Paper SAS1385-2015

Federated Security Domains with SAS® and SAML

Mike Roda, SAS Institute Inc.

ABSTRACT

From large holding companies with multiple subsidiaries to loosely affiliated state educational institutions, security domains are being federated to enable users from one domain to access applications in other domains and ultimately save money on software costs through sharing. Rather than rely on centralized security, applications must accept claims-based authentication from trusted authorities and support open standards such as Security Assertion Markup Language (SAML) instead of proprietary security protocols. This paper introduces SAML 2.0 and explains how the open source SAML implementation known as Shibboleth can be integrated with the SAS® 9.4 security architecture to support SAML. It then describes in detail how to set up Microsoft Active Directory Federation Services (AD FS) as the SAML identity provider, how to set up the SAS middle tier as the relying party, and how to troubleshoot problems.

INTRODUCTION

This paper will introduce the reader to federated identity and give a technical overview of what is perhaps the most important standard in this space, the Security Assertion Markup Language (SAML) 2.0 standard. Designed as layers of standards on top of each other, at the outer-most layer are SAML profiles that implement the use cases we are interested in, single sign-on, federated identity, and others. We will discuss the layers that make up the SAML 2.0 standard and look at how SAML distinguishes between the roles of the identity provider and service provider.

With some background on SAML 2.0, we will examine how the SAS 9.4 middle tier can integrate with existing security infrastructure in the enterprise by leveraging pluggable authentication modules for the Apache HTTP server that SAS Web Server is based. One such module comes from the open source Shibboleth project and implements a SAML 2.0 service provider. We will walk through the configuration required to use Shibboleth with the SAS middle tier in order to support authentication through an external identity provider.

On the other end of the federation is Microsoft Active Directory Federation Services (AD FS) 2.0. We will show how to use Microsoft's implementation of the SAML 2.0 standard to pull user attributes out of the Active Directory LDAP and send them as claims in the SAML assertion to the SAS middle tier, where they can be consumed for authentication.

SAML 2.0 is a very flexible standard that supports multiple communication mechanisms known as bindings, and ways of including information about users in the assertions. This unfortunately means that some troubleshooting will probably be required to get things working. In addition to showing what to look for in the log files, we will explain how to use a web browser to capture, decode and view the SAML assertions and look at examples of the XML data.

BACKGROUND ON FEDERATED IDENTITY AND SAML 2.0

The system of international travel passports used around the world is an example of federated security domains and claims-based authentication used in the physical world. People can prove their identity and citizenship within their own country using local documentation such as a birth certificate but this documentation is not accepted anywhere else. In order to collaborate on international travel, countries issue standardized passports to their citizens and accept the passports from other countries. The passport contains claims about the person's name, birthdate, and so on. Watermarks and other security devices prevent tampering and ensure authenticity. It would be impractical for a single government authority to issue passports for everyone in the world.

The same holds for computer systems. Collaboration and sharing applications between affiliated but distinct enterprises or institutions is becoming increasingly common. This extends the challenge of single sign-on to multiple security domains. Centralization of security is no longer possible. Duplication of

credentials is problematic and possibly insecure. The problem can be solved using a topology of federated security domains together with claims-based authentication as the mechanism for passing identity data between them. With claims-based authentication, an authority in the federation issues a token with claims about the bearer of said token. The token usually contains a digital signature to verify its integrity. With multiple security domains exchanging tokens, standardization becomes critical.

The Security Assertion Markup Language (SAML) 2.0 standard defines a framework for exchanging security information between partners in a federation. This security information is packaged in the form of portable XML assertions that applications working across security domain boundaries can trust. The standard was approved by the Organization for the Advancement of Structured Information Standards (OASIS) in March 2005.

KEY TERMS AND DEFINITIONS FOR SAML 2.0

Table 1 provides a table of terms.

Term	Concept
Federation	Two or more security domains with trust established between them.
Assertion	XML document that is created and sent during a federated access request and contains claims about a user.
Claims	Information a federated member is asserting to be true.
Identity provider	A federation member that creates assertions for its users. Also referred to as the asserting party.
Service provider	A federation member that consumes assertions to make access control decisions for its applications. Also referred to as a relying party.
Attributes	Information about the user that is contained in the assertion.
Metadata	XML document produced by a SAML provider to describe its service endpoint URLs, x.509 certificate and other information in a standard way for consumption by partners in the federation.

Table 1. SAML 2.0 Terminology

The SAML 2.0 standard consists of layers that, at the highest level, describes use cases to be supported. Single sign-on and federated identity, the focus of this paper, are included as well as security exchange between SOAP-based web services.

An exchange of security information in SAML takes place between an asserting party and a relying party. For multiple domain single sign-on with federated identity, the asserting party is referred to as an identity provider and the relying party is the service provider. SAML assertions are XML documents that contain a subject and include information about the subject in the form of attributes that the identity provider claims to be true.

When a user attempts to access service URL, the service provider initiates the exchange with an authentication request and the identity provider sends a response which encapsulates the assertion. The SAML protocol defines the structure and content of these request and response messages.

The method for which standard network communication protocols such as HTTP are used to send SAML protocol messages between parties is defined by the SAML bindings. This paper looks at the HTTP bindings but there are also more specialized bindings for SOAP. The three types of HTTP bindings are:

- HTTP Redirect Binding – The web browser receives a 302 redirect to the partner endpoint with the SAML message usually deflated (compressed), Base-64 encoded, and appended to the URL as a query string parameter.
- HTTP POST Binding – The web browser receives an HTML form containing the SAML message encoded into a hidden field and javascript to immediately subject the form and to the partner endpoint.

- HTTP Artifact Binding – Either of the previous two except the contents of the message contain only an artifact ID. The partner then must make a separate out-of-band call with the artifact to obtain the full SAML message.

Note that the SAML request and response use these bindings independently. It is common for the SAML authentication request to flow over HTTP Redirect Binding and the larger SAML response, that contains the assertion, to come back using HTTP POST Binding.

Finally, SAML profiles refer to the specific use case that is being addressed. Profiles can impose constraints on the protocol, bindings, and assertions in order to maximize interoperability.

Figure 1 illustrates the relationship between these SAML layers.

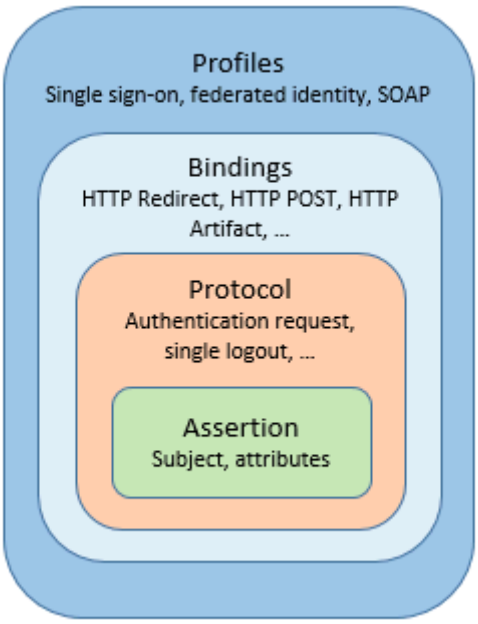


Figure 1. Layers of the SAML 2.0 standard

MICROSOFT AD FS 2.0

Microsoft Active Directory Federation Services (AD FS) 2.0 is a software component for Microsoft Windows operating systems that can be installed on the Active Directory domain controller to support federated identity and SAML 2.0. AD FS can act as the asserting party, asserting identity claims from the Active Directory LDAP, or as a relying party consuming claims from another party.

SAS AUTHENTICATION WITH SAML AND MICROSOFT AD FS

The SAS web application tier, otherwise known as the middle tier, centralizes security at the SAS Logon Manager, which is actually an implementation of the JASIG Central Authentication Service (CAS). CAS achieves single sign-on (SSO) using a protocol that loosely mimics Kerberos. In the typical scenario, this works as follows. When a user visits a SAS web application, their web browser is redirected to the SAS Logon Manager where they are prompted for credentials to log in. After authentication, the Logon Manager issues a Ticket Granting Cookie (TGC) to the user's web browser and redirects it back to the original web application with a one-time-use Service Ticket attached to the URL in the form of a query string parameter. The web application makes a back channel call to the Logon Manager to validate the service ticket and then establishes a web session for the user. If the user then visits another SAS web application for which they do not have a web session yet, they are again redirected to the Logon Manager. However, this time the web browser provides the previously issued TGC and instead of being

prompted again for credentials is immediately redirected back to the application with a service ticket for the other web application.

ENTERPRISE SECURITY INTEGRATION

While this technology for single sign-on works well among the SAS web applications, enterprises typically have their own centralized security infrastructure and identity store. In many cases, enterprise single sign-on can be extended over the SAS middle tier. There are several ways of accomplishing this but what all of these have in common is they pre-establish an authenticated principal in the login request to the Logon Manager and CAS is configured to trust this. The specific means for pre-establishing the principal in the login request is how the methods differ but they can be approximately grouped into those that use built-in features of the web application server, and those that perform authentication upstream in the web server.

While some of the most common authentication methods are built into the Tomcat-based SAS Web Application Server, many more are available as pluggable authentication modules to the popular open source Apache HTTP server that SAS Web Server is based on. Credentials established in the web server are rewritten into HTTP headers and passed to the web application server, where they are intercepted and used to pre-establish a principal in the login request. This is depicted in Figure 2.

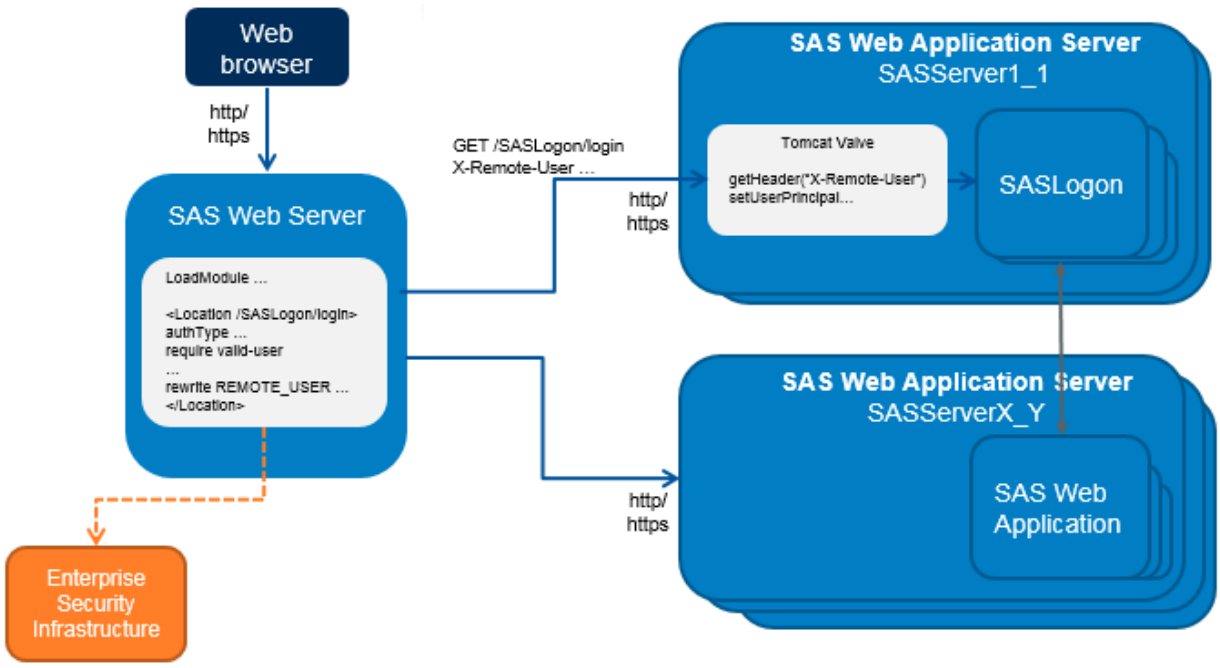


Figure 2. Enterprise Security Integration with Web Server Authentication

Note that the web server is performing authentication only on the login request. The other SAS web applications are unaware of enterprise security and always defer authentication to the SAS Logon Manager.

SHIBBOLETH SP

A standard SAS deployment can be expressed in SAML terminology, with the SAS Logon Manager acting as the identity provider and the other SAS web applications as service providers. This does not change. However, when integrating with AD FS 2.0 for the identity provider, the SAS login page must be seen by AD FS as a service provider address. You can think of it as the login page is providing the service of logging you into the local identity provider. This can seem convoluted but the login page is actually transparent to the user with web authentication.

This can be accomplished using the open source SAML 2.0 service provider implementation from the Shibboleth project. The Shibboleth SP is implemented as native compiled software that is divided into a pair of components: a daemon process named "shibd", and an Apache HTTP server loadable authentication module that can be plugged into SAS Web Server. Since the Apache web server runs as a pool of separate processes, the shibd process is used to maintain a common cache of session information. On Windows the daemon is installed as a service and named shibd_Default by default. The two components can be restarted independently of each other.

The Shibboleth Native SP module for Apache is supported on 64-bit Windows, Linux, and Solaris, but binaries are available only for Windows and Linux. The Windows binaries come with an installer. For installation on Linux, you can use yum or install from RPMs. It must be built from source for Solaris SPARC or x64.

HTTPS COMMUNICATION

It is recommended that SAS Web Server be configured for HTTPS. Since the SAML 2.0 endpoints provided by the Shibboleth service provider are surfaced through SAS Web Server, the assertions passed to Shibboleth will go over an encrypted channel. Make sure the Trusted Roots store of the AD FS 2.0 machine contains the certificate used by SAS Web Server or the CA certificate chain that signed it.

SAML 2.0 was designed to work over unencrypted communications so the identity provider will also encrypt the assertions passed to the service provider using the public key from the service provider's Metadata. AD FS 2.0 does this by default but it can be disabled for debugging or if considered too redundant. Shibboleth automatically generates RSA keys and a self-signed certificate for itself and includes the public key in the metadata file that is shared with the identity provider.

With authentication being performed in SAS Web Server, trust must be established back to SAS Web Application Server. A few options are presented later in this paper. It is recommended that SAS Web Application Server be configured for HTTPS to secure the communication.

SAS Web Server can be configured for HTTPS at deployment time using the SAS Deployment Wizard. There are instructions for manually configuring HTTPS to SAS Web Server and SAS Web Application Server in the SAS Middle Tier Administration Guide.

SHIBBOLETH INSTALLATION

SAS does not ship Shibboleth software so the Shibboleth Native SP must be downloaded and installed separate from SAS. Instructions can be found on the Shibboleth 2 installation page. There are links under Native Service Provider (SP) for all the supported platforms, and on each of those pages is a list of web servers including Apache. By default the software is installed under `/opt/shibboleth-sp` on UNIX and `C:\opt\shibboleth-sp` on Windows.

Configuration files are found in the `etc` subdirectory. The most important configuration file is `shibboleth2.xml`. The application entityID should be edited. After installation the file should be edited and the entityID set to the URL of the shibboleth service running on the web server. For example if SAS Web Server is running on sas.company.com:

```
<ApplicationDefaults entityID="https://sas.company.com/shibboleth"
                     REMOTE_USER="eppn persistent-id targeted-id">;
```

APACHE CONFIGURATION

Most of the configuration needed for Apache comes packaged from Shibboleth in add-on files named to coincide with the supported versions of Apaches. At the time of this writing, SAS Web Server is based on Apache HTTP server 2.2 so the `apache22.config` file that comes with Shibboleth needs to be included in the `httpd.conf` file in the SAS Web Server configuration directory. This should go at the bottom of the file. The example below is for Windows:

```
Include C:/opt/shibboleth-sp/etc/shibboleth/apache22.config
```

Then Apache is configured to require Shibboleth authentication for the SAS login page. To prevent other parts of the environment from failing, it is important that only this location is protected by Shibboleth:

```
<Location /SASLogon/login>
 AuthType shibboleth
 ShibRequestSetting requireSession 1
 require valid-user
</Location>
```

If SAS Web Server is configured for HTTPS, the above should be added within the `VirtualHost` defined in `httpd-ssl.conf`, which is located in the extra subdirectory.

Apache must be configured to pass the authenticated principal name to the web application server. We would like to rewrite the standard Apache REMOTE_USER environment variable into an HTTP header; however, since REMOTE_USER is determined late in the Apache processing flow, it cannot simply be set in a header using a RequestHeader directive. This problem was discussed on the Server Fault website for administrators and a solution posted using a look-ahead. The resulting configuration is shown below:

```
<Location /SASLogon/login>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user

  RewriteEngine On
  RewriteCond %{LA-U:REMOTE_USER} (.+)
  RewriteRule . - [E=RU:%1,NS]
  RequestHeader set X-Remote-User "%{RU}e" env=RU
</Location>
```

Send Secret Password

Malicious clients must be prevented from bypassing the web server and spoofing the `X-Remote-User` header. There are two ways this can be accomplished:

- A secret password known only to the web server and web application server can be sent in every login request using a Basic Authorization header.
- The web server can be configured to authenticate with the web application server with client certificate authentication.

This paper will use the secret password. A Base64-encoded Authorization string of the format user name:password is created. The user name is not used. Many online sites exist to Base64 encode and decode the string. For example ":Password1" is encoded as "OlBhc3N3b3JkMQ==". This is added to the request headers:

```
<Location /SASLogon/login>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user

  RewriteEngine On
  RewriteCond %{LA-U:REMOTE_USER} (.+)
  RewriteRule . - [E=RU:%1]
  RequestHeader set X-Remote-User "%{RU}e" env=RU
  RequestHeader set Authorization "Basic OlBhc3N3b3JkMQ=="
</Location>
```

As mentioned earlier, SAS Web Application Server should be configured for HTTPS. This will protect the secret password from network sniffers.

TOMCAT VALVE CONFIGURATION

SAS Web Application Server is Pivotal tc Server, which itself is based on Apache Tomcat. Tomcat allows for valves that sit in the request processing pipeline to perform some action. A number of valves come built into Tomcat and the default SAS configuration uses them for such things as logging. SAS has extended the container with valves for authentication and other purposes. To support authentication performed in SAS Web Server, SAS provides a valve that pulls the user name out of an HTTP request header and sets the principal in request. For security purposes the valve is not configured by default so it must be added.

The valve only needs to be exposed to SASLogon to it is added to `Web/WebAppServer/SASServer1_1/conf/Catalina/localhost/SASLogon.xml` under the config directory:

```
<Valve
className="com.sas.vfabrictcsvr.authenticator.PrincipalFromRequestHeadersV
alve" />
```

Check Secret Password

The secret password used in the Apache HTTP server configuration needs to be specified in the valve configuration also. However, rather than put it in plaintext, it is good practice to encrypt it. Tc Server uses a password-based encryption scheme to encrypt passwords used in the configuration. The same process can be used here for consistency.

To encrypt the plaintext password (Password1), a shell is opened and the following command is executed from the SASHome/SASWebApplicationServer/9.4 directory to get the encrypted value:

Windows:

```
java -cp tomcat-6.0.35.B.RELEASE\lib\tcServer.jar;tomcat-
6.0.35.B.RELEASE\bin\tomcat-juli.jar;tomcat-6.0.35.B.RELEASE\lib\tomcat-
coyote.jar com.springsource.tcserver.security.PropertyDecoder -encode
"This!sTheSpringSourcePassphrase" Password1
```

UNIX:

```
java -cp './tomcat-6.0.35.B.RELEASE/lib/tcServer.jar:./tomcat-
6.0.35.B.RELEASE/bin/tomcat-juli.jar:./tomcat-6.0.35.B.RELEASE/lib/tomcat-
coyote.jar' com.springsource.tcserver.security.PropertyDecoder -encode
'This!sTheSpringSourcePassphrase' Password1
```

Keeping with the convention used elsewhere, rather than include the encrypted value directly in the valve configuration, many of the settings used by the Tomcat server are specified as properties. The following is added to the `catalina.properties` file in the configuration area under `Web/WebAppServer/SASServer1_1/conf/`:

```
pw.sas.valve.PrincipalFromRequestHeadersValve=s2enc://OZBRllprnGj4J/YPqnj5
Bw==
```

The secret password property is then added to the valve configuration as a parameter. This is shown in bold font below:

```
<Valve
className="com.sas.vfabrictcsvr.authenticator.PrincipalFromRequestHeadersV
alve" secretPassword="${pw.sas.valve.PrincipalFromRequestHeadersValve}" />
```

Table 2 shows all the options supported by the valve:

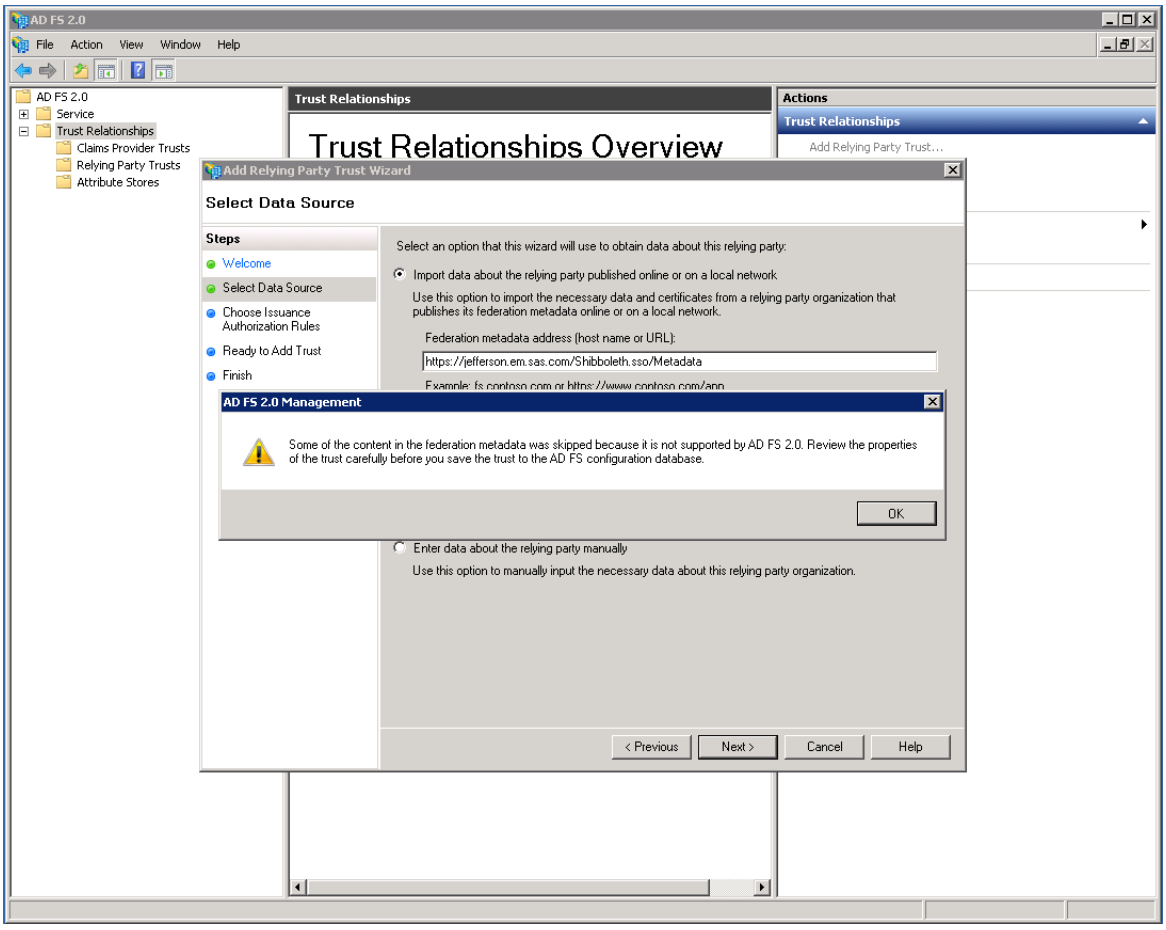
Option	Default	Description
uriPattern	/SASLogon/login.*$	URI regular expression to process requests. Includes the query string.
fallThrough	false	Controls flow upon unsuccessful authentication. If true, control passes to the next valve in the pipeline; if false, a 401 error code is returned.
secretPassword		Secret password to expect in the Basic Authorization header (optional).
userHeader	X-Remote-User	HTTP header containing the authenticated principal name.
roleName	ROLE_USER	Role to associate with the principal. Normally not used.

Table 2. PrincipalFromRequestHeadersValve Options

SAS Web Application Server must be restarted after making these changes.

MICROSOFT AD FS 2.0 CONFIGURATION

On the AD FS 2.0 server we need to add information about the Shibboleth service provider and configure what claims should be sent in the assertions. This is done from the AD FS 2.0 Management Console. Under Trust Relationships, we right-click on Relying Party Trusts and select Add Relying Party Trust to bring up the wizard. After entering the URL for the Shibboleth SP Metadata, AD FS displays a warning indicating that some of the content in the Metadata is not supported. This is shown in Display 1.

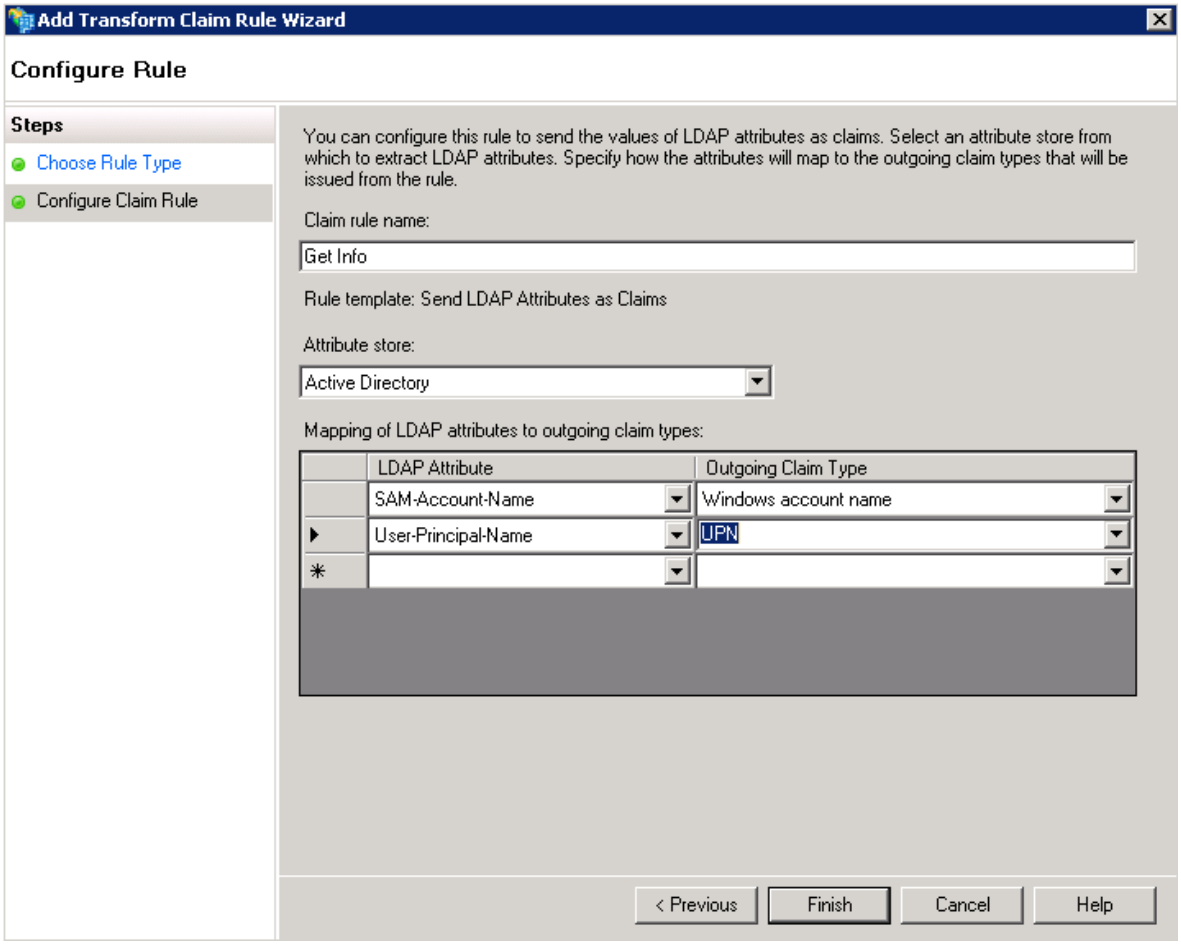


Display 1. Adding Relying Party Trust in AD FS 2.0 Management Console

We give the relying party a display name and check the box to 'Permit all users to access this relying party'. After adding the relying party, a window pops up to edit the claim rules for the relying party. Claim rules control what information will be passed to Shibboleth in the SAML assertion. This can be brought up

later by right clicking on the Relying party to the right of the Relying Party Trusts and selecting Edit Claim Rules.

From within the claim rules, the Issuance Transform Rules tab is selected and we click on Add Rule to bring up another wizard. A list of options appears. Most administrators will choose ‘Send LDAP Attributes as Claims’. On the next page we select Active Directory for the attribute store and choose which attributes to include in the assertions. The LDAP Attribute column displays the name of the attribute in the LDAP attribute store and the Outgoing Claim Type column displays the SAML attribute that should carry the value. SAM-Account-Name would normally be mapped to Windows account name, User-Principal-Name would map to UPN, and so on. This is shown in Display 2. Multiple claims can be sent in the assertion for flexibility and we can decide which one to use when completing the Shibboleth configuration.



Display 2. Editing Claim Rules for the Relying Party in AD FS 2.0 Management Console

There are a couple of important URLs that are need to be noted from the ADFS Management console. These will be required later when we complete the Shibboleth configuration. Under “AD FS 2.0” in the Management console, under Service and then Endpoints, the URL to the Metadata should be shown in the list of endpoints. It will be something like https://host.company.com/FederationMetadata/2007-06/FederationMetadata.xml. After expanding Service and clicking on Claim Descriptions, we can find the claim that was selected in the last step for the outgoing claim (for example UPN) and copy the Claim Type, which is in the form of a URI. A double click on the row in the table will bring it up in a separate window which is easier to copy from.

COMPLETE SHIBBOLETH CONFIGURATION

Shibboleth must be told about the AD FS 2.0 identity provider. Inside the `shibboleth2.xml` file we update the entityID attribute of the SSO element to match the entityID in the AD FS Metadata. The AD FS Metadata can be viewed using a web browser.

```
<SSO entityID="http://host.company.com/adfs/services/trust">
     SAML2 SAML1
</SSO>
```

There are examples in the file showing how the identity provider Metadata can be provided. The following loads it from the URL determined earlier and specifies a local filename to store the Metadata in between reloads:

```
<MetadataProvider type="XML" uri=
"https://host.company.com/FederationMetadata/2007-
06/FederationMetadata.xml" backingFilePath="FederationMetadata.xml"
reloadInterval="180000"/>
```

There are situations where modifications need to be made to the Metadata. If this is the case, it can be downloaded using a web browser and specified as follows:

```
<MetadataProvider type="XML" file="FederationMetadata.xml"/>
```

Attribute Scoping

Attributes passed in the SAML 2.0 assertion are usually scoped with a '@' character. Scoping makes it clear which institution or enterprise in the federation the attribute belongs to. By default Shibboleth will verify that the scope in each attribute matches the scope specified in the identity provider Metadata. This is a security feature when using multiple identity providers that prevents one identity provider of asserting claims in another identity provider's scope. Unfortunately AD FS does not add a scope to attributes automatically, but there are a few workarounds. The User-Principal-Name in Active Directory will be scoped with the Active Directory Realm so this can be used. Shibboleth can also be configured to not expect attributes to be scoped. This paper will cover both cases.

Attributes in the assertions passed from the identity provider are *mapped* in the Shibbooth `attribute-map.xml` file.

Scoped Attributes

By default Shibboleth will validate scoped attributes. This includes attributes from Active Directory such as User-Principal-Name and E-mail address that are naturally scoped. In the AD FS management console we set up the claim to pass in the assertion and made note of the URI. This needs to be put in the `attribute-map.xml` file. The example below is for User Principal Name:

```
<Attribute
name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn" id="eppn"
>
   <AttributeDecoder xsi:type="ScopedAttributeDecoder"
caseSensitive="false" />
</Attribute>
```

There is an extra step required since AD FS does not automatically add the scope (it is part of the value itself). The AD FS Metadata must be downloaded and manually edited to explicitly state that scoped attributes will be passed. Specific instructions are in the Microsoft guide under 'Add the Scope Element to AD FS 2.0 Metadata'.

Un-scoped Attributes

For un-scoped attributes we add the same thing to attribute-map.xml but specify a StringAttributeDecoder instead of a ScopedAttrributeDecoder. The example below is for Windows Account Name (SAM-Account-Name):

```
<Attribute
name="http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccou
ntname" id="eppn" >
   <AttributeDecoder xsi:type="StringAttributeDecoder"
caseSensitive="false"/>
</Attribute>
```

Shibboleth requires scopes by default for some attributes, including eppn. These are specified in the `attribute-policy.xml` file. For eppn it is changed to permit ANY (not just scoped) values:

```
<afp:AttributeRule attributeID="eppn">
    <afp:PermitValueRule xsi:type="ANY"/>
</afp:AttributeRule>
```

SAS Web Server and the Shibboleth daemon must be restarted after making changes to the Shibboleth configuration files.

TROUBLESHOOTING

When everything is setup correctly, a user attempting to access a SAS web application will be redirected to the AD FS identity provider to log in, and back to the SAS web application without seeing the SAS login page. However, whenever configuring Shibboleth for a new identity provider, there is usually some amount of troubleshooting required. The most common problem is figuring out what part of the assertion should be used for the authenticated user name. Some identity providers include a Subject NameId element while others simply send attribute claims and the service provider is left to decide which one to use for the user name. Even with the NameID, there are multiple formats.

VIEWING THE ASSERTIONS

It is very difficult to tell what is going on without seeing the actual XML being sent in the SAML request and response. Most modern web browsers have a way of displaying the HTTP traffic. In Internet Explorer 10 and later, F12 Developer Tools is available from the Options menu. On the Network tab is a button to Start Capturing. Navigating to any of the SAS web applications should results in a number of redirects and finally end up on the identity provider login, which is an IIS server by default. For SAML 2.0 Redirect binding, the SAML request XML will be compressed and encoded into the query string with the SAMLRequest parameter. The value must be URL decoded and then Base64 Decoded and inflated. Several online sites exist to do this. Output 2 shows an XML request.

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="https://jefferson.em.sas.com/Shibboleth.sso/SA
ML2/POST" Destination="https://washington.em.sas.com/saml2"
ID="_ffb034169b63d5b062a0b358a9ad42cf" IssueInstant="2015-01-10T17:21:49Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Version="2.0"><saml:Issuer
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">https://jefferson.na.sas
.com/shibboleth</saml:Issuer><samlp:NameIDPolicy
AllowCreate="1"/></samlp:AuthnRequest>
```

Output 1. SAML request

AD FS 2.0 encrypts assertions it sends to service providers by default. This can be disabled on the AD FS server by running the Administrative Tools from the Start menu and bringing up the Windows PowerShell Modules. At the Windows PowerShell command prompt, the following command will disable

encryption. The TargetName must match the name of the relying party as shown in the AD FS management console earlier when adding relying party trust:

```
set-ADFSRelyingPartyTrust –TargetName "relying-party" –EncryptClaims
$False
```

After authenticating at the identity provider, the web browser is sent back to the service provider with a POST request to /Shibboeth.sso/SAML2/POST. In Internet Explorer a double click on that request and viewing the Request Body tab reveals the POST content sent in the request. The content is Base64-encoded. Using any online Base64 decoders, the XML can be viewed. Output 2 shows a snippet of the response.

```
<samlp:Response...
      <AttributeStatement>
         <Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn">
            <AttributeValue>miroda@em.sas.com</AttributeValue>
         </Attribute>
      </AttributeStatement>
   ...</Assertion>
</samlp:Response>
```

Output 2. SAML response showing the attribute containing the UPN claim

After debugging the assertions, encryption can be re-enabled on the AD FS 2.0 server from the Windows PowerShell command prompt:

```
set-ADFSRelyingPartyTrust –TargetName "relying-party" –EncryptClaims
$True
```

LOG FILES

Shibboleth creates several log files under `/opt/shibboleth-sp/var/log/shibboleth` (same path on Windows). The `shibd.log` will contain messages indicating when attributes could not be mapped successfully. For example, Output 3 shows an attribute from the assertion that was not mapped.

```
2015-01-09 16:00:33 INFO Shibboleth.AttributeExtractor.XML [1]: skipping
unmapped SAML 2.0 Attribute with Name: email,
Format:urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified
```

Output 3. Shibd.log entry showing unmapped assertion

Conversely, the `transaction.log` will output messages indicating which attributes were successfully mapped. Output 4 shows that our eppn id was successfully mapped.

```
2015-01-09 16:00:33 INFO Shibboleth-TRANSACTION [1]: New session (ID:
_a15736ee46882ab09a87a7272eefa1f7) with (applicationId: default) for
principal from (IdP: http://host.company.com/adfs/services/trust) at
(ClientAddress: 10.23.13.155)
2015-01-09 16:00:33 INFO Shibboleth-TRANSACTION [1]: Cached the following
attributes with session (ID: _a15736ee46882ab09a87a7272eefa1f7) for
(applicationId: default) {
2015-01-09 16:00:33 INFO Shibboleth-TRANSACTION [1]:      eppn (1 values)
2015-01-09 16:00:33 INFO Shibboleth-TRANSACTION [1]: }
```

Output 4. Transaction log shows attribute mapped to eppn

When Shibboleth is properly parsing the attributes and mapping one of them to the Apache REMOTE_USER variable, the user name can be viewed in the SAS Web Server access logs. This happens by default for HTTP requests. HTTPS requests go to the ssl_request logs and are configured in `httpd-ssl.conf`. The user name can be output by adding a %u to the log directive near the end of the file. Output 5 shows the user name in the HTTP access log:

```
fe80::4dee:d4b2:cfbe:f1d6 - - [11/Jul/2014:10:06:31 -0400] "GET
/SASLogon/login HTTP/1.1" 302 774
fe80::4dee:d4b2:cfbe:f1d6 - - [11/Jul/2014:10:07:41 -0400] "GET
/SASLogon/login HTTP/1.1" 302 778
fe80::4dee:d4b2:cfbe:f1d6 - - [11/Jul/2014:10:07:48 -0400] "POST
/Shibboleth.sso/SAML2/POST HTTP/1.1" 302 227
fe80::4dee:d4b2:cfbe:f1d6 – miroda@em.sas.com [11/Jul/2014:10:07:48 -0400]
"GET /SASLogon/login HTTP/1.1" 200 2730
```

Output 5. User name displayed in the SAS Web Server access log

Finally, if the principal is being received by the Tomcat valve in SAS Web Application Server and set in the request, the user name should be visible in the localhost access log under SASServer1_1/logs. Output 6 shows the user name appearing in the web application server logs.

```
10.121.19.81 – miroda@em.sas.com [09/Jan/2015:15:43:07 -0500] "GET
/SASLogon/login HTTP/1.1" 200 2787
```

Output 6. User name displayed in the SAS Web Application Server access log

CONCLUSION

Federated identity and multi-domain single sign-on, the two most important use cases or profiles described by the SAML 2.0 standard, make it possible for organizations in entirely separate security domains to collaborate with applications like SAS.

We have discussed what federated identity is and taken a technical overview of the layers that make up the SAML 2.0 standard. We have looked at how the SAS 9.4 middle tier can support a wide variety of authentication methods using widely available Apache pluggable modules with SAS Web Server and a Tomcat valve in SAS Web Application Server. And we introduced the Shibboleth SP project that provides an Apache module for SAML-based authentication.

Then we went through the steps necessary to configure an end-to-end solution with SAS as the relying party and Microsoft AD FS as the asserting party or identity provider. Finally we discussed how to view the assertions and what to look for in the various log files for troubleshooting purposes.

REFERENCES

Organization for the Advancement of Structured Information Standards. “Security Assertion Markup Language (SAML) v2.0”. March 2005. Available at https://www.oasis-open.org/standards#samlv2.0.

Organization for the Advancement of Structured Information Standards. “SAML Wiki Knowledgebase.” Accessed Jan 2015. Available at http://saml.xml.org/wiki/saml-wiki-knowledgebase.

Microsoft. “AD FS 2.0 Step-by-Step Guide: Federation with Shibboleth 2 and the InCommon Federation”. Accessed Jan 2015. Available at http://technet.microsoft.com/en-us/library/gg317734(WS.10).aspx.

Shibboleth. “Understanding Shibboleth.” Accessed Jan 2015. Available at https://wiki.shibboleth.net/confluence/display/SHIB2/UnderstandingShibboleth.

Shibboleth. “NativeSPConfiguration.” Accessed Jan 2015. Available at https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPConfiguration.

SAS Institute Inc. (2014). “SAS 9.4 Intelligence Platform: Middle-Tier Administration Guide, Second Edition.” Available at http://support.sas.com/94administration.

Server Fault. “Get the authenticated user under apache”. Accessed Jan 2015. Available at http://serverfault.com/questions/207301/get-the-authenticated-user-under-apache.

ACKNOWLEDGMENTS

I would like to thank the following people for taking the time to review and contribute to this paper:

- Phil Hopkins
- Stuart Rogers
- Adam Smith

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mike Roda
SAS Institute Inc.
919-362-9707
mike.roda@sas.com