MT AG

SINGLE SIGN-ON

FOR (APEX) APPLICATIONS

WITH AZURE AD

USING SAML V2

Author: Niels de Bruijn
Version: 1.4
Date: 06-DEC-2018

1 INTRODUCTION

Most APEX environments run inside the corporate network. In some cases, you would like to also give registered external users (like customers or partners) access over the internet to specific APEX apps running on the internal APEX instance. To prevent ending up building your own user/password management system including a registration process, most companies already utilize Azure AD, which is the cloud version of Active Directory. The question here is: how can we securely authenticate external users that are registered in Azure AD? There are various ways how you can achieve this (ie. SAMLv2 or OAUTH2 being two of these). We will utilize the SAMLv2 standard as we only want to trust Azure AD as Identity Provider using a secure channel (SSL) and "automatically" get the user ID together with selected user attributes back as part of the HTTP header. This doesn't mean that using OAUTH2 for Single Sign-On isn't an option. Each way has its own advantages and disadvantages.

This document will show you how to setup Single Sign-On using SAMLv2 against Azure AD.

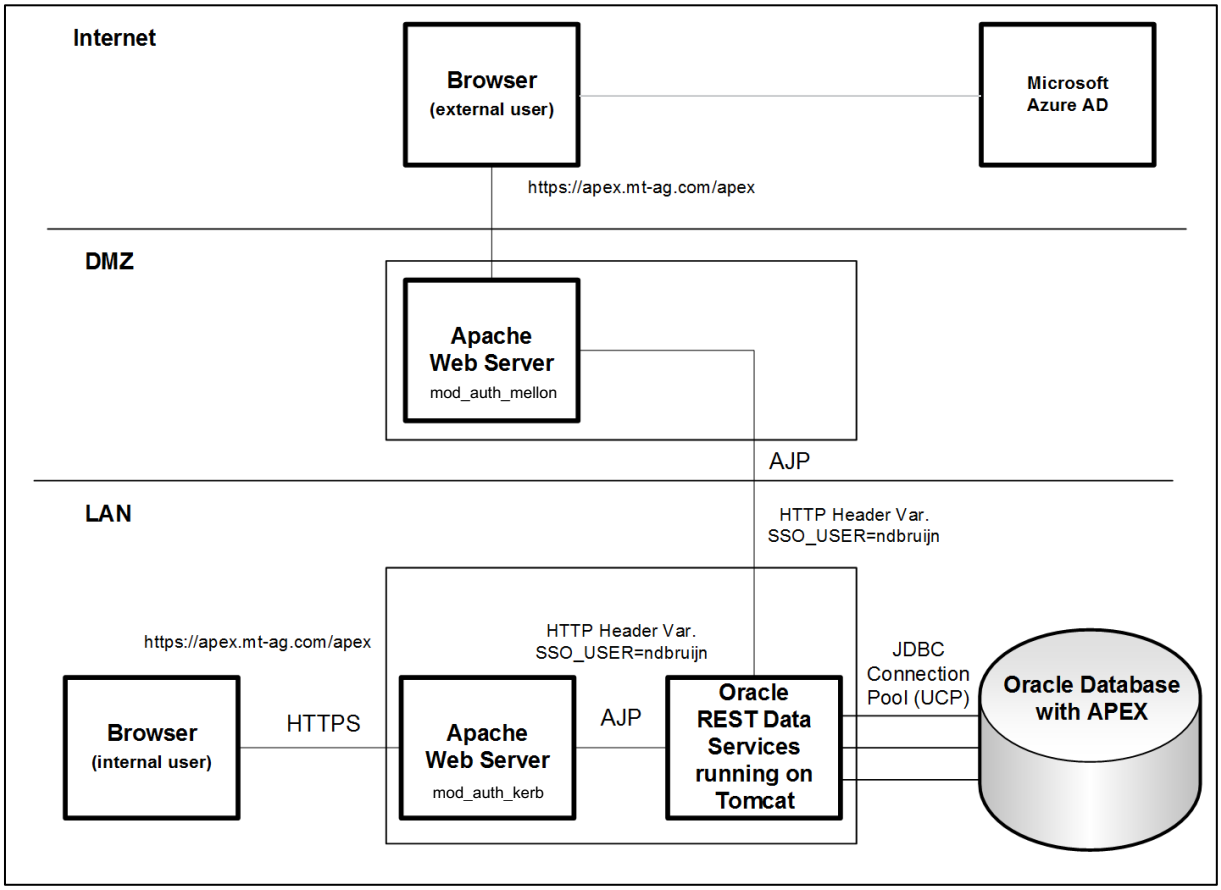

Image 1: Enabling secure access for external users, registered in Azure AD.

Before starting the implementation, make sure that you already have setup an APEX environment as well as an Azure AD subscription. Access to the APEX environment should be given through ORDS on Tomcat. We will need Apache running on a server in the DMZ to enable access to the APEX environment.

Remarks:

- Use a firewall to restrict communications from the Internet with the Apache web server through port 443 (HTTPS).
- Use a firewall to restrict communication from Apache web server to Tomcat through port 8009/8010 (AJP).
- By default, Tomcat already runs on port 8009 using AJP. Configure it in `server.xml` so that ORDS runs on `127.0.0.1:8009` (AJP) for local requests and `xx.xx.xx.xx:8010` (AJP) for requests coming from DMZ. Replace the `xx` with the IP address of your server in DMZ.
- Once finished, you can use any device and browser that is certified with the APEX version you are running. The process flow is the same each time: you start a browser instance, enter the APEX URL, login at Azure AD and start working with any authorized APEX app without having the logon again (unless you close the browser instance, your session has expired or the Apache web server was restarted).
- Please be aware that this document is not about hardening your environment or optimizing the configuration for ie. ORDS. For instance, you might not need all Apache modules that are installed by default.
- In this case, the APEX URL (/apex) and all its static files (/i) will be protected, but you can protect any other web application with this approach that lies behind the Apache web server.
- The static files of APEX are expected to be on the Tomcat server.
- We will use Apache with mod_auth_mellon in this document, but you could also use a hardware load balancer instead if you expect high traffic.
- After enabling SSO using this document, the app is automatically displayed as a tile on the Office 365 dashboard.
- The features outlined in this document are included in the standard subscription of Azure AD. If you want to customize the metadata provided by Azure AD using its interface portal.azure.com, you can do this with a premium subscription.
- If you have the money, consider replacing the Apache Web Server with a hardware load balancer.

Further links:

- To enable Single Sign-On for employees using Kerberos, I suggest you have a look at this document: https://www.edocr.com/v/lv1rvxvx/nielsdebruijn/Single-Sign-On-for-Oracle-Application-Express-APEX.
- Here is a good blog post about SAMLv2 versus OAUTH2: https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2.
- Good read about customizing metadata returned by Azure AD: https://github.com/MicrosoftDocs/azure-docs/blob/master/articles/active-directory/develop/active-directory-optional-claims.md

2 CONFIGURATION OF APEX

2.1 SESSION TIMEOUT

Session timeout is already taken care of by the mod_auth_mellon module, so it makes no sense to have this active for an APEX session as well. We can disable the session timeouts for the APEX session cookie by setting it to 0. You can find this setting in the “security settings” section after logging on in the internal workspace, but be aware that this setting can be overridden on workspace and application level.

2.2 AUTHENTICATION SCHEME

For each APEX application (including APEX itself), configure a authentication scheme that reads out the HTTP header variable „SSO_USER“.

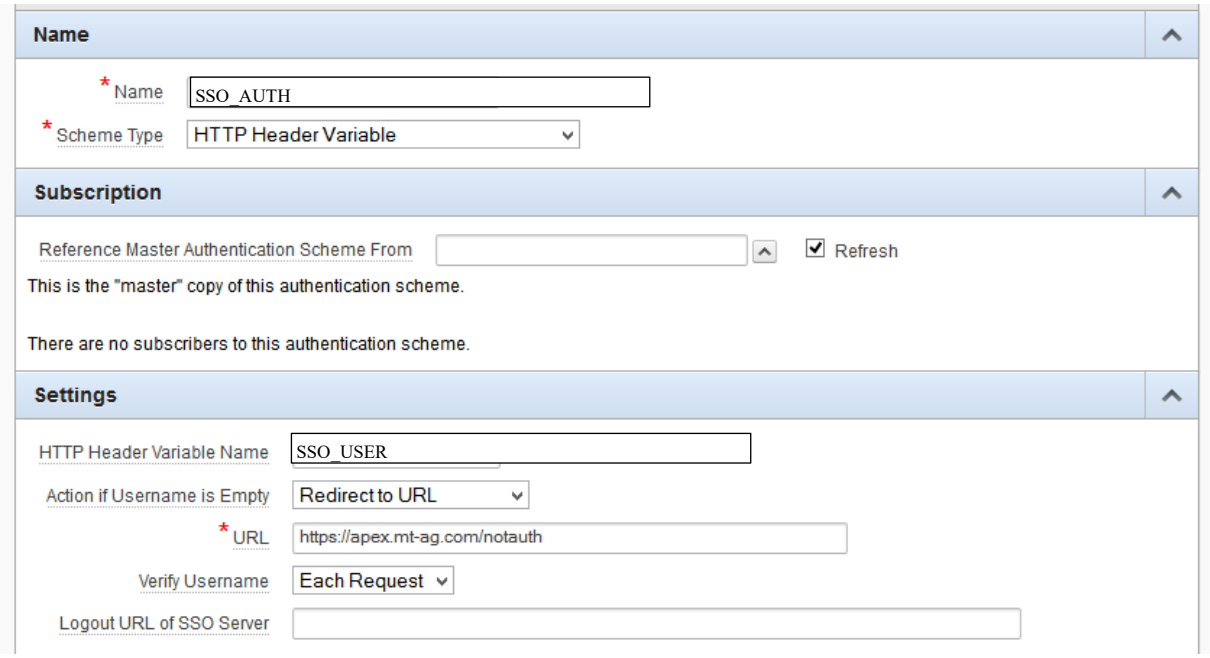

Image 2: Changing the authentication method for an APEX application to HTTP header.

When SSO_USER is empty, the user will be redirected to a public static HTML page (index.html) hosted by Apache. This page will inform the user that he or she is currently not logged on.

Remarks:

- If you would like to strengthen security a bit, you could take a long series of random characters and numbers for the Header Variable Name instead of using SSO_USER.

3 AZURE AD CONFIGURATION

We need to register APEX in Azure AD through portal.azure.com:

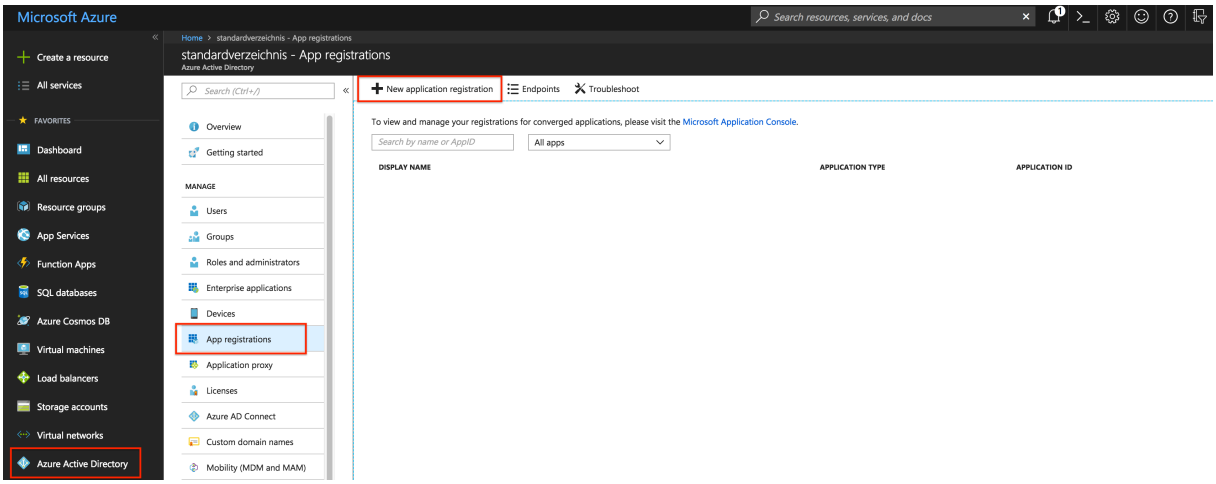

Image 3: Adding a new application to Azure AD.

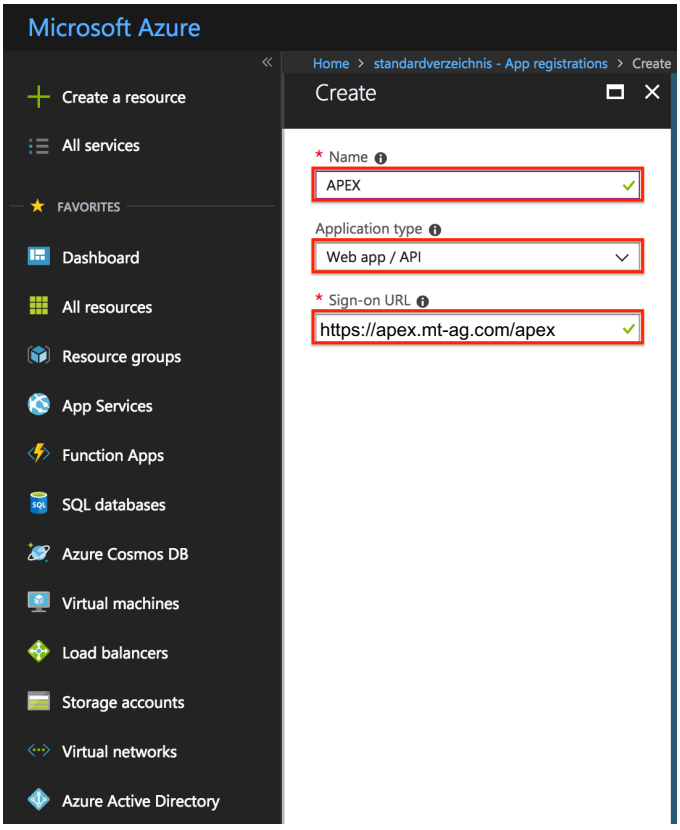

Image 4: Adding a new application to Azure AD.

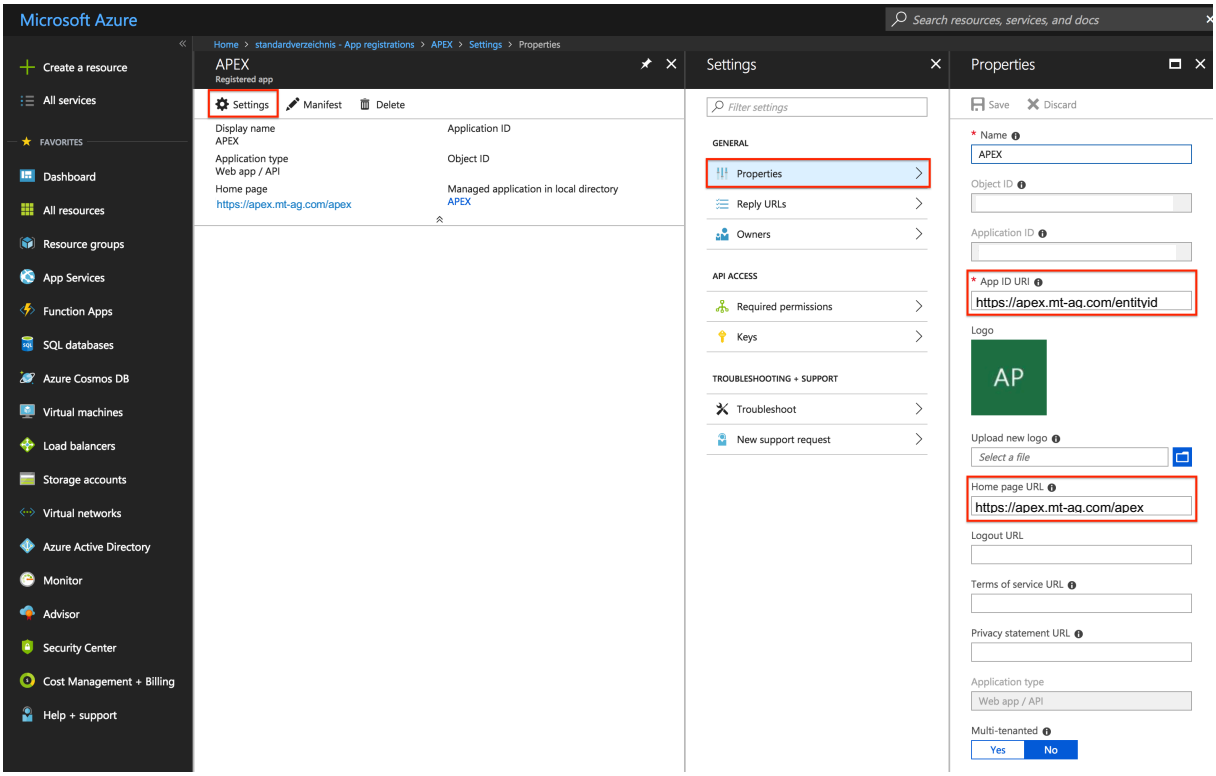

Image 5: Changing the properties of the registration.

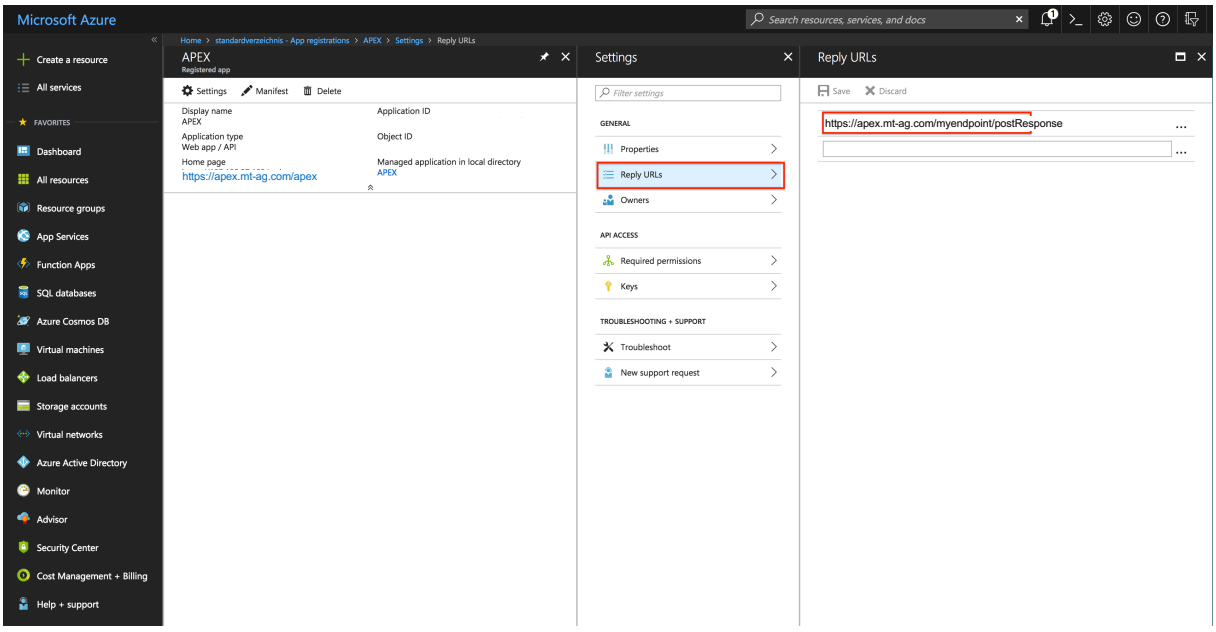

Image 6: Changing the properties of the registration.

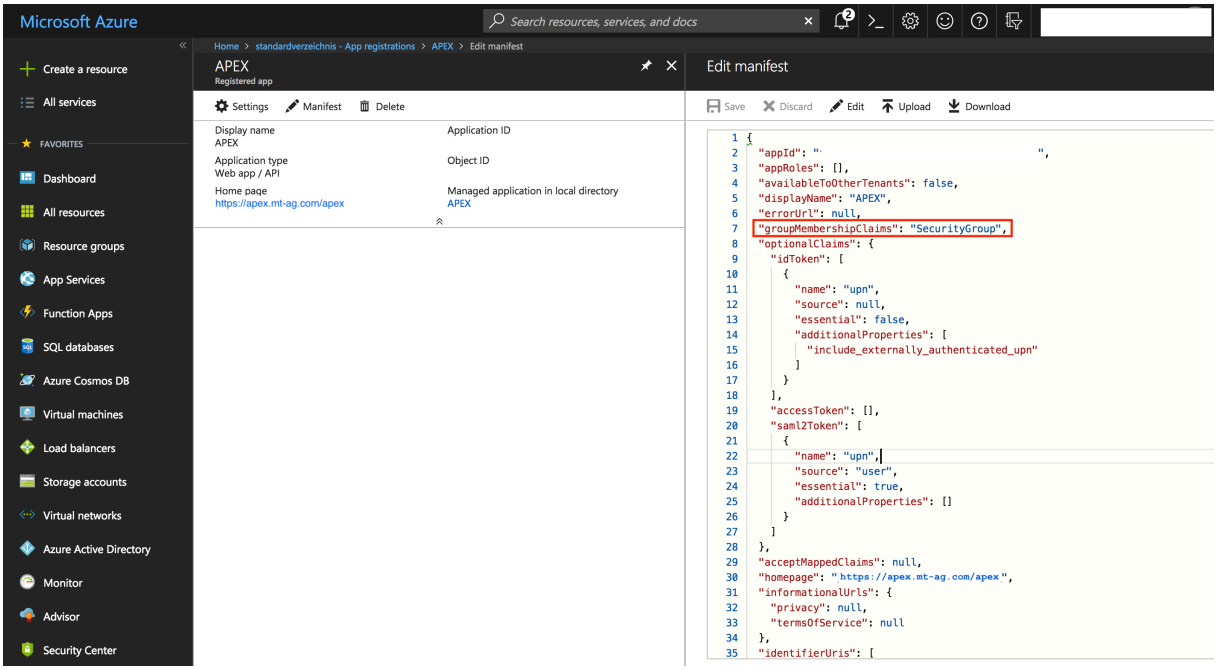

Image 7: Optional - Pass back (unique) Object IDs of all groups the user belongs to (up to 150).

4 APACHE CONFIGURATION

4.1 INSTALL MOD_AUTH_MELLON

By installing the module mod_auth_mellon, Apache will be installed as well:

```
yum install mod_auth_mellon
```

Make sure that Apache starts upon server reboot:

```
chkconfig httpd on
```

This document uses the default SSL certificate for secure communication between the servers. Although we can still test single sign-on using this default certificate, you should install a valid SSL certificate (not only) to prevent a security warning in the browser.

4.2 CREATE SAML CONFIGURATION FILES FOR APACHE

Create a directory for the configuration files:

```
cd /etc/httpd
mkdir mellon
cd mellon
```

Download and execute a script with which the configuration files can be generated:

```
wget
https://raw.githubusercontent.com/UNINETT/mod_auth_mellon/master/mellon_c
reate_metadata.sh
chmod 755 mellon_create_metadata.sh
./mellon_create_metadata.sh https://apex.mt-ag.com/entityid
https://apex.mt-ag.com/myendoint
```

We need the username in the HTTP header to let APEX create a session based on it. By default, the username, called MELLON_NAME_ID, is not presented in a readable format. We need to specify the format to make it readable by editing the file `https_apex.mt_ag.com_entityid.xml` and add the line marked yellow:

```
<EntityDescriptor entityID="https://apex.mt-ag.com/entityid"
xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <SPSSODescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress</NameIDFormat>
    <KeyDescriptor use="signing">
...
```

Download the metadata from Microsoft with details about the Identity Provider. You can copy the URL for your environment out of portal.azure.com:

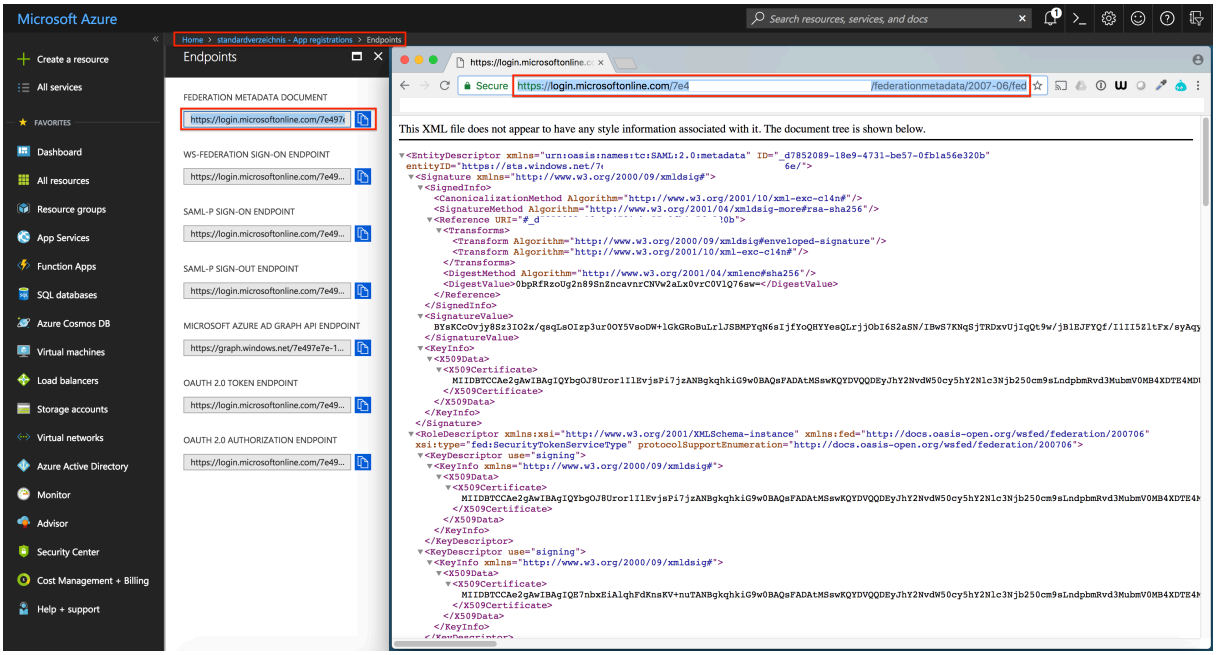

Image 7: Copy the metadata document to clipboard.

```
wget https://login.microsoftonline.com/this-string-is-specific-to-your-ad-domain/federationmetadata/2007-06/federationmetadata.xml
```

4.3 SSL.CONF

Add the following lines in the file `/etc/httpd/conf.d/ssl.conf` just before `</VirtualHost>`:

```
..
# Adds Mellon session information to all requests to the web server
<Location />
  MellonEnable "info"

  # Use the filenames provided by the mellon_create_metadata.sh script
  MellonSPPrivateKeyFile
/etc/httpd/mellon/https_apex.mt_ag.com_entityid.key
  MellonSPCertFile /etc/httpd/mellon/https_apex.mt_ag.com_entityid.cert
  MellonSPMetadataFile
/etc/httpd/mellon/https_apex.mt_ag.com_entityid.xml

  # Add the full path to the IdP metadata that was downloaded when
configuring the IdP
  MellonIdPMetadataFile /etc/httpd/mellon/federationmetadata.xml

  RequestHeader set APEX_USER %{MELLON_NAME_ID}e

  # The endpoint path used when generating the SP metadata
  MellonEndpointPath /myendpoint
</Location>

SSLProxyEngine On
SSLProxyVerify none
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
```

```
SSLProxyCheckPeerExpire off

# Enable Mellon for the ORDS path
<Location /apex>
  MellonEnable "auth"
  ProxyPreserveHost On
  RewriteEngine On

#
# START of optional Azure AD group check
#
  # Display all groupmemberships in one variable using ; as separator
  MellonMergeEnvVars On

  # By using OR you can specify multiple groups
  MellonCond "
http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"
"12345f0d-8d15-4e02-ac8a-3cab5cdd1234" [OR]
  MellonCond "
http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"
"12345f0d-8d15-4e02-ac8a-3cab5cdd5678"
#
# END of optional Azure AD group check
#

  # A registered external user may only call a specific APEX app ID.
  # An additional APEX app can be allowed as follows: RewriteCond
%{QUERY_STRING} !p=(102055|102056)
  # Any other app ID in the URL will lead to a "forbidden" page.
  # URLs to reference a static file or JS messages are allowed as well.
  RewriteCond %{QUERY_STRING} !p=102055
  RewriteCond %{REQUEST_URI} !/<Enter Workspace Name here>/r/102055
  RewriteCond %{REQUEST_URI} !wwv_flow.js_messages
  RewriteRule .* / [F]

  ProxyPass ajp://apex-lan.mt-ag.com:8010/apex
</Location>

<Location /i>
  MellonEnable "auth"
  ProxyPreserveHost On
  ProxyPass ajp://apex-lan.mt-ag.com:8010/i
</Location>
</VirtualHost>
```

Instead of using MellonCond, you can also use RewriteCond:

```
...
#
# START of optional Azure AD group check
#
  # Display all groupmemberships in one variable using ; as separator
  MellonMergeEnvVars On

  # Copy the variable so it can be referenced by RewriteCond.
  # Remark: RequestHeader doesnt work here as it is interpreted after
RewriteCond.
  RewriteRule .* -
[E=AZURE_GROUPIDS:%{ENV:MELLON_http://schemas.microsoft.com/ws/2008/06/id
entity/claims/groups}]
```

```
  # Check if the user belongs to a specific group in Azure AD, if not, a
Forbidden page is shown
  # Azure AD returns the object id of the group by default, in this
example the group id stands for the group name "My APEX App"
  # Remark: the group id is unique, whereas the group name can be changed
in Azure AD.
  RewriteCond %{ENV:AZURE_GROUPIDS} !12345f0d-8d15-4e02-ac8a-3cab5cdd1234
  RewriteRule .* / [F]
#
# END of optional Azure AD group check
#
...
```

Save the file and restart Apache:

```
systemctl restart httpd
```

Now open up your browser and access `https://apex.mt-ag.com/apex/f?p=102055`. You should now first logon to Azure AD, either with a guest or normal account, before you can access the APEX application. Any other APEX URL, like `/apex`, will be blocked by Apache and the user is presented the default “forbidden” page.

5 AUTHENTICATION FLOW

Here is what happens under the hood when you access an APEX app protected by SAMLv2:

1. The user accesses `https://apex.mt-ag.com/apex/f?p=102055`.

2. The HTTP GET request is received by Apache. Since all URLs with /apex are protected, mod_auth_mellon uses the metadata in the XML file to redirect the user to a logon page of Azure AD where the authentication takes place.

3. After authentication, a HTML form containing an encrypted XML (SAML Token) is sent back to the client (browser) and is then automatically posted to the Service Provider (Apache) using `https://apex.mt-ag.com/myendpoint/postResponse` as the endpoint.

4. Apache decrypts the XML message and finds out if the Azure AD authentication succeeded. It does not have to contact Azure AD to check this. mod_auth_mellon now creates a session cookie “mellon-cookie” which is sent back to the browser.
The Apache configuration also checks if the App ID in the URL is generally accessible and, optional, checks if the user belongs to a specific group in Azure AD.

5. The user is now automatically redirected back to `https://apex.mt-ag.com/apex/f?p=102055`, but now the session-cookie is sent with the request. mod_auth_mellon checks if the session-cookie is valid and passes the request to APEX running in the local area network together with the username in the HTTP header variable SSO_USER.

6. The authentication scheme of the APEX application now reads out the value of the HTTP variable and creates a session cookie which is sent back to the browser. The user context is now known and can be requested by the developer by calling `:APP_USER` in APEX.

7. A subsequent browser request now includes two session cookies, one for Apache and one for APEX.