
Configuring SSO Applications for Mobius

Notices

Copyright

© 1996-2023 Rocket Software, Inc. or its affiliates. All Rights Reserved.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: www.rocketsoftware.com/about/legal. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

Examples

This information might contain examples of data and reports. 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.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc. or its affiliates, are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters

77 4th Avenue, Suite 100

Waltham, MA 02451-1468

USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country and Toll-free telephone number

- United States: 1-855-577-4323
- Australia: 1-800-823-405
- Belgium: 0800-266-65
- Canada: 1-855-577-4323
- China: 400-120-9242
- France: 08-05-08-05-62
- Germany: 0800-180-0882
- Italy: 800-878-295
- Japan: 0800-170-5464
- Netherlands: 0-800-022-2961
- New Zealand: 0800-003210
- South Africa: 0-800-980-818
- United Kingdom: 0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support. In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Table of contents

IIS.	5
Configuring Server Manager.	5
LDAP.	9
Setting up users in LDAP.	9
Configuring Apache reverse proxy.	9
OIDC.	12
Deploying OIDC authorization server.	12
SAML.	13
SSO using SAML authentication.	13
Authentication flow.	13
Keycloak.	15
Setting up Keycloak.	15
Creating SAML client.	15
Adding keys.	17
Creating user and role.	17
Generating metadata files.	18
Apache HTTPD.	18
Setting up Apache HTTPD Server.	18
MellonSPMetadataFile.	22
MellonIdPMetadataFile.	22
HTTPD docker-compose configuration.	32

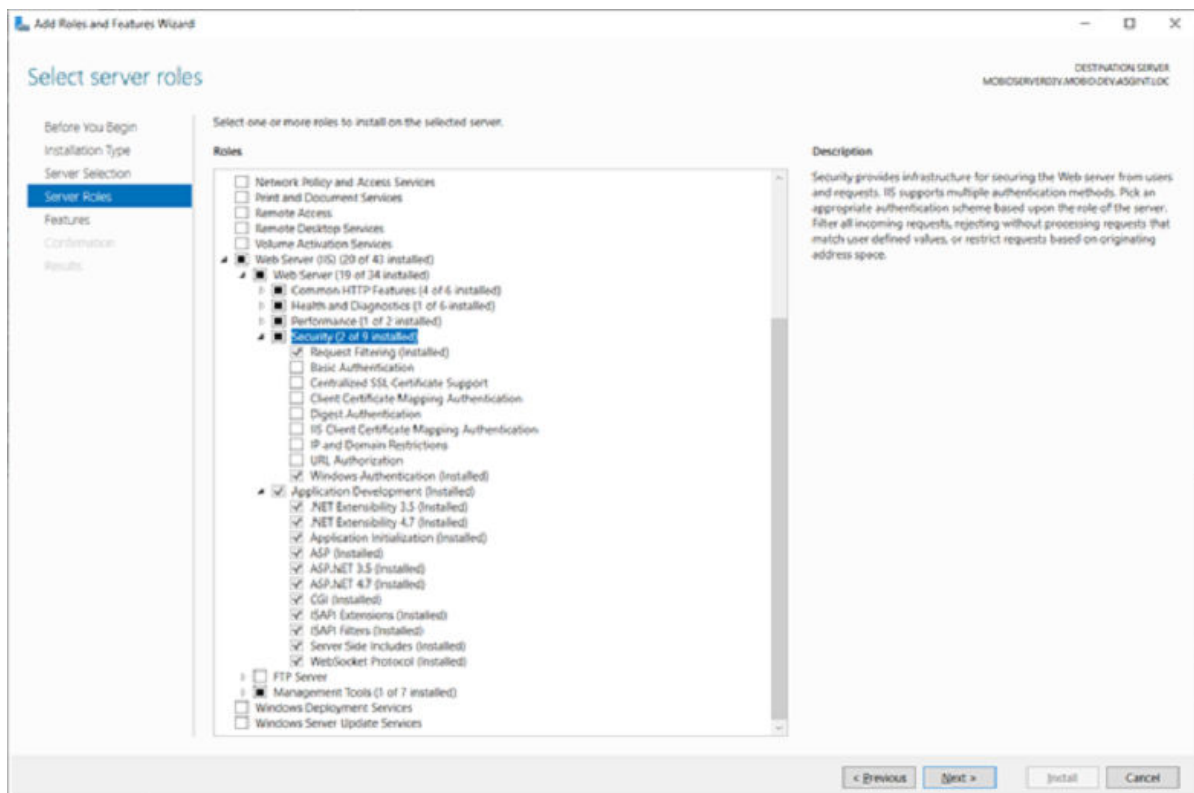
IIS

Configuring Server Manager


Configure the Server Manager to create and add a custom module.

1. In the Server Manager,
 - a. Click **Manage > Add Roles and Features**.
 - b. Select **Server Roles** for role-based installation or **Features** for feature-based installation.
 - c. Select the required server.

Note: By default, the local server is selected.



- d. Select **Windows Authentication** under **Security**.
 - e. Select all .NET and ASP options under **Application Development**.
 - f. Click **Next** to complete the installation.
2. Install Application Request Routing module. See <https://www.iis.net/downloads/microsoft/application-request-routing> for information.
This installs URL Rewrite module as well. Both the modules are required to use IIS as reverse proxy.
3. Enable proxy and set Windows authentication for your site in the IIS manager.
4. Set inbound rule to direct all URLs that start with Mobius, to your Mobius installation.



Edit Inbound Rule

Name:

Match URL

Requested URL:

Matches the Pattern

Using:

Regular Expressions

Pattern:

Test pattern...

☒ Ignore case

Conditions

Server Variables

Action

Action type:

Rewrite

Action Properties

Rewrite URL:

- Right-click on the site and select **Explore menu item** to open the Windows folder with all the files for the site.
- Add a new folder with the name `App_Code`.
- Provide the code for custom module in this directory.
A sample code for custom module that sets the user and group information in HTTP request headers after successful authentication is given below.

SampleCustomHeaderModule.cs

```

SampleCustomHeaderModule.cs
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Security.Principal;
using System.Web;

//Set custom request headers using this module
public class SampleCustomHeaderModule : IHttpModule
{
    public void Dispose()
    {
    }
}

```

```

public void Init(HttpApplication app)
{
    //Add event handler post authorization and authentication
    app.PostAuthorizeRequest += new EventHandler(AddCustomHeaders);
}

/*Event handler that will run post authorization. Set Mobius related custom headers
aders
    * X-PROXY-USER - Current logged in user
    * X-PROXY-GROUP - Authenticated user's groups
    *
    * Please note these header names can be anything.
    * Mobius doesn't require them to be of specific name.
    * However, same header names need to be configured in Mobius
    * ASG_BOOTSTRAP_VENDOR_TYPE=CUSTOM
    * ASG_BOOTSTRAP_VENDOR_ATTRIBUTES_USERNAME=X-PROXY-USER
    * ASG_BOOTSTRAP_VENDOR_ATTRIBUTES_GROUPNAMES=X-PROXY-GROUP
    * ASG_BOOTSTRAP_VENDOR_ATTRIBUTES_GROUPSEPARATOR=,
    */
public void AddCustomHeaders(object sender, EventArgs e)
{
    HttpApplication app = (HttpApplication)sender;
    app.Context.Request.Headers.Set("X-PROXY-USER", app.Context.Request.ServerVariables.Get("LOGON_USER"));
    app.Context.Request.Headers.Set("X-PROXY-GROUP", GetGroups(app.Context.Request));
}

//Get groups of logged in user
public string GetGroups(HttpRequest request)
{
    string groups = string.Empty;

    foreach (IdentityReference group in request.LogonUserIdentity.Groups)
    {
        groups = groups + (group.Translate(typeof (System.Security.Principal.NTAccount)).ToString()) + ",";
    }

    return groups;
}
}

```

Once this custom module is available, all requests directed to Mobius from this site includes user and groups in headers.

8. Add the custom module to the web.config file to register it under the site.

Sample web.config

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <modules>
      <add name="SampleCustomHeaderModule" type="SampleCustomHeaderModule" >
    </modules>
  </system.webServer>
</configuration>

```

```
    <system.webServer>  
  <configuration>
```

LDAP

Setting up users in LDAP

1. Set up a user to be a member of the mobiusadmin group.
2. Execute the `ldapsearch` command with OpenLDAP.

The result of the command is given below.

```
root@dldapserver:/# ldapsearch -x -D cn=admin,dc=example,dc=org -w admin -b uid=john,ou=people,dc=example,dc=org dn memberof
berof
# extended LDIF
#
# LDAPv3
# base <uid=john,ou=people,dc=example,dc=org> with scope subtree
# filter: (objectclass=*)
# requesting: dn memberof
#
# john, people, example.org
dn: uid=john,ou=people,dc=example,dc=org
memberof: cn=mygroup,ou=groups,dc=example,dc=org
memberof: cn=mobiusadmin,ou=groups,dc=example,dc=org
# search result
search: 2
result: 0 Success
# numResponses: 2
# numEntries: 1
```

Here, John is a user defined in LDAP and a member of two groups, mygroup and mobiusadmin. When John logs in, only mobiusadmin, the default administrator group in Mobius deployment, provides access rights.

Configuring Apache reverse proxy

1. Run Apache using the below sample docker-compose file to copy the `httpd.conf` file

```
version: '2'
services:
  apache:
    container_name: apache
    image: docker.io/bitnami/apache:2.4
    ports:
      - 7080:8080
      - 7443:8443
    #volumes:
    # - ./httpd.conf:/opt/bitnami/apache/conf/httpd.conf
```

This loads the local `httpd.conf` file.

2. Start Apache using the below command.

```
docker-compose up -d
```

3. Download a copy of the default httpd.conf file for editing using the below command.

```
docker cp apache:/opt/bitnami/apache/conf/httpd.conf .
```

4. Remove the comments for volume in the docker-compose file to allow Apache to accept the latest changes, using the below command.

```
volumes:
  - ./httpd.conf:/opt/bitnami/apache/conf/httpd.conf
```

5. Add the below changes to the default httpd.conf file.

```
#cat httpd.conf

#1. Enable modules
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LoadModule headers_module modules/mod_headers.so
LoadModule proxy_module modules/mod_proxy.so
...

<LocationMatch >


#2. reverse proxy configuration
ProxyPass http://<mobius http endpoint>/
ProxyPassReverse http://<mobius http endpoint>/
Order allow,deny
Allow from all
AuthName "Private"
AuthType Basic

#3. LDAP configuration
AuthBasicProvider ldap
AuthLDAPBindDN " cn=admin,dc=example,dc=org "
AuthLDAPBindPassword "admin"
AuthLDAPURL "ldap://ldap_server:389/DC=example,DC=org?uid,memberOf?sub?(objectClasses=*)"
AuthLDAPBindAuthoritative On
LDAPReferrals Off

#4. set information returned from LDAP
RequestHeader set PROXYUSER "%{AUTHENTICATE_UID}e"
RequestHeader set PROXYGROUP "%{AUTHENTICATE_MEMBEROF}e"

#clean up group listing PROXYGROUP is
#cn=mygroup,ou=groups,dc=example,dc=org; cn=mobiusadmin,ou=groups,dc=example,dc=org
RequestHeader edit* PROXYGROUP ",.*?;" ""
RequestHeader edit* PROXYGROUP ",.*" ""
RequestHeader edit* PROXYGROUP "cn=" ""
RequestHeader edit* PROXYGROUP " " ",,"
#PROXYGROUP is now mygroup,mobiusadmin
Require valid-user
</LocationMatch>
```

-
- a. Enable `mod_authnz_ldap`, `mod_headers`, `mod_proxy` modules if they are not already enabled.
 - b. Set the headers to the information put into the Apache environment variables by LDAP authentication.

 **Note:** In this example, the headers for groups come back with LDAP organizational unit and domain information. Add a few lines to strip out the unnecessary information.

6. Restart the Apache instance to obtain the `httpd.conf` file changes and log in as the configured user, by accessing the Apache proxy running at, `http://localhost:7080/mobius/admin`.

OIDC

Deploying OIDC authorization server

1. Deploy Keycloak OIDC provider using the instructions given in the Keycloak Installation Guide.
You can also deploy Keycloak using the Docker image and the `docker-compose.yaml` file.
2. Deploy Keycloak authorization server with internal Postgres database using the below docker-compose script.

```
version: '3'
#Create Volume
volumes:
  keycloak-data:

services:
  #Create database container
  oidc-keycloak-db:
    image: postgres:11-alpine
    container_name: oidc-keycloak-db
    ports:
      - "${DB_PORT:-2101:5432}"
    environment:
      POSTGRES_DB: keycloak
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    volumes:
      - "keycloak-data:/var/lib/postgresql/data"

  #OIDC provider
  oidc-keycloak:
    image: jboss/keycloak
    container_name: oidc-keycloak
    volumes:
      - ./tls.crt:/etc/x509/https/tls.crt
      - ./tls.key:/etc/x509/https/tls.key
    ports:
      - "${KEYCLOAK_PORT:-80}:8080"
      - "${KEYCLOAK_SSLPORT:-443}:8443"
    environment:
      - KEYCLOAK_USER=admin
      - KEYCLOAK_PASSWORD=mobius123
      - DB_VENDOR=postgres
      - DB_ADDR=oidc-keycloak-db
      - DB_USER=postgres
      - DB_PASSWORD=postgres
    depends_on:
      - oidc-keycloak-db
```

3. Generate a server certificate, `tls.crt` and a server key file, `tls.key`.
4. Add the files to the same location as the `docker-compose.yaml` file to enable SSL.



Note: You can also use `openssl` to generate self-signed certificates and key for a given host or request CA signed certificates.

SAML

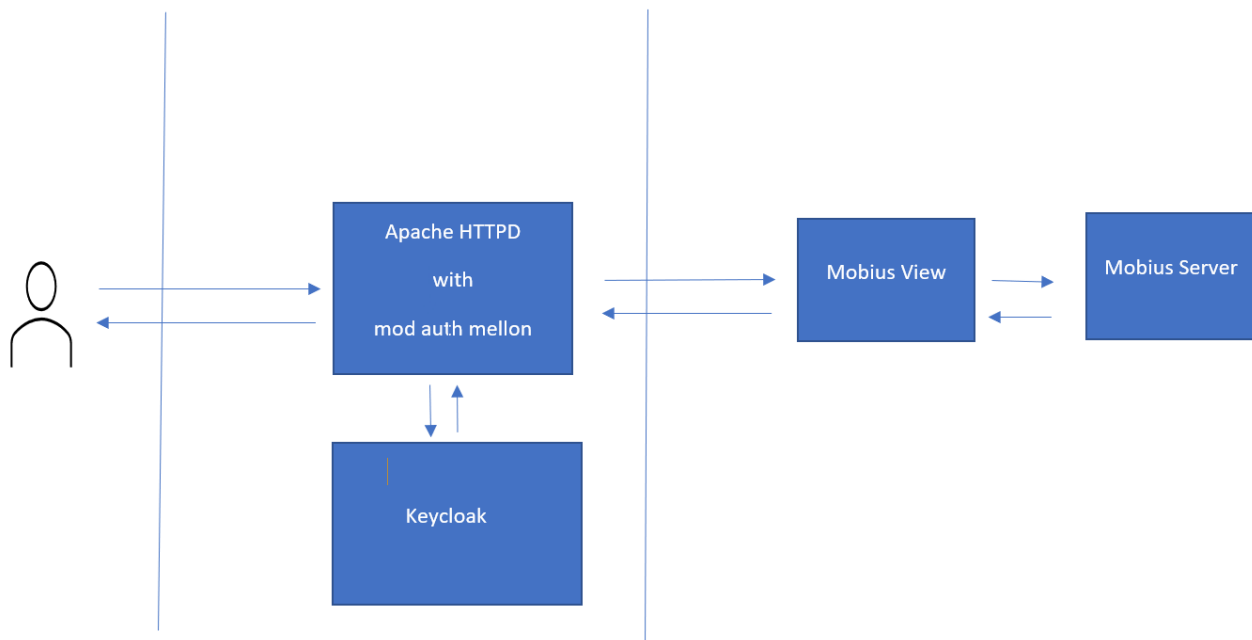
SSO using SAML authentication

Configure SAML authentication for Mobius using Apache HTTPD server and Keycloak.

Security Assertion Markup Language (SAML) is an open standard that allows Identity Providers (IdP) to pass authorization credentials to Service Providers (SP) through XML.

In this set up,

- SP - Apache HTTPD server (used as proxy for Mobius View)
- Idp - Keycloak server
- Authorization provider - Keycloak

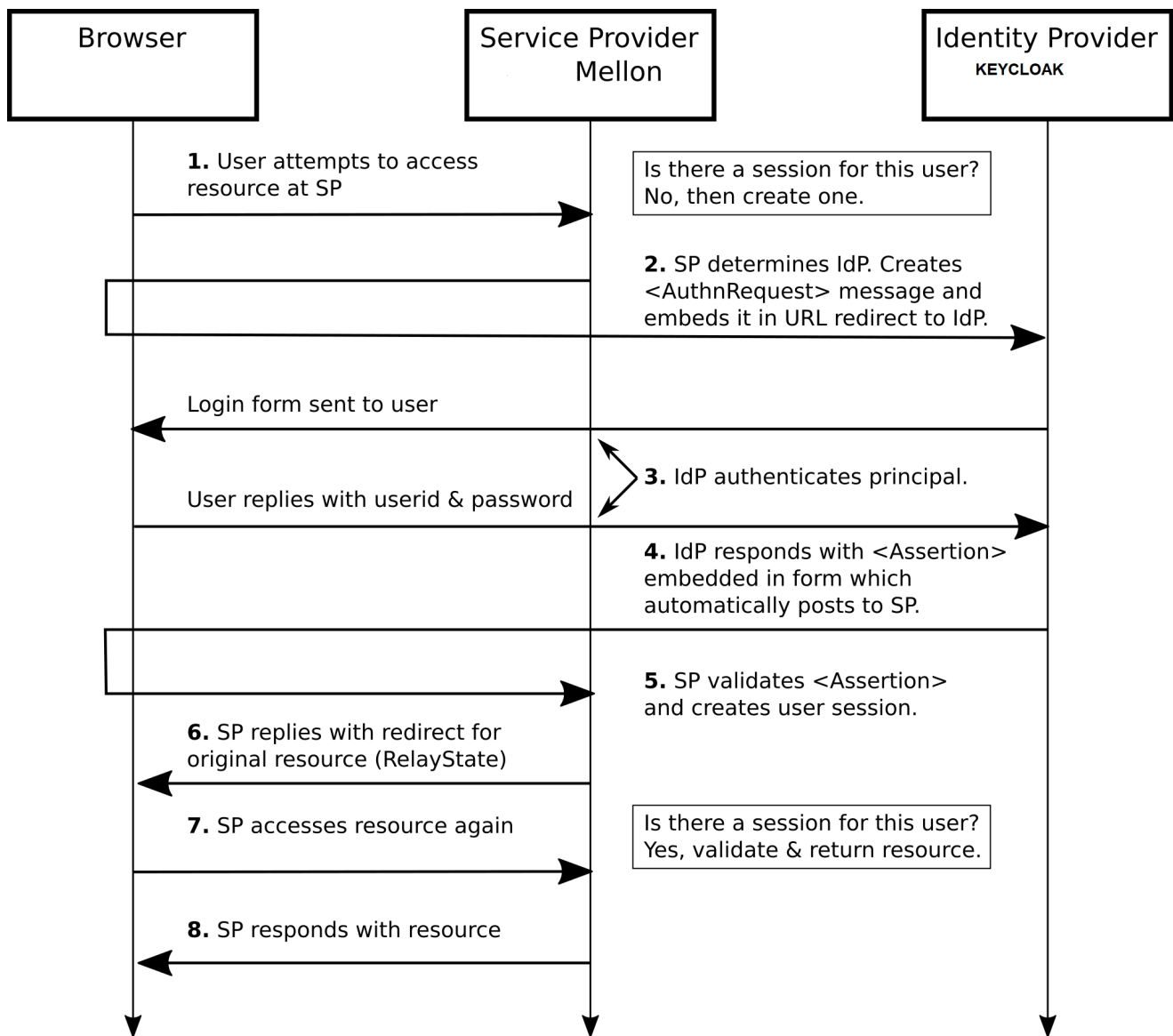


The `mod_auth_mellon` module provides SAML processing capabilities to Apache HTTPD. For more information about `mod_auth_mellon` module, see [mod_auth_mellon user guide](#).

The HTTPD server is configured as a proxy for Mobius View and interacts with Keycloak to provide SAML authentication. The SP (Apache HTTPD with mellon) and IdP agree to a contract through the metadata files (`sp-metadata.xml` and `idp-metadata.xml`) to identify the client, endpoints to be used for SAML authentication, and public keys to verify the signature. The SP and IdP communicate with each other to authenticate, using the contract.

Authentication flow

The authentication process follows the below approach.



1. When you access Mobius UI through the browser, Apache HTTPD proxy intercepts the request and mellon module determines if there is a session. If there is no session, the request is redirected to IdP (Keycloak) to authenticate and create a new session.
2. The mellon module checks the metadata files to identify Keycloak as the IdP, creates an authentication request method, and sends it to the URL identified in the IdP metadata file.
3. The IdP determines whether the session is available and displays the login page if the session is unavailable.
4. You are now required to provide the username and password for IdP to validate.
5. IdP sends SAML assertion to the URL identified in SP metadata or client configuration after successful login. The SAML assertion contains the username, user groups and roles.
6. The SP (mellon) then validates the signature of the assertion using IdP's public key and creates a user session. The SP also stores the username, user groups and roles in session as Apache environment variables. This information is also copied to custom HTTP headers in the request with Apache rewrite header module.
7. The SP redirects the request to the requested Mobius page.
8. A session is established. Further requests are allowed without login redirection.

Keycloak

Setting up Keycloak

Use Keycloak to generate metadata files and configure SP (HTTPD with mellon module) with metadata.

To set up Keycloak you must,

- Create a SAML client
- Add keys for SP
- Create users and roles
- Generate metadata files for SP

Creating SAML client

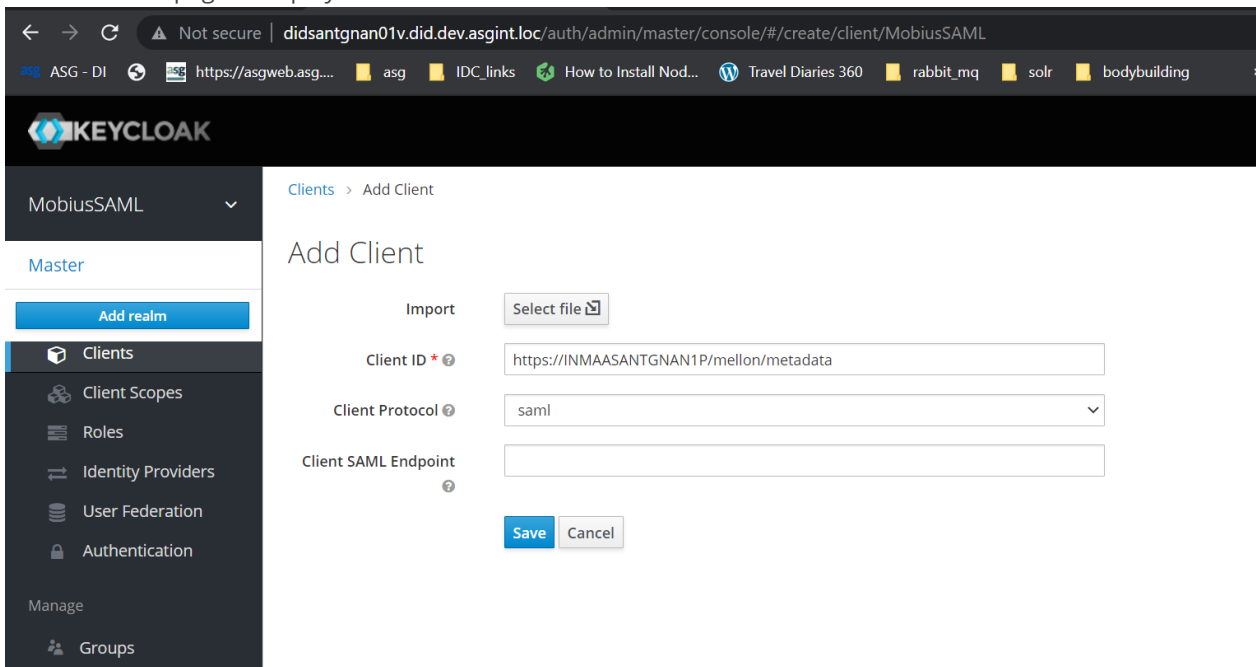
Create SAML client and configure the client settings.

1. Access the below Keycloak admin URL through a web browser and log in using admin credentials.

```
http://<keycloak host>/auth/admin
```

2. Select the realm to create a SAML client.
3. Click **Clients > Create**.

The Add Client page is displayed.






4. Enter the client identifier in the **Client ID** field.


The client ID is a URL which is the expected issuer value in SAML requests sent by the application and is of the below format.

```
https://<proxy host>/mellon/metadata
```

5. Select SAML from the **Client Protocol** drop-down.
6. Click **Save**.
The client is created and the client settings tab is displayed.

7. Enter the following details in the settings tab.

Field	Description
Client ID	Value that matches the issuer value sent with AuthNRequests by SP.
Name	Client name to be displayed in a Keycloak UI screen.
Enabled	Allows the client to request authentication.
Consent Required	Allows you to grant access to the application. It also displays the metadata that the client is interested in.
Include AuthnStatementSAML	<p>By default, this is enabled, allowing AuthStatement element to be included in login responses.</p> <div> Note: If this setting disabled, it prevents the client from determining the maximum session length which could result in a never-expiring client session.</div>
Force Artifact Binding	<p>Allows you to use artifact binding with Idp-initiated login, if enabled. To use artifact messages during logout, you must configure Logout Service Redirect Binding URL.</p> <div> Note: It forces Keycloak to send artifact messages instead of SAML messages through POST and Redirect, even when the client did not ask for binding during login.</div>
Sign Documents	Allows Keycloak to sign the document using the realm's private key. Ensure this is enabled.
Client Signature Required	Confirms that documents coming from a client are signed. Keycloak validates this signature using the client public key or cert set up in the Keys tab.
Force POST Binding	Allows Keycloak to always respond using SAML POST binding even if the original request was the Redirect binding. By default, Keycloak responds using the initial SAML binding of the original request.
Front Channel Logout	Allows the application to require a browser redirect to perform a logout. For example, the application may require a cookie to be reset which could only be done through a redirect. Ensure this is enabled.
Root URL	Prepends any configured Keycloak relative URLs.
Valid Redirect URIs	<p>Set of valid redirect URIs the client can request after login or logout.</p> <div> Note: Wildcards (*) are only allowed at the end of a URI, that is, http://host.com/*. It is recommended to add all valid URIs under the domain using the wildcard, after the base domain URI. Add mellon paosResponse endpoint.</div>
Base URL	Allows Keycloak to link to the client. This is the base URL for the domain.

Field	Description
Master SAML Processing URL	Directs the response to the SP and is used for all SAML requests. It is used as the Assertion Consumer Service URL and the Single Logout Service URL.
Assertion Consumer Service POST Binding URL	POST binding URL for the Assertion Consumer Service. This is the SP endpoint to which SAML assertion must be sent and is the SP handler URL for assertions.
Logout Service POST Binding URL	POST binding URL for the Logout service. This is the SP endpoint to which SAML assertion must be sent and is the SP handler URL for logout. <div>  Note: The URL contains the ReturnTo parameter, which identifies the page to which the browser must be redirected after logout. </div>

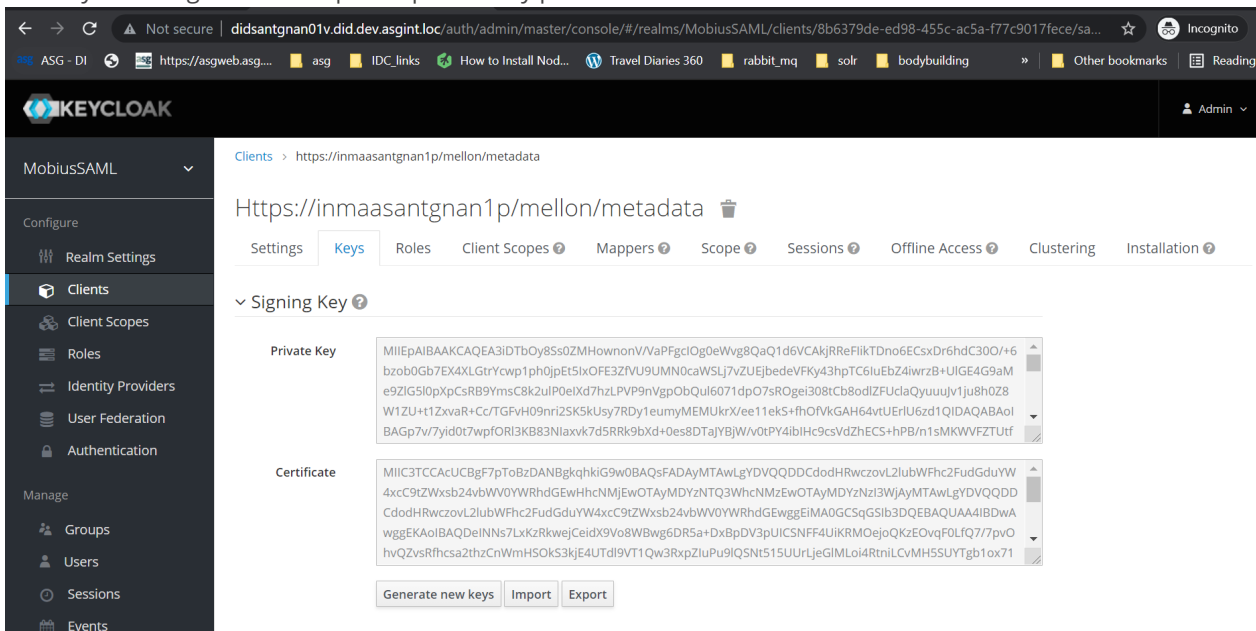
Adding keys

Add public and private keys used by SP to allow Keycloak to verify the signature on SAML request sent by SP and generate all required metadata files.

1. Add the public key used by SP [httpd with mod_auth_mellon].
2. Add the SP's private key to generate the metadata files (idp-metadata.xml and sp-metadata.xml).

or

Use Keycloak to generate the public/private key pair.



The screenshot shows the Keycloak Admin Console interface. The left sidebar contains navigation options like 'Realm Settings', 'Clients', 'Roles', etc. The main content area is titled 'Clients' and shows the 'Keys' tab for a specific client. Under the 'Signing Key' section, there are two fields: 'Private Key' and 'Certificate', each containing a long string of base64-encoded text. Below these fields are three buttons: 'Generate new keys', 'Import', and 'Export'.

 **Note:** The public/private key pair is used only for signing and not for matching the identity of host. A valid public/private key pair can also be imported.

Creating user and role

Create users and assign roles to provide access privileges.

1. Click **Roles > Add Role**.
The Add Role page is displayed.
2. Enter an admin name in the **Role Name** field and click **Save**.
3. Repeat the steps to create a Mobius user role.
4. Click **Users > Add user**.
5. Enter an admin user name in the **Username** field and click **Save**.
6. Click **Credentials**.
7. Select the admin role from the available realm role.
8. Repeat the steps to create a guest user.



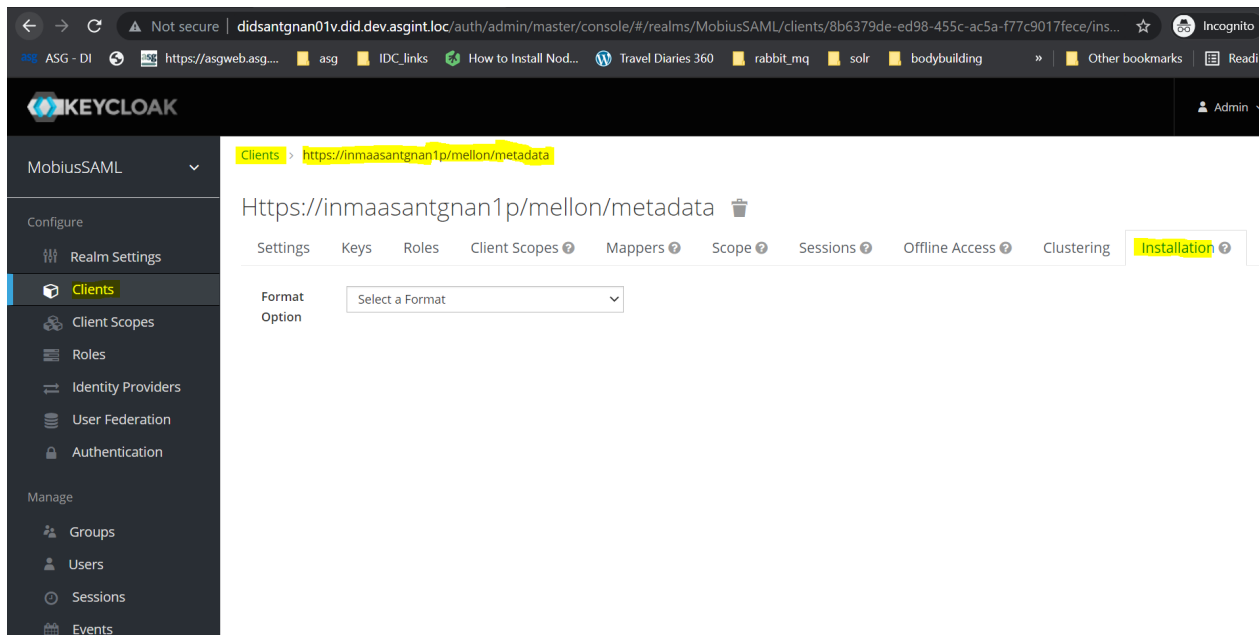
Note:

- The admin user has the privilege to access the Mobius Administrator and View page.
- The guest user has the privilege to access Mobius View page only.

Generating metadata files

After the client and key information is added, use keycloak to generate the metadata files (sp-metadata.xml and idp-metadata.xml).

1. Navigate to **Clients > Installation tab**.
2. Select Mod Auth Mellon files from the **Format Option** drop-down.
3. Click **Download**.



Apache HTTPD

Setting up Apache HTTPD Server

To set up Apache HTTPD Server with mod_auth_mellon module using a docker image.

1. Obtain the base image for HTTPD server.
2. Add mod_auth_mellon module to the base image.

The following base image is an official image for HTTPD in Docker hub. The docker file configuration for it is shown below:

```
FROM httpd:2.4.48

RUN apt-get update && apt-get install -y /
libapache2-mod-auth-mellon
```

3. Provide the configuration required for Apache to enable required modules to act as proxy:
 - Rewrite the headers.
 - Enable mod_auth_mellon module.
4. Enable the SSL module. The SSL module is enabled using the httpd.conf file located at /usr/local/apache2/conf/httpd.conf directory.
 - Add the following lines in httpd.conf file for updating the locations appropriately.

```
#update mod_auth_mellon.so location as appropriate
LoadModule auth_mellon_module /usr/lib/apache2/modules/mod_auth_mellon.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule headers_module modules/mod_headers.so
LoadModule rewrite_module modules/mod_rewrite.so
```

5. Add configuration required for HTTPD to act as reverse proxy for Mobius View and Mobius Administrator pages.
6. Replace the value of mobiusView:8080 with actual host and port name if Docker is not used for deploying Mobius View.

The following code is provided based on Docker compose file referenced below and mobiusView is name of the Docker container for Mobius View.

```
ProxyRequests off

<Proxy *>
    Order deny,allow
    Deny from all
    Allow from all
</Proxy>
ProxyTimeout 300

ProxyPass /mobius/logout !
ProxyPass /mobius http://mobiusView:8080/mobius
ProxyPassReverse /mobius http://mobiusView:8080/mobius
```

- a. The ProxyPass is not enabled for logout. Use a rewrite directive to redirect the logout request to SP logout URL as there is no logout endpoint on Mobius.
- b. Ensure to replace the hostname with host name value in RedirectTo parameter. The logout function will then be available in http://<APACHE_HTTPD_HOST>/mobius/logout.

```
#Create a logout endpoint
Redirect "/mobius/logout" "https://inmaasantgnan1p/mellon/logout?ReturnTo=http
s%3A%2F%2Finmaasantgnan1p%2Fmobius"
```

- c. The user's information needs to be added to the HTTP headers.

On successful authentication, the user's information is stored as Apache environment variable and it can be added to HTTP request header using the `mod_header` module directives.

- `PROXYUSER` header provides the username. `NAME_ID` is attribute in SAML assertion defining username
- `PROXYGROUP` header provides the role information. `ROLE` is attribute in SAML assertion defining the role.

The values should be substituted as per the environment. The attributes must be prefixed with `MELLON_` as shown below.

Capture the SAML assertion using "Developer Tools" and base decode to get the XML form.

```
# set information returned from IDP - keycloak
RequestHeader set PROXYUSER "%{MELLON_NAME_ID}e"
RequestHeader set PROXYGROUP "%{MELLON_ROLE}e"
```

- d. Configure mellon module with required information in `httpd.conf` file. It is recommended to create a separate configuration file for mellon.

The following example includes a file named `mellon.conf` to the `httpd` configuration. It is placed in `conf/extra` directory relative to the `httpd` base directory.

```
#mellon configuration
Include conf/extra/mellon.conf
```

The contents of the mellon configuration file defines the following:

- metadata files used by mellon
- Keycloak for SAML authentication
- public/private key pair used by mellon to sign assertions

It also defines the list of location that need to be protected.

The contents of `mellon.conf` file is shown below:

```
<Location ^
    MellonEnable info
    MellonEndpointPath /mellon/
    MellonSPMetadataFile /usr/local/apache2/conf/saml2/sp-metadata.xml
    MellonSPPrivateKeyFile /usr/local/apache2/conf/saml2/client-private-key.pem
    MellonSPCertFile /usr/local/apache2/conf/saml2/client-cert.pem
    MellonIdPMetadataFile /usr/local/apache2/conf/saml2/idp-metadata.xml
```

```

<Location>

<Location /mobius>
  AuthType Mellon
  MellonEnable auth
  Require valid-user
  MellonMergeEnvVars On
</Location>

```

Variable	Description
<Location/>	The Location directive on the / root is a convenient place to locate common configuration directives that will be shared by all mellon protected locations. In this instance, it defines the metadata files, certificates, and keys. It is not necessary to locate this on the / root URL, in fact in a real world deployment, you may want to locate the common shared set of Mellon directives lower in the hierarchy. The only requirement is that all of the protected locations are positioned below it so they may inherit those values.
MellonEnable info	Mellon does not process any directives unless it is enabled for that location either explicitly or through inheritance. If the user is authorized to access the resource, then Mellon populates the environment with information about the user. If the user is not authorized, then Mellon does not populate the environment, but Mellon does not deny the user access either.
MellonEndpointPath / mellon/	Defines where the Mellon endpoints are located in URL space. This is a critical value to properly specify and is one of the most common Mellon configuration errors leading to a failed deployment. The best way to think of these Mellon endpoints is as a way of binding a URL to a handler. When an HTTP request arrives at one of these Mellon endpoints, a dedicated handler processes the request. The way Mellon identifies a URL as being one of its endpoints is by looking at the beginning of the URL path. If everything in the path except the last path component matches the MellonEndpointPath then Mellon recognizes the URL as being one of its endpoints. Then the request is redirected to corresponding handler.
MellonSPMetadataFile	The SAML metadata for this provider (that is, Mellon's metadata). This metadata plays two important roles: Mellon reads it at start-up to initialize itself, and you provide the IdP specified in MellonIdPMetadataFile with this metadata. Both Mellon (the SP) and the IdP MUST have loaded exactly the same Mellon metadata in order to interoperate. Out of sync metadata is a very common deployment error. See Keycloak section for how Mellon metadata is created.
Include AuthnStatementSAML	Login responses may specify the authentication method used (password), timestamps of the login, and the session expiration. By default, this is enabled, which means that AuthStatement element is included in login responses. Note that setting this to off would prevent the client from determining the maximum session length which could result into never expiring client session.
MellonSPPrivateKeyFile	The private cryptographic key used by Mellon to sign its SAML data.
MellonSPCertFile	The public cryptographic key associated with the private key. This public key is embedded in Mellon's metadata so that an IdP can validate Mellon's signed

Variable	Description
	data. The certificates are used to only sign assertion and not to identify host machines.
MellonIdPMetadataFile	The IdP used to authenticate is specified by its metadata file. This file contains details about Keycloak and the URLs associated with the client.
Location /mobius	A URL location protected by Mellon. In the example, we have used the / mobius URL. Note that this <Location> block is simple and does not contain many of the necessary Mellon directives, because those other Mellon directives are inherited from an ancestor location, in the example /. The only Mellon directives in this location block are those necessary to turn on Mellon authentication. This configuration strategy permits to define many subordinate protected locations all sharing the same common Mellon directives via inheritance.
AuthType	AuthType is an Apache directive specifying which Apache authentication module performs the authentication for this location. Obviously, Mellon has to be used.
MellonEnable	Instruct Mellon that this location (and all its descendants) will be authenticated.
Require	An Apache directive that instructs Apache's authentication and authorization sub-system that it must successfully authenticate the user.
MellonMergeEnvVars	Attributes can be multi-valued. MellonMergeEnvVars controls whether each value is added to the environment by appending an index to the attribute name or whether the values are listed together under the bare attribute name with each value separated by a separator character.

MellonSPMetadataFile

The SP metadata files defines the SAML related configuration for mellon module. Keycloak provides an automated way to generate the SP metadata. For information on how to generate metadata, see [Setting up Keycloak](#).

MellonIdPMetadataFile

The IdP metadata file contain details about Keycloak and the URLs associated with it. Keycloak provides an automated way to generate the SP metadata. For information on how to generate metadata, see [Setting up Keycloak](#).

Sample Reference of httpd.conf File

```
#
# This is the main Apache HTTP server configuration file. It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.4> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>
# for a discussion of each configuration directive.
#
# Do NOT simply read the instructions in here without understanding
# what they do. They're here only as hints or reminders. If you are unsure
# consult the online docs. You have been warned.
#
# Configuration and logfile names: If the filenames you specify for many
```

```

# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path. If the filenames do not begin
# with "/", the value of ServerRoot is prepended -- so "logs/access_log"
# with ServerRoot set to "/usr/local/apache2" will be interpreted by the
# server as "/usr/local/apache2/logs/access_log", whereas "/logs/access_log"
# will be interpreted as '/logs/access_log'.

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path. If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used. If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "/usr/local/apache2"

#
# Mutex: Allows you to set the mutex mechanism and mutex file directory
# for individual mutexes, or change the global defaults
#
# Uncomment and change the directory if mutexes are file-based and the default
# mutex file directory is not on a local disk or is not appropriate for some
# other reason.
#
# Mutex default:logs

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding 'LoadModule' lines at this location so the
# directives contained in it are actually available before they are used.
# Statically compiled modules (those listed by 'httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule mpm_event_module modules/mod_mpm_event.so
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
#LoadModule mpm_worker_module modules/mod_mpm_worker.so
LoadModule authn_file_module modules/mod_authn_file.so
#LoadModule authn_dbm_module modules/mod_authn_dbm.so

```

```
#LoadModule authn_anon_module modules/mod_authn_anon.so
#LoadModule authn_dbd_module modules/mod_authn_dbd.so
#LoadModule authn_socache_module modules/mod_authn_socache.so
LoadModule authn_core_module modules/mod_authn_core.so
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule authz_user_module modules/mod_authz_user.so
#LoadModule authz_dbm_module modules/mod_authz_dbm.so
#LoadModule authz_owner_module modules/mod_authz_owner.so
#LoadModule authz_dbd_module modules/mod_authz_dbd.so
LoadModule authz_core_module modules/mod_authz_core.so
#LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
#LoadModule authnz_fcgi_module modules/mod_authnz_fcgi.so
LoadModule access_compat_module modules/mod_access_compat.so
LoadModule auth_basic_module modules/mod_auth_basic.so
#LoadModule auth_form_module modules/mod_auth_form.so
#LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule auth_mellon_module /usr/lib/apache2/modules/mod_auth_mellon.so
#LoadModule allowmethods_module modules/mod_allowmethods.so
#LoadModule isapi_module modules/mod_isapi.so
#LoadModule file_cache_module modules/mod_file_cache.so
#LoadModule cache_module modules/mod_cache.so
#LoadModule cache_disk_module modules/mod_cache_disk.so
#LoadModule cache_socache_module modules/mod_cache_socache.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
#LoadModule socache_dbm_module modules/mod_socache_dbm.so
#LoadModule socache_memcache_module modules/mod_socache_memcache.so
#LoadModule socache_redis_module modules/mod_socache_redis.so
#LoadModule watchdog_module modules/mod_watchdog.so
#LoadModule macro_module modules/mod_macro.so
#LoadModule dbd_module modules/mod_dbd.so
#LoadModule bucketeer_module modules/mod_bucketeer.so
#LoadModule dumpio_module modules/mod_dumpio.so
#LoadModule echo_module modules/mod_echo.so
#LoadModule example_hooks_module modules/mod_example_hooks.so
#LoadModule case_filter_module modules/mod_case_filter.so
#LoadModule case_filter_in_module modules/mod_case_filter_in.so
#LoadModule example_ipc_module modules/mod_example_ipc.so
#LoadModule buffer_module modules/mod_buffer.so
#LoadModule data_module modules/mod_data.so
#LoadModule ratelimit_module modules/mod_ratelimit.so
LoadModule reqtimeout_module modules/mod_reqtimeout.so
#LoadModule ext_filter_module modules/mod_ext_filter.so
#LoadModule request_module modules/mod_request.so
#LoadModule include_module modules/mod_include.so
LoadModule filter_module modules/mod_filter.so
#LoadModule reflector_module modules/mod_reflector.so
#LoadModule substitute_module modules/mod_substitute.so
#LoadModule sed_module modules/mod_sed.so
#LoadModule charset_lite_module modules/mod_charset_lite.so
#LoadModule deflate_module modules/mod_deflate.so
#LoadModule xml2enc_module modules/mod_xml2enc.so
#LoadModule proxy_html_module modules/mod_proxy_html.so
#LoadModule brotli_module modules/mod_brotli.so
LoadModule mime_module modules/mod_mime.so
#LoadModule ldap_module modules/mod_ldap.so
LoadModule log_config_module modules/mod_log_config.so
```



```
#LoadModule log_debug_module modules/mod_log_debug.so
#LoadModule log_forensic_module modules/mod_log_forensic.so
#LoadModule logio_module modules/mod_logio.so
#LoadModule lua_module modules/mod_lua.so
LoadModule env_module modules/mod_env.so
#LoadModule mime_magic_module modules/mod_mime_magic.so
#LoadModule cern_meta_module modules/mod_cern_meta.so
#LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
#LoadModule ident_module modules/mod_ident.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule version_module modules/mod_version.so
#LoadModule remoteip_module modules/mod_remoteip.so
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
#LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
#LoadModule proxy_uwsgi_module modules/mod_proxy_uwsgi.so
#LoadModule proxy_fdpass_module modules/mod_proxy_fdpass.so
#LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#LoadModule proxy_express_module modules/mod_proxy_express.so
#LoadModule proxy_hcheck_module modules/mod_proxy_hcheck.so
#LoadModule session_module modules/mod_session.so
#LoadModule session_cookie_module modules/mod_session_cookie.so
#LoadModule session_crypto_module modules/mod_session_crypto.so
#LoadModule session_dbd_module modules/mod_session_dbd.so
#LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
#LoadModule slotmem_plain_module modules/mod_slotmem_plain.so
LoadModule ssl_module modules/mod_ssl.so
#LoadModule optional_hook_export_module modules/mod_optional_hook_export.so
#LoadModule optional_hook_import_module modules/mod_optional_hook_import.so
#LoadModule optional_fn_import_module modules/mod_optional_fn_import.so
#LoadModule optional_fn_export_module modules/mod_optional_fn_export.so
#LoadModule dialup_module modules/mod_dialup.so
#LoadModule http2_module modules/mod_http2.so
#LoadModule proxy_http2_module modules/mod_proxy_http2.so
#LoadModule md_module modules/mod_md.so
#LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
#LoadModule lbmethod_bytraffic_module modules/mod_lbmethod_bytraffic.so
#LoadModule lbmethod_bybusyness_module modules/mod_lbmethod_bybusyness.so
#LoadModule lbmethod_heartbeat_module modules/mod_lbmethod_heartbeat.so
LoadModule unixd_module modules/mod_unixd.so
#LoadModule heartbeat_module modules/mod_heartbeat.so
#LoadModule heartmonitor_module modules/mod_heartmonitor.so
#LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
#LoadModule asis_module modules/mod_asis.so
#LoadModule info_module modules/mod_info.so
#LoadModule suexec_module modules/mod_suexec.so
<IfModule !mpm_prefork_module>
```

```

#LoadModule cgid_module modules/mod_cgid.so
<IfModule>
<IfModule mpm_prefork_module>
#LoadModule cgi_module modules/mod_cgi.so
</IfModule>
#LoadModule dav_fs_module modules/mod_dav_fs.so
#LoadModule dav_lock_module modules/mod_dav_lock.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
#LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
#LoadModule imagemap_module modules/mod_imagemap.so
#LoadModule actions_module modules/mod_actions.so
#LoadModule speling_module modules/mod_speling.so
#LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so

<IfModule unixd_module>
#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# It is usually good practice to create a dedicated user and group for
# running httpd, as with most system services.
#
User daemon
Group daemon

</IfModule>

# 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition. These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents. e.g. admin@your-domain.com
#
ServerAdmin you@example.com

#
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#

```

```

#ServerName www.example.com:80

#
# Deny access to the entirety of your server's filesystem. You must
# explicitly permit access to web content directories in other
# <Directory> blocks below.
#
<Directory >
    AllowOverride none
    Require all denied
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/apache2/htdocs"
<Directory "/usr/local/apache2/htdocs">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    # Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important. Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    # AllowOverride FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#

```

```

<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
    Require all denied
</Files>

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here. If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog /proc/self/fd/2

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

<IfModule log_config_module>
#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t /"%r/" %>s %b /"%{Referer}i/" /"%{User-Agent}i/" " combined
LogFormat "%h %l %u %t /"%r/" %>s %b" common

<IfModule logio_module>
# You need to enable mod_logio.c to use %I and %O
LogFormat "%h %l %u %t /"%r/" %>s %b /"%{Referer}i/" /"%{User-Agent}i/" %I %O" combi
nedio
</IfModule>

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog /proc/self/fd/1 common

#
# If you prefer a logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog "logs/access_log" combined
</IfModule>

```

```

<IfModule alias_module>
#
# Redirect: Allows you to tell clients about documents that used to
# exist in your server's namespace, but do not anymore. The client
# will make a new request for the document at its new location.
# Example:
# Redirect permanent /foo http://www.example.com/bar

#
# Alias: Maps web paths into filesystem paths and is used to
# access content that does not live under the DocumentRoot.
# Example:
# Alias /webpath /full/filesystem/path
#
# If you include a trailing / on /webpath then the server will
# require it to be present in the URL. You will also likely
# need to provide a <Directory> section to allow access to
# the filesystem path.

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the target directory are treated as applications and
# run by the server when requested rather than as documents sent to the
# client. The same rules about trailing "/" apply to ScriptAlias
# directives as to Alias.
#
ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"

</IfModule>

<IfModule cgid_module>
#
# ScriptSock: On threaded servers, designate the path to the UNIX
# socket used to communicate with the CGI daemon of mod_cgid.
#
#Scriptsock cgisock
</IfModule>

#
# "/usr/local/apache2/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>

<IfModule headers_module>
#
# Avoid passing HTTP_PROXY environment to CGI's on this or any proxied
# backend servers which have lingering "httproxy" defects.
# 'Proxy' request header is undefined by the IETF, not listed by IANA
#
RequestHeader unset Proxy early

```

```

<IfModule>

<IfModule mime_module>
#
# TypesConfig points to the file containing the list of mappings from
# filename extension to MIME-type.
#
TypesConfig conf/mime.types

#
# AddType allows you to add to or override the MIME configuration
# file specified in TypesConfig for specific file types.
#
#AddType application/x-gzip .tgz
#
# AddEncoding allows you to have certain browsers uncompress
# information on the fly. Note: Not all browsers support this.
#
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz
#
# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

#
# AddHandler allows you to map certain file extensions to "handlers":
# actions unrelated to filetype. These can be either built into the server
# or added with the Action directive (see below)
#
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
#
#AddHandler cgi-script .cgi

# For type maps (negotiated resources):
#AddHandler type-map var

#
# Filters allow you to process content before it is sent to the client.
#
# To parse .shtml files for server-side includes (SSI):
# (You will also need to add "Includes" to the "Options" directive.)
#
#AddType text/html .shtml
#AddOutputFilter INCLUDES .shtml
</IfModule>

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type. The MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
#MIMEMagicFile conf/magic

```

```

#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#

#
# MaxRanges: Maximum number of Ranges in a request before
# returning the entire resource, or one of the special
# values 'default', 'none' or 'unlimited'.
# Default setting is to accept 200 Ranges.
#MaxRanges unlimited

#
# EnableMMAP and EnableSendfile: On systems that support it,
# memory-mapping or the sendfile syscall may be used to deliver
# files. This usually improves server performance, but must
# be turned off when serving from networked-mounted
# filesystems or if support for these functions is otherwise
# broken on your system.
# Defaults: EnableMMAP On, EnableSendfile Off
#
#EnableMMAP off
#EnableSendfile on

# Supplemental configuration
#
# The configuration files in the conf/extra/ directory can be
# included to add extra features or to modify the default configuration of
# the server, or you may simply copy their contents here and change as
# necessary.

# Server-pool management (MPM specific)
#Include conf/extra/httpd-mpm.conf

# Multi-language error messages
#Include conf/extra/httpd-multilang-errordoc.conf

# Fancy directory listings
#Include conf/extra/httpd-autoindex.conf

# Language settings
#Include conf/extra/httpd-languages.conf

# User home directories
#Include conf/extra/httpd-userdir.conf

# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf

# Virtual hosts
#Include conf/extra/httpd-vhosts.conf

```

```

# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Various default settings
#Include conf/extra/httpd-default.conf

# Configure mod_proxy_html to understand HTML4/XHTML1
<IfModule proxy_html_module>
Include conf/extra/proxy-html.conf
</IfModule>

#mellon configuration
Include conf/extra/mellon.conf

# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
# Note: The following must must be present to support
# starting without SSL on platforms with no /dev/random equivalent
# but a statically compiled-in mod_ssl.
#
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>

#Create a logout endpoint
Redirect "/mobius/logout" "https://inmaasantgnan1p/mellon/logout?ReturnTo=https%3A%2F%
2Finmaasantgnan1p%2Fmobius"

ProxyRequests off

<Proxy *>
Order deny,allow
Deny from all
Allow from all
</Proxy>
ProxyTimeout 300

ProxyPass /mobius/logout !
ProxyPass /mobius http://mobiusView:8080/mobius
ProxyPassReverse /mobius http://mobiusView:8080/mobius
ProxyPass /echo http://echo:8080/echo
ProxyPassReverse /echo http://echo:8080/echo

# set information returned from IDP - keycloak
RequestHeader set PROXYUSER "%{MELLON_NAME_ID}e"
RequestHeader set PROXYGROUP "%{MELLON_ROLE}e"

```

HTTPD docker-compose configuration

The docker-compose configuration used for Apache HTTPD server is shown below:

```
#proxy in front of mobius
proxy:
# image: httpd:2.4
build: './httpd'
container_name: proxy
ports:
  - 80:80
  - 443:443
volumes:
  - ./public-html:/usr/local/apache2/htdocs/
  - ./saml-httpd.conf:/usr/local/apache2/conf/httpd.conf
# Add cert for your domain in below two lines
  - ./certificate/localhost/localhost.key:/usr/local/apache2/conf/server.key
  - ./certificate/localhost/localhost.crt:/usr/local/apache2/conf/server.crt
  - ./https__inmaasantgnan1p_mellon_metadata:/usr/local/apache2/conf/saml2
  - ./mellon.conf:/usr/local/apache2/conf/extra/mellon.conf
```