SP user registry because they will no longer be relevant.

You can manage ISAM user using the LMI, by script, or manually. At this point in the cookbook it is expected you know how to do this, and no auto-configuration script is provided to do so. We will show "isam admin" commands here to complete this task if you are logged in to the SP's administration command line:

```
user delete -registry testuser
```

Perform a SAML SSO and observe closely the format of attributes in the final user credential at the service provider. The test should work regardless of whether or not the username in the credential is a user in the ISAM registry.

19.5 Scenario: "EXTUSER" authentication

In this scenario we will first enable authentication headers to support returning username and group names, plus extended attributes. We will then replace the mapping rule at the SP SAML federation with one that inserts a couple of dynamic groups names (testgroup, testgroup2) into the STSUU such that these groups will be returned in the am-ext-user-groups header to the reverse proxy. Performing SSO at this point will fail due to testgroup and testgroup2 not being in the user registry. We will then add testgroup and testgroup2 to the registry and show successful SSO with multiple groups in the credential.

SCRIPT-START:

A script is available for this section as an alternative to following the manual steps. This script performs operations against the SP image only.

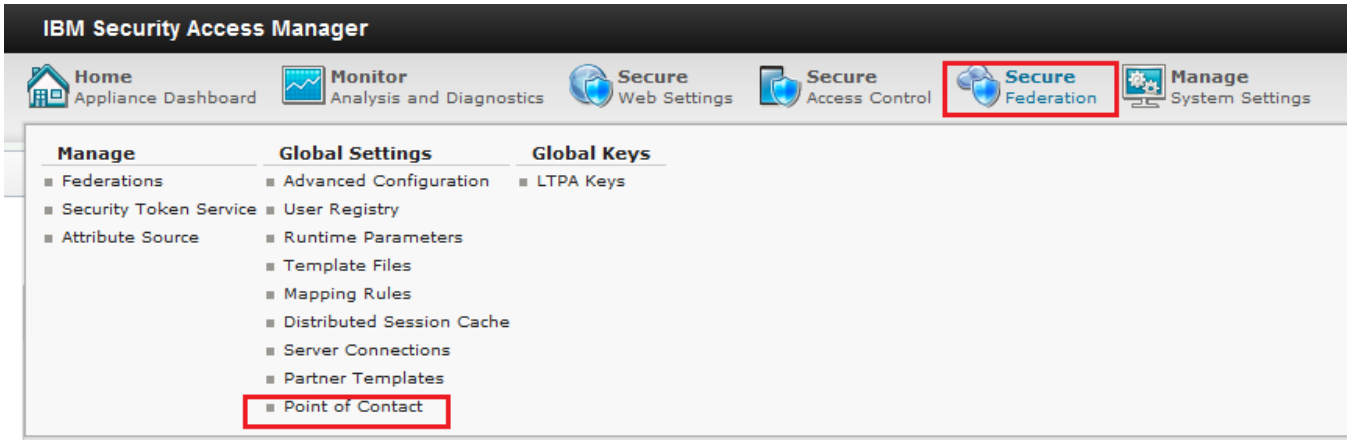
This is only for the SP, run this script: **SAMLSPConfig.py –configure PoC_Use_EXTUSER**

If you use this script, skip to the corresponding SCRIPT-END notice

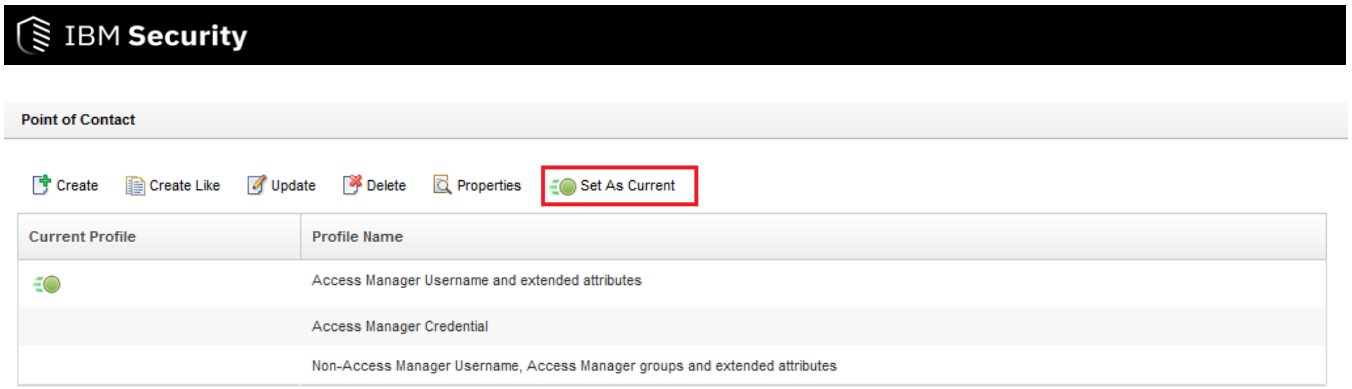
The following manual steps are only valid for SAM 9.0.1.0 onwards. For SAM 9.0.0.0 and 9.0.0.1, see section 0 - .

Go to the LMI Admin console of the SP using URL: https://isam.mysp.ibm.com

Authenticate with **admin** and **Passw0rd**.



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



Select the "Non-Access Manager Username, Access Manager groups and extended attributes" PoC Profile and Click "**Set As Current**" button to set this profile as Current Profile.

Deploy pending changes.

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Set Advanced Configuration Parameters to use EXTUSER
INFO:FederationManager:Successfully set Advanced Configuration Parameters to use EXTUSER

If you were to perform SAML SSO at this time, using the default sp_saml20.js mapping rule in the SAML federation, SSO will succeed regardless of whether or not testuser exists in the user registry. As the mapping rule does not populate any groups into the STSUU, the am-eai-ext-groups response header will not be populated and the net effect is very similar to using the PAC authentication method.

Verify SAML SSO is working.

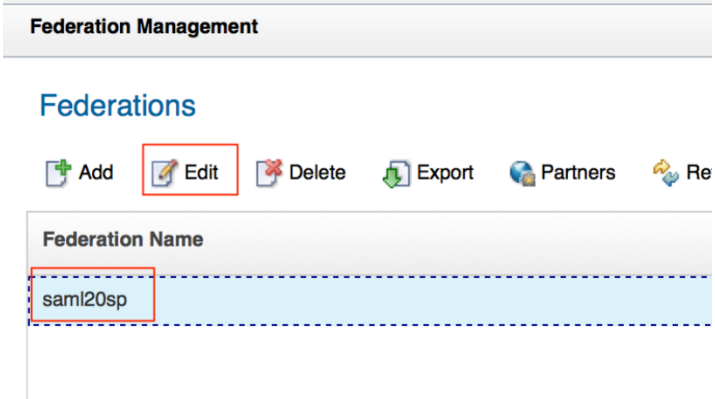
Now replace the **sp_saml20.js** mapping rule with the **sp_saml20_dynamic_group.js** mapping rule:

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the SP image only

Run this script: **UploadSPMappingRule.py –configure dynamicGroupMapping**

If you use this script, skip to the corresponding SCRIPT-END notice

Using the administration console on the SP, navigate to **Secure Federations -> Manage: Federations**.



Select the **saml20sp** federation and press edit:

Keep clicking **Next** until you reach the *Identity Mapping Rule screen* in the wizard.

Select the **sp_dynamic_group.js** mapping rule.

Press **Next**, then **OK** to save the federation configuration updates.

Deploy pending changes.

The **sp_dynamic_group.js** mapping rule adds two groups to the STSUU during identity mapping at the Service Provider. These are:

- testgroup
- testgroup2

If you have been completing these cookbook exercises in order, neither of these groups will exist in the SP user registry at this stage. As a result, attempts to perform SAML SSO at this point will fail:

Server Error

Access Manager WebSEAL could not complete your request due to an unexpected error.

Diagnostic Information

Method: *POST*

URL: */isam/sps/saml20sp/saml20/login*

Error Code: *0x13212072*

Error Text: *HPDIA0114E Could not acquire a client credential.*

Use the isam administration utility to create both testgroup and testgroup2:

```
group create testgroup cn=testgroup,dc=iswga testgroup
group create testgroup2 cn=testgroup2,dc=iswga testgroup2
```

> There is no need to add any users to this group. They exist so that they can be added to user credentials at runtime.

SCRIPT-END:
The script should display the following:
INFO:UploadSPMappingRule:Configuring Dynamic Group Mapping
INFO:FederationManager:Modifying SP to change mapping rule
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully modified the Federation using PUT
INFO:FederationManager:Creating testgroup and testgroup2
INFO:UploadSPMappingRule:Successfully configured Dynamic Group Mapping

Now perform SAML SSO again and notice that the credential contains both testgroup and testgroup2 group membership:

Access Manager Credential:
User: **testuser**

AZN_CRED_GROUP_UUIDS[0]	f3b112ca-8c9b-11e5-bdd4-000c29775921
AZN_CRED_GROUP_UUIDS[1]	f82c650c-8c9b-11e5-bdd4-000c29775921
AZN_CRED_AUTHZN_ID[0]	testuser
AudienceRestrictionCondition.Audience[0]	https://www.mysp.ibm.com/isam/sps/saml20sp/saml20
AZN_CRED_PRINCIPAL_DOMAIN[0]	Default
AZN_CRED_REGISTRY_ID[0]	cn=testuser,cn=ExternalUser
am_eai_xattr_session_lifetime[0]	1447707334
AZN_CRED_PRINCIPAL_NAME[0]	testuser
tagvalue_login_user_name[0]	testuser
AZN_CRED_NETWORK_ADDRESS_STR[0]	192.168.42.1
NotOnOrAfter[0]	2015-11-16T20:00:34Z
AZN_CRED_GROUPS[0]	testgroup
AZN_CRED_GROUPS[1]	testgroup2

The group names are human readable (and can be asserted downstream in HTTP headers. The Group UUIDs are how Access Manager references groups in Access Control Lists.

Advanced exercise: Try attaching an ACL to the /isam/mobile-demo application and allowing only users of testgroup2 to access that application. Change the authentication method back to PAC and observe how access is denied because the PAC does not contain ISAM group information for testgroup2.

<u>This concludes exercises related to the different ways authentication be achieved between the ISAM federation runtime and the ISAM reverse proxy.</u>

20 Advanced OIDC: Configuring access policies to showcase prompt=login during OpenID Connect flow

This section is completed only for the Identity Provider.

In this section, an access policy is created to show case a scenario where the user is forced to authenticate every time is the prompt parameter is set to login. This section involves activating Advanced Access Control, configuring username password authentication mechanism and configuring an access policy for the API Definition created in the previous.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

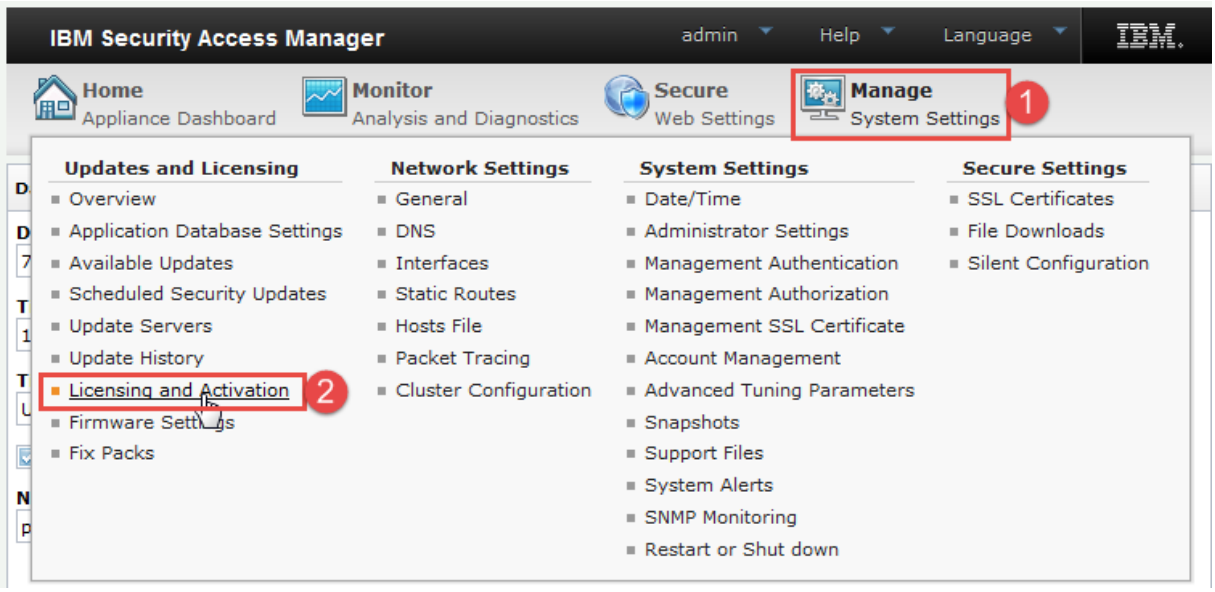
For the IdP, run this script: **OIDCWithAccessPolicy.py -configure All**

20.1 Activation Advanced Access Control

In this section, the Advanced Access Control module is activated on the appliance

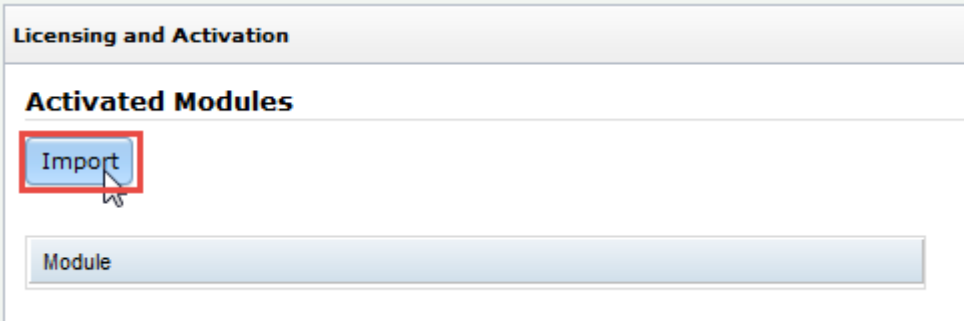
SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

For the IdP, run this script: **OIDCWithAccessPolicy.py -configure Activate_AAC**



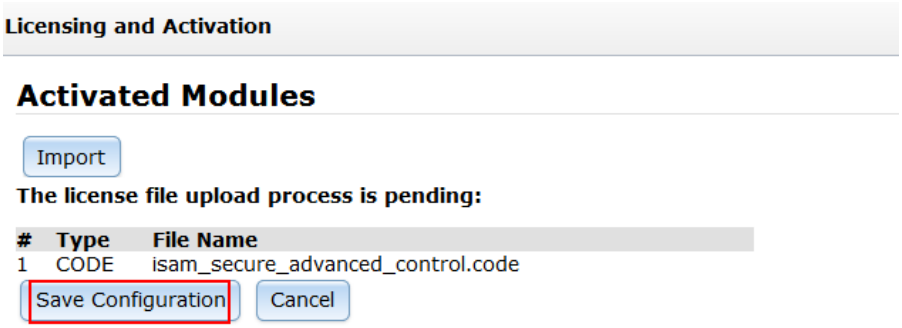
Click on the ***Manage System Settings*** icon to open the "mega-menu" and click the ***Licensing and Activation*** item - as shown above.

The licensing and Activation screen is displayed. Currently there are no activated modules.

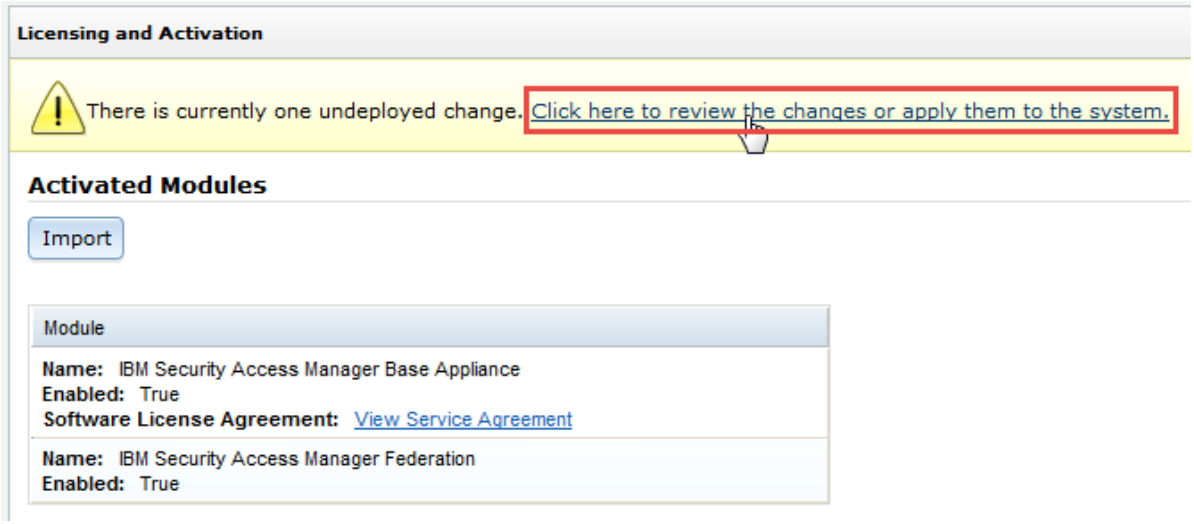


Click the ***Import*** button. A file selector dialog is displayed.

Select the ISAM 9.0.6 Advanced Access Control Activation File that you downloaded from IBM

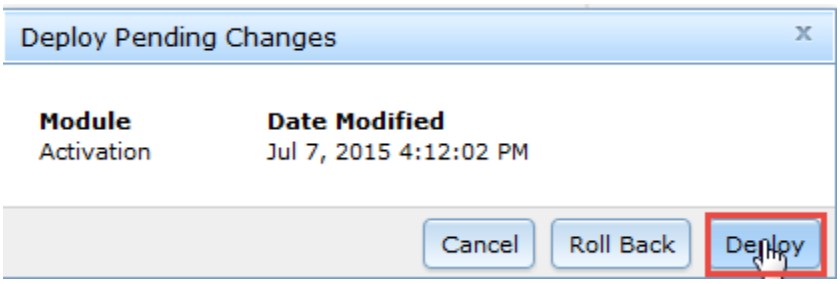


The activation code is processed. Now both IBM Security Access Manager Base Appliance ,IBM Security Access Manager Federation and IBM Security Access Manager Base Advanced Access Control are listed:



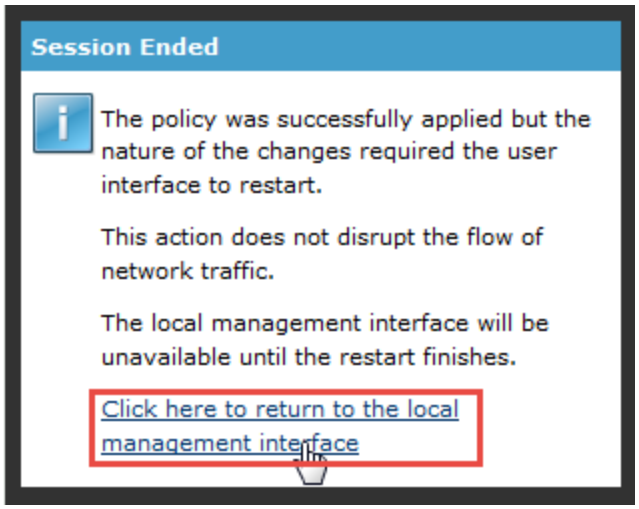
To complete the activation process we must deploy the changes we have made.

Click the ***Click here to review the changes or apply them to the system*** link in the warning message - as shown above.



Click ***Deploy*** to confirm the deployment of the changes.

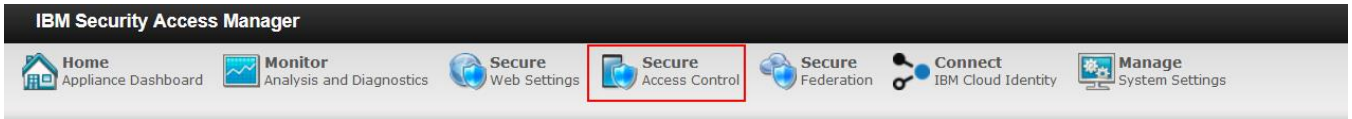
The activation process can take a few minutes to complete because a number of new components are started and initialized within the appliance. Once it is complete, the following message is displayed:



Click on the link in the message to reconnect to the appliance management interface (it may take a few seconds for this to work).

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Activating Advanced Access Control
INFO:BaseManager:Successfully activated Advanced Access Control

You should now see that **Secure Federation** mega-menu is available in the LMI Web Console:

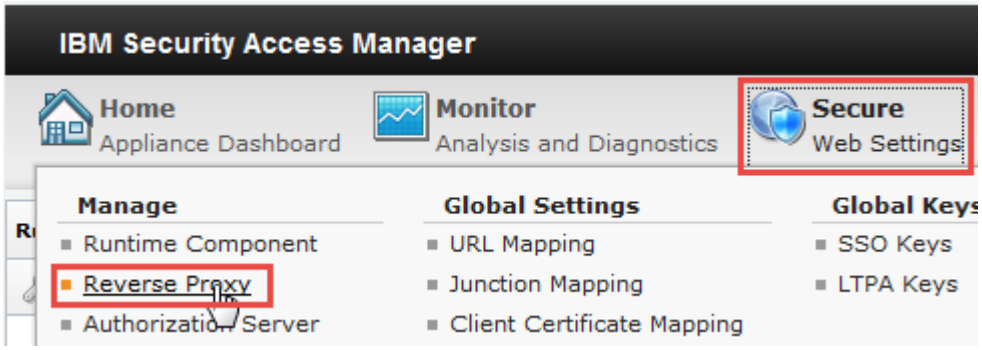


20.2 Configuring Reverse Proxy for Advanced Access Control

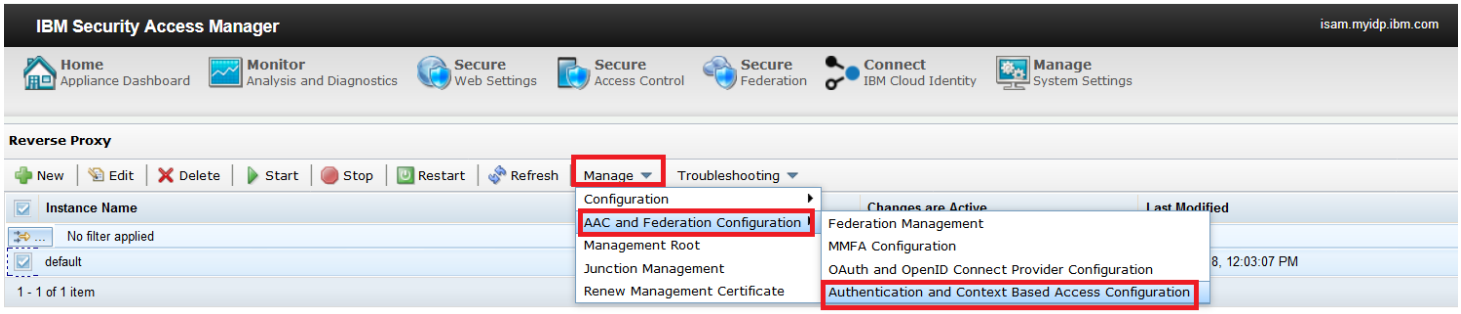
In this section, we configure the reverse proxy instance for Authentication and Context Based Access Configuration.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

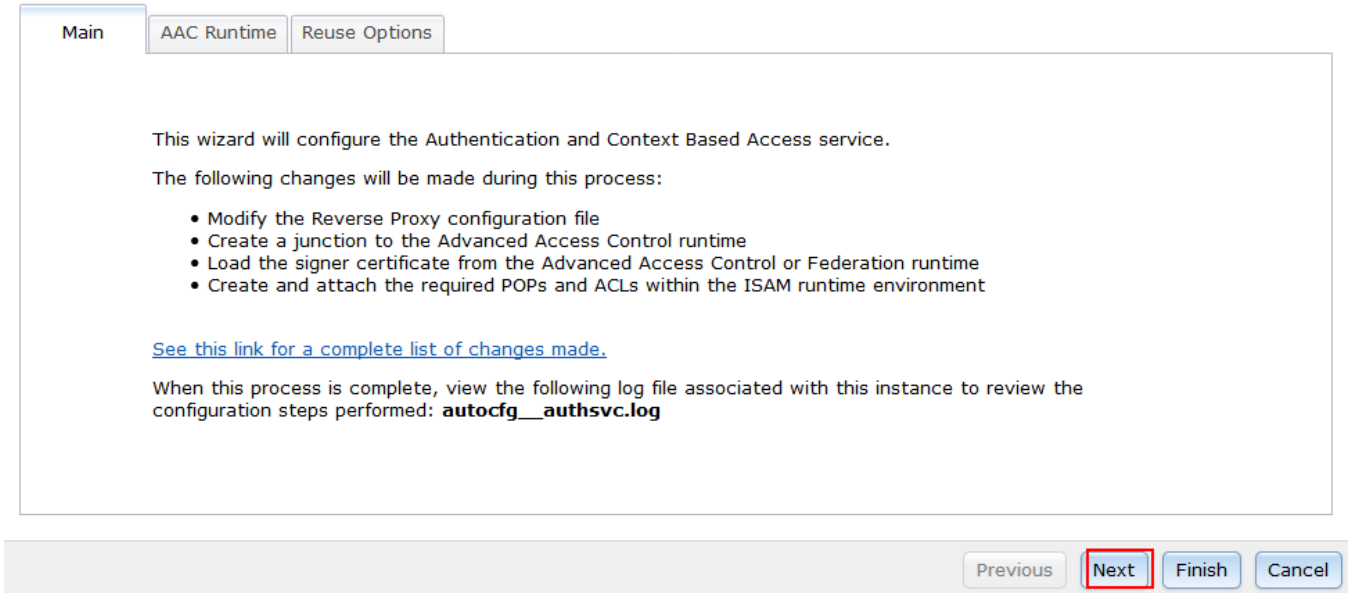
For the IdP, run this script: **OIDCWithAccessPolicy.py -configure WebSEAL_AAC**



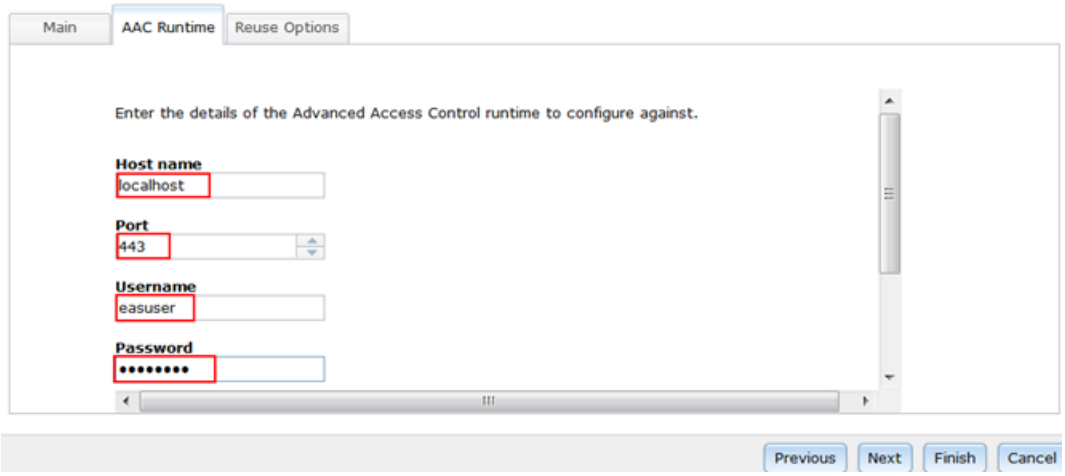
In the mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.



Select the Reverse Proxy instance, and click on **Manage -> AAC and Federation Configuration -> Authentication and Context Based Access Configuration.**

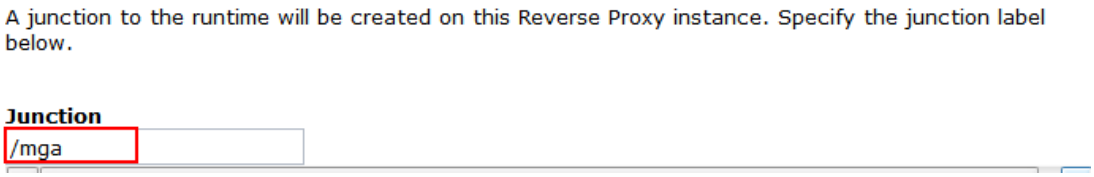


There are three panels which need to be filled out. In the **Main** panel, select all the checkboxes **Configure for browser interaction** – the /authorize and the /session endpoints are made accessible, **Configure for API Protection** – this configures the oauth-auth and oauth-cluster stanza, **Require authentication to register a client** – this sets an anyauth ACL to the client registration endpoint.



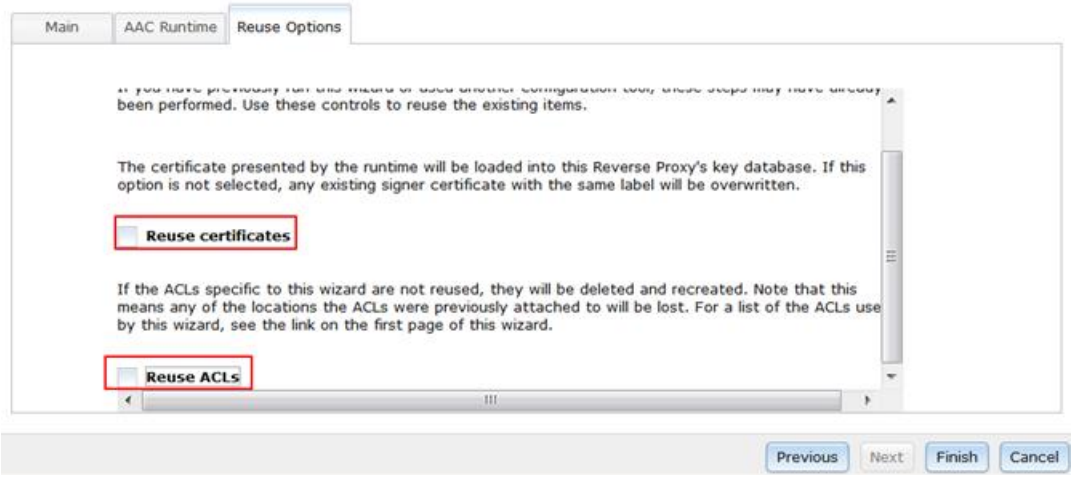
Click **Next.**

Inside the **AAC Runtime** pane, user has to provide the details to authenticate with federation runtime. The details include the host, port, user name and password. All of them are required. When you move to the next pane, these details are used to connect to the Runtime.



The default junction name used is /mga.

Click **Next.**



The next tab is the ACLs and Certificates panel, you can choose to reuse ACLs and Certificates if they exist or create new ones.

Once all the panels are done, click on **Finish** and then **Deploy** the Pending changes.

20.2.1 Configuring ACL

For prompt=login scenario we use the username password authentication mechanism to authenticate the user instead of WebSEAL login.

Hence, we change from authenticated access to unauthenticated access for '/mga/sps/auth' endpoint, so that the access policy is triggered to call the username password mechanism.

Note the unauthenticated access to '/mga/sps/auth' should be reset to authenticated for the other scenarios.

Open an SSH session to the appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

SSH to **isam.myidp.ibm.com** and authenticate using the administrator credentials:

```
The authenticity of host 'isam.myidp.ibm.com (192.168.42.101)' can't be established.
ECDSA key fingerprint is SHA256:hXml4xBfov+C9/4pxAgxh5IDh7BR4JUBbbbMnibPNPM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'isam.myidp.ibm.com,192.168.42.101' (ECDSA) to the list of known
hosts.
admin@isam.myidp.ibm.com's password: Passw0rd
Last login: Wed Sep 23 13:41:05 2015
Welcome to the IBM Security Access Manager
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.myidp.ibm.com>
```

Navigate to **isam** and start the **admin** utility:

```
isam.myidp.ibm.com> isam
isam.myidp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Create an unauth ACL using the commands:

```
acl create unauth-oidc
acl modify unauth-oidc set group iv-admin TcmdbsvaBRrxl
acl modify unauth-oidc set group webseal-servers Tgmdbsrxl
acl modify unauth-oidc set user sec_master TcmdbsvaBRrxl
acl modify unauth-oidc set any-other Tr
acl modify unauth-oidc set unauthenticated Tr
```

Attach the ACL to the SAML endpoints using the commands:

```
acl attach /WebSEAL/isam.myidp.ibm.com-default/mga/sps/auth unauth-oidc
```

Run server replicate command to save the changes using the command:

```
server replicate
```

SCRIPT-END:

The script should display the following for IdP:

INFO:WGAManager:Configure WebSEAL for AAC
INFO:WGAManager:Successfully configured WebSEAL for AAC
INFO:WGAManager:Configure ACLs for OIDC
INFO:WGAManager:Successfully configured ACLs OIDC

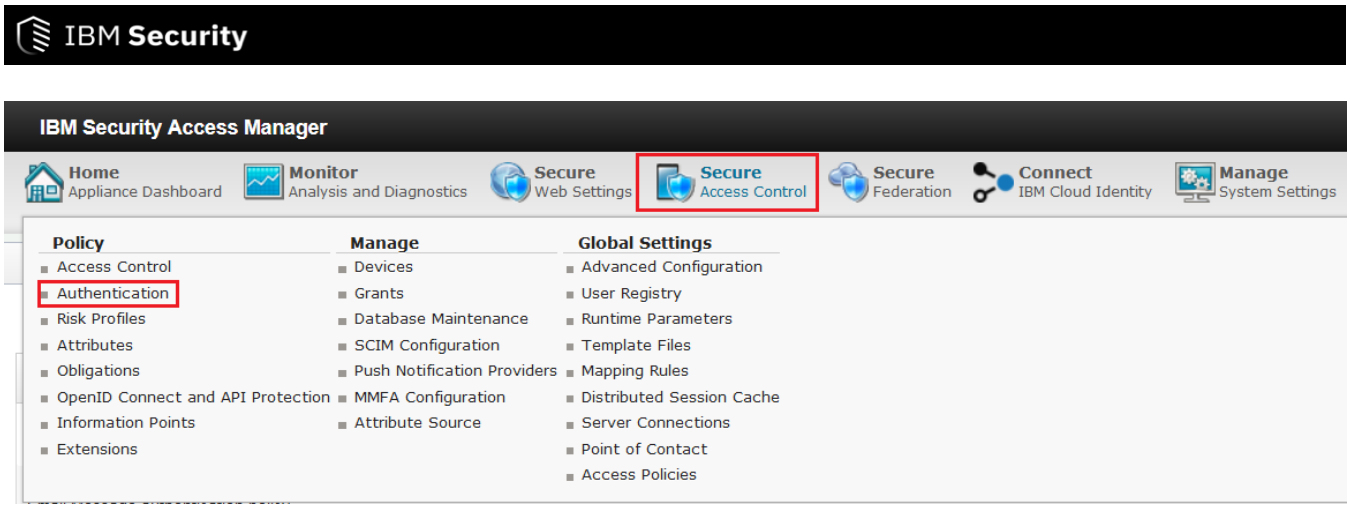
20.3 Configuring Username Password Authentication Mechanism

In this section we configure the Username Password Authentication Mechanism.

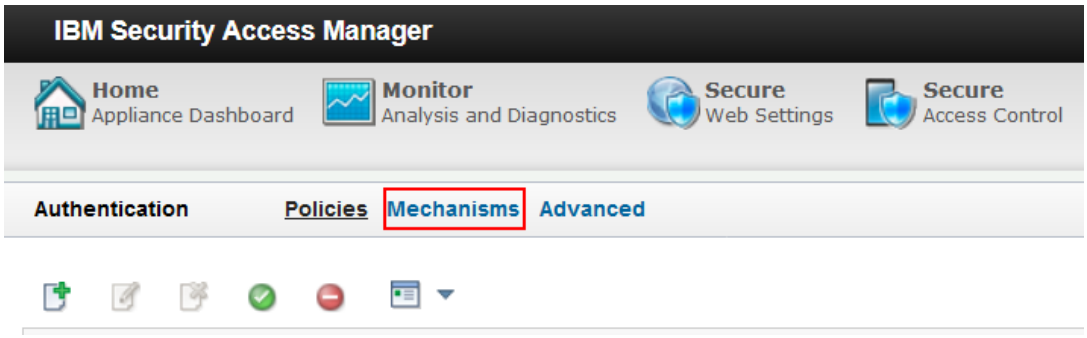
SCRIPT-START:

A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

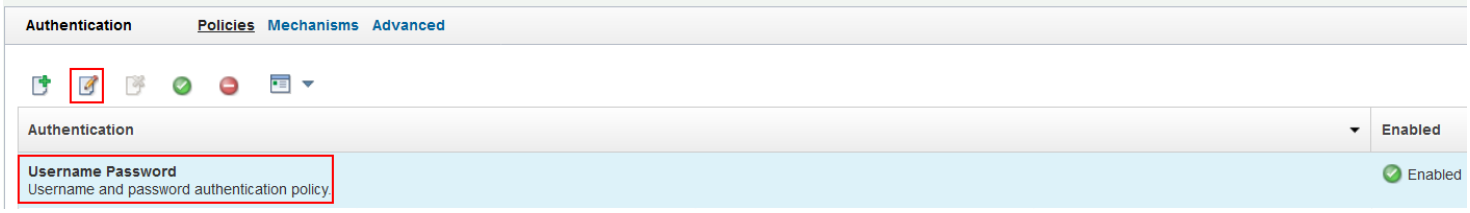
For the IdP, run this script: **OIDCWithAccessPolicy.py -configure UsernamePassword**



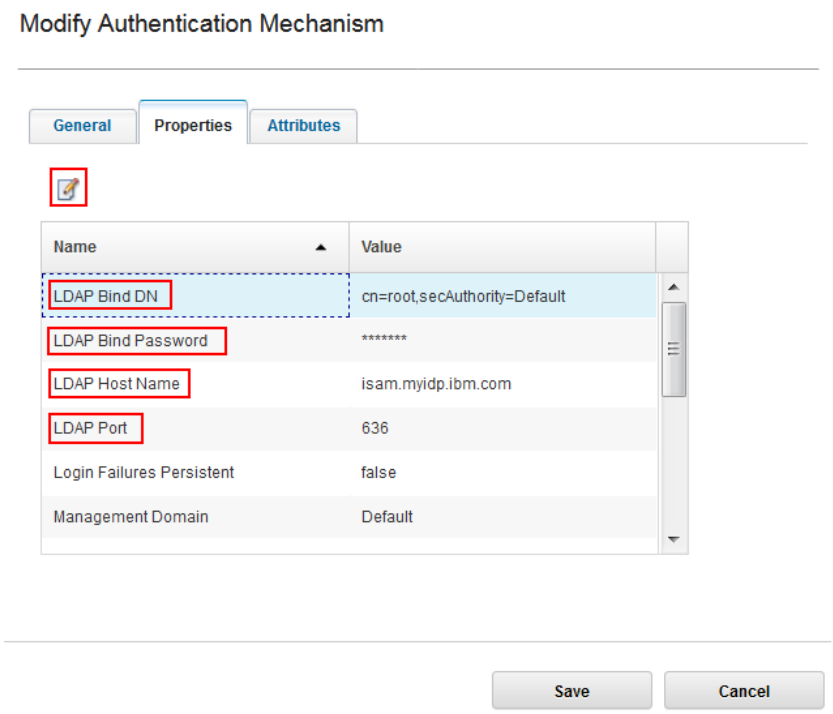
Navigate to Secure Access Control -> **Authentication** under the Policy menu.



Navigate to **Mechanisms**

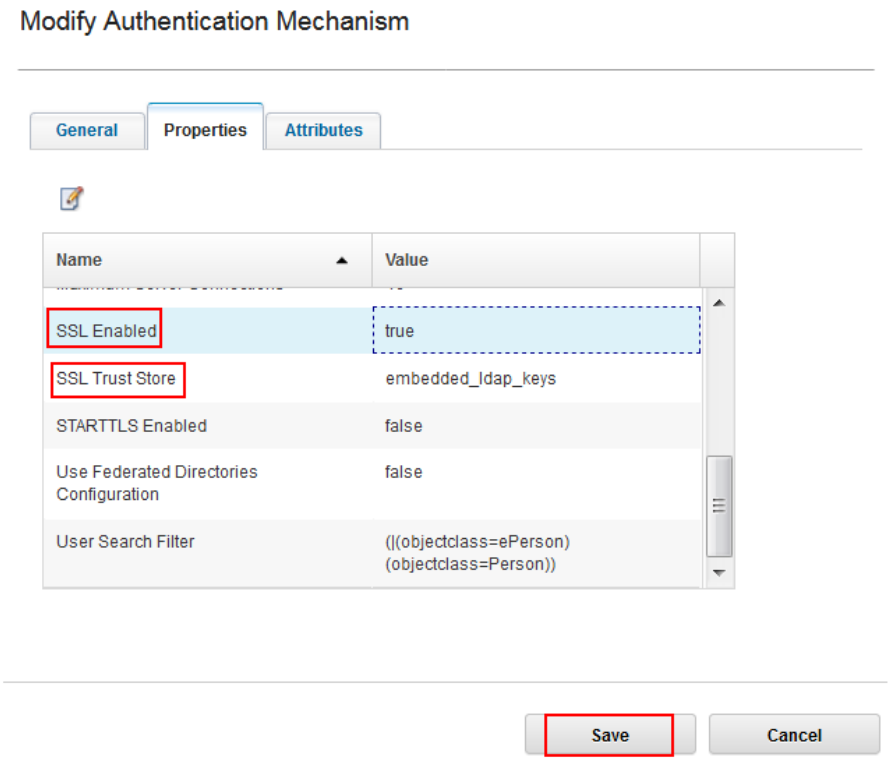


Select the **Username Password** mechanism and click the **Edit** button



Navigate to the **Properties** panel and edit the following entries

Enter **cn=root,secAuthority=Default** for LDAP Bind DN value, enter **Passw0rd** for LDAP Bind Password, enter **isam.myidp.ibm.com** for LDAP Host Name, enter **636** for LDAP Port value



Set SSL Enabled to **true** and select **embedded_ldap_keys** as the SSLTruststore.

Click on **Save** and deploy pending changes.

SCRIPT-END:
The script should display the following:
INFO:AACManager:Configuring Username Password Mechanism
INFO:AACManager:Successfully configured Username Password Mechanism

20.4 Configuring Definition with Access Policy

In this section an access policy is configured and the definition created in the previous section is updated to used the access policy.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

For the IdP, run this script: **OIDCWithAccessPolicy.py -configure AccessPolicy**

20.4.1 Configuring Access Policy

This scenario makes use of access policy which is written in JavaScript.

When using the appliance console to create Access Policy, cut-and-paste is used to load the JavaScript content of the policy. Before we get started, we need to open our first rule in a text editor so we can copy it.

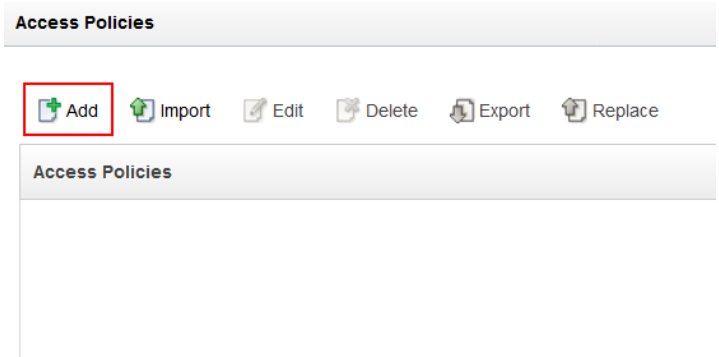
Go to the **…/providedfiles/idp_files** directory and open the **accesspolicy_prompt.js** file in a text editor.

```
importClass(Packages.com.ibm.security.access.policy.decision.Decision);
importClass(Packages.com.ibm.security.access.policy.decision.HtmlPageDenyDecisionHandler);
importClass(Packages.com.ibm.security.access.policy.decision.RedirectDenyDecisionHandler);
importClass(Packages.com.ibm.security.access.policy.decision.HtmlPageChallengeDecisionHandler);
importClass(Packages.com.ibm.security.access.policy.decision.RedirectChallengeDecisionHandler);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);

/*
 * Redirects a user to the authentication service, invoking the
 * username/password mechanism
 */
function getRedirectToAuthSvc() {
  var handler = new RedirectChallengeDecisionHandler();
  IDMappingExtUtils.traceString("rediect to authSvc");
  handler.setRedirectUri("
  https://www.myidp.ibm.com/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:password&Target=https://www.myidp.ibm.com/mga@ACTION@");
  IDMappingExtUtils.traceString("returning challenge");
  return handler;
}
var prompt_login_demo = true;
if(prompt_login_demo) {
  context.setDecision((function() {
    var request = context.getRequest();

    var user = context.getUser();
    if (user == null) {
      IDMappingExtUtils.traceString("User isnt authenticated");
      return Decision.challenge(getRedirectToAuthSvc());
    } else {
      // The current HTTP request contains authentication request.
      IDMappingExtUtils.traceString("Currently authenticated user: " + user);
```

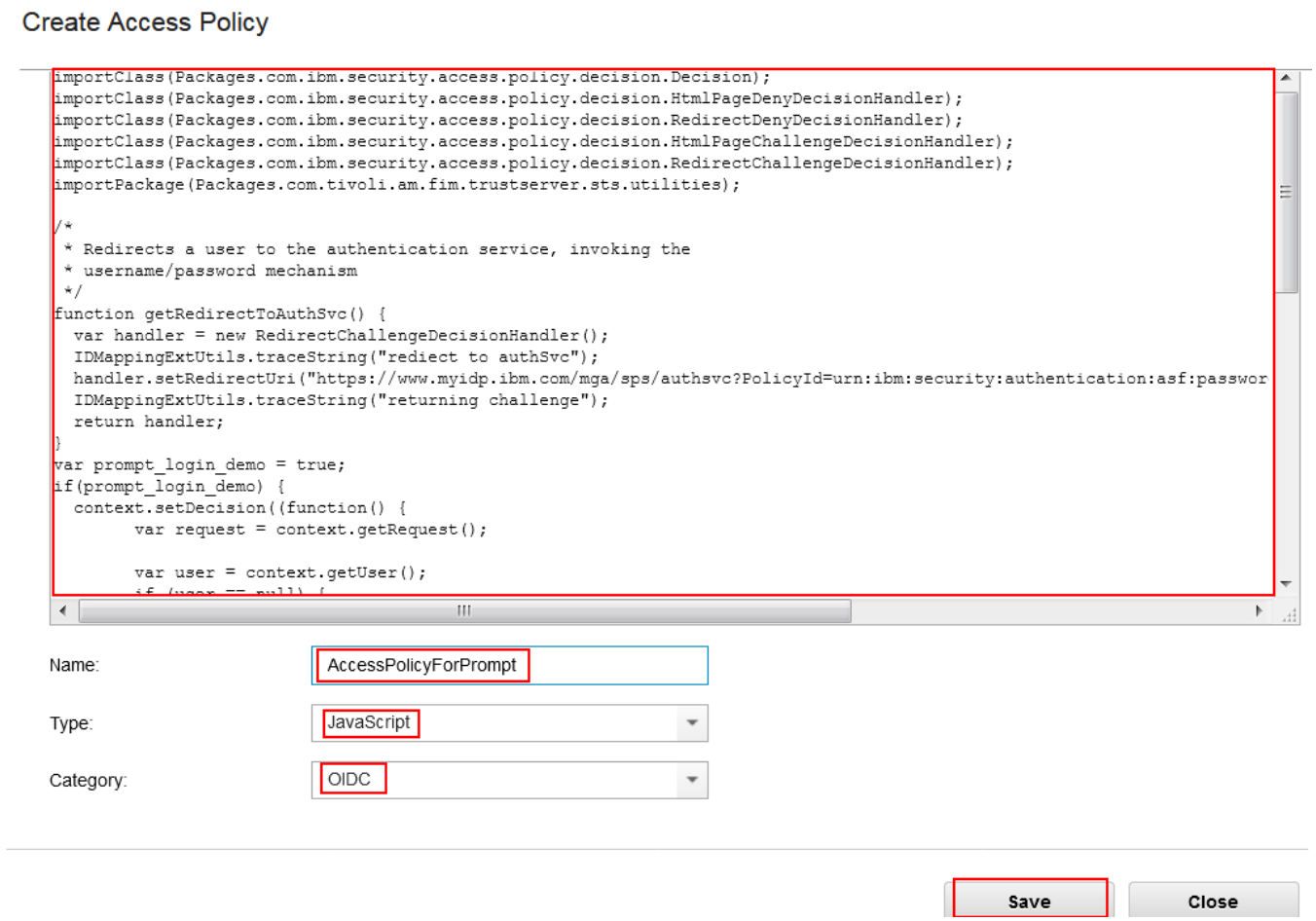
Select all the text in the file and then copy it. On Windows you can use **Ctrl-a** to select all and **Ctrl-c** to copy.

Now we're ready to create a Mapping Rule on the appliance with this content.

In the LMI Administration console, navigate to **Secure Federation→Global Settings: Access Policies**.



Click **Add** to add a new access policy.



Paste the rule text into the *Content* box. On Windows you can use **Ctrl-c** to paste.

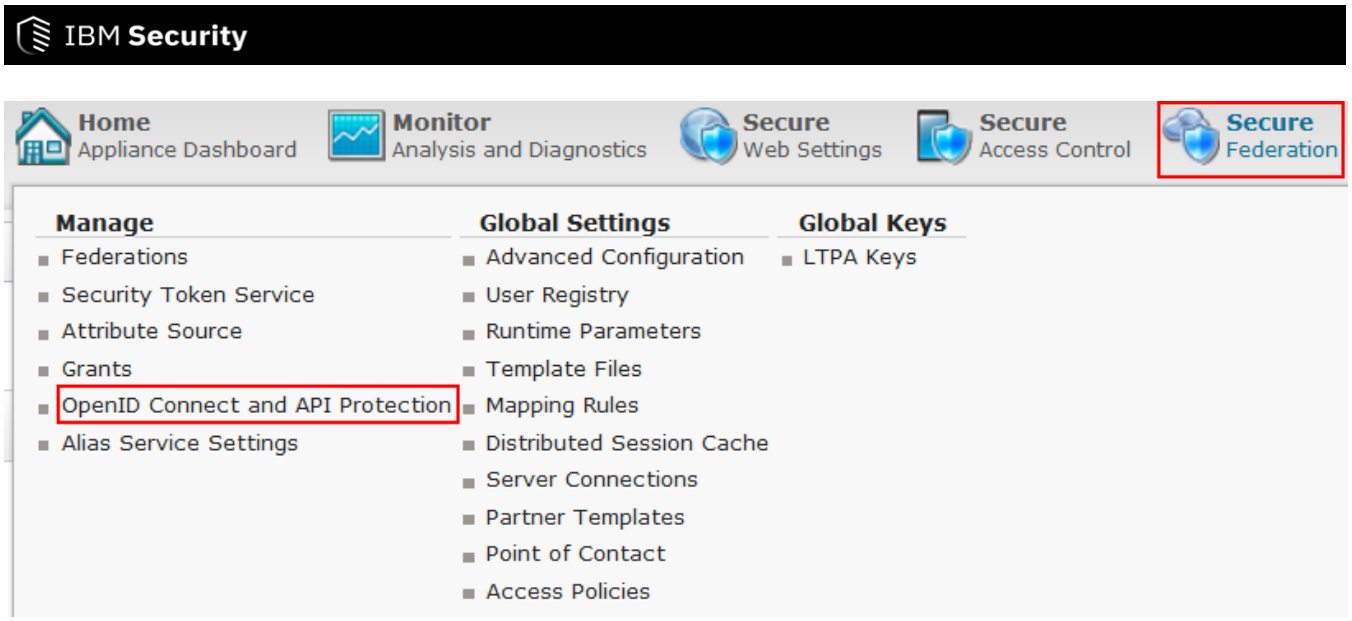
Enter **AccessPolicyForPrompt** as the *Name,* select **JavaScript** as the *Type* and enter **OIDC** as the *Category*.

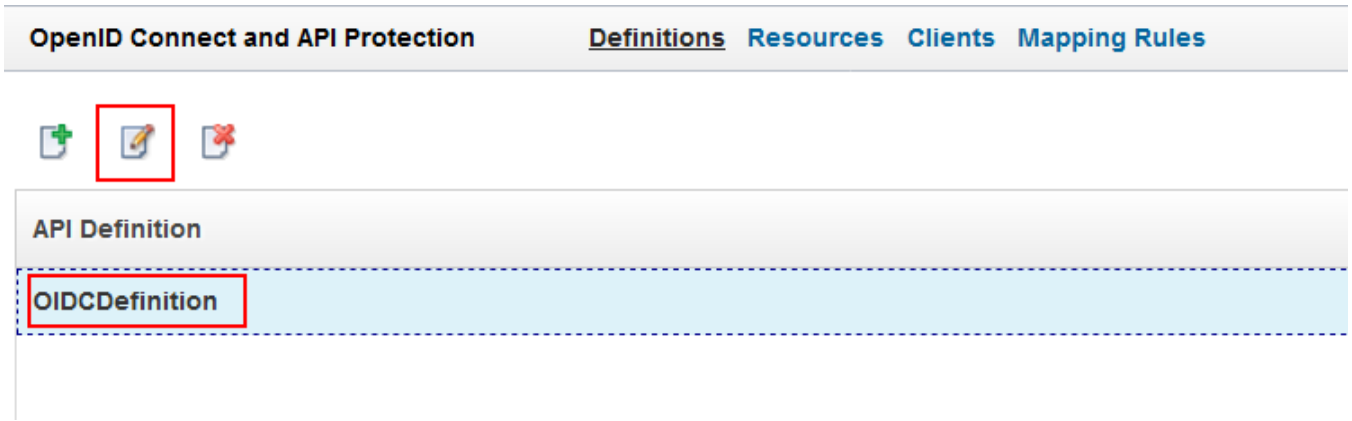
Click **Save** to save the new Access Policy.

Deploy the pending changes.

20.4.1 Updating Definition with Access Policy

Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**





Select **OIDCDefinition** and Click on the Edit button.

Select **AccessPolicyForPormpt** from the drop-down list for Access Policy.

Click on **Save** and **Deploy** pending changes

SCRIPT-END:
The script should display the following:
INFO:FederationManager:Configuring the Access Policy
INFO:FederationManager:Successfully configured the Access Policy
INFO:FederationManager:Update the OIDC Definition
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Successfully updated the OIDC Definition

If the configure -All option was used the script end should look like this

SCRIPT-END:
The script should display the following:
INFO:BaseManager:Activating Advanced Access Control
INFO:BaseManager:Successfully activated Advanced Access Control
INFO:WGAManager:Configure WebSEAL for AAC
INFO:WGAManager:Successfully configured WebSEAL for AAC
INFO:AACManager:Configuring Username Password Mechanism
INFO:AACManager:Successfully updated Username Password Mechanism
INFO:FederationManager:Configuring the Access Policy
INFO:FederationManager:Successfully configured the Access Policy
INFO:FederationManager:Update the OIDC Definition
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the Access Policy
INFO:FederationManager:Successfully updated the OIDC Definition

20.5 Testing OpenID Connect flow with prompt=login

We are now ready to test the OpenID Connect configuration with prompt=login as a parameter.

> **Note:** It is recommended that you restart your browser to remove all session cookies at both IdP and SP between each of the tests below.

A RP(SP) initiated OIDC flow can be triggered using

https://<Relying Party reverse proxy:port>/<junction name>/sps/oidc/rp/< Relying Party federation name>/kickoff/< Relying Party partner>?Target=https://<TargetURL>&prompt=login

The Relying party advanced mapping rule can pick up query string parameter and add it as a parameter to the authorize url.

Based on values previously set by following this document, the URL will be:

https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/partner?Target=/isam/mobile-demo/diag&prompt=login

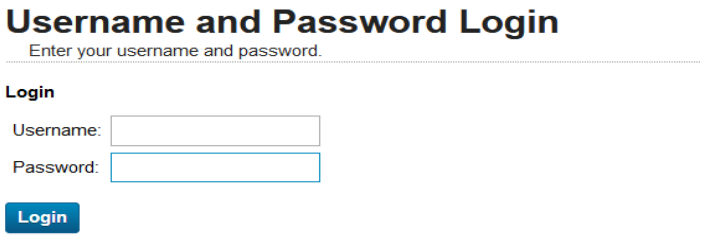
Trigger the flow using a browser.

If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the IDP by hitting the authorize URL.

An example of the authorize URL
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?nonce=HS8qF166ty&redirect_uri=https%3A%2F%2Fwww.mysp.ibm.com%2Fisam%2Fsps%2Foidc%2Frp%2Fisamrp%2Fredirect%2Fpartner&response_type=code+id_token+token&response_mode=form_post&state=5uxsESNh5e&scope=openid&client_id=clientID&prompt=login

The authorize url then redirects to Username Password Mechanism login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP.

At the landing page, which is part of the live demo application that you configured earlier, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

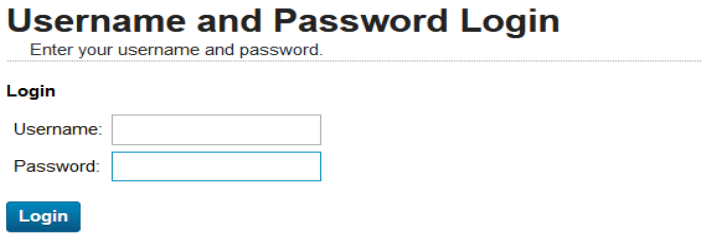
Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2IkQW1hdmhsaWYzaWhzMDNDN3JPbDI0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Open another tab in the same browser and initialize another flow with prompt=login

https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/partner?Target=/isam/mobile-demo/diag&prompt=login

The authorize url then redirects to username password mechanism login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



The login page is thrown again since prompt=login.

Open another tab in the same browser and initialize another flow without prompt=login

https://www.mysp.ibm.com/isam/sps/oidc/rp/isamrp/kickoff/partner?Target=/isam/mobile-demo/diag

The user is directly navigated to the landing page

At the landing page, the details of the user are displayed:

Access Manager Credential:
User: **https://www.myidp.ibm.com/testuser**

Name	Value(s)
AZN_CRED_PRINCIPAL_NAME[0]	https://www.myidp.ibm.com/testuser
tagvalue_login_user_name[0]	https://www.myidp.ibm.com/testuser
AZN_CRED_AUTH_METHOD[0]	trust
tagvalue_user_session_id[0]	aXNhbS5teXNwLmlibS5jb20tZGVmYXVsdAA=_W/tZegAAAAIAAAAweIn7WwhLAfgifwAAcHJHVFVkekcrNkRxR2lkQW1hdmhsaWYzaWhzMDNDN3JPbDl0Y3B5NXFCMUs2b3Vk:default
AZN_CRED_AUTHNMECH_INFO[0]	Federated trust
AZN_CRED_MECH_ID[0]	ITFIM_trust
access_token[0]	bytgVJZsycvBAP1JNJ62
AZN_CRED_CREATE_TIME[0]	2018-11-26T02:24:58Z
tagvalue_session_index[0]	30c6a178-f122-11e8-b58e-000c296cd683
scope[0]	openid

Once you are done testing, use the following script to update the API Definition to not use the access policy and to update the ACL attached to '/mga/sps/auth' to authenticated access.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. Remember, this step is for the IdP.

For the IdP, run this script: **OIDCWithAccessPolicy.py -configure Reset**

SCRIPT-END:
The script should display the following:
INFO:WGAManager:Configure ACLs for OIDC
INFO:WGAManager:Successfully configured ACLs OIDC
INFO:FederationManager:Update the OIDC Definition
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Retrieving the attribute source reference ID
INFO:FederationManager:Successfully updated the OIDC

21 Reverse Proxy Native OIDC Relying Party

WebSEAL provides a native OpenID Connect relying partner (RP) capability that is able to consume an identity token which has been provided by an OpenID Connect Provider in order to establish an authenticated session.

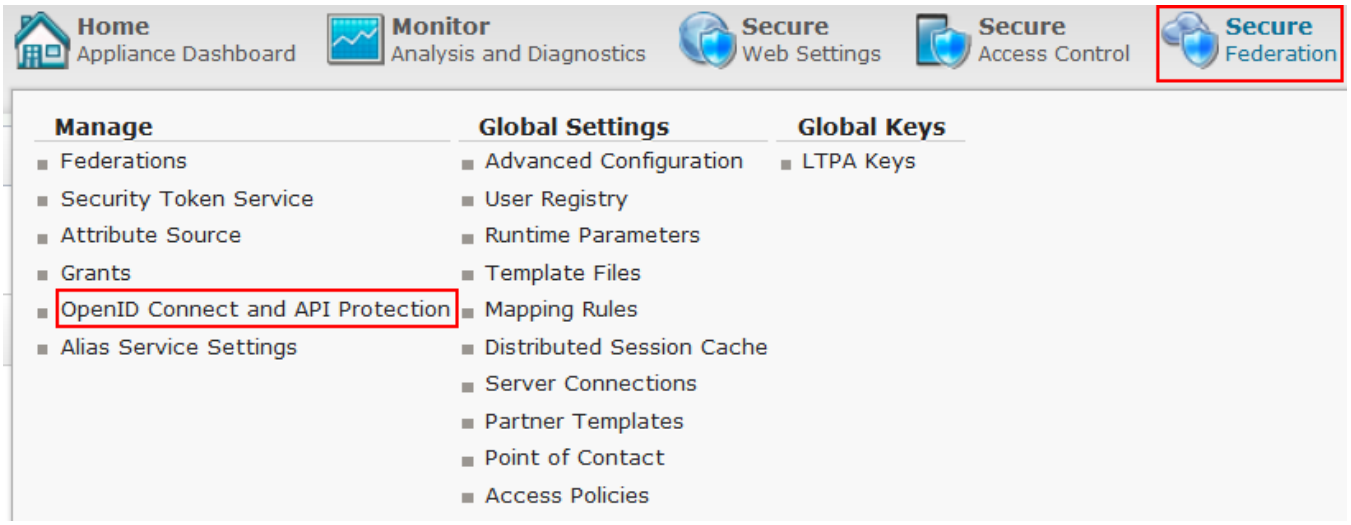
The WebSEAL implementation does not implement the complete specification for OIDC relying parties

SCRIPT-START:
Run this script: **WebSEALNativeRP.py -configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

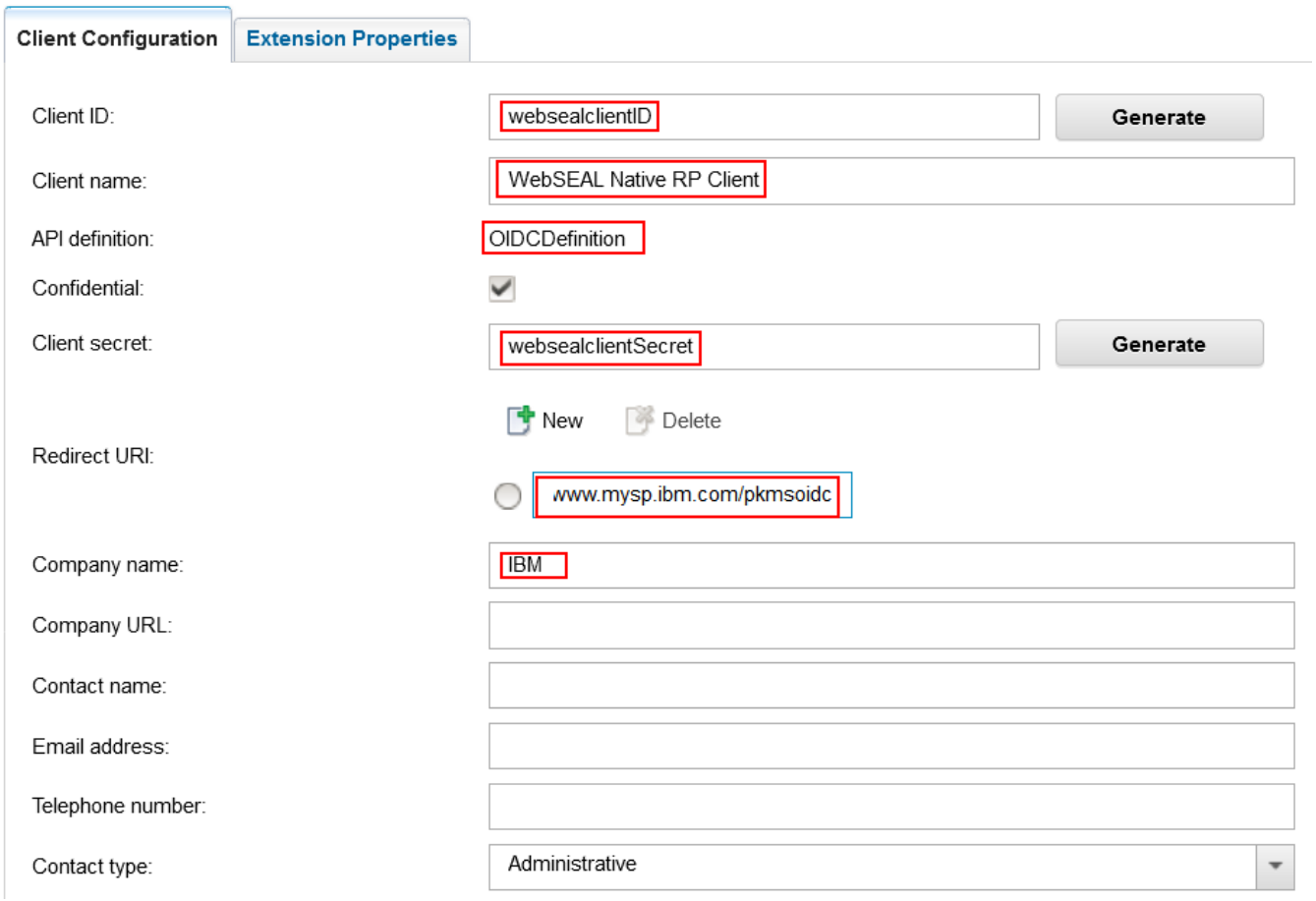
21.1 Configuring a Client

This step is done at the IdP machine.



Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**

Navigate to **Clients** and click on **Add** to create a New Client



On the Client Configuration panel, enter the **websealclientID** as Client ID, enter **WebSEAL Native RP Client** as the Client Name and select **OIDC Definition** as the API Definition, select checkbox **Confidential** and enter **websealclientSecret** as Client Secret**.**

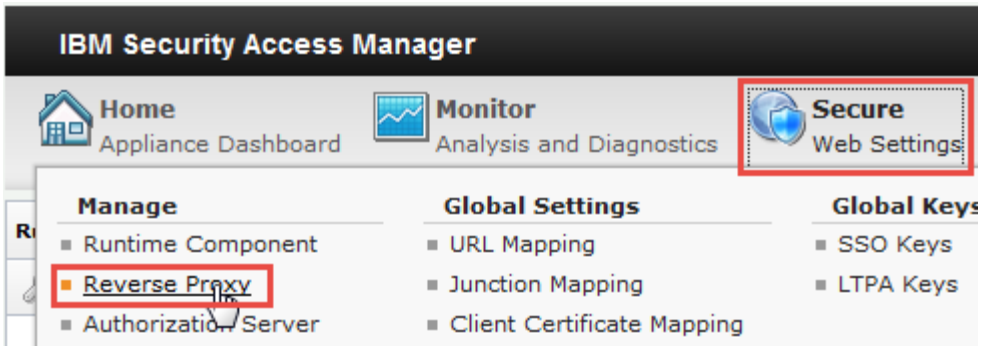
Enter **https://www.mysp.ibm.com/pkmsoidc** as a Redirect URI by clicking New**,** enter **IBM** as the Company Name**.**

Click on **OK** to create a Client.
Follow on-screen instructions to **deploy** pending changes

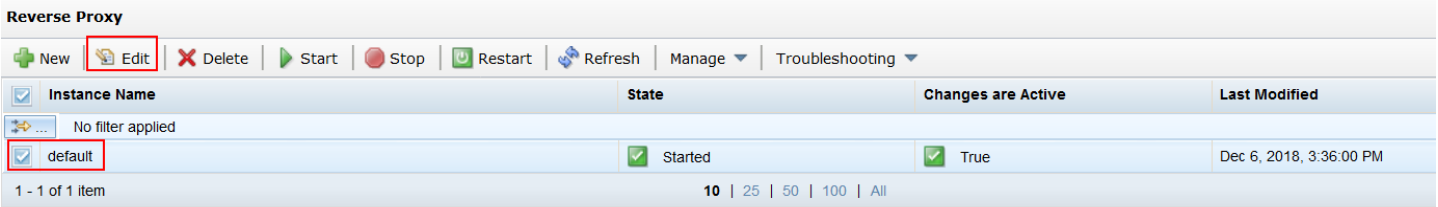
21.2 Configuring WebSEAL

This step is done at the SP machine.

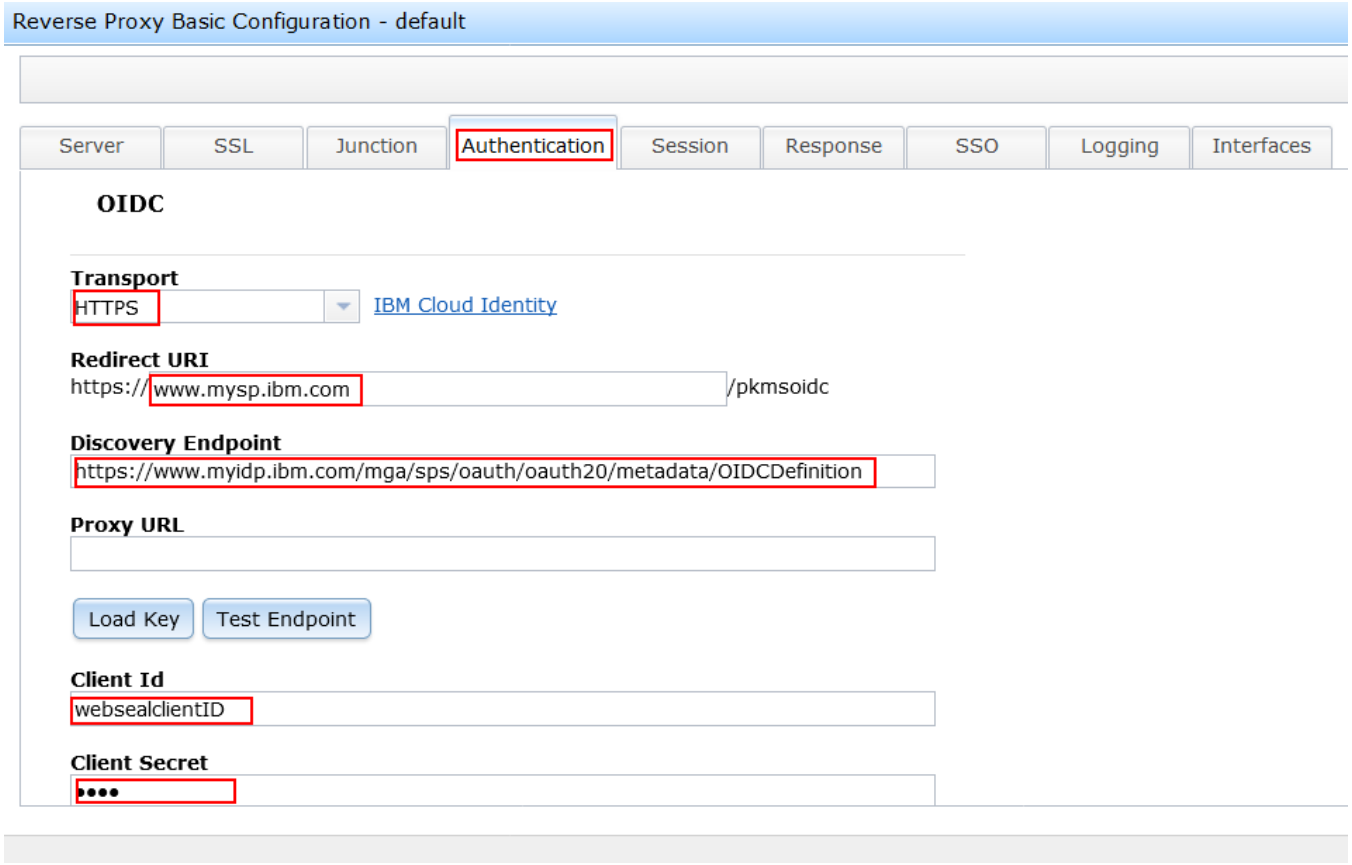
The RP functionality is configured using the '[oidc]' and '[oidc:<op-id>]' stanzas.



In the mega-menu, navigate to **Secure Web Settings > Manage: Reverse Proxy**.

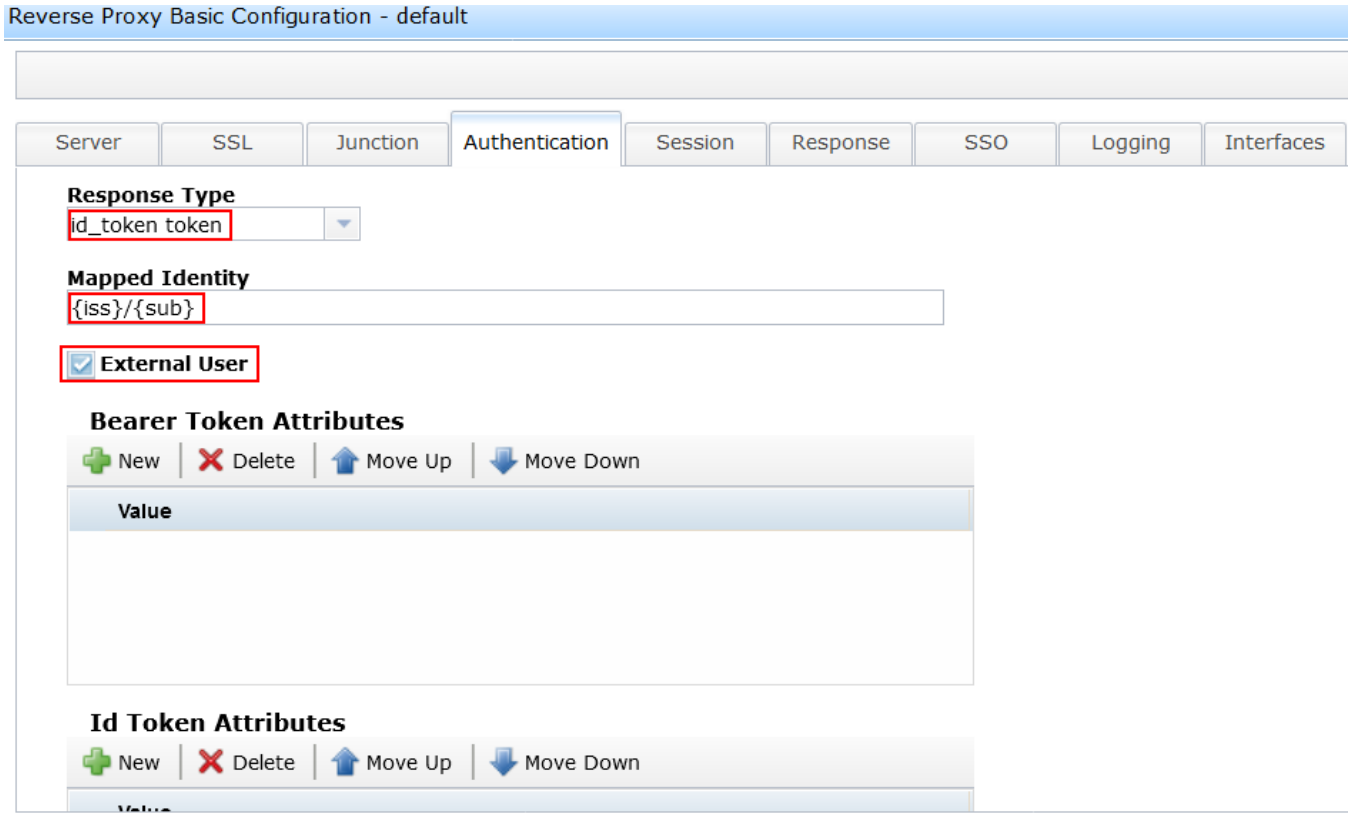


Select the **default** instance and Click on **Edit.**



Navigate to Authentication tab, and navigate to OIDC section, select **HTTPS** for Transport, enter **www.mysp.ibm.com** for the redirect uri, enter

https://www.myidp.ibm.com/mga/sps/oauth/oauth20/metadata/OIDCDefinition as the discovery endpoint, enter websealclientID as the client id and websealclientSecret as the client secret.



Select id_token token as the response type, enter mapped identity as {iss}/{sub}, check External User to true.

Click on Save and deploy pending changes and restart the reverse proxy instance.

SCRIPT-END:
The script should display the following:
INFO:WebSEALNativeRP:Begin configuring WebSEAL as native RP
INFO:FederationManager:Configuring the OIDC Client
INFO:FederationManager:Successfully configured OIDC Client
INFO:WGAManager:Configuring WebSEAL.conf file for native OIDC rp support
INFO:WGAManager:Successfully Configured WebSEAL.conf file for native OIDC rp support
INFO:WGAManager:Configuring Signer Certificates
INFO:WGAManager:Successfully configured Signer Certificates
INFO:WebSEALNativeRP:Successfully configured WebSEAL as native RP

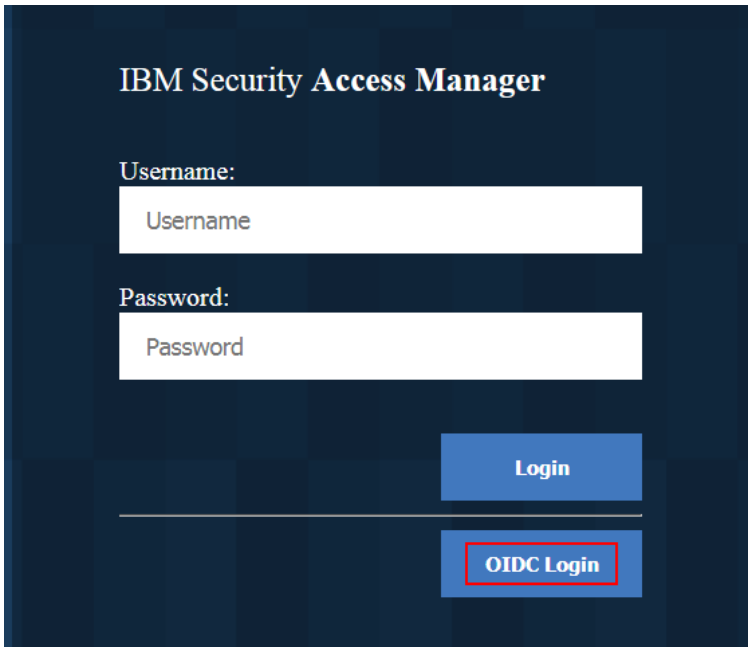
21.3 Testing OIDC Single Sign-on flow

We are now ready to test the OpenID Connect configuration.

Access the SP's reverse proxy instance URL

https://<Relying Party reverse proxy:port>

Based on values previously set by following this document, the URL will be:
https://www.mysp.ibm.com/



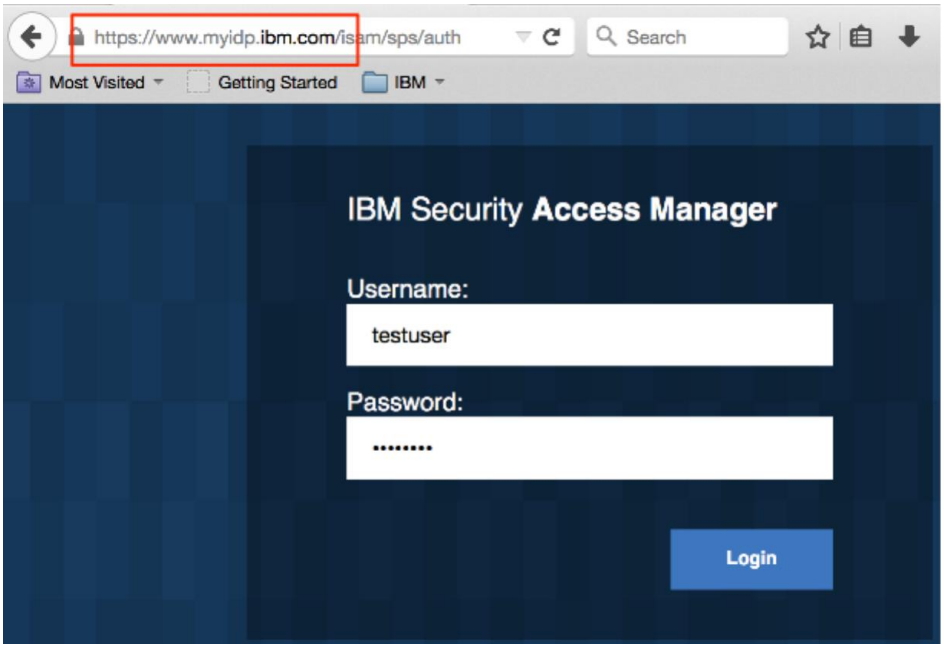
Click on the OIDC Login button.

The authorize url is formed with the following parameters

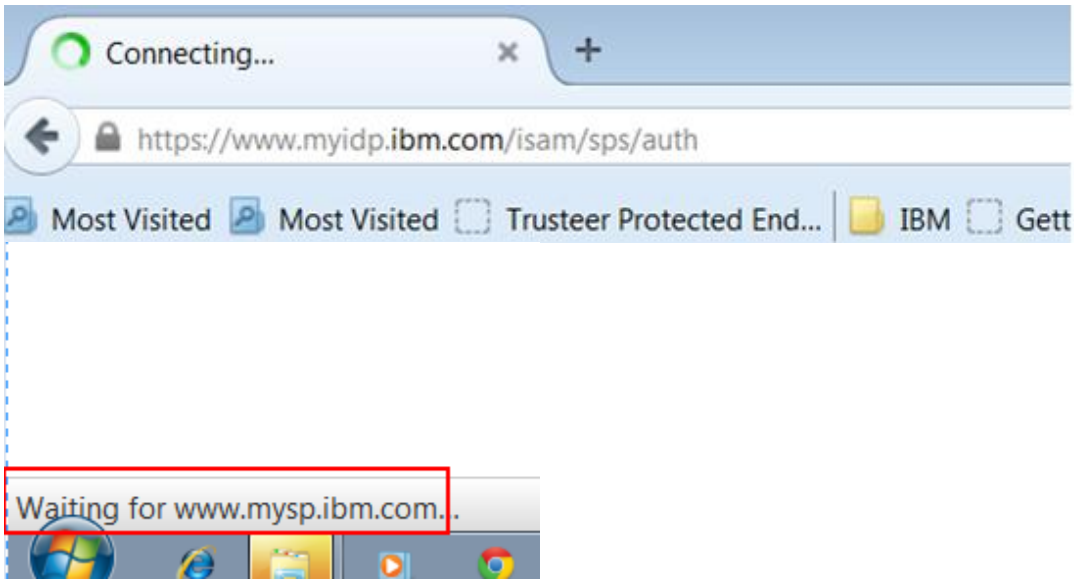
https://www.myidp.ibm.com/mga/sps/oauth/oauth20/authorize?scope=openid&response_type=id_token%20token&client_id=websealclientID&response_mode=form_post&redirect_uri=https://www.mysp.ibm.com/pkmsoidc&state=932a0fd4-f92f-11e8-ae46-000c29e4b31f&nonce=8e93b419-592a-0f65-809c-b1b42bf6c140

The authorize url then redirects to login page.

Login using **testuser** and **Passw0rd**, as created at the IDP in an earlier section.



If you notice the browser URL, page and footer you can see that the browser is now redirecting you to the SP's redirect uri.

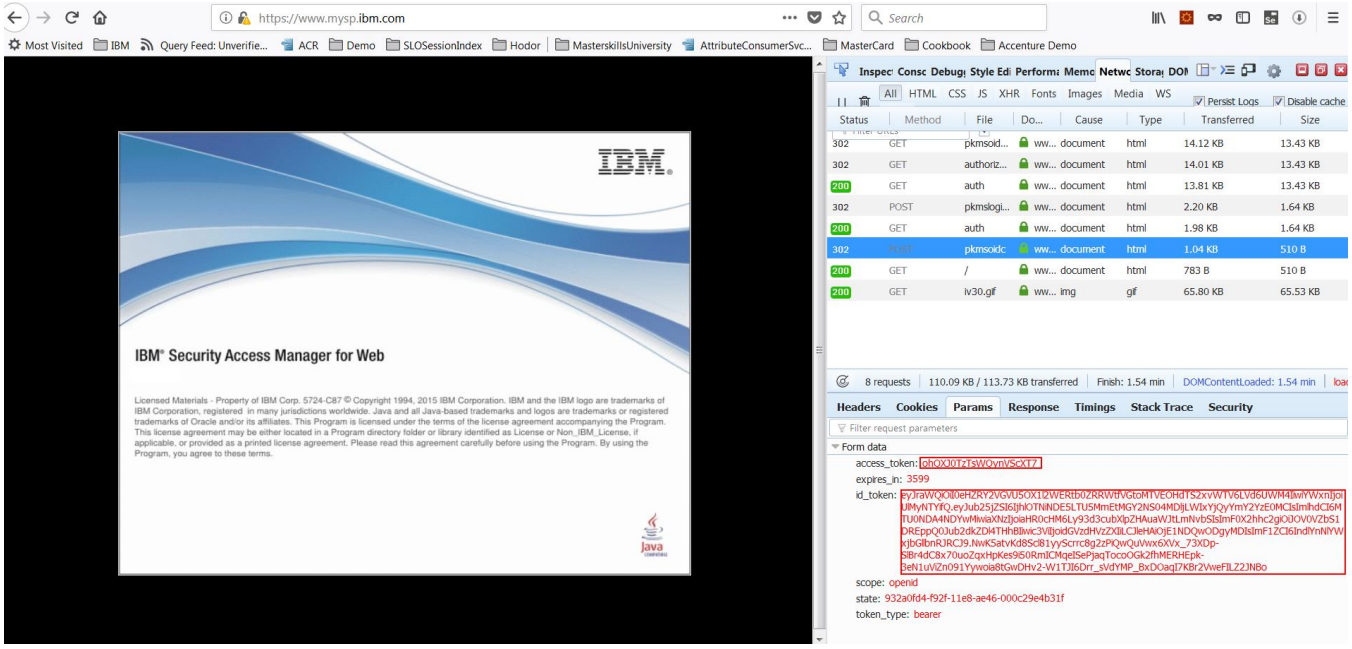


The landing page is now displayed, by default it is the WebSEAL splash screen.

Inspect the request and response by enabling Web Developer tools, look at section 16.2.

Since we ran a implicit flow, the OP will send access_token and id_token as a response.

Inspect the id_token using https://jwt.io/



Encoded PASTE A TOKEN HERE

eyJraWQiOiI0eHZRY2VGVU5OX1l2WERtb0ZRRWtf
VGtoMTVEOHdTS2xvWTV6LVd6UWM4IiwiYWxnIjoi
UlMyNTYifQ.eyJub25jZSI6IjhlOTNiNDE5LTU5M
mEtMGY2NS04MDljLWIxYjQyYmY2YzE0MCIsImlhd
CI6MTU0NDA4NDYwMiwiaXNzIjoiaHR0cHM6Ly93d
3cubXlpZHAuaWJtLmNvbSIsImF0X2hhc2giOiJOV
0VZbS1DREppQ0Jub2dkZDl4THhBIiwic3ViIjoid
GVzdHVzZXIiLCJleHAiOjE1NDQwODgyMDIsImF1Z
CI6IndlYnNlYWxjbGllbnRJRCJ9.NwK5atvKd8Sc
l81yyScrrc8g2zPiQwQuVwx6XVx_73XDp-
SlBr4dC8x70uoZqxHpKes9i50RmICMqeISePjaqT
ocoOGk2fhMERHEpk-
3eN1uViZn091Yywoia8tGwDHv2-
W1TJI6Drr_sVdYMP_BxDOaqI7KBr2VweFILZ2JNB
o

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "4xvQceFUNN_YvXDmoFQEk_Tkh15D8wSKloY5z-WzQc8",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
{
  "nonce": "8e93b419-592a-0f65-809c-b1b42bf6c140",
  "iat": 1544084602,
  "iss": "https://www.myidp.ibm.com",
  "at_hash": "NWEYm-CDJiCBnogdd9xLxA",
  "sub": "testuser",
  "exp": 1544088202,
  "aud": "websealclientID"
}
```


VERIFY SIGNATURE

To verify the access_token use the Userinfo endpoint, we could use a browser extension for a REST tool or use postman to make this request.

```
curl --request GET \
  --url https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo \
  --header 'authorization: Bearer CRaSsmobyYSobQTx0E5y' \
```

Userinfo endpoint - https://www.myidp.ibm.com/mga/sps/oauth/oauth20/userinfo
Header – Auhtorization: Bearer <Access Token>

22 Appendix A: Troubleshooting and Workarounds

22.1 Troubleshooting

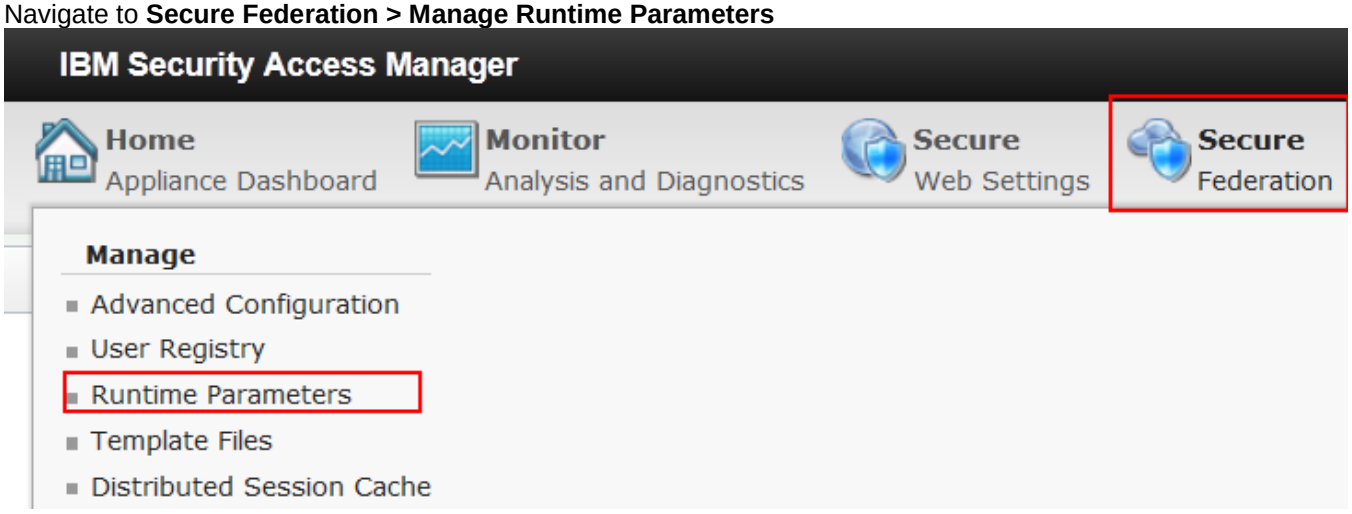
22.1.1 Enabling PD logs

Refer to the ISAM Help to enable logging for reverse proxy, policy director and other ISAM base components.

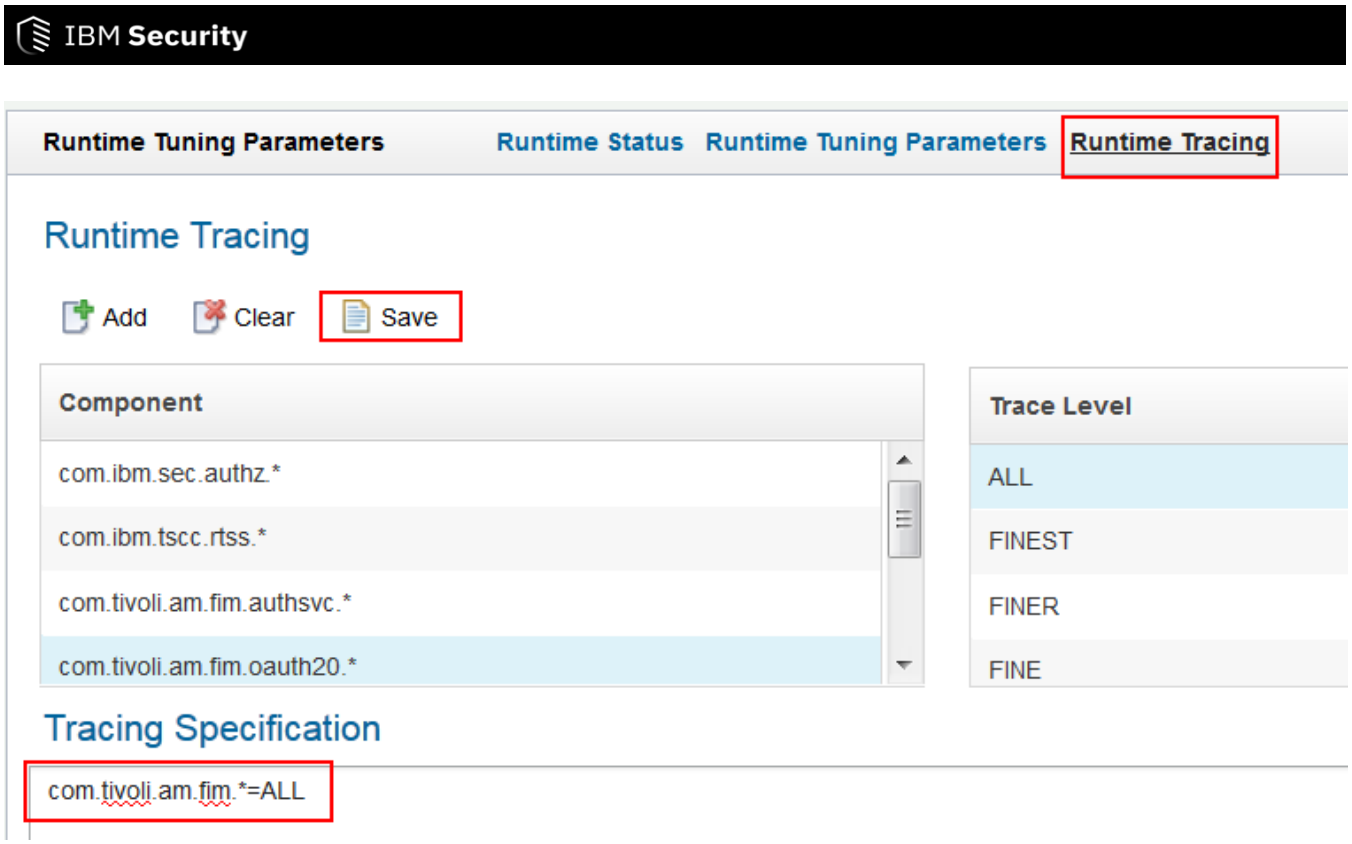
22.1.2 Enable Federation logs

You can enable trace logs for debugging. There is a script to enable trace logs. It is mentioned in section 23. You can also use the console / LMI to enable trace logs. The steps are mentioned below.

Navigate to **Secure Federation > Manage Runtime Parameters**

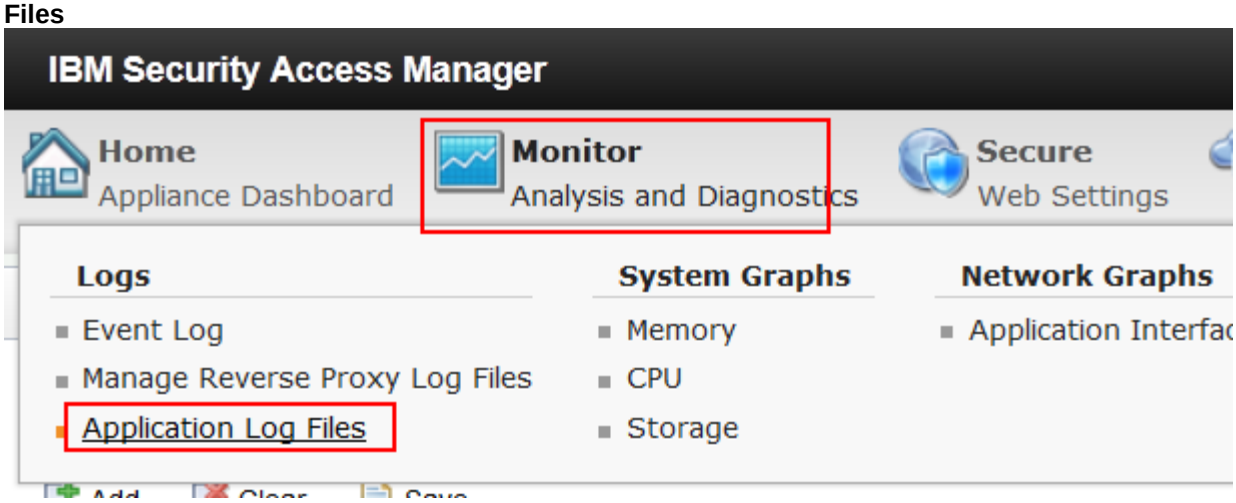


In the **Runtime Tracing** tab, set the **Tracing Specification** as shown below to *com.tivoli.am.fim.*=ALL* . Click **Save**

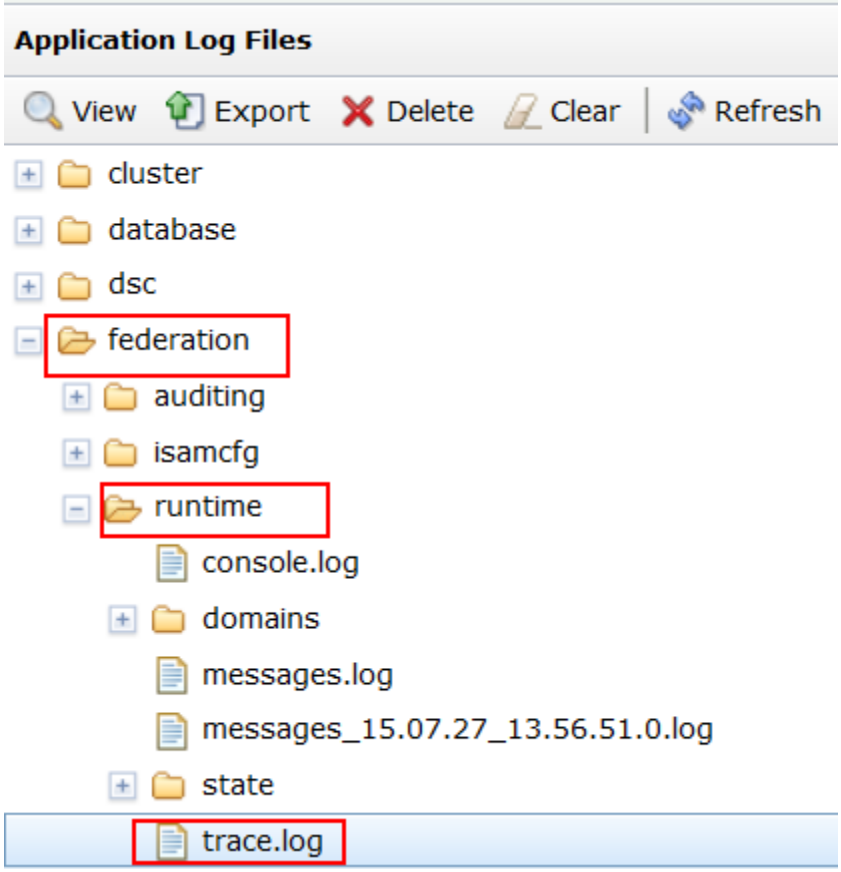


A warning will be displayed at the top of the window. Click the link to activate the configuration change you have just made. A pop-up dialog is displayed showing the pending changes. Click deploy.

During a SAML flow, if there is an error then navigate to **Monitor Analysis and Diagnostics > Application Log Files**

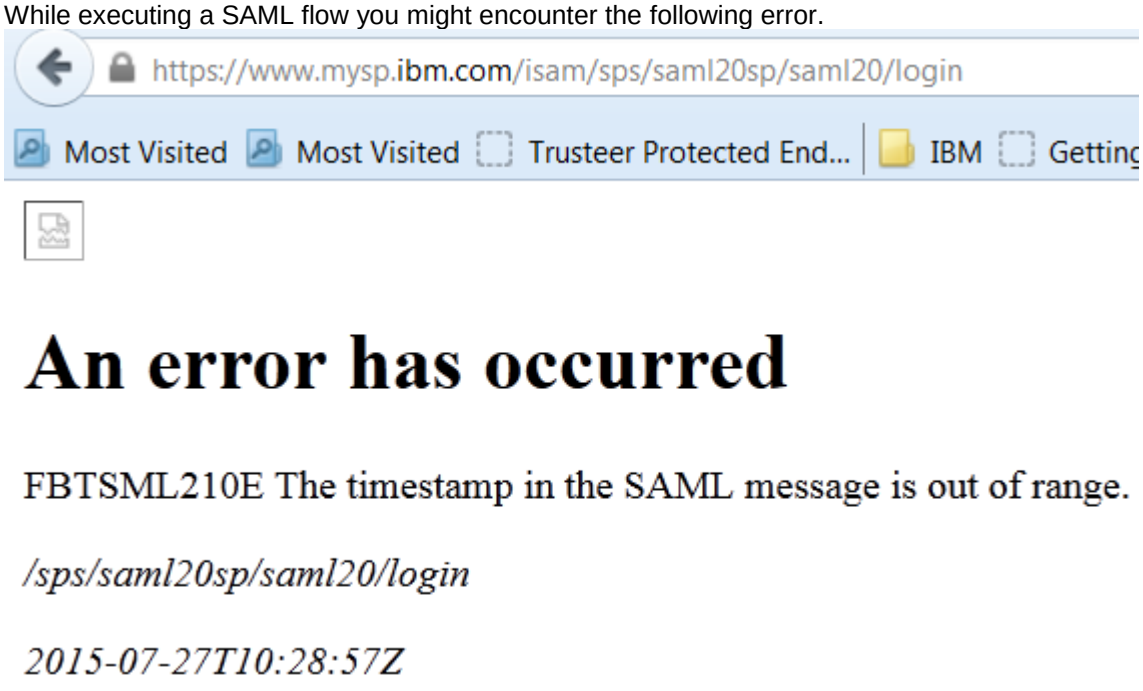


Navigate to federation runtime trace log as shown below. Export or view the file.



22.1.3 Time Sync error

While executing a SAML flow you might encounter the following error.

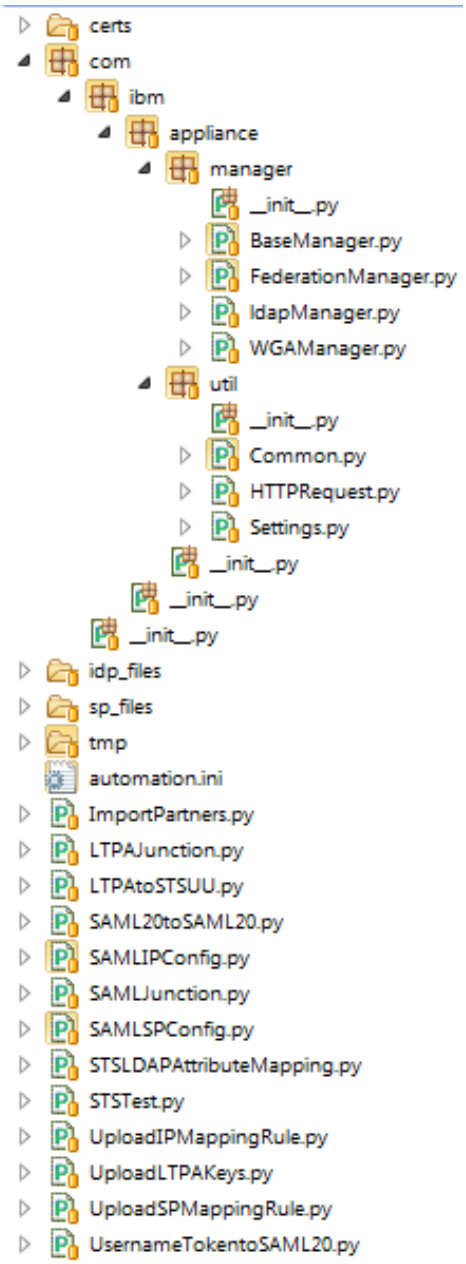


Make sure the IdP appliance and SP appliance time zones and time are not more than 300 seconds apart.

23 Appendix B – Python Automation Project

The automatic configuration scripts provided with this cookbook were written for python 2.7.

The …**/providedfiles/Automation** directory contains a Python project.
The directory structure looks as shown below:



In order to automate the steps mentioned in this document, use SAMLIPConfig.py, SAMLSPConfig.py and ImportPartner.py. SAMLIPConfig.py should be used to configure the IdP appliance. The SAMLSPConfig.py should be used to configure the SP appliance. The source code is available for editing. The code was developed to configure the steps mentioned in this document. Any change to the code would require thorough testing. Run the SAMLIPConfig.py and SAMLSPConfig.py without parameters to understand the usage.

In order to run the scripts your python installation will require the “requests”, and “python-ldap” libraries. These are installed using “pip install”, as follows (examples for a python 2.7 installation):

```
pip install requests
pip install python-ldap
```

Note that on Linux or MacOS systems, pip may need to be run as root:

```
sudo pip install requests
sudo pip install python-ldap
```

The following commands will be useful when a combination of automated code and manual steps are used. The examples below show SAMLIPConfig.py. The commands also work with SAMLSPConfig.py.

To restart the federation runtime use the following command:

```
SAMLIPConfig.py -action Restart_Federation_Runtime
INFO:FederationManager:Restarting federation runtime.
INFO:FederationManager:Successfully restarted federation runtime.
```

To set the runtime tracing for federation run the following command.

```
SAMLIPConfig.py -action Runtime_Trace_String
INFO:FederationManager:Setting runtime trace string.
INFO:FederationManager:Successfully set runtime trace string.
```

To deploy pending changes run the following command.

```
SAMLIPConfig.py -action Show_Pending_Changes
INFO:FederationManager:Displaying pending changes
INFO:FederationManager:[]
INFO:FederationManager:Successfully displayed pending changes.
```

To restart console / LMI run the following command.

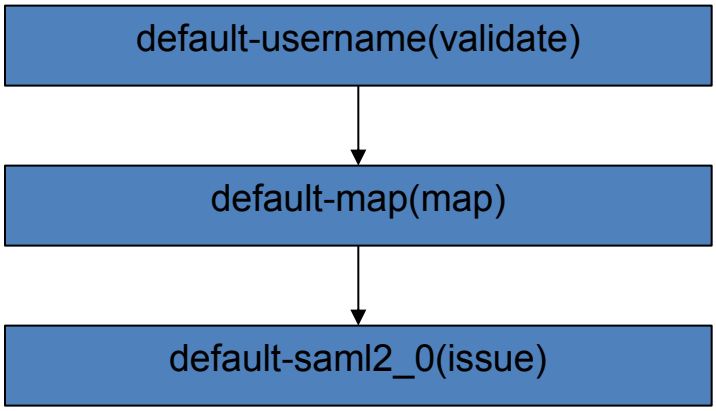
```
SAMLIPConfig.py -action Restart_LMI
INFO:FederationManager:Restarting LMI
INFO:FederationManager:Successful restart of LMI
```

24 Appendix C – Additional STS Examples

There are several exercises in this cookbook that demonstrate detailed use of the STS. The python automation code includes several extra examples for other STS use cases and modules. You may find these use cases useful for later reference, as they are relatively common customer adoption patterns. We will not outline the detailed use of the console to configure them, but instead just list what the examples are, and how you can configure and test them with the automation scripts. You can also explore the python scripts for more details on configuration properties, or you can create the examples using the scripts, then explore the configuration in the administration UI.

24.1 UsernameToken to SAML 2.0

This example establishes an STS chain that validates a UsernameToken, and issues a SAML 2.0 assertion, using a chain as depicted below:



24.1.1 Pre-requisites and Configuration

It is assumed that your IDP image already has the SAML IDP federation configured because the SAML 2.0 module uses the same signing key and the configuration script used here assumes it is already loaded into the appliance. The test case also assumes that the testuser/Passw0rd account exists in ISAM, and that is configured as part of the SAML IDP example.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **UsernameTokentoSAML20.py –configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:UsernameTokentoSAML20:Configuring the test UsernameTokentoSAML20 chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:BaseManager:Configuring the easuser password
INFO:BaseManager:Successfully configured the easuser password
INFO:BaseManager:Configuring the server connection
INFO:BaseManager:Successfully configured the server connection
INFO:FederationManager:Configuring the Username to SAML20 Module Chain Template
INFO:FederationManager:Successfully configured the Username to SAML20 Module Chain Template
INFO:FederationManager:Configuring the Username to SAML20 Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the Username to SAML20 Module Chain Mapping

24.1.2 Testing

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint.

The providedfiles/ststest directory includes an example soap message and shell command to invoke cUrl with the correct parameters for WS-Trust 1.2. The set of provided files related to this example are:

Filename	Description
rst12_ut.sh	Executable script to launch the request
rst12_ut.xml	XML of the request body containing the UsernameToken to validate
ut_to_saml20.js	Javascript mapping rule

24.1.3 Further Details

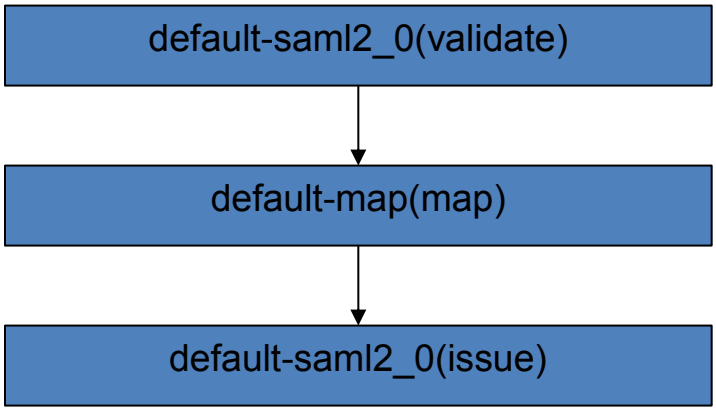
The validation of the password for a UsernameToken is performed against the ISAM registry. Only ISAM users (and not basic-users) can be used for password validation. Configuration of the UsernameToken module requires that a ServerConnection be created, which points to the ISAM LDAP registry. This is the same approach that was used to configure the LDAP Attribute Mapping Example. You can turn password validation off, and then the UsernameToken is really just treated as an ID assertion, with the option of validating the password yourself using an HTTPClient callout in the mapping rule, for example.

The mapping rule used in this example is very simple – it just re-writes the Principal name “type” to *urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress* in preparation for issuing the SAML assertion. It also adds a multi-valued attribute so that the resulting SAML assertion will contain an example AttributeStatement.

The SAML 2.0 module is configured to use the same signing keys as were used in the SAML federation. Other module configuration properties control exist to control a variety of properties of the SAML assertion including the Issuer URI, the NotBefore and NotOnOrAfter times, and many others. Take a look at the script, or first configure the chain using the script then explore the properties in the Security Token Service administration UI.

24.2 SAML 2.0 to SAML 2.0

This example establishes an STS chain that validates a SAML 2.0 assertion, and issues a new SAML 2.0 assertion, using a chain as depicted below:



This use case could be used to validate a SAML assertion signed by one key, and issue another SAML assertion signed with another key, or just re-issue a new SAML assertion with the same key and updated attributes in the AttributeStatement. The other reason for showing this use case is so that you have a reference of setting up automatic configuration for the SAML 2.0 token module in both validate and issue modes.

24.2.1 Pre-requisites and Configuration

It is assumed that your IDP image already has the SAML IDP federation configured because the SAML 2.0 module uses the same signing key and the configuration script used here assumes it is already loaded into the appliance.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **SAML20toSAML20.py -configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:SAML20toSAML20:Configuring the SAML20 to SAML20 chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:BaseManager:Configuring the easuser password
INFO:BaseManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the SAML20 to SAML20 Module Chain Template
INFO:FederationManager:Successfully configured the SAML20 to SAML20 Module Chain Template
INFO:FederationManager:Configuring the SAML20 to SAML20 Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the SAML20 to SAML20 Module Chain Mapping
INFO:SAML20toSAML20:Successfully configured the SAML20 to SAML20 chain

24.2.2 Testing

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint.

The providedfiles/ststest directory includes an example soap message and shell command to invoke cUrl with the correct parameters for WS-Trust 1.2. The set of provided files related to this example are:

Filename	Description
rst12_saml20.sh	Executable script to launch the request
rst12_saml20.xml	XML of the request body. Note that you will certainly have to update the SAML assertion contained within the request body before it will successfully validate, as the one that is in there by default will surely have expired.
saml20_to_saml20.js	Javascript mapping rule

In order to obtain a “current” SAML assertion to insert into rst12_saml20.xml to be successfully validated, use either the SAML Junction use case documented in section 18 of this cookbook, or run the UsernameToken to SAML 2.0 use case documented in the previous section of this cookbook. You should paste that in as a replacement to the old SAML assertion that you will find in rst12_saml20.xml.

24.2.3 Further Details

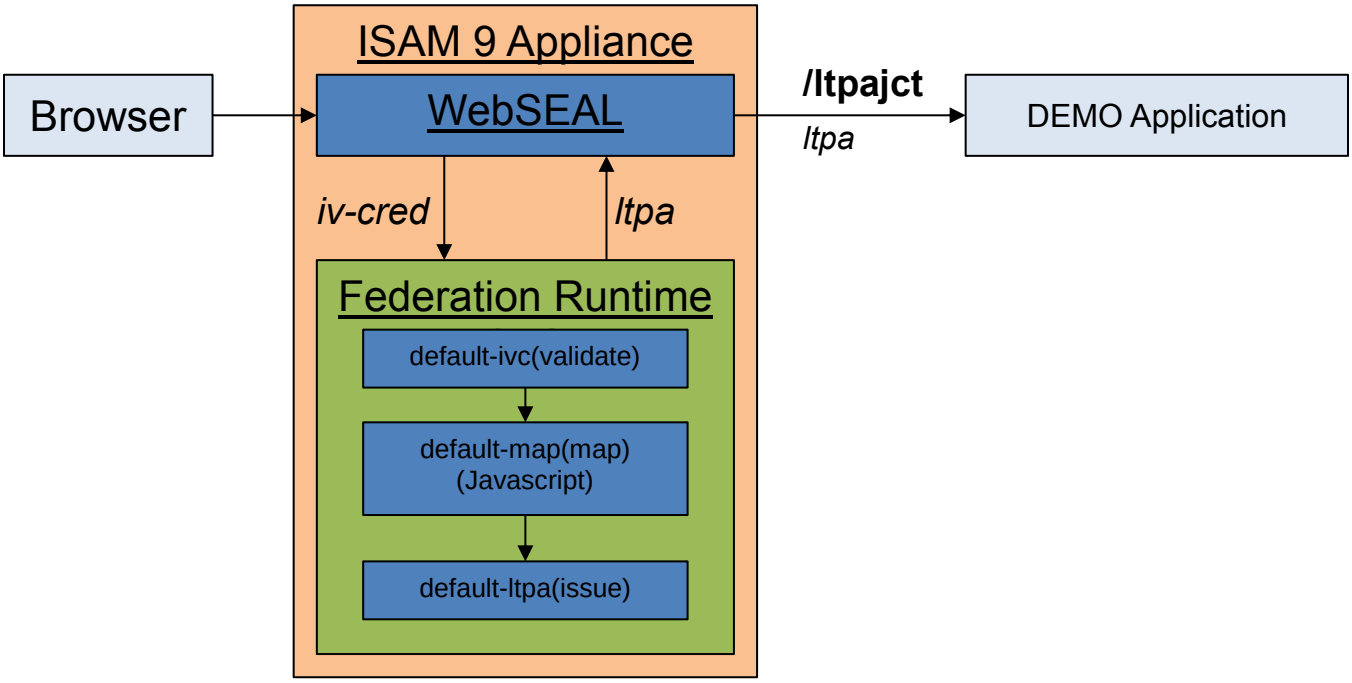
Validation of a SAML assertion has a number of parameters that controls validity period, signature verification keys, decryption keys (if encryption is used on the assertion – not used in our examples), etc.

The mapping rule used in this example is very simple – it just re-writes the Principal name “type” to *urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress* in preparation for issuing the SAML assertion. It also adds a multi-valued attribute so that the resulting SAML assertion will contain an example AttributeStatement.

The SAML 2.0 module for issuing the final assertion is also configured to use the same signing keys as were used in the SAML federation. Other module configuration properties control exist to control a variety of properties of the SAML assertion including the Issuer URI, the NotBefore and NotOnOrAfter times, and many others. Take a look at the script, or first configure the chain using the script then explore the properties in the Security Token Service administration UI.

24.3 LTPA Junction

This example establishes an STS chain and WebSEAL configuration that allows WebSEAL to downstream an LTPA BinarySecurityToken across a WebSEAL junction. This is very similar to the SAML Junction configuration that is performed in section 18 of this document except instead of down streaming a SAML assertion to the junctioned application we downstream an LTPAv2 BinarySecurityToken. The integration looks like this:



24.3.1 Pre-requisites and Configuration

This use case has the same pre-requisites as the SAML Junction use case. It is assumed that your IDP image already has the SAML IDP federation configured because mapping rules and the LTPA keys used in the use case are loaded during the appliance setup done by that application.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **LTPAJunction.py –configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:

The script should display the following:

INFO:LTPA Junction:Configuring LTPA Junction
INFO:FederationManager:Configuring the IVCred to LTPA Module Chain Template
INFO:FederationManager:Successfully configured the IVCred to LTPA Module Chain Template
INFO:FederationManager:Configuring the IVCred to LTPA Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the IVCred to LTPA Module Chain Mapping
INFO:FederationManager:Restarting federation runtime.
INFO:FederationManager:Successfully restarted federation runtime.
INFO:WGAManager:Configuring WebSEAL.conf file for LTPA Junction at IdP
INFO:WGAManager:Successfully configured WebSEAL.conf file for LTPA junction at IdP
INFO:WGAManager:Configuring junction
INFO:WGAManager:Successfully configured junction
INFO:LTPA Junction:End LTPA junction creation and configuration

Configuration of this use case leverages the following files (included in the providedfiles accompanying this cookbook):

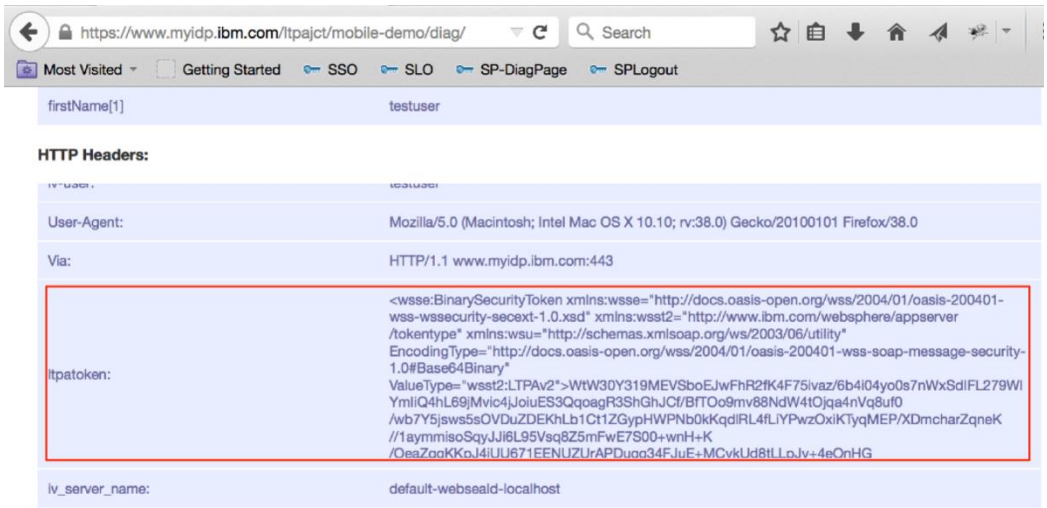
Filename	Description
ltpasso.keys	Issuing an LTPA token makes use of uploaded LTPA keys that have to come from a WebSphere server or other source outside the appliance. In this case we used a traditional WebSphere application server to obtain an LTPA keyfile.
ivc_to_ltpa.js	Javascript mapping rule

24.3.2 Testing

To test this use case, use a browser to visit the URL:

https://www.myidp.ibm.com/ltpajct/mobile-demo/diag/

Following authentication as the testuser, you should be able to scroll down and see the LTPA BinarySecurityToken that was sent as a HTTP header:



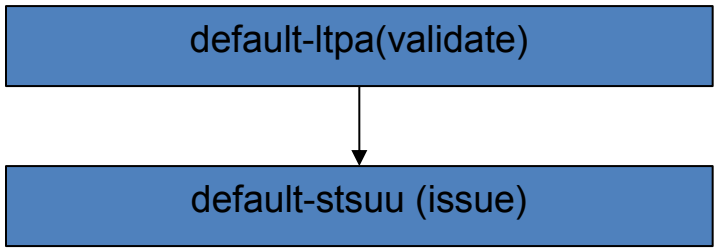
24.3.3 Further Details

The mapping rule used in this example does some manipulation of the STSUU to modify the Principal name “type” attribute to that required for LTPA issue to work. The attribute list is also pruned to a small subset of credential attributes.

Issuing an LTPA token includes configuration for the LTPA realm, token expiry and LTPA version (1 or 2) that the LTPA token will be issued with. As with all these examples, you can explore LTPA configuration in the UI after creating the STS chain with the script.

24.4 LTPA to STSUniversalUser

This example establishes an STS chain that validates an LTPA BinarySecurityToken, and issues an STSUniversalUser XML token, using a chain as depicted below (notice no mapping rule):



This primary reason for including this example use case is to show you how to validate and see the attributes in an LTPA token like that from the previous exercise.

24.4.1 Pre-requisites and Configuration

It is assumed that your IDP image already has the SAML IDP federation configured because that is when the LTPA keys are uploaded.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **LTPAtoSTSUU.py –configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:LTPA to STSUU:Configuring LTPA to STSUU chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:BaseManager:Configuring the easuser password
INFO:BaseManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the LTPA to STSUU Module Chain Template
INFO:FederationManager:Successfully configured the LTPA to STSUU Module Chain Template
INFO:FederationManager:Configuring the LTPA to STSUU Module Chain Mapping
INFO:FederationManager:Successfully configured the LTPA to STSUU Module Chain Mapping
INFO:FederationManager:Restarting federation runtime.
INFO:FederationManager:Successfully restarted federation runtime.
INFO:LTPA to STSUU:End LTPA to STSUU creation and configuration

24.4.2 Testing

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint.

The providedfiles/ststest directory includes an example soap message and shell command to invoke cUrl with the correct parameters for WS-Trust 1.2. The set of provided files related to this example are:

Filename	Description
rst12_ltpa.sh	Executable script to launch the request
rst12_ltpa.xml	XML of the request body. Note that you will certainly have to update the LTPA BinarySecurityToken contained within the request body before it will successfully validate, as the one that is in there by default will surely have expired.

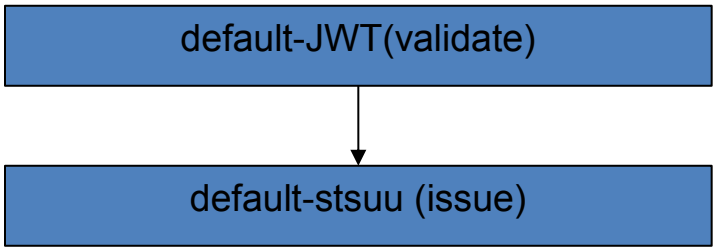
In order to obtain a “current” LTPA BinarySecurityToken to insert into rst12_ltpa.xml to be successfully validated, use the LTPA Junction use case documented in the previous section of this cookbook. You should paste that in as a replacement to the old LTPA BinarySecurityToken that you will find in rst12_ltpa.xml.

24.4.3 Further Details

Validation of an LTPA BinarySecurityToken has parameters for the LTPA SSO keys to use, You will need to know the password associated with the LTPA key file. In our provided files that is “passw0rd”.

24.5 JWT to STSUniversalUser

This example establishes an STS chain that validates a Json Web Token(JWT), and issues an STSUniversalUser XML token, using a chain as depicted below (notice no mapping rule):



24.5.1 Pre-requisites and Configuration

It is assumed that your IDP image already has the OpenID Connect provider configured.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **JWTtoSTSUU.py –configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:JWT to STSUU:Configuring JWT to STSUU chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:BaseManager:Configuring the easuser password
INFO:BaseManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the JWT to STSUU Module Chain Template
INFO:FederationManager:Successfully configured the JWT to STSUU Module Chain Template
INFO:FederationManager:Configuring the JWT to STSUU Module Chain Mapping
INFO:FederationManager:Successfully configured the LTPA to STSUU Module Chain Mapping
INFO:JWT to STSUU:End JWT to STSUU creation and configuration

24.5.2 Testing

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint.

The providedfiles/ststest directory includes an example soap message and shell command to invoke cUrl with the correct parameters for WS-Trust 1.2. The set of provided files related to this example are:

Filename	Description
rst12_jwt_to_stsuu.sh	Executable script to launch the request
rst12_jwt_to_stsuu.xml	XML of the request body. Note that you will certainly have to update the JWT BinarySecurityToken contained within the request body before it will successfully validate, as the one that is in there by default will surely have expired.

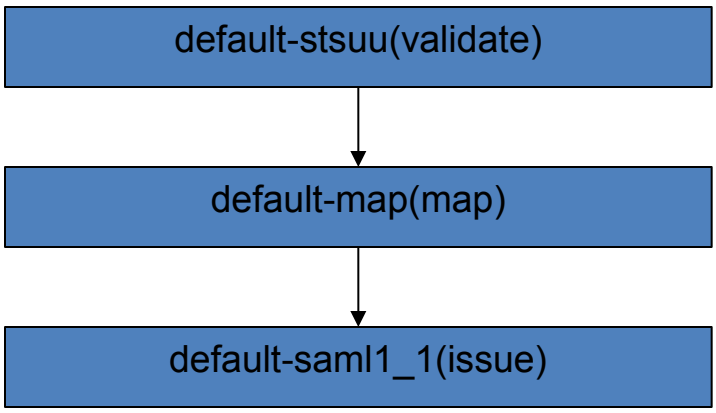
To obtain a “current JWT BinarySecurityToken to insert into rst12_jwt_to_stsuu.xml to be successfully validated, use the id_token from the OIDC Single Sign On flow from the previous section of this cookbook. You should paste that in as a replacement to the old JWT BinarySecurityToken that you will find in rst12_jwt_to_stsuu.xml.

24.5.3 Further Details

Validation of a JWT has several parameters that controls issuer, subject, signature, decryption keys (if encryption is used on the assertion – not used in our examples), etc.

24.6 STSUniversalUser to SAML 1.1

This example establishes an STS chain that validates a STSUniversalUser, and issues a SAML 1.1 assertion, using a chain as depicted below:



24.6.1 Pre-requisites and Configuration

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **STSUUtoSAML11.py -configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:STSUUtoSAML11:Configuring the test STSUU to SAML11 chain
INFO:WGAManager:Configure WGA for STS Chains
INFO:WGAManager:Successfully configured ACLs for STS Chain.
INFO:FederationManager:Configuring the easuser password
INFO:FederationManager:Successfully configured the easuser password
INFO:FederationManager:Configuring the STS Module Chain Template
INFO:FederationManager:Successfully configured the STS Module Chain Template
INFO:FederationManager:Configuring the STSUU to SAML11 Module Chain Mapping
INFO:FederationManager:Successfully configured the STSUU to SAML11 Module Chain Mapping
INFO:STSUUtoSAML11:Successfully configured the STSUU to SAML11 chain

24.6.2 Testing

To invoke the STS runtime we POST a formatted WS-Trust XML SOAP request to STS endpoint.

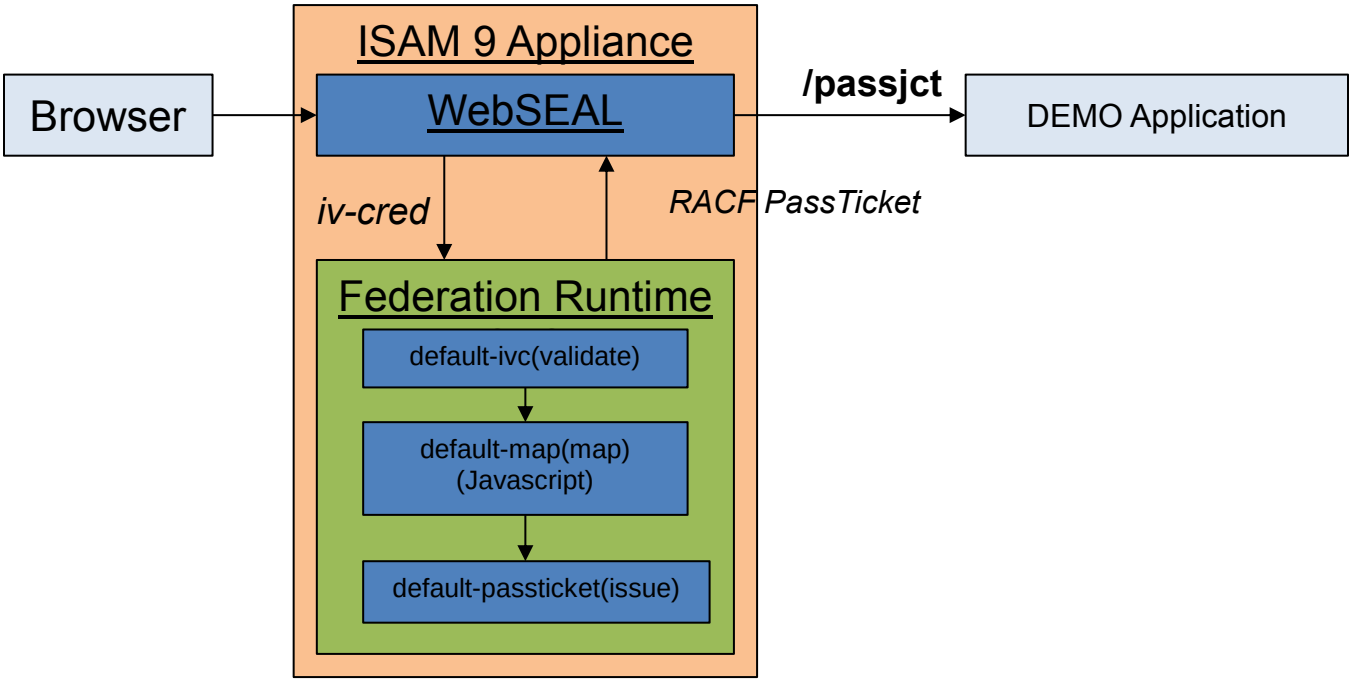
The providedfiles/ststest directory includes an example soap message and shell command to invoke cUrl with the correct parameters for WS-Trust 1.2. The set of provided files related to this example are:

Filename	Description
rst12_saml11.sh	Executable script to launch the request
rst12_stsuu_to_saml11.xml	XML of the request body containing the UsernameToken to validate
stsuu_to_saml11.js	Javascript mapping rule

24.6.3 Further Details

24.7 RACF PassTicket Junction

This example establishes an STS chain and WebSEAL configuration that allows WebSEAL to downstream an Passticket BinarySecurityToken across a WebSEAL junction. This is very similar to the SAML Junction configuration that is performed in section 18 of this document except instead of down streaming a SAML assertion to the junctioned application we downstream a RACF PassTicket. The integration looks like this:



24.7.1 Pre-requisites and Configuration

This use case has the same pre-requisites as the SAML Junction use case. It is assumed that your IDP image already has the SAML IDP federation configured because mapping rules are loaded during the appliance setup done by that application.

The following script may be used to configure the example against the IDP image used in this cookbook.

SCRIPT-START:
A script is available for this section as an alternative to following the manual steps. This script performs operations against the IdP image only.

Run this script: **PassticketJunction.py -configure All**

If you use this script, skip to the corresponding SCRIPT-END notice

SCRIPT-END:
The script should display the following:
INFO:LTPA Junction:Configuring Passticket Junction
INFO:FederationManager:Configuring the IVCred to Passticket Chain Template
INFO:FederationManager:Successfully configured the IVCred to Passticket Chain Template
INFO:FederationManager:Configuring the IVCred to Passticket Module Chain Mapping
INFO:FederationManager:Retrieving the mapping rule reference ID
INFO:FederationManager:Successfully configured the IVCred to Passticket Module Chain Mapping
INFO:WGAManager:Configuring WebSEAL.conf file for LTPA Junction at IdP
INFO:WGAManager:Successfully configured WebSEAL.conf file for LTPA junction at IdP
INFO:Passticket Junction:End LTPA junction creation and configuration

Configuration of this use case leverages the following files (included in the providedfiles accompanying this cookbook):

Filename	Description
ivc_to_passticket.js	Javascript mapping rule

24.7.2 Testing

To test this use case, use a browser to visit the URL:

https://www.myidp.ibm.com/passjct/mobile-demo/diag/

Following authentication as the testuser, you should be able to scroll down and see the RACF PassTicket BinarySecurityToken that was sent as a HTTP header:

Latest Risk Score Report for user: testuser

The risk score calculation report property (riskEngine.reportsEnabled) in the advanced configuration panel isn't enable.
To display risk score reports, you will need to enabled them in the LMI.

Access Manager Credential:

User: **testuser**

tagvalue_user_session_id[0]	SHRhTzBTNWxjV2tudXYwaVM4OFoyK084czZvVWpMc3BEZ1RjaWFxYVo3UGUxQ2R5:default
AZN_CRED_NETWORK_ADDRESS_STR[0]	192.168.42.1
AZN_CRED_AUTHNMECH_INFO[0]	LDAP Registry
AZN_CRED_MECH_ID[0]	IV_LDAP_V3.0
firstName[0]	Test
firstName[1]	testuser
tagvalue_session_index[0]	45f96714-f83a-11e8-b597-000c29279998
AZN_CRED_IP_FAMILY[0]	AF_INET
AZN_CRED_PRINCIPAL_UUID[0]	c02ef2ae-f2e2-11e8-8489-000c29279998
AZN_CRED_QOP_INFO[0]	SSK: TLSV12: 0A
AZN_CRED_VERSION[0]	0x00000907
AZN_CRED_BROWSER_INFO[0]	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:60.0) Gecko/20100101 Firefox/60.0

HTTP Headers:

iv-creds:	PMAkwBwIBBAwABAAwJwwQQVpOX0NSRURfVkVSU0IPTjATMBECAQQMCjB4MDAwMDA5MDcEADArDAImaXJzdE5hbWUwHjALAgEEDARUZXN0BAAwDwIBBAwIdGVzdHVzZXIEADAZDAhsYXN0TmFtZTANMAsCAQQMBFVzZXIEADAtDBh0YWd2YWx1ZV9sb2dpbI91c2VyX25hbWUwETAPAgEEDAh0ZXN0dXNIcgQAMDYMJHRhZ3ZhbHVIX21heF9jb25jdXJyZW50X3dIYI9zZXNzaW9uczAOMAwCAQQMBXVuc2V0BAAwRwwWdGFndmFs
iv-user:	testuser
User-Agent:	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:60.0) Gecko/20100101 Firefox/60.0
Via:	HTTP/1.1 www.myidp.ibm.com:443
upgrade-insecure-requests:	1
iv_server_name:	default-webseald-isam.myidp.ibm.com
passtickettoken:	<wss:UsernameToken xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="username7c51aa03-0167-16ac-986f-81a979c56922"><wss:Username>testuser</wss:Username><wss:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">RHmUa+OJUjQcead4dDtOIw==</wss:Nonce><wsu:Created>2018-12-05T03:03:05Z</wsu:Created><wss:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">0ACGF2HN</wss:Password></wss:UsernameToken>

25 Appendix D – Manual ISAM Configuration steps for IdP and SP

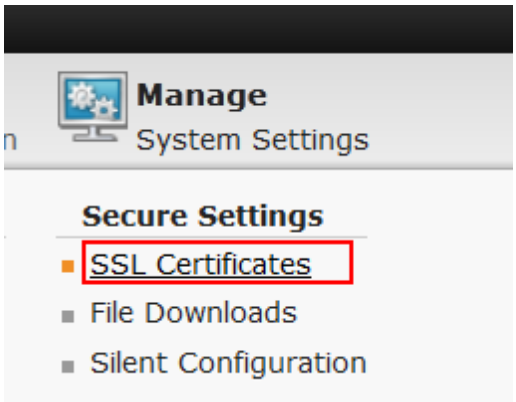
This section documents the step by step guide to ISAM configuration for IdP and SP. The recommended approach is to use the new Reverse Proxy->Federation Management UI.

25.1 ISAM Configuration for the IdP

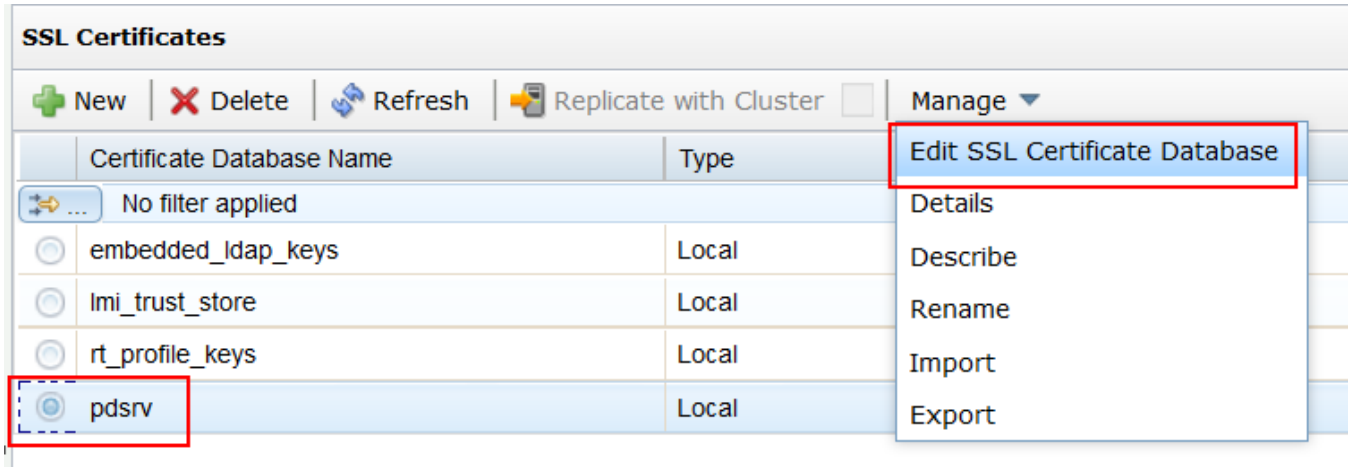
This section is completed only for the Identity Provider.
You will configure the Service Provider in a later section.

25.1.1 Load Federation Runtime SSL certificate into pdsrv trust store

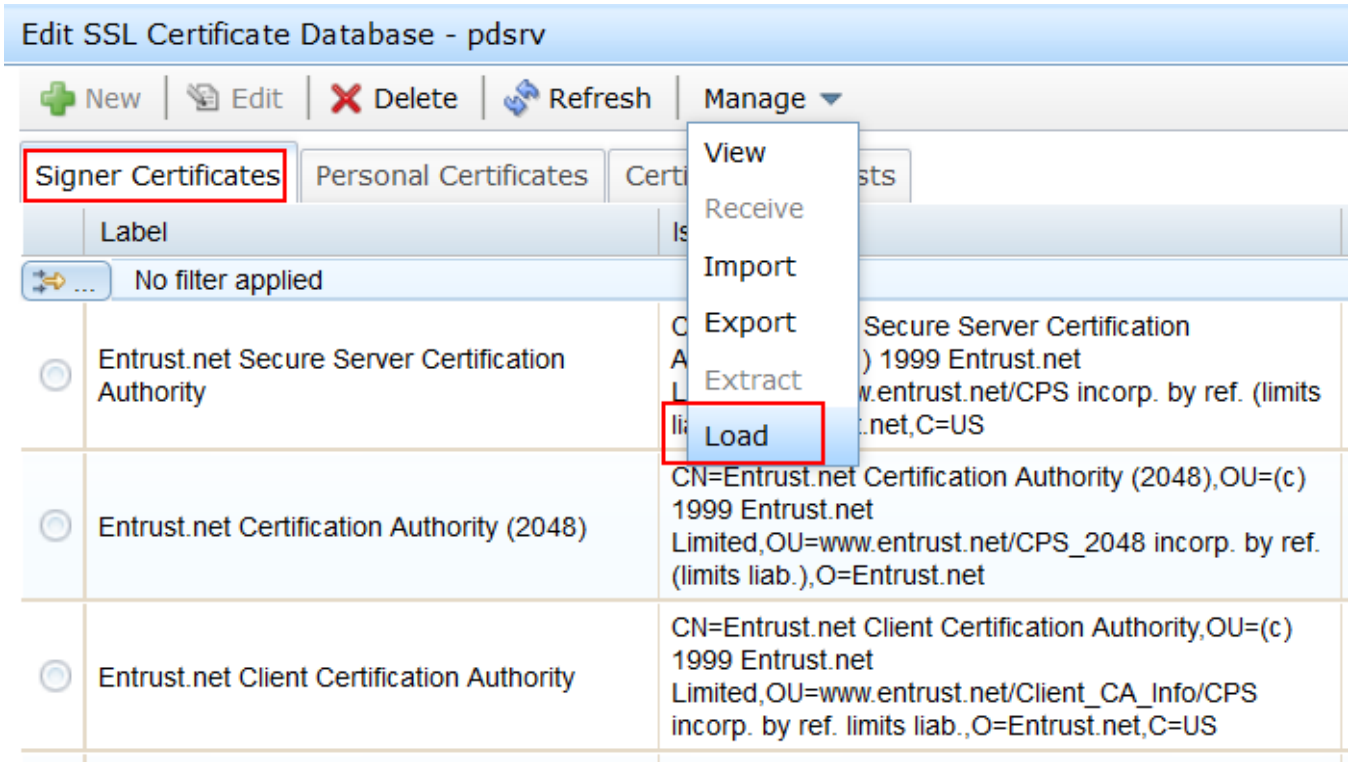
You need to import the certificate that the runtime server uses into the pdsrv keystore. It is needed so that the runtime junction creation will not fail.



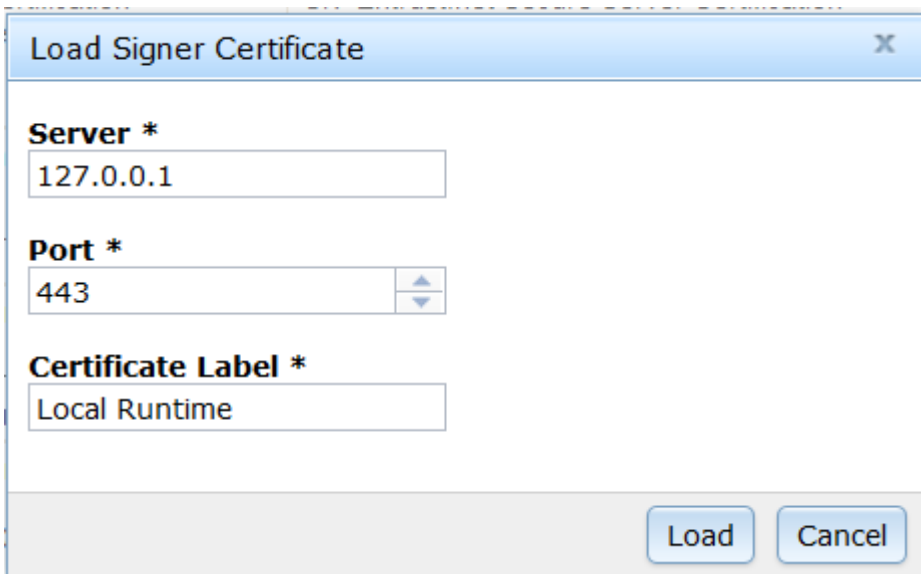
Navigate to **Manage System Settings > Secure Settings > SSL Certificates** in the IdP appliance console / LMI



Select the pdsrv certificate database. Click **Manage > Edit SSL Certificate Database**



Click on the **Signer Certificates** tab. Click **Manage > Load**

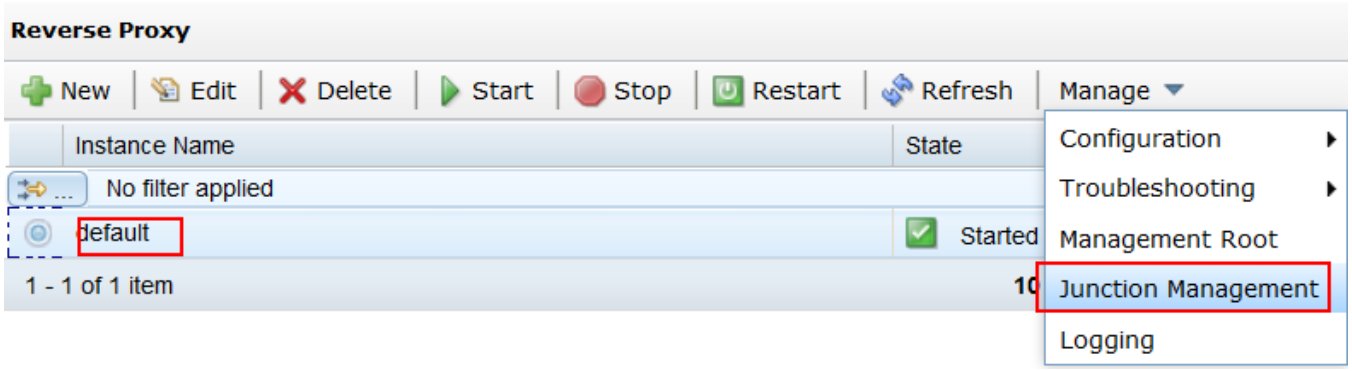


Load the certificate from localhost as shown above.

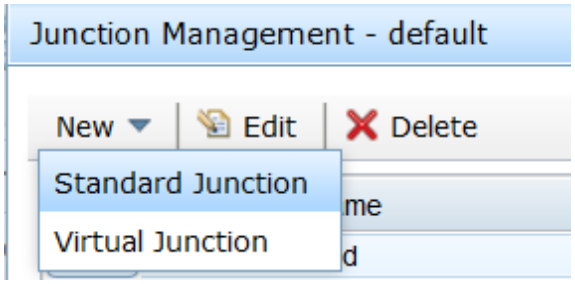
Save and deploy the changes. Navigate to the Secure Web Settings | Reverse Proxy menu. Select and restart the reverse proxy instance and ensure that the changes are active after restarting.

25.1.2 Configure runtime junction for the IdP

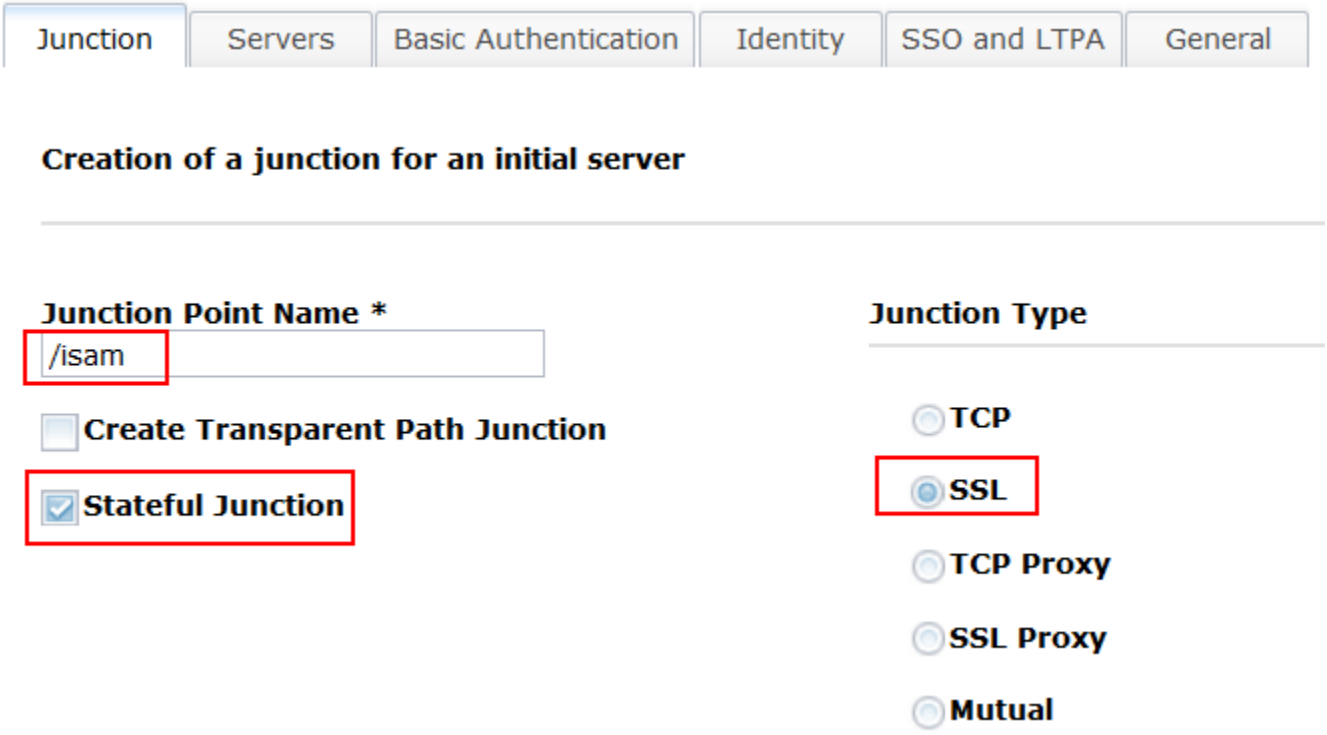
Navigate to **Secure Web Settings > Manage > Reverse Proxy**.



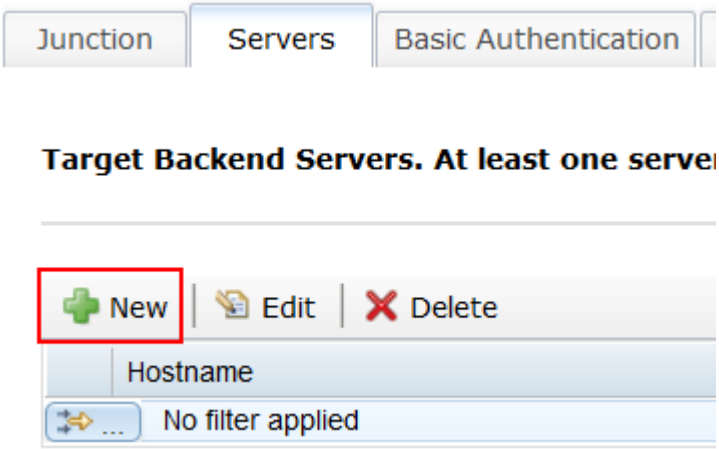
Select the reverse proxy instance and click on **Manage > Junction Management**



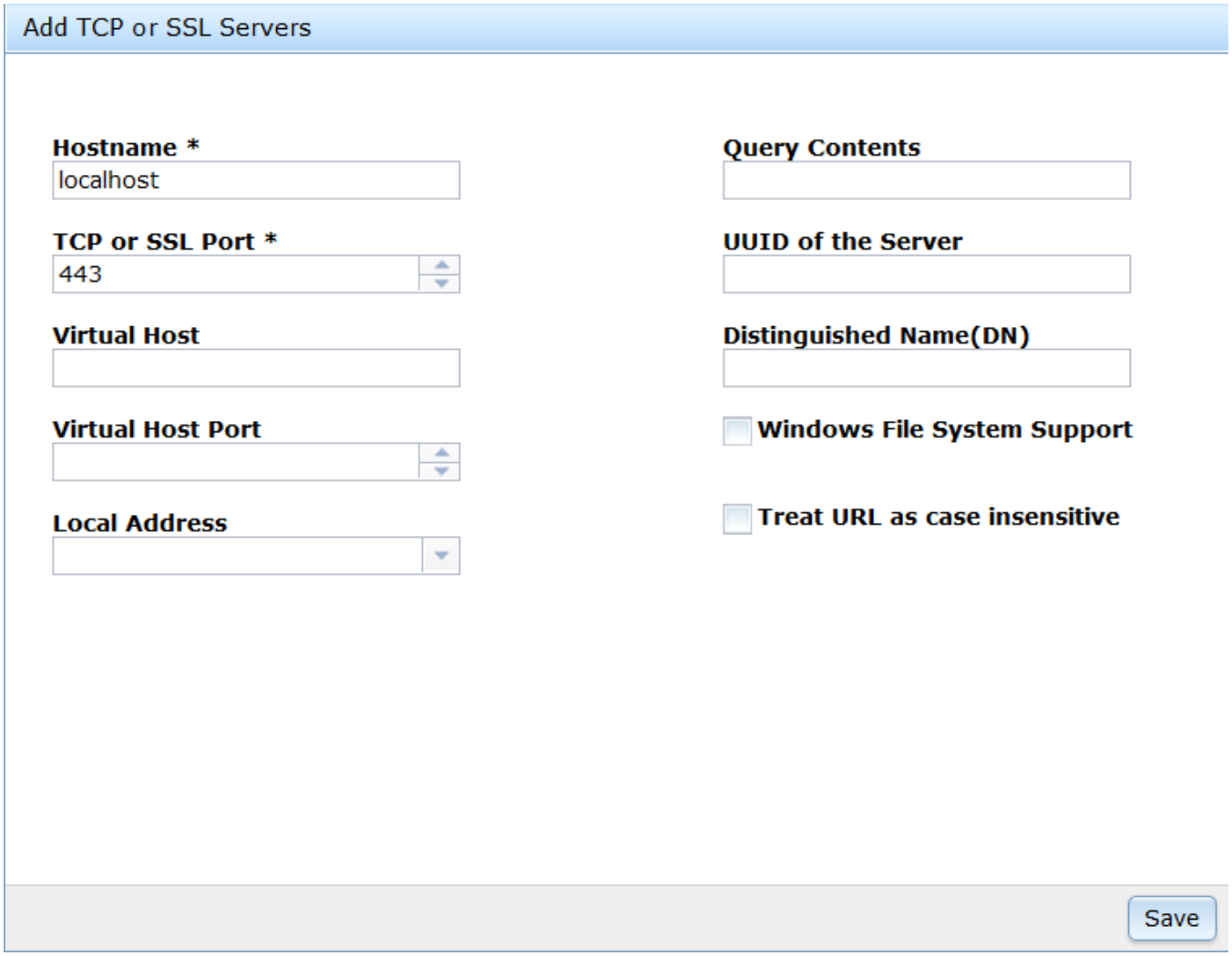
Click on **New > Standard Junction**



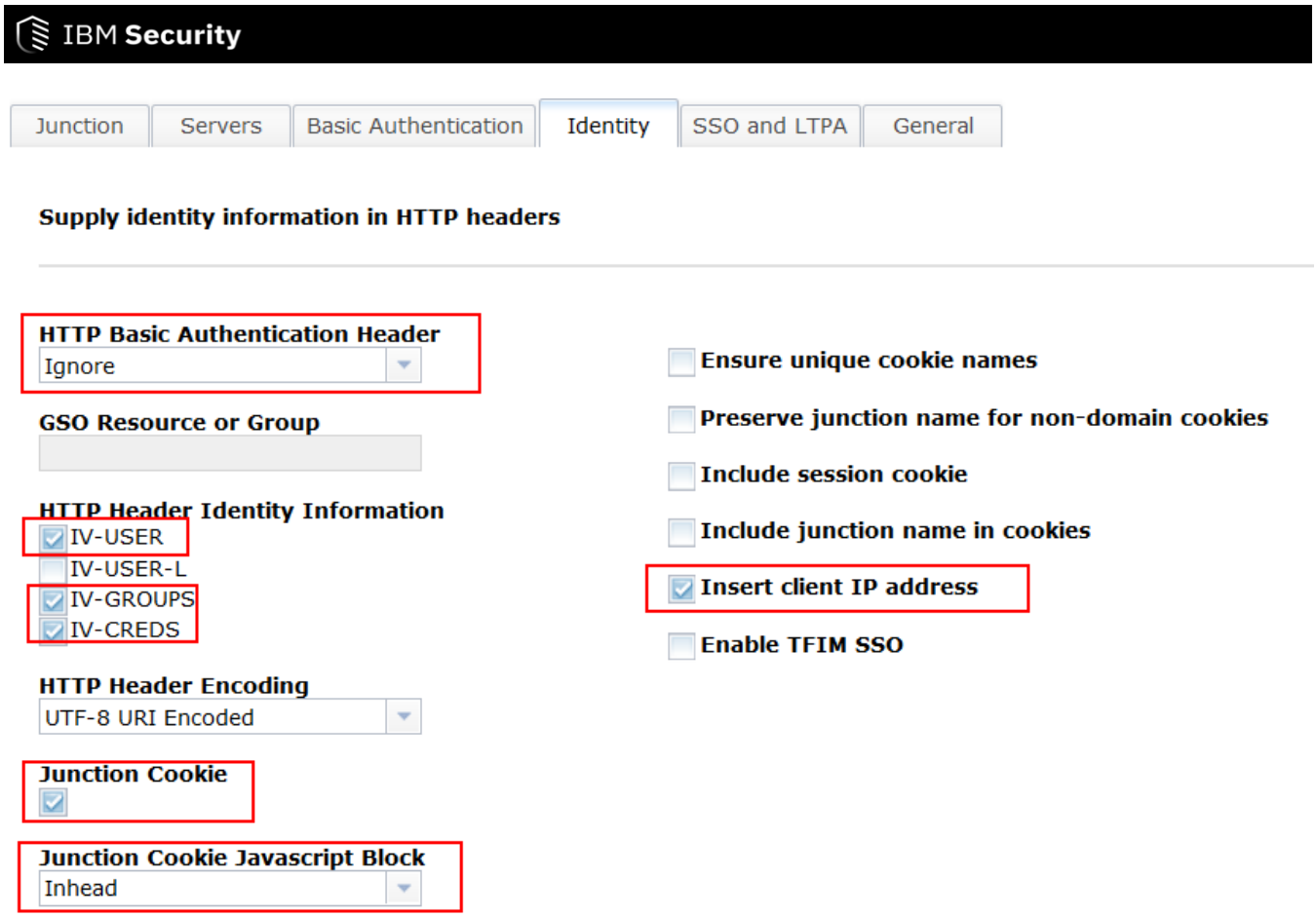
In the **Junction** tab, enter */isam* as the junction name. Select **SSL** and **Stateful**



Junction In the **Servers** tab, click on **New**.



Enter the **Hostname*** as *localhost* and **TCP or SSL Port*** as *443*. Click on **Save**.



In the **Identity** tab, set the values as shown below.

Click on **Save** and close the Junction dialog after the Save has completed.

25.1.3 Configure ACL policy for IdP

Open an SSH session to the appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

SSH to **isam.myidp.ibm.com** and authenticate using the administrator credentials:

```
The authenticity of host 'isam.myidp.ibm.com (192.168.42.101)' can't be established.
ECDSA key fingerprint is SHA256:hXml4xBfov+C9/4pxAgxh5IDh7BR4JUBbbbMnibPNPM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'isam.myidp.ibm.com,192.168.42.101' (ECDSA) to the list of known
hosts.
admin@isam.myidp.ibm.com's password: Passw0rd
Last login: Wed Sep 23 13:41:05 2015
Welcome to the IBM Security Access Manager
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.myidp.ibm.com>
```

Navigate to **isam** and start the **admin** utility:

```
isam.myidp.ibm.com> isam
isam.myidp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Create an unauth ACL using the commands:

```
acl create saml20idp-unauth
acl modify saml20idp-unauth set group iv-admin TcmdbsvaBRrxl
acl modify saml20idp-unauth set group webseal-servers Tgmdbsrxl
acl modify saml20idp-unauth set user sec_master TcmdbsvaBRrxl
acl modify saml20idp-unauth set any-other Tr
acl modify saml20idp-unauth set unauthenticated Tr
```

Attach the ACL to the SAML endpoints using the commands:

```
acl attach /WebSEAL/isam.myidp.ibm.com-default/favicon.ico saml20idp-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/saml20idp/saml20/login saml20idp-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/saml20idp/saml20/logininitial saml20idp-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/saml20idp/saml20/slo saml20idp-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/saml20idp/saml20/sloinitial saml20idp-unauth
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/static saml20idp-unauth
```

Create an anyauth ACL and attach it to the SAML endpoints using the following commands:

```
acl create saml20idp-anyauth
acl modify saml20idp-anyauth set group iv-admin TcmdbsvaBRrxl
acl modify saml20idp-anyauth set group webseal-servers Tgmdbsrxl
acl modify saml20idp-anyauth set user sec_master TcmdbsvaBRrxl
acl modify saml20idp-anyauth set any-other Tr
acl modify saml20idp-anyauth set unauthenticated T
acl attach /WebSEAL/isam.myidp.ibm.com-default/isam/sps/saml20idp/saml20/auth saml20idp-anyauth
```

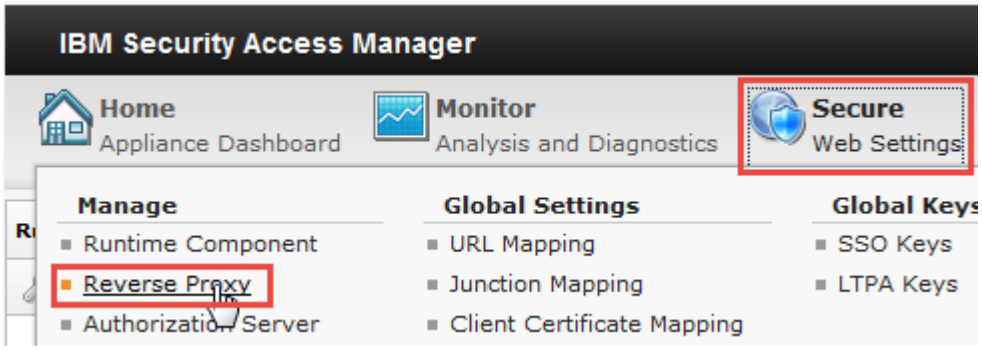
Run the object modify command to configure HTTP-Tag-Value so that Reverse Proxy passes user_session_id to federation runtime:

```
object modify /WebSEAL/isam.myidp.ibm.com-default/isam/ set attribute HTTP-Tag-Value
user_session_id=user_session_id
```

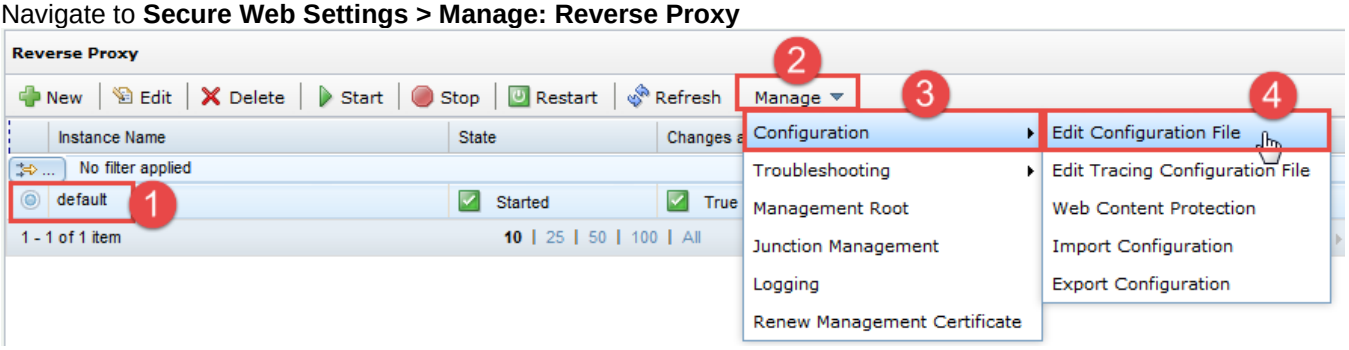
Run server replicate command to save the changes using the command:

```
server replicate
```

25.1.4 Configure the IdP reverse proxy



Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

In the eai stanza modify *eai-auth* and add *eai-redir-url-priority* as highlighted in red:

```
[eai]
…
eai-auth = https
…
eai-redir-url-priority = yes
```

In the eai-trigger-urls stanza add the following entries highlighted in red.

```
# EAI TRIGGER URLS
[eai-trigger-urls]
# If eai-auth is not 'none', then WebSEAL will examine the URLs of incoming
# requests to determine if they match one of the entries in this list.
# If they do, then WebSEAL will examine the corresponding server response to
# determine if it contains authentication data.
…
trigger = /isam/sps/saml20idp/saml20/login*
trigger = /isam/sps/saml20idp/saml20/slo*
trigger = /isam/sps/saml20idp/saml20/soap*
trigger = /isam/sps/auth*
```

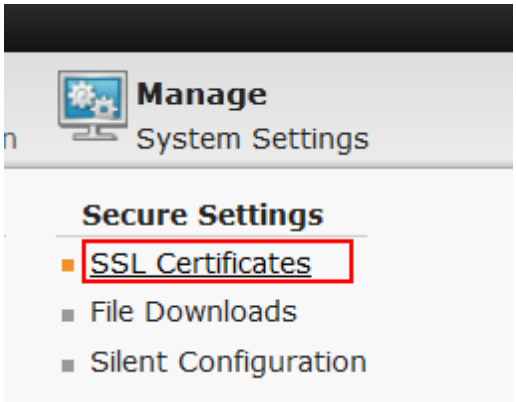
Save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

25.2 ISAM Configuration for SP

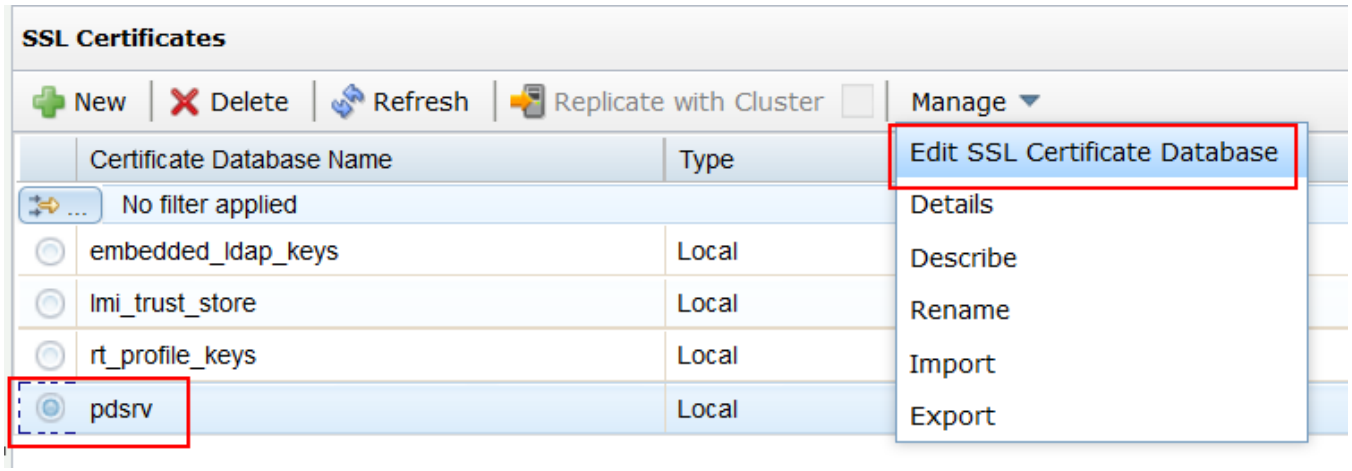
This section is completed only for the Service Provider.
You should have configured the Identity Provider in the previous section.

25.2.1 Load Federation Runtime SSL certificate into pdsrv trust store

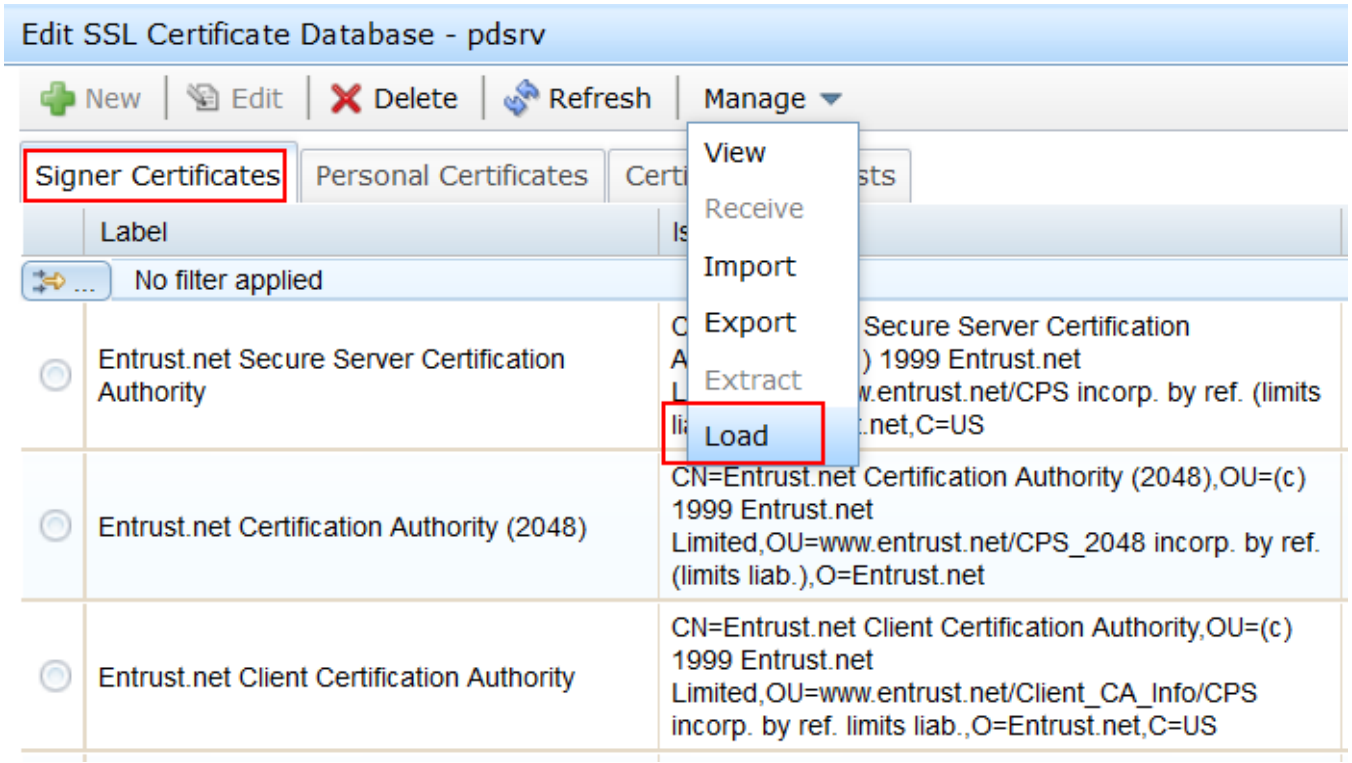
You need to import the certificate that the runtime server uses into the pdsrv keystore. It is needed so that the runtime junction creation will not fail.



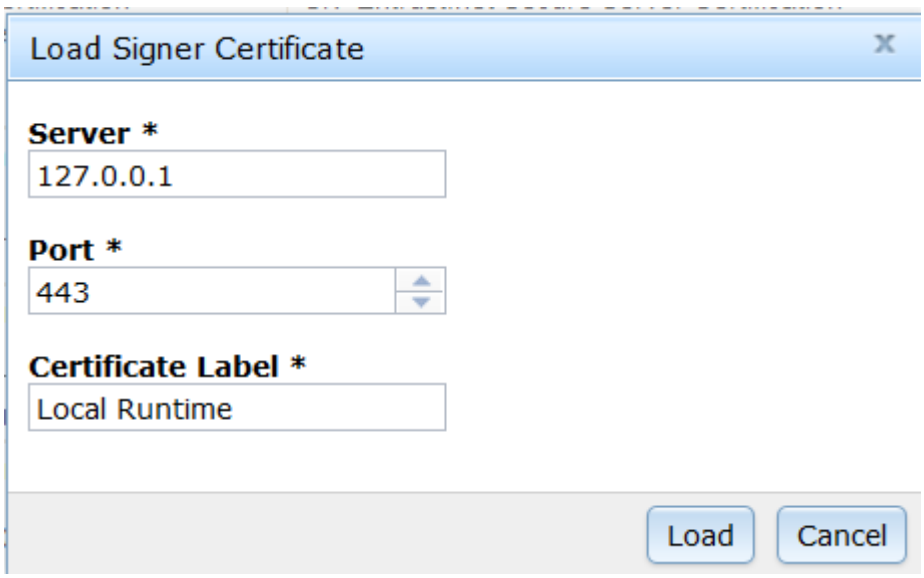
Navigate to **Manage System Settings > Secure Settings > SSL Certificates** in the SP appliance console / LMI



Select the pdsrv certificate database. Click **Manage > Edit SSL Certificate Database**



Click on the **Signer Certificates** tab. Click **Manage > Load**

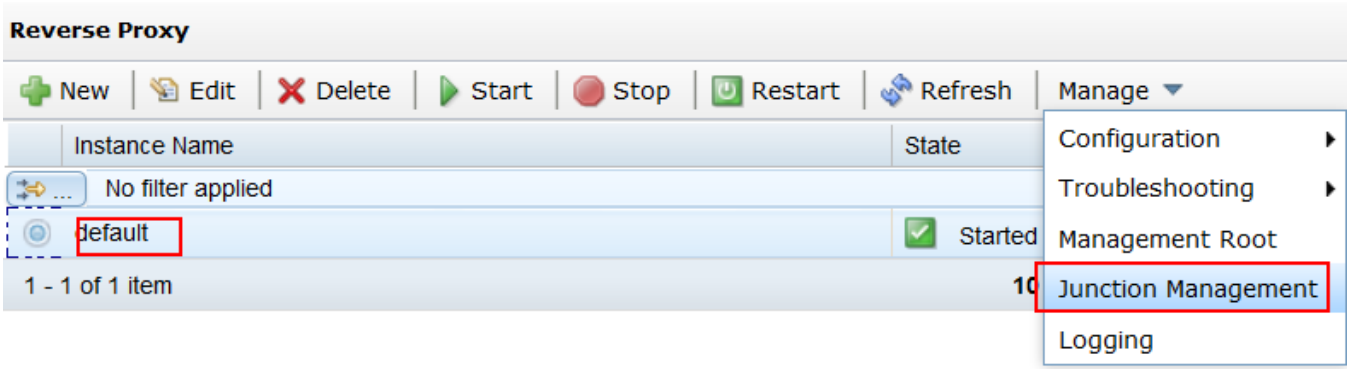


Load the certificate from localhost as shown above.

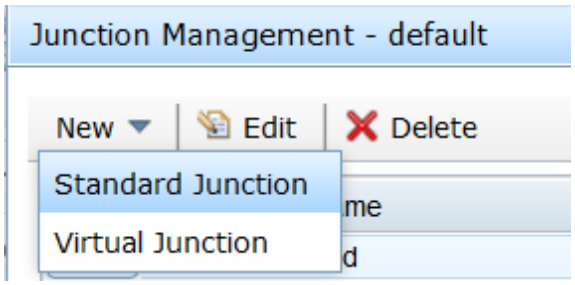
Save and deploy the changes. Navigate to the **Secure Web Settings→Reverse Proxy** menu. Select and restart the reverse proxy instance and ensure that the changes are active after restarting.

25.2.2 Configure runtime junction for the SP

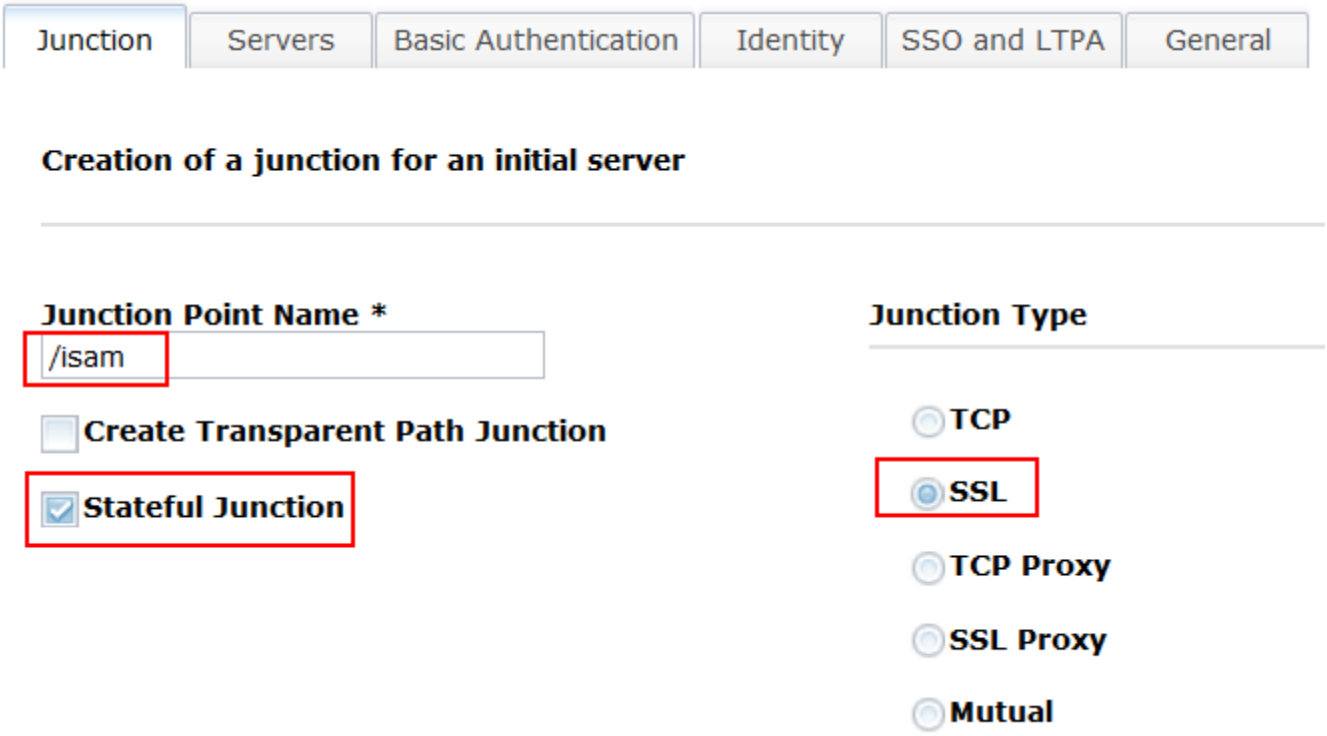
Navigate to **Secure Web Settings > Manage > Reverse Proxy**.



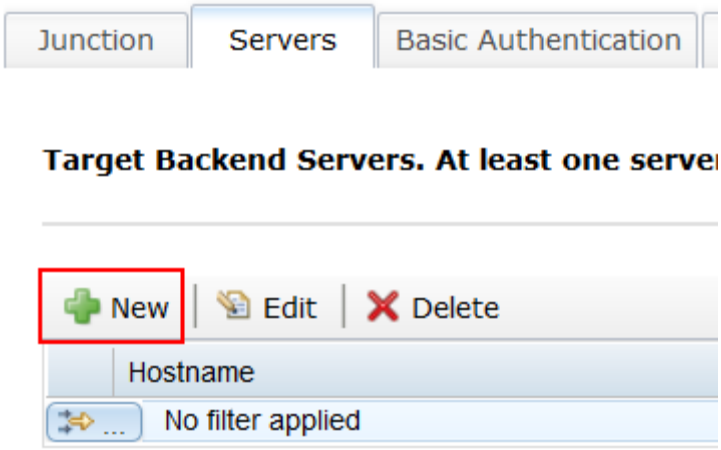
Select the reverse proxy instance and click on **Manage > Junction Management**



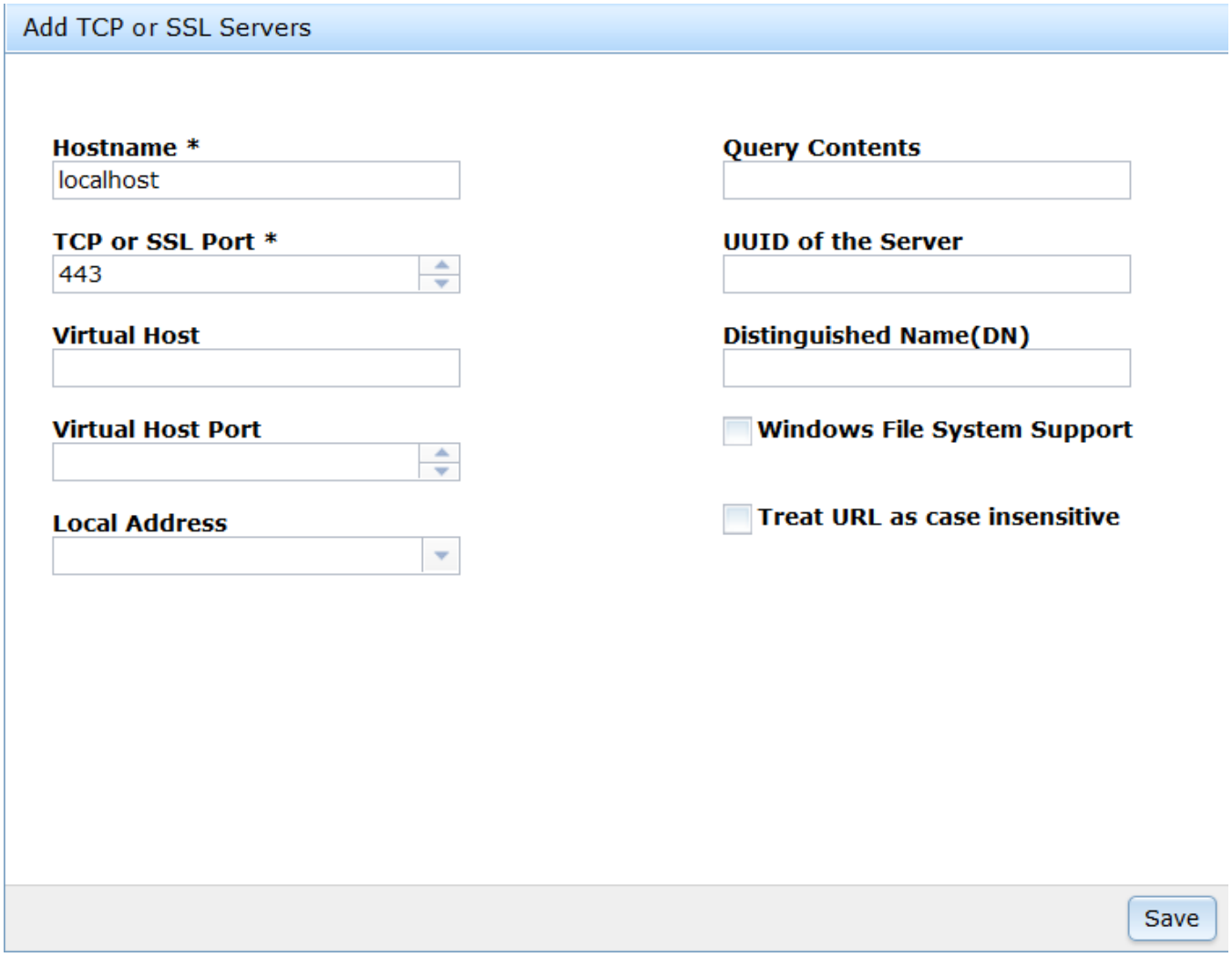
Click on **New > Standard Junction**



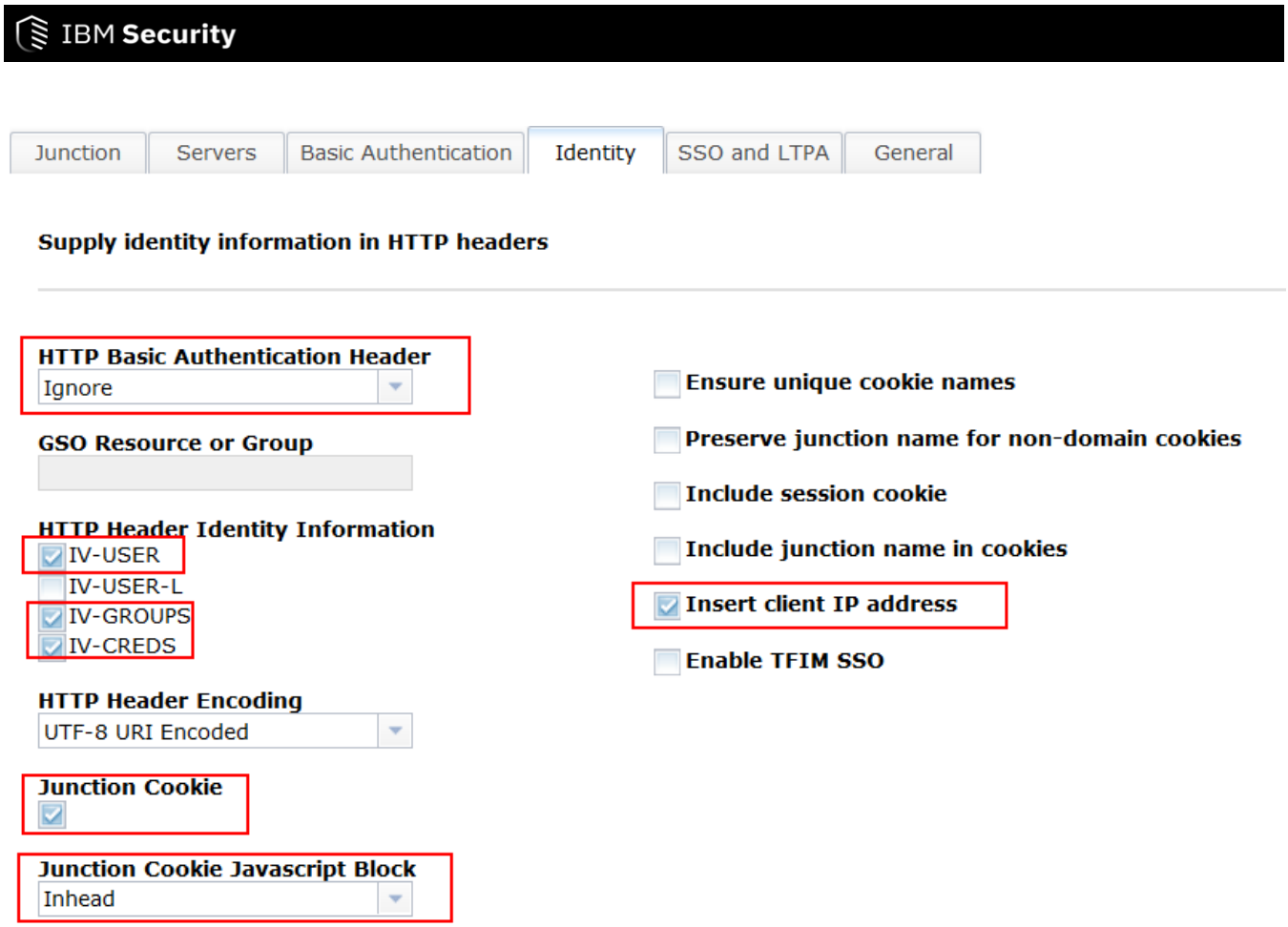
In the **Junction** tab, enter */isam* as the junction name. Select **SSL** and **Stateful**



Junction In the **Servers** tab, click on **New**.



Enter the **Hostname*** as *localhost* and **TCP or SSL Port*** as *443*. Click on **Save**.



In the **Identity** tab, set the values as shown below.

Click on **Save** and close the Junction dialog after the Save has completed.

25.2.3 Configure ACL policy for SP

Open an SSH session to the appliance. You could use ssh command-line (on a Linux system or in Cygwin) or you could use PuTTY. You could also connect directly to the console of the appliance via VMWare.

SSH to **isam.mysp.ibm.com** and authenticate using the administrator credentials.

```
The authenticity of host 'isam.mysp.ibm.com (192.168.42.201)' can't be established.
ECDSA key fingerprint is SHA256:RSDlRkEy+lcZhtd8N53yR8mKiE0llDGx303gQ2IwWRg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'isam.mysp.ibm.com,192.168.42.201' (ECDSA) to the list of known
hosts.
admin@isam.mysp.ibm.com's password: Passw0rd
Last login: Wed Sep 23 13:44:26 2015
Welcome to the IBM Security Access Manager
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.mysp.ibm.com>
```

Navigate to **isam** and start the **admin** utility:

```
isam.mysp.ibm.com> isam
isam.mysp.ibm.com:isam> admin

pdadmin>
```

Login to the pdadmin console using the command **: login -a sec_master -p Passw0rd** . The password was set for the user sec_master in one of the earlier sections.

```
pdadmin> login -a sec_master -p Passw0rd
```

Create an unauth ACL using the commands and attach it to SAML endpoints:

```
acl create saml20sp-unauth
acl modify saml20sp-unauth set group iv-admin TcmdbsvaBRrxl
acl modify saml20sp-unauth set group webseal-servers Tgmdbsrxl
acl modify saml20sp-unauth set user sec_master TcmdbsvaBRrxl
acl modify saml20sp-unauth set any-other Tr
acl modify saml20sp-unauth set unauthenticated Tr
acl attach /WebSEAL/isam.mysp.ibm.com-default/favicon.ico saml20sp-unauth
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/saml20sp/saml20/login saml20sp-unauth
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/saml20sp/saml20/logininitial saml20sp-unauth
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/saml20sp/saml20/slo saml20sp-unauth
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/saml20sp/saml20/sloinitial saml20sp-unauth
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/static saml20sp-unauth
```

Create an anyauth ACL and attach it to the SAML endpoints using the following commands:

```
acl create saml20sp-anyauth
acl modify saml20sp-anyauth set group iv-admin TcmdbsvaBRrxl
acl modify saml20sp-anyauth set group webseal-servers Tgmdbsrxl
acl modify saml20sp-anyauth set user sec_master TcmdbsvaBRrxl
acl modify saml20sp-anyauth set any-other Tr
acl modify saml20sp-anyauth set unauthenticated T
acl attach /WebSEAL/isam.mysp.ibm.com-default/isam/sps/saml20sp/saml20/auth saml20sp-anyauth
```

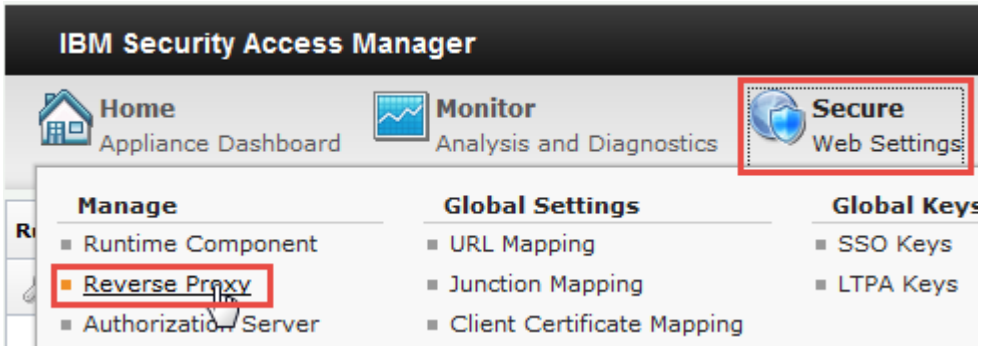
Run the object modify command to pass HTTP-Tag-Value attribute.

```
object modify /WebSEAL/isam.mysp.ibm.com-default/isam/ set attribute HTTP-Tag-Value
user_session_id=user_session_id
```

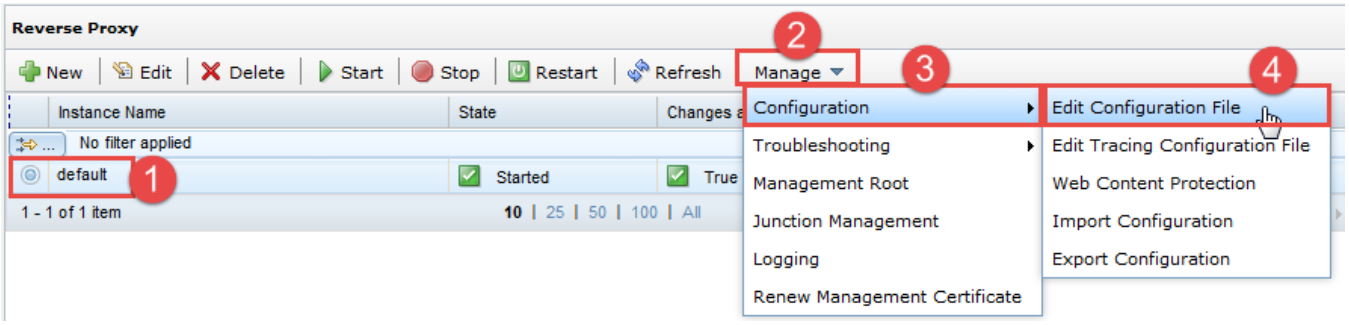
Save the changes by running the following command.

```
server replicate
```

25.2.4 Configure the SP reverse proxy



Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the checkbox for the **default** Reverse Proxy instance. Click on **Manage** and select **Configuration→Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

In the authentication-levels stanza add the following entries highlighted in green.
You should also remove the password level although it won't matter for these use cases.

```
[authentication-levels]
…
level = unauthenticated
level = password
level = ext-auth-interface
```

In the eai stanza modify *eai-auth* and add *eai-redir-url-priority* as highlighted in red:

```
[eai]
…
eai-auth = https
…
eai-redir-url-priority = yes
```

In the eai-trigger-urls stanza add the following entries highlighted in red.

```
# EAI TRIGGER URLS
[eai-trigger-urls]
# If eai-auth is not 'none', then WebSEAL will examine the URLs of incoming
# requests to determine if they match one of the entries in this list.
# If they do, then WebSEAL will examine the corresponding server response to
# determine if it contains authentication data.
…
trigger = /isam/sps/auth*
trigger = /isam/sps/saml20sp/saml20/soap*
trigger = /isam/sps/saml20sp/saml20/slo*
trigger = /isam/sps/saml20sp/saml20/login*
```

Save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

26 Appendix E – Using cURL to call POC Configuration REST

26.1 Configuration for the IdP

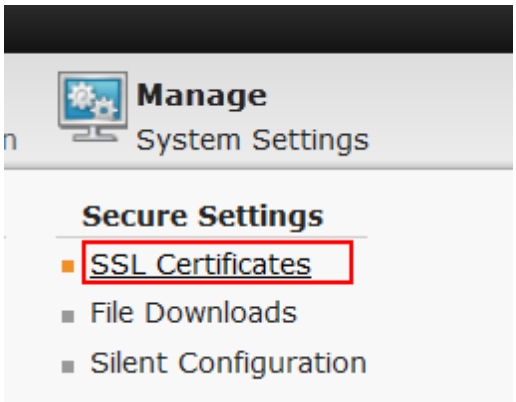
A junction on the Reverse Proxy instance for the IdP should be created, such that the federation runtime can be accessed through the Reverse Proxy. In order to do this, the SSL Certificate presented by the federation runtime must be added to **pdsrv**, the default trust store for the Reverse Proxy instances.

Once this certificate has been loaded, the auto-configuration endpoint will be invoked, making changes to the Reverse Proxy configuration file and ACLs, and adding the junction to the federation runtime.

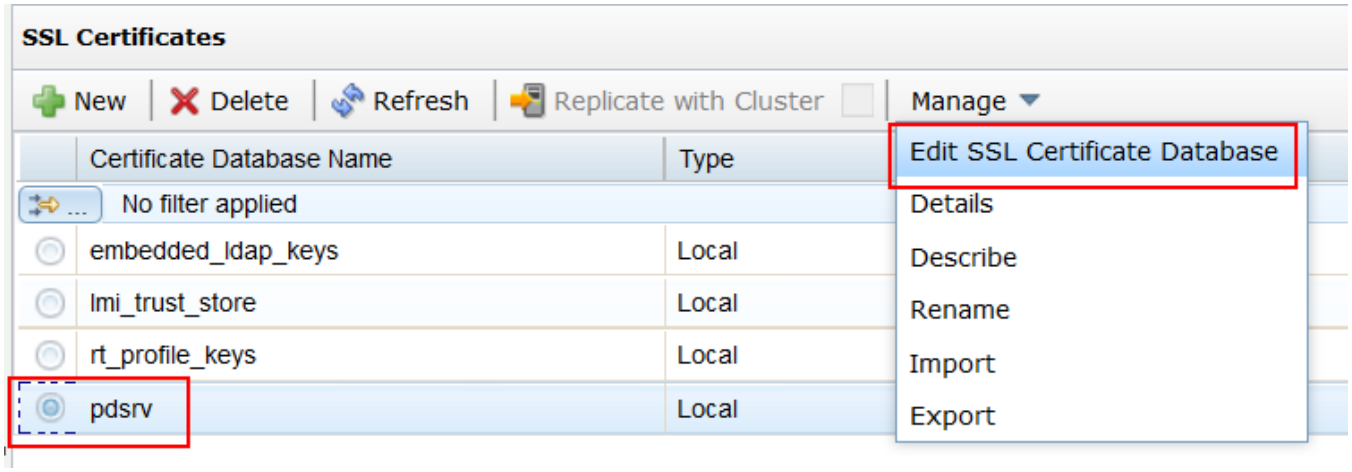
This section is completed only for the Identity Provider.
You will configure the Service Provider in a later section.

26.1.1 Load Federation Runtime SSL certificate into pdsrv trust store

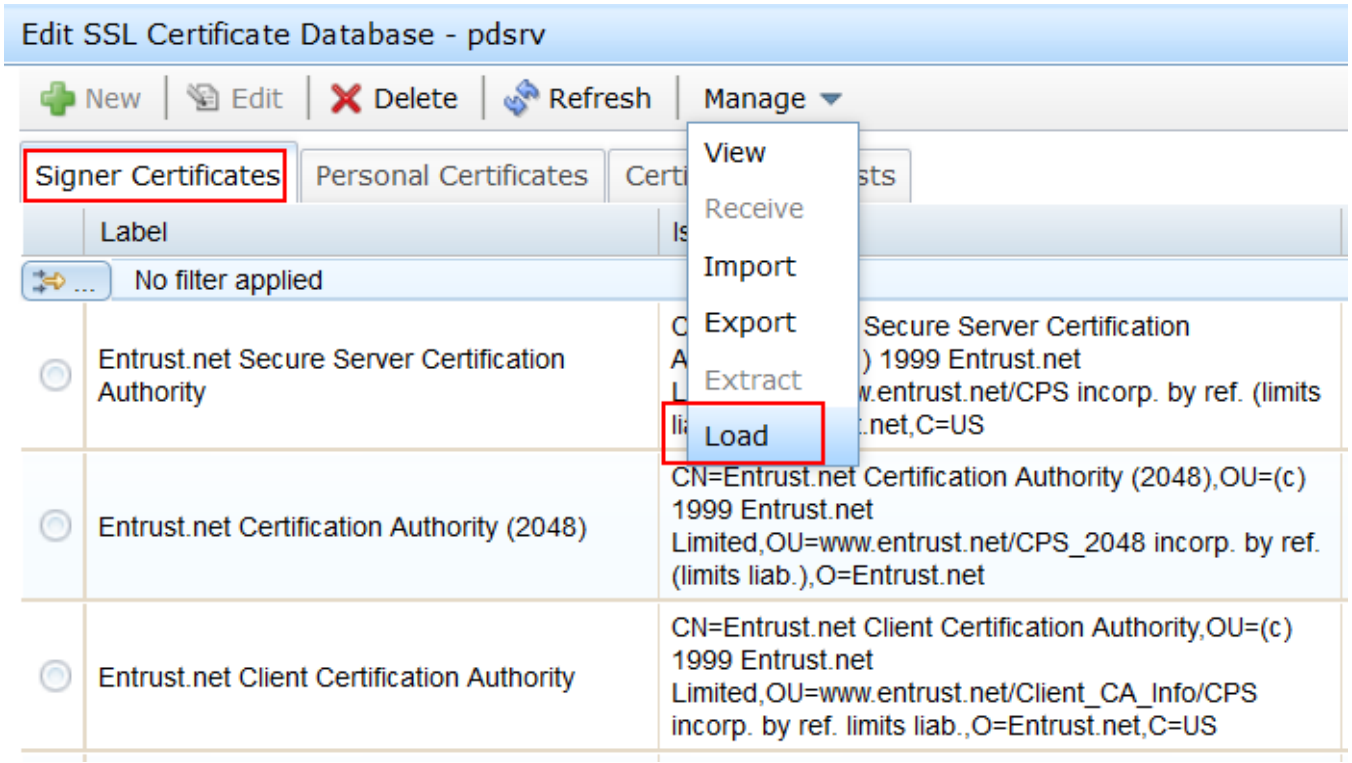
You need to import the certificate that the runtime server uses into the pdsrv keystore. It is needed so that the runtime junction creation will not fail.



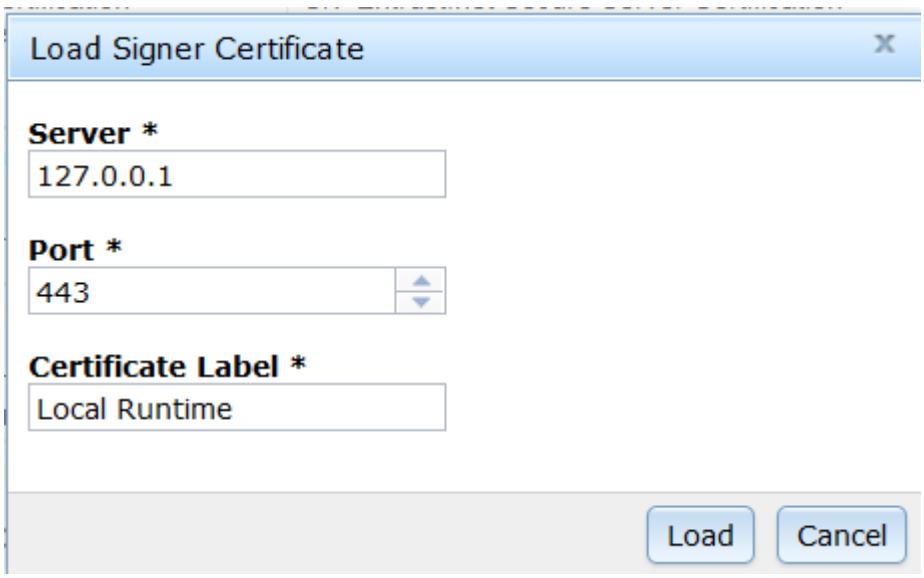
Navigate to **Manage System Settings > Secure Settings > SSL Certificates** in the IdP appliance console / LMI



Select the pdsrv certificate database. Click **Manage > Edit SSL Certificate Database**



Click on the **Signer Certificates** tab. Click **Manage > Load**



Load the certificate from 127.0.0.1 as shown above.

Save and deploy the changes. Navigate to the **Secure Web Settings → Reverse Proxy** menu. Select and restart the reverse proxy instance and ensure that the changes are active after restarting.

26.1.2 Federation auto-configuration endpoint

The 'fed_config' endpoint is an API endpoint which you will access via cURL.

Issue a GET request to the federations endpoint to view the id of the ISAMOP federation

```
$ curl -k -u admin:Passw0rd -H 'Accept: application/json' -H 'Content-Type: application/json'
https://isam.myidp.ibm.com/iam/access/v8/federations
[{"protocol":"OIDC","role":"op","templateName":"","configuration":{"authorizationCodeLength":
30,"authorizationGrantLifetime":604800,"grantTypesSupported":["authorization_code","implicit"
],"issuerIdentifier":"https:\/\/www.myidp.ibm.com","attributeMapping":{"map":[]},"authorizati
onCodeLifetime":30,"accessTokenLength":40,"identityMapping":{"activeDelegateId":"default-
map","properties":{"identityMappingRuleReference":"5","ruleType":"JAVASCRIPT"}},"idTokenLifet
ime":7200,"signatureAlgorithm":"HS256","accessTokenLifetime":7200,"refreshTokenLength":50},"n
ame":"ISAMOP","id":"uuidd36144cb-0152-1a77-9a02-febccb94da75"}]
```

In this example, there is only one federation, the id of which is **uuidd36144cb-0152-1a77-9a02-febccb94da75**. Issue a POST request containing the following data, substituting the federation id with the one from your environment.

```
$ curl -k -v -u admin:Passw0rd -H 'Accept: application/json' -H 'Content-Type:
application/json' https://isam.myidp.ibm.com/wga/reverseproxy/default/fed_config -d '{
"runtime":{"hostname":"localhost","port":"443","username":"easuser","password":"Passw0rd"
},"federation_id":"uuidd36144cb-0152-1a77-9a02-febccb94da75"}'
*   Trying 192.168.42.101...
...
* Server auth using Basic with user 'admin'
> POST /wga/reverseproxy/default/fed_config HTTP/1.1
> Host: isam.myidp.ibm.com
> Authorization: Basic YWRtaW46UGFzc3cwcmQ=
> User-Agent: curl/7.43.0
> Accept: application/json
> Content-Type: application/json
> Content-Length: 157
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 204 No Content
...
<
* Connection #0 to host isam.myidp.ibm.com left intact
```

An HTTP response code of 204 indicates that the request completed successfully. The changes have been made to the policy server, the reverse proxy configuration file, and the reverse proxy junctions

Open the LMI, and save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

26.2 ISAM Configuration for SP

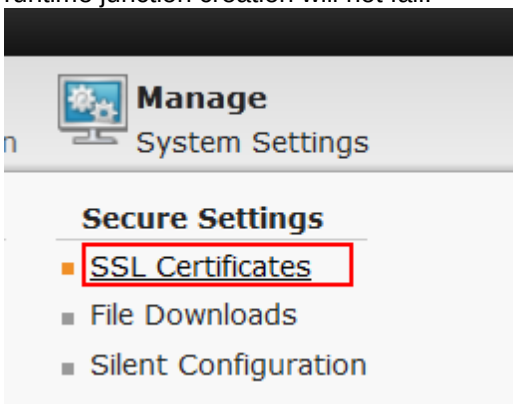
A junction on the Reverse Proxy instance for the SP should be created, such that the federation runtime can be accessed through the Reverse Proxy. In order to do this, the SSL Certificate presented by the federation runtime must be added to **pdsrv**, the default trust store for the Reverse Proxy instances.

Once this certificate has been loaded, the auto-configuration endpoint will be invoked, making changes to the Reverse Proxy configuration file and ACLs, and adding the junction to the federation runtime.

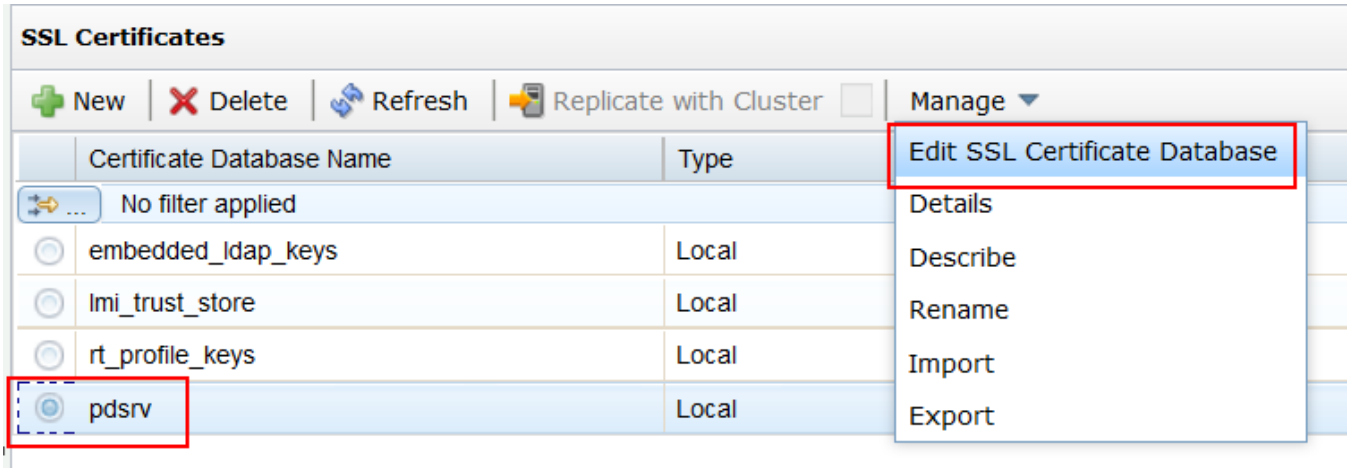
This section is completed only for the Service Provider.
You should have configured the Identity Provider in the previous section.

26.2.1 Load SSL certificates

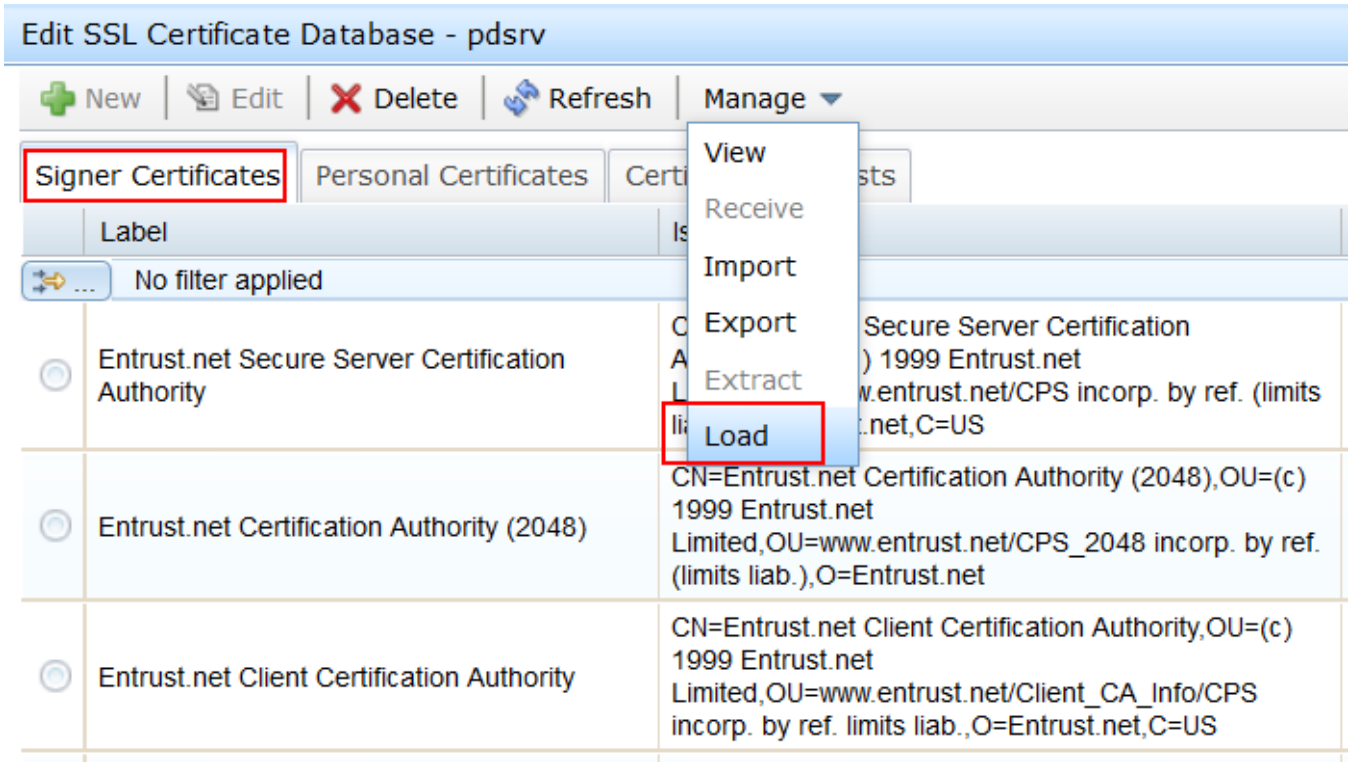
You need to import the certificate that the runtime server uses into the pdsrv keystore. It is needed so that the runtime junction creation will not fail.



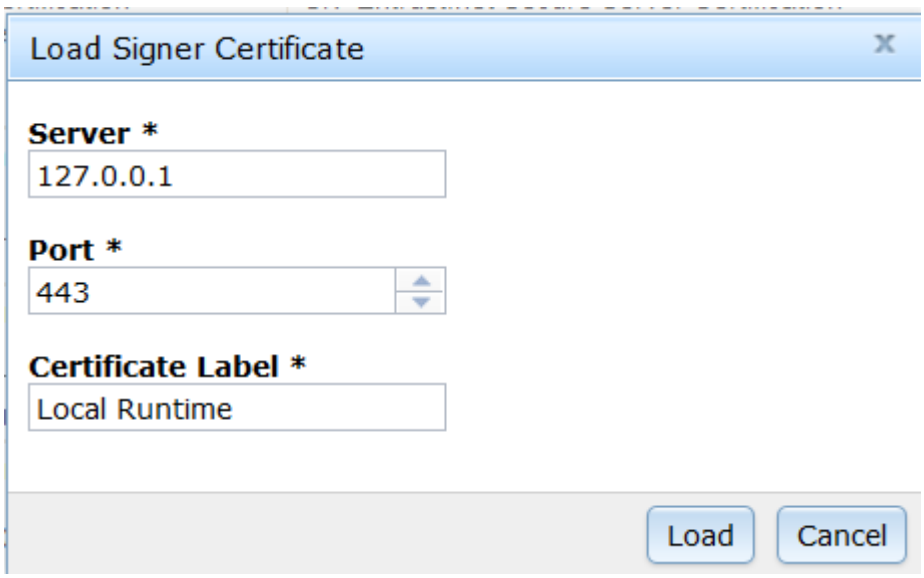
Navigate to **Manage System Settings > Secure Settings > SSL Certificates** in the SP appliance console / LMI



Select the pdsrv certificate database. Click **Manage > Edit SSL Certificate Database**



Click on the **Signer Certificates** tab. Click **Manage > Load**



Load the certificate from 127.0.0.1 as shown above. Click **Close**.

Save and deploy the changes. Navigate to the **Secure Web Settings→Reverse Proxy** menu. Select and restart the reverse proxy instance and ensure that the changes are active after restarting.

26.2.2 Federation auto-configuration endpoint

The 'fed_config' endpoint is an API endpoint which you will access via cURL.

Issue a GET request to the federations endpoint to view the id of the ISAMRP federation

```
$ curl -k -u admin:Passw0rd -H 'Accept: application/json' -H 'Content-Type: application/json'
https://isam.mysp.ibm.com/iam/access/v8/federations
[{"protocol":"OIDC","role":"rp","templateName":"","configuration":{"attributeMapping":{"map":
[]},"identityMapping":{"activeDelegateId":"default-
map","properties":{"identityMappingRuleReference":"6","ruleType":"JAVASCRIPT"}}},"name":"ISAM
RP","id":"uuidd3854bc5-0152-1ee3-af6a-b227acdee231"}]
```

In this example, there is only one federation, the id of which is **uuidd3854bc5-0152-1ee3-af6a-b227acdee231**. Issue a POST request containing the following data, substituting the federation id with the one from your environment.

```
$ curl -k -v -u admin:Passw0rd -H 'Accept: application/json' -H 'Content-Type:
application/json' https://isam.mysp.ibm.com/wga/reverseproxy/default/fed_config -d '{
"runtime":{ "hostname":"localhost", "port":"443", "username":"easuser", "password":"Passw0rd"
},"federation_id":"uuidd3854bc5-0152-1ee3-af6a-b227acdee231"}'
*   Trying 192.168.42.201...
...
* Server auth using Basic with user 'admin'
> POST /wga/reverseproxy/default/fed_config HTTP/1.1
> Host: isam.mysp.ibm.com
> Authorization: Basic YWRtaW46UGFzc3cwcmQ=
> User-Agent: curl/7.43.0
> Accept: application/json
> Content-Type: application/json
> Content-Length: 157
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 204 No Content
...
* Connection #0 to host isam.mysp.ibm.com left intact
```

An HTTP response code of 204 indicates that the request completed successfully. The changes have been made to the policy server, the reverse proxy configuration file

Open the LMI, and save and deploy the changes. Then select and restart the reverse proxy instance and ensure that the changes are active after restarting.

27 Appendix F – Configuring OAuth2.0 Device Flow

This OAuth 2.0 authorization flow for browser less and input constrained devices is called device flow.

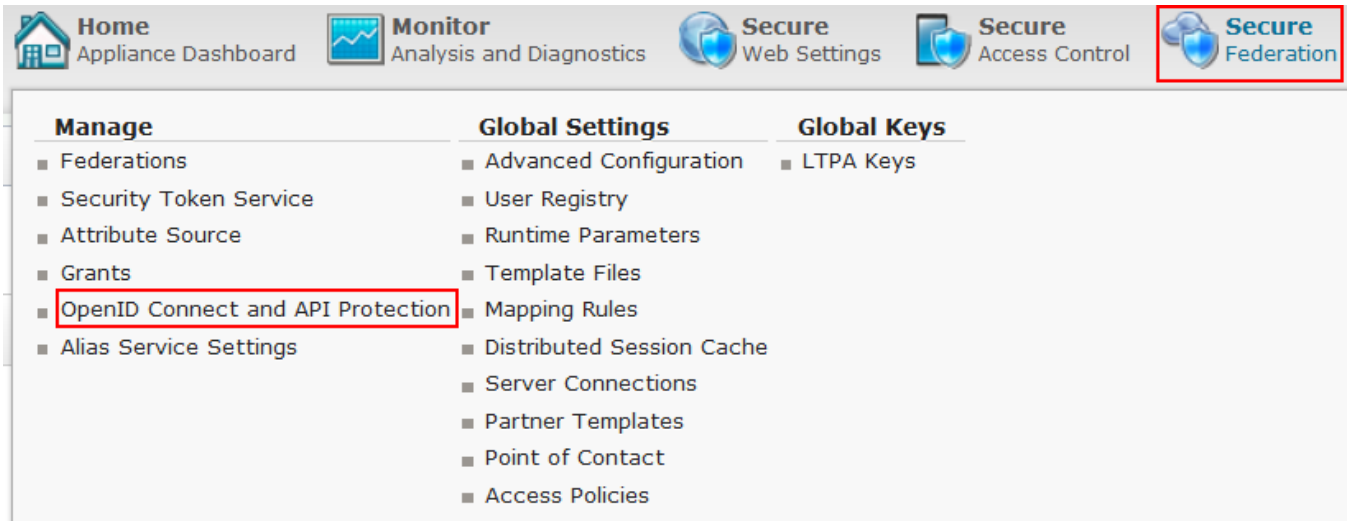
The OAuth device flow is intended for use where the OAuth client is unable to provide any input mechanism to the user and is only able to broadcast information.

Such applications would be smart devices which can display (for example, a smart device plugged into a TV) content, but not provide a user-agent. This means the flow of information is one way from the client to the resource owner.

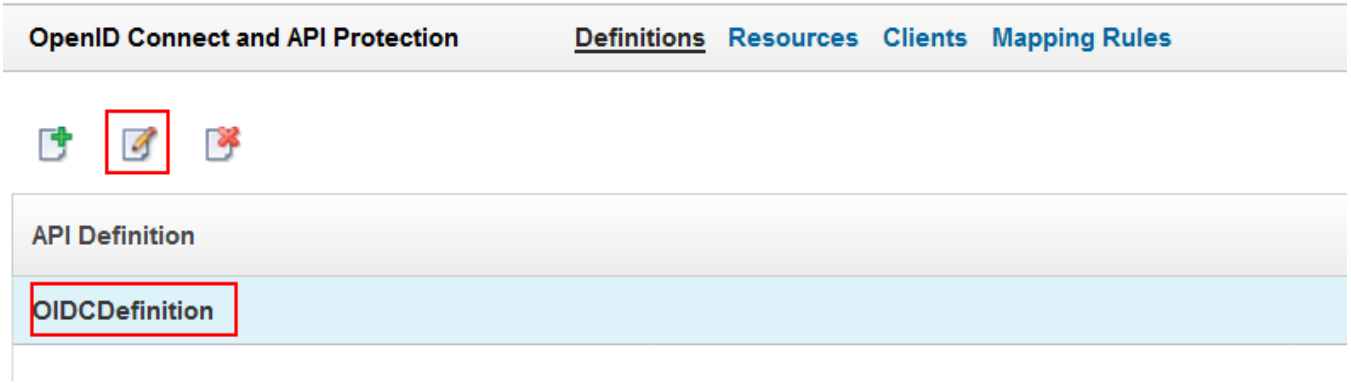
27.1 Pre-requisites and configuration

- The OpenID Connect Provider should have enable device grant as a supported grant type. We can check that.

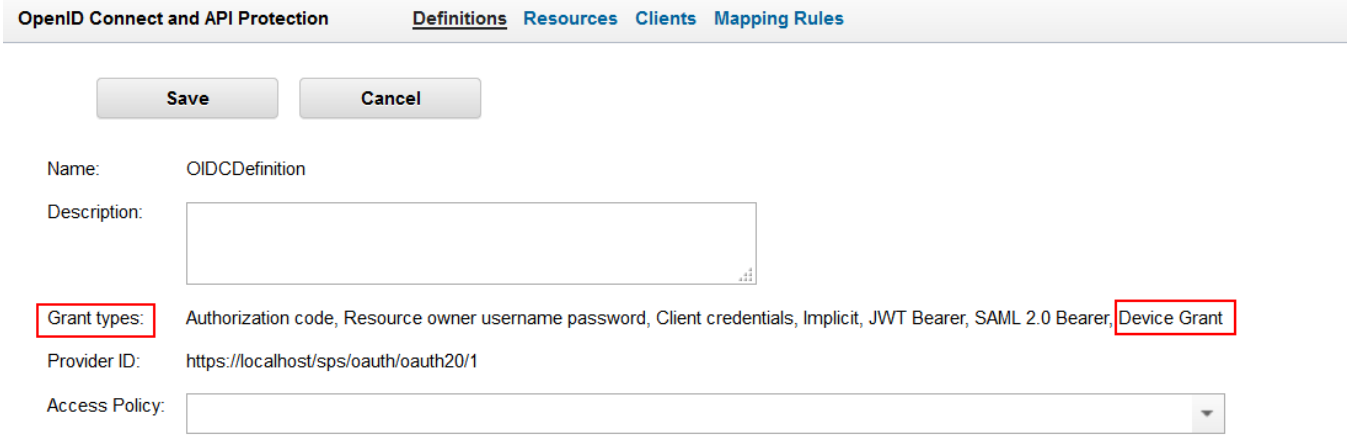
  Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**



Click on the API Definition **OIDCDefinition** and Edit.



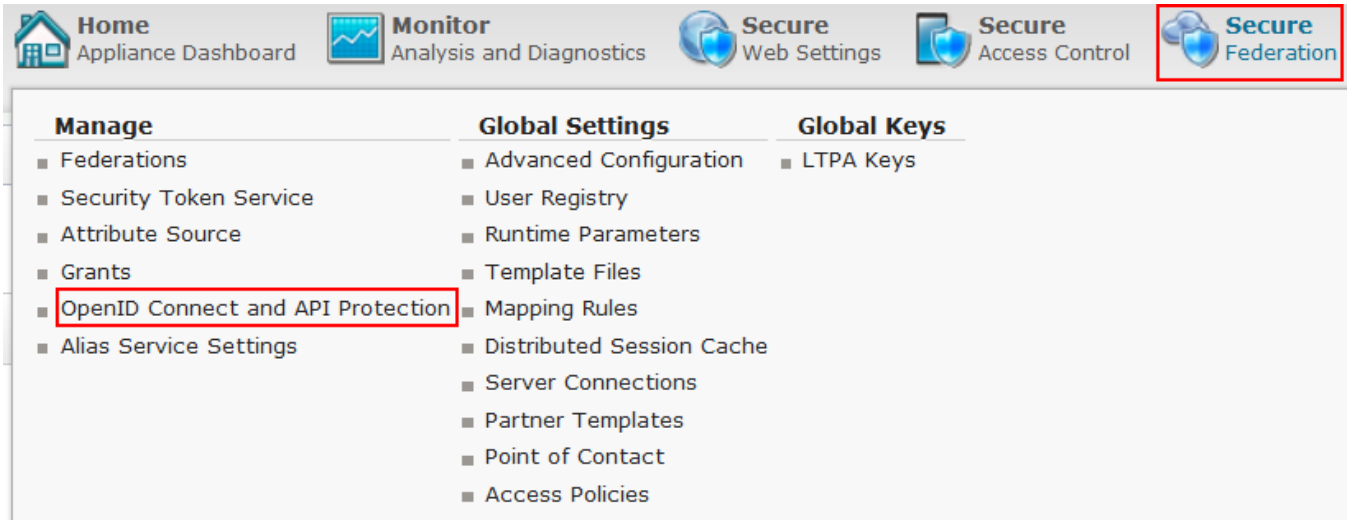
Check the Grant Types for the definition.



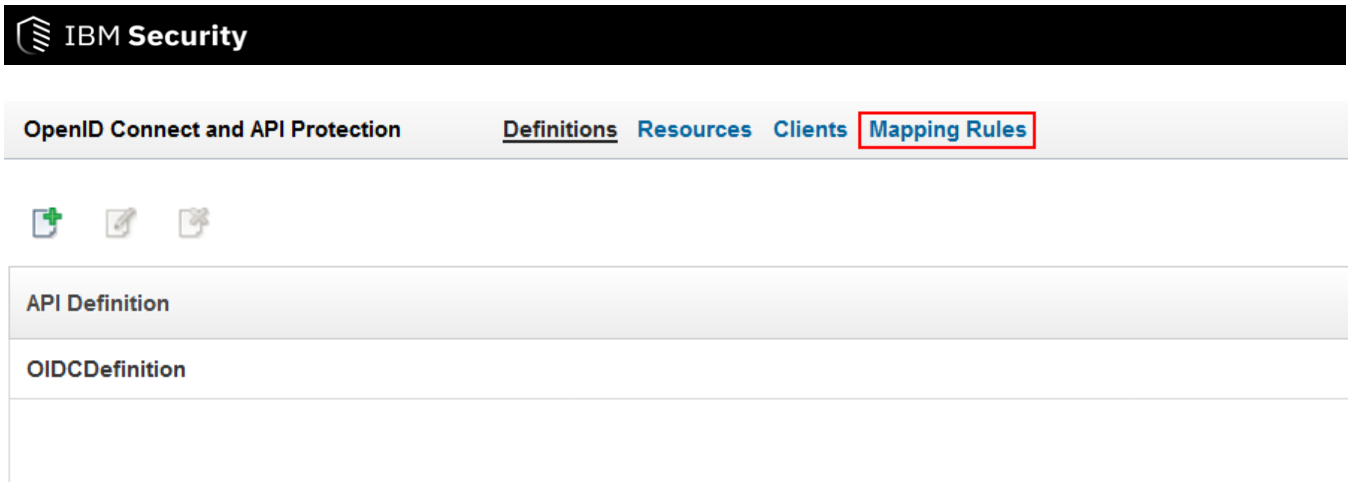
- Updating the Post Token Mapping Rule

In the LMI Administration console, navigate to **Secure Federation→Global Settings: Mapping Rules**.3

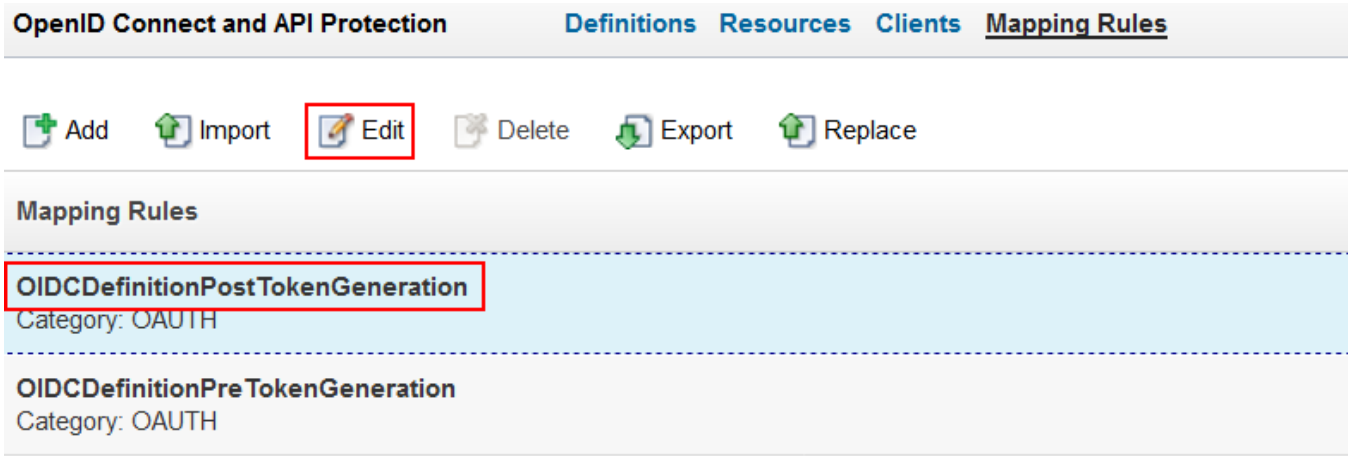
Using the administration console on the Identity Provider, navigate to **Secure Federation -> OpenID Connect and API protection**



Navigate to **Mapping Rules**



Select the **OIDCDefinitionPostTokenGeneration** mapping rule and Click on **Edit**.



Look for var webseal_portion = “https://localhost/isam” and replace it with var webseal_portion = "https://www.myidp.ibm.com/mga";

Click on Save and Deploy Pending Changes.

27.2 Testing OAuth2.0 Device Flow

SCRIPT-START:
Run this script: **DeviceFlow.py**

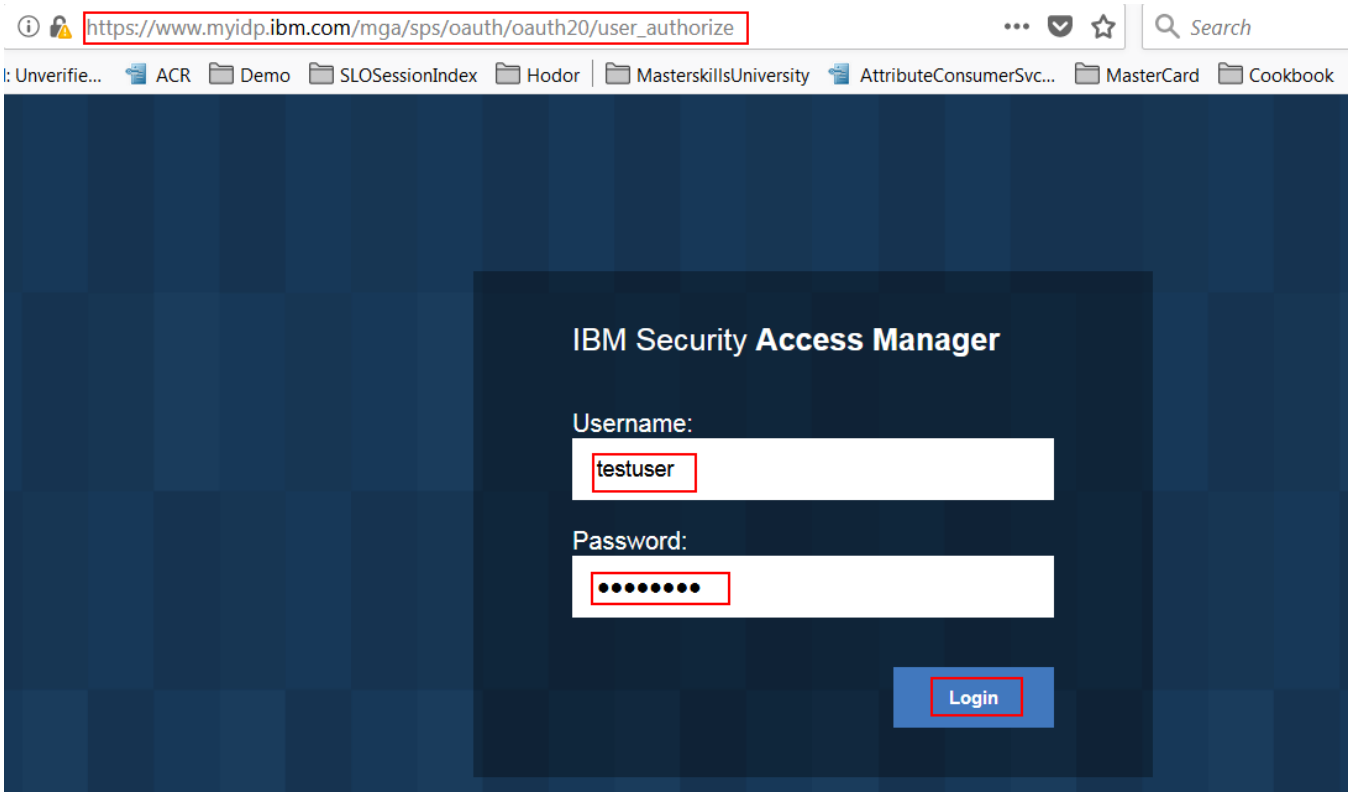
If you use this script, skip to the corresponding SCRIPT-END notice

Output

```
Initiating device flow.

Received:
        Device Code(A secret, not usually shown): 0XN30eFy4L21hesJ45NydhAErqwQRz
        User Code: ecsr-eraz
        Verification_uri: https://www.myidp.ibm.com/mga/sps/oauth/oauth20/user_authorize

 Visit the verification uri and input the user code




Polling for the token to be validated......
```

The device flow starts with a device requesting authorization. This results in two codes, one which is kept by the device(**device_code**) and the **user_code** which is displayed with a **verification_uri** to the end user.

Access the **verification_uri** and login with username password, testuser/Passw0rd.



Enter the user_code that device script output and Click on Submit

OAuth 2.0 - Device flow authorization pending
The following error was encountered while processing your OAuth request:

Error Code: **invalid_request**

Error Description: **+FBTOAU202E The required parameter: [user_code] was not found in the request.**

User Code: ecsr-eraz

Submit

If the user_code is correct, we see a authorization success message.

OAuth 2.0 - Device flow authorization successful
The Device **ISAM Client** showing **ecsr-eraz** has been authorized.

DeviceFlow script output once the user code is entered

```
Initiating device flow.

Received:
        Device Code(A secret, not usually shown): 0XN30eFy4L21hesJ45NydhAErqwQRz
        User Code: ecsr-eraz
        Verification_uri: https://www.myidp.ibm.com/mga/sps/oauth/oauth20/user_authorize

 Visit the verification uri and input the user code




Polling for the token to be validated......

Flow successful

        Access Token: 8W6iaz3LZuESPKOLPv9w

        Refresh Token: jtq5n8tB1oYikf6jRFuFm40R2wuo7kdZUHiHjC33

        Scope: scope1 scope2
```

27.3 Testing OAuth2.0 Device Flow using curl

The client makes a request to device authorize and receives a device_code, a user_code and a verification_uri.

```
curl --request POST \
--url
'https://www.myidp.ibm.com/mga/sps/oauth/oauth20/device_authorize?scope=scope1%20scope2&clien
t_id=clientID'\
--header 'accept: application/json'
```

Response

```
{
  "user_code": "8meq-bjb2",
  "device_code": "Ak1EHlyTjfxuCYQbp010BUw5qhSijv",
  "scope": "scope1scope2",
  "interval": 5,
  "verification_uri_complete": "https:
\/\/www.myidp.ibm.com\/mga\/sps\/oauth\/oauth20\/user_authorize?user_code=8meq-bjb2",
  "verification_uri": "https:
\/\/www.myidp.ibm.com\/mga\/sps\/oauth\/oauth20\/user_authorize",
  "expires_in": 299
}
```

The client begins polling the token endpoint with the device_code, it will receive errors of 'authorization_pending' or 'slow_down' while it waits for a user to verify the user code

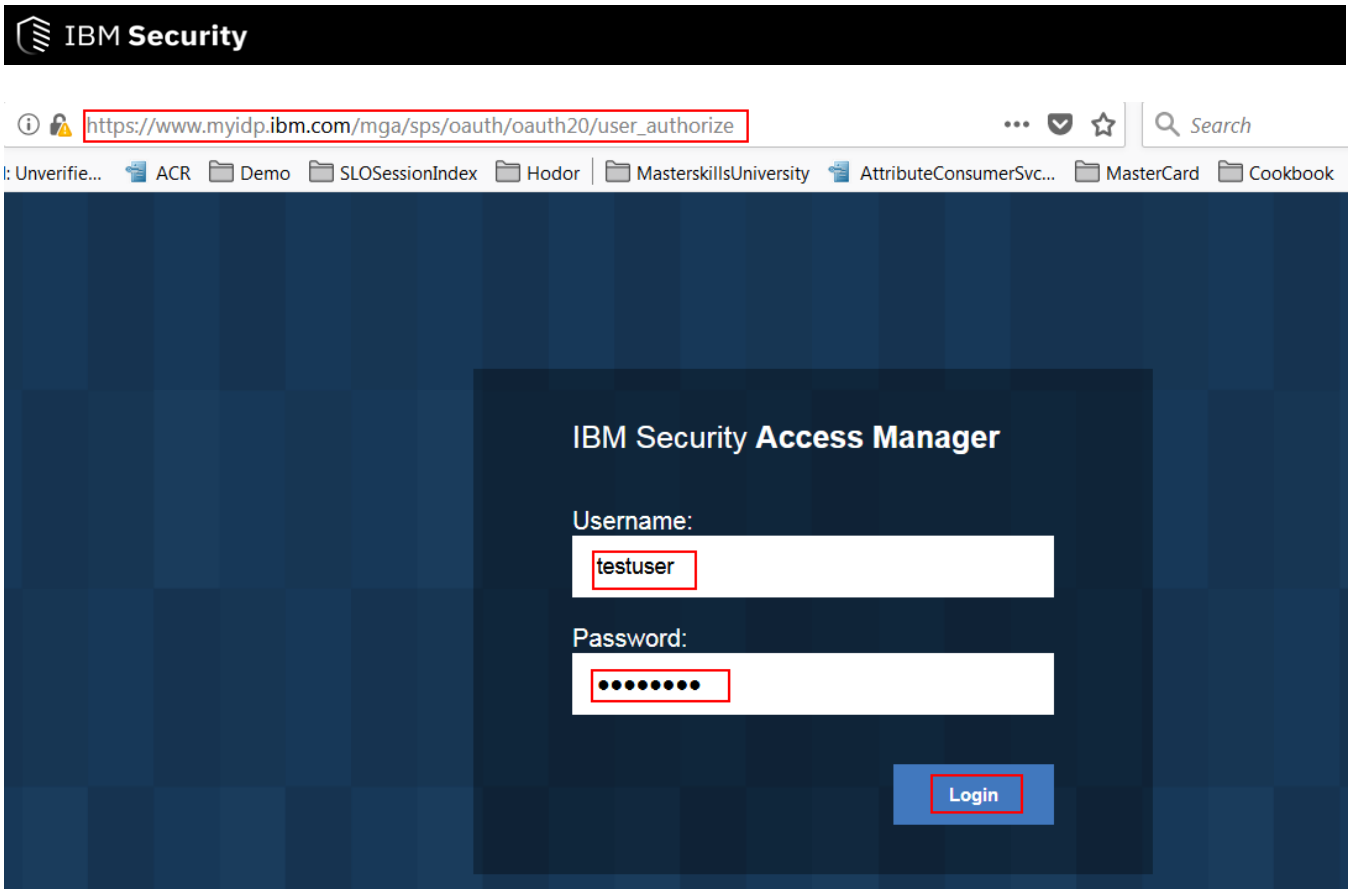
Polling the token endpoint

```
curl --request POST \
--url 'https://www.myidp.ibm.com/mga/sps/oauth/oauth20/token'\
--header 'accept: application/json'\
--header 'content-type: application/x-www-form-urlencoded' \
--data
'client_id=clientID&client_secret=clientSecret&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant
-type%3Adevice_code&device_code=qwMYmmCd1RNrAAF8j70sKOAHqjPpi8'
```

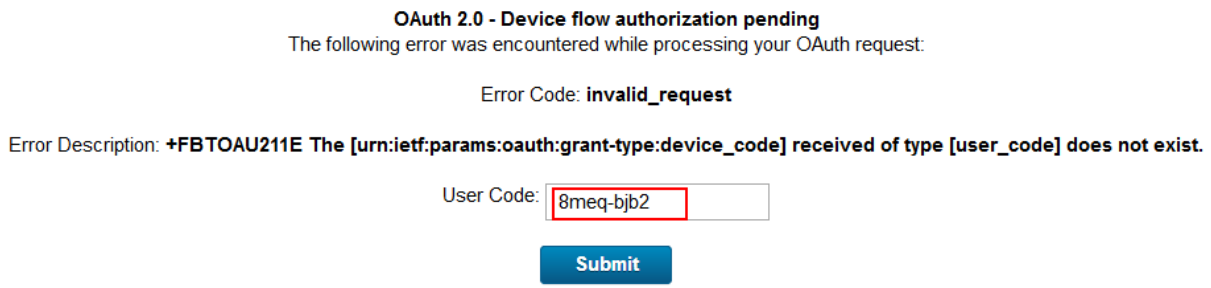
Response

```
{
  "error_description": "FBTOAU256E Pending. The user code is not yet verified.",
  "error": "slow_down"
}
```

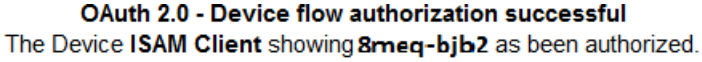
The user visits the verification uri presenting the user_code. The user will then be prompted to authenticate.



After the user has authenticated and entered the user_code.



If the code is correct a success page is shown



Retrieving the access token

```
curl --request POST \
--url 'https://www.myidp.ibm.com/mga/sps/oauth/oauth20/token'\
--header 'accept: application/json'\
--header 'content-type: application/x-www-form-urlencoded' \
--data
'client_id=clientID&client_secret=clientSecret&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant
-type%3Adevice_code&device_code=qwMYmmCd1RNrAAF8j70sKOAHqjPpi8'
```

Response

```
{
  "access_token": "anguBpUYmpOQQ21EXcvj",
  "refresh_token": "F2w6H10BdLcbpBeiFzmteOP2PjnLWaoreQFhulWH”,
  "scope": "scope1 scope2",
  "token_type": "bearer",
  "expires_in": 3599
}
```

28 Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

International Business Machines Corporation
New Orchard Road Armonk, NY 10504
Produced in the United States of America 01-2019