# Deploying Keystone with Federated Identity Management middleware

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 0.1 | 6th November 2012 | Kristy Siu | Initial Document Creation |
| 0.2 | 8th November 2012 | Kristy Siu | Added Config.xml configuration instructions |
| 0.3 | 12 November 2012 | David/Kristy | Updated XML |

## Introduction

This guide does not seek to provide the user with instructions for installing the Keystone service, for information on how to do this users should refer to the documentation found on the Openstack keystone website:

<http://docs.openstack.org/developer/keystone/installing.html>

The purpose of this document is to describe the configuration steps required to enable federated identity management (FIM) on your keystone server.

## Getting the code

The code is held in a Git repository supplied by Github and can be downloaded using the following command:

In order to use FIM in keystone, the service must be accessed with modified clients, which are available in the following repositories:

git clone git://github.com/kwss/keystone.git

[git://github.com/kwss/python-swiftclient.git](https://github.com/kwss/python-swiftclient.git)

[git://github.com/kwss/python-novaclient.git](https://github.com/kwss/python-novaclient.git)

[git://github.com/kwss/python-glanceclient.git](https://github.com/kwss/python-glanceclient.git)

## If deploying into an existing Keystone installation

Copy the federatedAccess folder from the downloaded code into the middleware folder of your keystone installation.

## If deploying a fresh a installation of Keystone

Following the Openstack documentation to install the Keystone server from the downloaded source. \*

\*Please note that it is not required to copy any files for FIM unless working with an existing deployment.

## Configuring Keystone to use FIM

1. Modify the keystone.conf file:
   1. Add the filter section for the federated middleware (Figure 1)
   2. Add the federated middleware filter to the pipeline – this allows the middleware to intercept requests coming into keystone. (Figure 2)
2. Configure your directory and request issuing services. (Section: Configuring services)
3. Configure the attribute / tenant mappings in the config.xml file (Section: Configuring Attributes)
4. Restart your keystone server.

[filter:auth\_federated]**1**

paste.filter\_factory = keystone.middleware.federatedAccess.auth\_federated:filter\_factory

validationClass = keystone.middleware.federatedAccess.permis\_validator.PermisValidator**2**

discoverClass = keystone.middleware.federatedAccess.directory\_service.ExampleDS**3**

requestIssuingClass = keystone.middleware.federatedAccess.request\_issuing\_service.ExampleRIS**4**

requestSigningKey = cert/privkey.pem**5**

SPName = KeystoneClientSecure**6**

attributeFile = config.xml**7**

## 1 Section Header

Figure - Example filter section

## 2 Credential Validation Class\*

## 3 Directory Service Class\*

## 4 Request Issuing Service Class\*

## 5 Key used to sign requests.

## 6 The name your keystone server will use to identify itself to the Identity Providers.

## 7 The file containing the attributes used to determine which tenants should be created and which users can access them.

\* Unless you are overriding these classes with custom classes the values shown should not need to be modified.

Figure : Example Modified Pipeline

[pipeline:public\_api]

pipeline = auth\_federated token\_auth admin\_token\_auth xml\_body json\_body debug ec2\_extension public\_service

## Configuring services

### Directory Service Configuration

The directory service configuration file should contain a list of available IdPs and their metadata location. (Figure 3)

<ExampleDiscoveryServiceConfiguration>

<IdentityProvider>

<Name>ProxyIdP</Name>

<EntityID>https://l4l-idp-01.kent.ac.uk</EntityID>

<MetadataLocation>metadata.xml</MetadataLocation>

</IdentityProvider>

<IdentityProvider>

<Name>OtherIdP</Name>

<EntityID>https://l4l-idp-01.kent.ac.uk</EntityID>

<MetadataLocation>metadata.xml</MetadataLocation>

</IdentityProvider>

</ExampleDiscoveryServiceConfiguration>

### Metadata

Figure : Example directory service configuration

The Example Service uses SAML 2.0 XML Metadata, the metadata provided to IdPs should include the endpoint: <https://localhost:8080>. More information about this can be found here: <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>

### Overriding the Directory Service Class

The following functions should be implemented in the overriding class:

* discovery

This function takes the following parameters:

1. Realm – the name of the realm to issue a request for.

And returns a string endpoint for the requested IdP

* getIdPList

This function takes no parameters and returns a list of “Name”:idpname mappings.

e.g. [{‘Name”:”MyIdP”}, {“Name”, “MyOtherIdp}]

### Configuring the Request Issuing Service

The only configuration required for the example request issuing service (ExampleRIS) is to set the signing key and service provider name in the keystone.conf file. The values specified should match the information in the metadata provided to the Identity Providers configured into your directory service.

### Overriding the Request Issuing Service

Only one function needs to be overridden in a custom request issuing service:

* getIdpRequest

This function takes the following parameters:

1. key – the key used to sign requests
2. issuer – name used to identify your keystone to the identity providers

And returns the signed request to be sent to the identity provider.

## Configuring Attributes

Access to tenants is determined by adding entries to the config.xml file and the persistent identifier (pid) for each identity provider (IdP) should be defined here. A default pid can be defined which will be used for any IdP which has not been defined specifically. An example Config.xml is given in Figure 4. The schema for the Config.xml file can been seen in Appendix A.

<Config>

<IdPpidMapping Name="ProxyIdP" PID="eduPersonTargettedID" />**1**

<SetOfTenants> **2**

<Tenant DisplayName="Kent Personal Account">**3**

<!-- Kent accounts -->

<Attribute Name="idp” Value=”kent" DisplayName="Account" />**4**

<Attribute Name="uid" DisplayName="ID" />

<Attribute Name="permisRole" DisplayName="Role" />

<MappedAttribute>**5**

<Attribute Name="permisRole" Value="staff"/>**6**

<RoleGranted>admin</RoleGranted>**7**

</MappedAttribute>

</Tenant>

</SetOfTenants>

</Config>

**1** The Persistent Identifier to Identity Provider Mapping

Figure : An example config.xml

**2** The Set of Attributes element – contains one or more Set elements

**3** The Set element – has an attribute to determine the name displayed to the user to represent the tenant available when the nested attribute are present

**4** An AttributeType element – has two attributes:

* Name – the name of the attribute and optionally an assignment to denote the required attribute value, if omitted then any value is valid.
* FN – the friendly, easy to understand name for this attribute

**5** The mapped attribute element – determines which keystone roles can be assigned to a user if the nested attributes are present

**6** UserAttribute – an attribute which must be present for the associated RoleGranted to be assigned to the user, has two attributes:

* Name – the type of the attribute required
* Value – the value of the attribute required

**7** RoleGranted – the keystone role which will be assigned if the associated UserAttribute(s) are present in the users credentials.

Each set of attributes represents one or more tenants which will be automatically created by the middleware. If values are specified for each of the attributes in the set, then only one tenant will be created by the middleware. If a set of attributes has one or more attributes with no values specified, then one tenant will be created for each unique combination of values that are presented by the IdP. A tenant will be personal i.e. only accessible to one user, if one of the required attributes uniquely identifies the user e.g. uid or email address. A tenant will be shared by multiple users if the set of required attributes can be possessed by multiple users e.g. role or organization.

## Appendix A

<?xml version="1.0" encoding="ISO-8859-1" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Simple Types -->

<xs:element name="RoleGranted" type="xs:string" />

<xs:element name="CertificateFile" type="xs:string"/>

<xs:element name="KeyFile" type="xs:string"/>

<xs:element name="pid" type="xs:string" />

<!-- Attributes -->

<xs:attribute name="DisplayName" type="xs:string" />

<xs:attribute name="Name" type="xs:string" />

<xs:attribute name=”Value” type="xs:string" />

<!-- Complex Types -->

<!-- A mapping between an Identity Provider and the Persistent Identifier it uses -->

<xs:element name="IdPpidMapping">

<xs:complexType>

<xs:sequence>

<xs:element ref="pid" />

</xs:sequence>

<xs:attribute ref="Name" />

</xs:complexType>

</xs:element>

<!-- Element representing one attribute with a Name and a display name-->

<xs:element name="Attribute">

<xs:complexType>

<xs:attribute ref="Name" />

<xs:attribute ref="DisplayName" use=optional />

<xs:attribute ref="Value" use=optional/>

</xs:complexType>

</xs:element>

<!-- Element representing an optional set of attributes that grant a role on Openstack services

Attribute – the set of attributes required to grant the role

RoleGranted - the Openstack role that is granted if all Attribute(s) are present -->

<xs:element name="AttributeMapping">

<xs:complexType>

<xs:sequence>

<xs:element ref="Attribute" maxOccurs="unbounded" />

<xs:element ref="RoleGranted" />

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Element representing a set of attributes required for access to a tenant/ account DisplayName - friendly tenant/account identifier -->

<xs:element name="Tenant">

<xs:complexType>

<xs:sequence>

<xs:element ref="Attribute" maxOccurs="unbounded" />

<xs:element ref="AttributeMapping" maxOccurs="unbounded" />

</xs:sequence>

<xs:attribute ref="DisplayName" use="required" />

</xs:complexType>

</xs:element>

<!-- Element representing a list of Tenants -->

<xs:element name="SetofTenants">

<xs:complexType>

<xs:sequence>

<xs:element ref="Tenant" maxOccurs="unbounded" />

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Element representing a configuration element for the Credential Validation Middleware

CertificateFile / KeyFile - the key/certificate pair used for signing SAML assertions -->

<xs:element name="Config">

<xs:complexType>

<xs:sequence>

<xs:element ref="CertificateFile" />

<xs:element ref="KeyFile"/>

<xs:element ref="IdPpidMapping" minOccurs="1" maxOccurs="unbounded" />

<xs:element ref="SetofTenants" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>