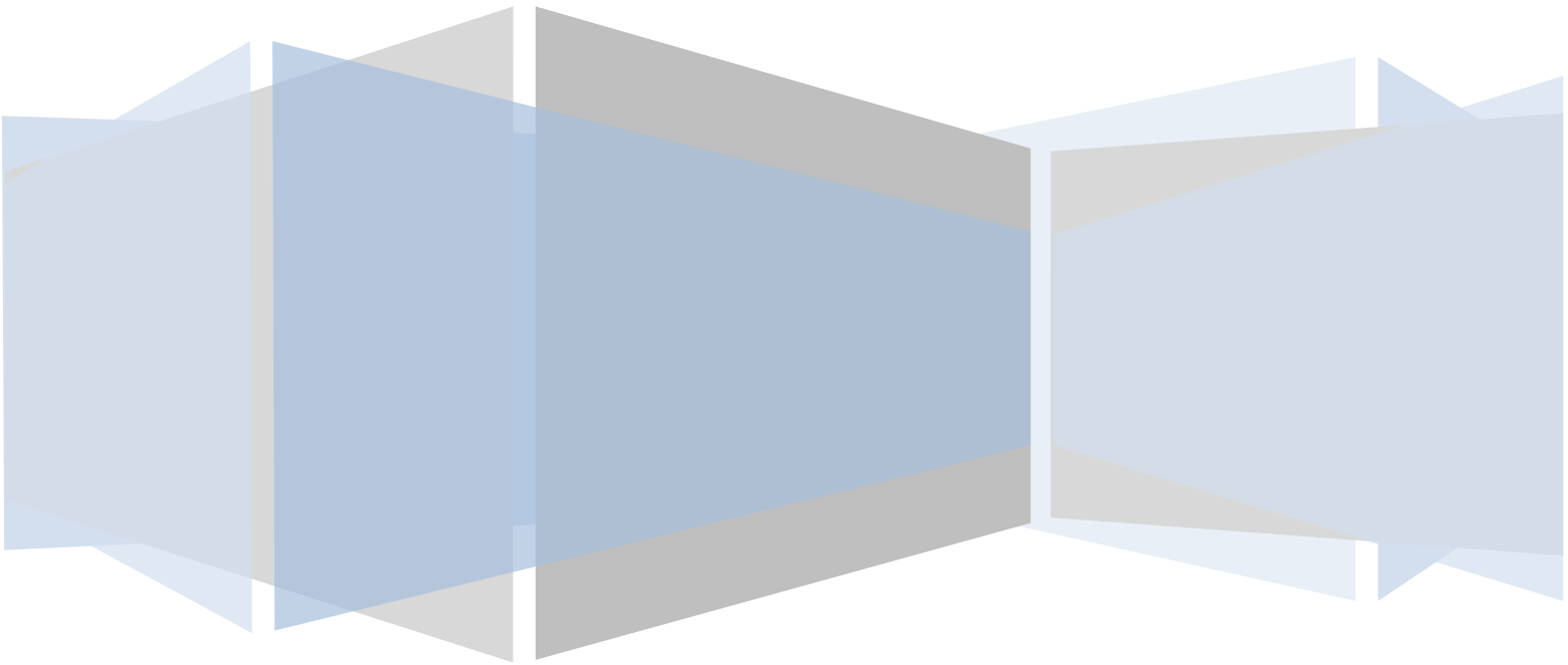
SSO Plugin

Configuration and integration with SSO systems

J System Solutions

http://www.javasystemsolutions.com

Version 5.1



Introduction

This document covers the configuration of SSO Plugin for BMC Mid-Tier, HP Web Tier and products supported by the SSO Plugin authentication service.

The JSS support website contains documents for other components and there are also many videos to assist with installing the SSO Plugin.

Terminology

Throughout the document, the following key terms are used:

- **Third party product** refers to BMC Mid Tier or HP Service Manager.
- **Java web server** refers to the product running the product, ie Apache Tomcat, JBoss, Oracle Weblogic, Websphere, etc.

Java version support

The product is designed to support versions of the Java Virtual Machine from 1.8+ and we recommend you use the latest version available from the Oracle website, compatible with the third party product. We'd prefer you used Java 11.

We currently test on OpenJDK 11

BMC AR System

There are no further patches required to the BMC Mid Tier when running with an IBM Java.

HP Service Manager

Websphere 8+ is supported by SSO Plugin. HP supply an ear file distribution of Web Tier but this packaging isn't necessary and a war file can be deployed.

When using the IBM JVM (ie with Websphere) and HP Service Manager Web Tier, the jars from the ibm-jre directory must be placed into the Web Tier WEB-INF/lib directory prior to deployment.

IBM Websphere and Oracle Weblogic

The SSO Plugin automatic web.xml patching tool does not work within Websphere so the application must be setup in a standalone environment. Therefore, prior to deploying a patched war file to Websphere, we recommend that you deploy to temporary Tomcat instance and configure SSO Plugin. This ensures the web.xml file is correctly patched (by SSO Plugin), and it is much easier to deploy and test applications within a Tomcat environment.

If in doubt, contact JSS for assistance.

Java web server support

The product is designed to work across all modern Java web servers including Apache Tomcat and "Java application servers" such as Oracle Weblogic, IBM Websphere, RedHat JBoss, etc.

The product requires write access to a configuration file jss-ssoplugin.properties on all deployments apart from BMC Mid Tier.

Reading and writing to local files can be problematic on all but Apache Tomcat when the application is deployed from a war file, because the Java application servers typically deploy the application to a temporary directory each time it is started. Hence, if SSO Plugin updates a local file, the changes are lost on the next restart.

Therefore, when using a Java application server, ensure the application is deployed in an exploded fashion and will not be copied elsewhere during deployment - consult the application server documentation for more details on the deployment process.

It is also possible to set the JVM environment variable jss-ssoplugin.properties to a path that points at a copy of the configuration file that is in a safe location, such as inside the web server root folder, and manually patch the Java application, ie the contents of the war file. This will allow a fully patched war file to be deployed with the only local file dependency being the jss-ssoplugin.properties file.

Given the range of application integrations supported by SSO Plugin, please consult JSS for further advice.

Authentication service

The authentication service is a generic version of SSO Plugin that can be used to provide SSO to a wide range of web applications (such as HP Asset Manager and Kinetic Request). The authentication service installation document covers the installation steps for each supported product but the configuration of SSO Plugin is covered in this document.

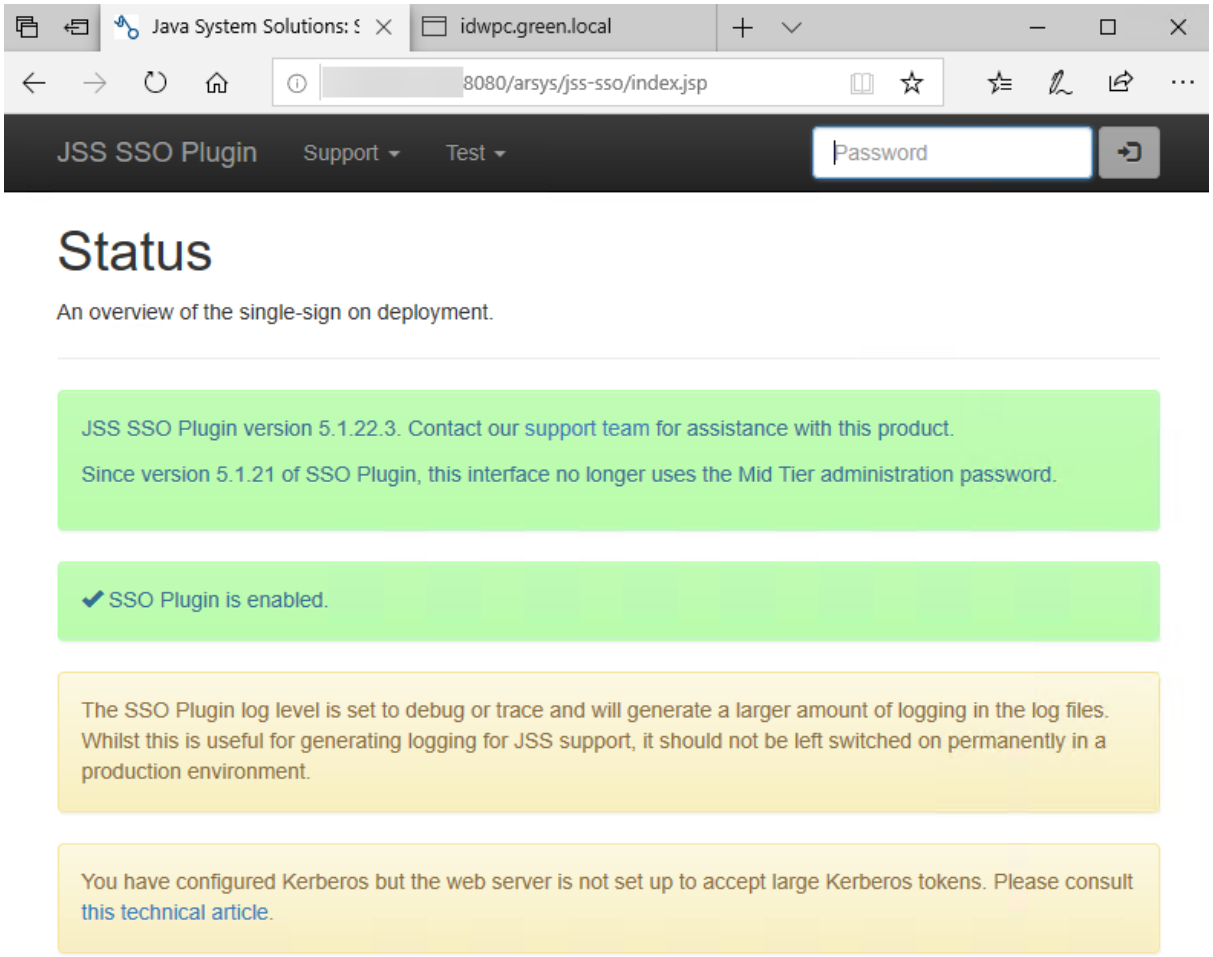
Whilst this document contains a lot of BMC Mid Tier / MyIT and HP Web Tier specific documentation, the configuration interface and the authentication methods sections contain relevant configuration information.

Accessing web interface

After the installation files have been copied to the web component (see the relevant product installation guide) and the web server (ie Tomcat) has been restarted, you should be able to access the web interface at the following URL:

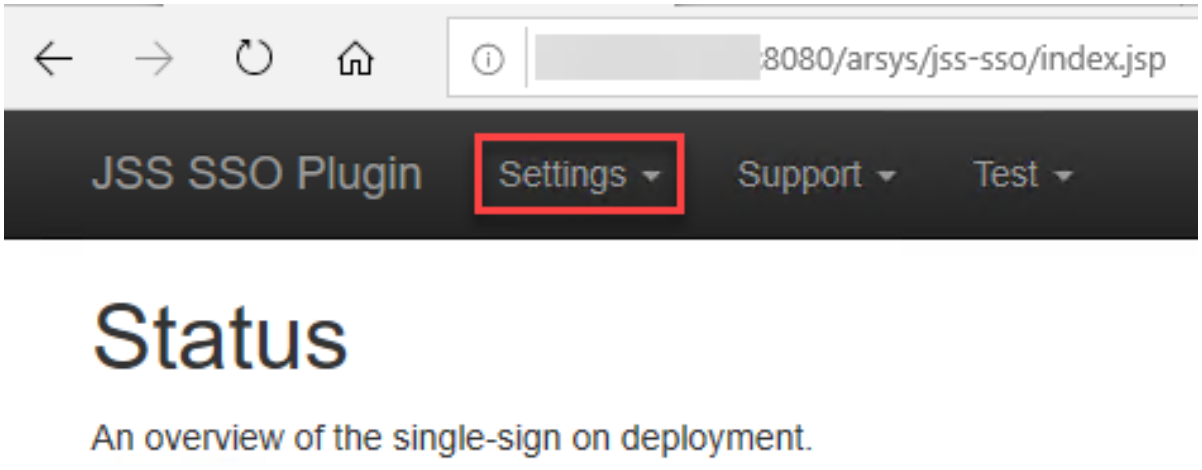
- BMC Mid Tier: http://host/arsys/jss-sso/index.jsp
- BMC My IT: http://host/ux/jss-sso/index.jsp
- HP Web Tier: http://host/webtier/jss-sso/index.jsp
- Authentication service: The URL is present in the relevant section of the authentication service installation document.

If you can not gain access to the SSO status page below, please consult JSS support for advice.



Logging into the web interface

The SSO interface is protected by a password and a default value of jss. Since version 5.1.21 of SSO Plugin, this interface no longer uses the Mid Tier administration password. Once logged in, the configuration link is available allowing the product to be configured.



Configuration interface

The configuration interface provides a range of features to configure SSO Plugin in the web component, including configuring the SSO integration, mapping the SSO username to the product username, actions to perform when a user has no SSO enabled account in the product, and setting the log level. Each section is described in detail below.

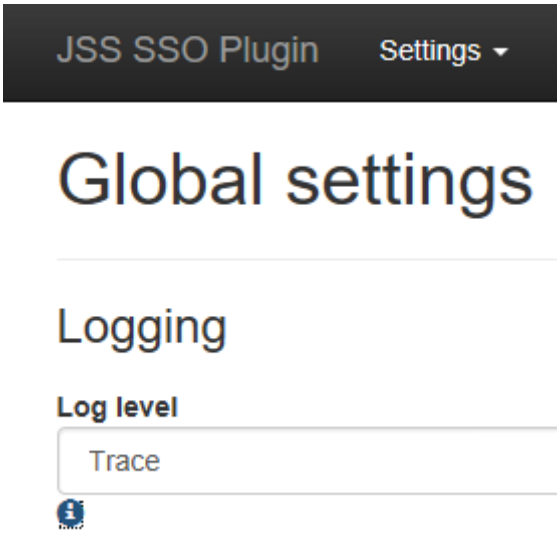
Logging

This controls the SSO Plugin log level which writes to the Java webserver standard out log file, which on Tomcat is called tomcat_stdout.log (Windows) or catalina.out (Unix).

We recommend you select information or 'warnings only' for production use, debugging when configuring the SSO Plugin, and trace when you're trying to resolve an issue with the help of JSS.

Trace will generate a lot of low level logging and is required by JSS to resolve issues. It is not suitable for production use.

Settings > Global > Logging



License

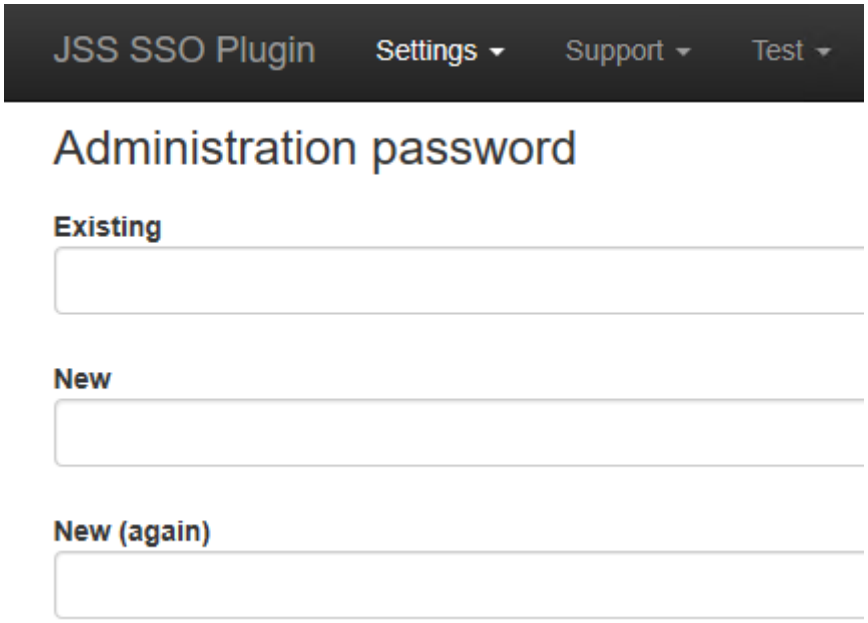
If you require a trial license, please contact JSS support.

If you've purchased an SSO Plugin license, you can login to the JSS website and generate one.

Administrative password

The SSO Plugin administration password can be changed and we recommend you do so. The default password is jss.

Settings > Global > Administration Password



Connecting to HP Service Manager

For HP Web Tier deployments, a Service Manager administrator user account is required. The product defaults to falcon with no password.

The URL of the Service Manager server to be used for webservice calls can also be set.

If self-service users should be re-directed to the Service Request Catalog product, the URL of SRC can be provided.

Identity Federation Service

This feature allows SSO Plugin to be configured in third party products that use an existing SSO Plugin instance for authenticating users. This is documented in the relevant third party integration documents, such as enabling SSO for BMC Dashboards and Analytics.

BMC Multi Service Providers

When configuring multiple SSO integrations, they can be mapped to a hostname and a JSS cookie value. The MSP page allows the user to select from a set of values to populate the cookie. The administrator can select the form to query and the field to retrieve when populating the MSP page.

Anti-idle feature

The anti-idle feature injects Javascript into the page and can prevent the user's session timing out, or presents the user with a warning (a clock) as their session is due to expire.

This feature can be tested by setting the application session timeout to 2 minutes, enabling the anti-idle feature to 'prompt', logging in and waiting 90 seconds. If the user acknowledges the anti-idle notification, the browser will make a call to the server and reset the session timeout.

Integration methods

There are a number of ways to integrate the product into your network and each are described in detail in this document, however an overview follows:

1. Windows Authentication (IWA) without IIS. When using a load balancer, please consult this technical article.
2. Windows Authentication performed by IIS. For security and simplicity, we recommend Windows Authentication without IIS wherever possible.
3. Using a third party authentication system that provides the username as an HTTP header or cookie.
4. Native CA SiteMinder, RSA ClearTrust or RSA SecurID integrations, allowing SSO Plugin to connect directly to the third party authentication service.
5. OAuth2, for integrations with Facebook, Google and others.
6. Generic REMOTE_USER or a JAAS pluign.
7. X509 client certificates, also used for DoD CAC.
8. SAMLv2 for integration with Microsoft ADFS2, Ping Federate, Symphony Identity Federation Manager and other third party products.

When you set the SSO Plugin configuration, it will update the product web.xml file to add the instructions that allow SSO Plugin to secure the product.

In rare cases where this fails, the web.xml.patch file (located in the product WEB-INF directory) can be manually applied to the web.xml file.

Please note, the Java web server may need restarting once SSO Plugin has patched the web.xml file and this will be communicated to the user.

Windows Authentication

The product contains an implementation of the Microsoft Windows Authentication protocol. This allows users to open a browser, commonly IE, and navigate to the product without being prompted to login.

There are two parts of the Windows Authentication protocol, NTLM (required) and Kerberos (optional).

If you are using the Apache Tomcat webserver and configure the Kerberos mechanism, you must enable large SSO tokens.

Rapid configuration on a Windows server

The product can make use of the internal Windows computer account to process NTLM tokens, so if the Java web server is running on a Windows server that is a **member of a Windows domain**, follow these steps:

1. Select Windows Authentication.
2. Select Permit NTLM.
3. Select the Windows native radio button.
4. Press set configuration and test using the Test SSO page.

If the browser does not perform SSO, check the browser is configured correctly.

Configuring native NTLM

This option is only available when the Java web server is running on a Windows server operating system. It makes use of the local Windows computer account to validate NTLM tokens, but domain and local policies must be configured to allow this process. Typically, the default policies allow this feature to work but if it fails, check the following:

1. The Network Security: Restrict NTLM: NTLM authentication in this domain policy setting, which is used to deny or allow NTLM authentication to a domain controller.
2. The group policies Deny access to this computer from the network and Access this computer from the network. Users must have access to the computer for the NTLM tokens to be processed. A typical mis-understanding is to deny domain users access to the computer, when denying interactive logins would be sufficient.

There is however one caveat when using this authentication method: native NTLM uses native memory, hence the data used to authenticate clients is not serializable. This means that you cannot use this authentication method with Java webserver session replication.

Configuring NTLM

The NTLM mechanism is the core part of the Windows Authentication protocol. We highly recommend it is configured. On a Windows server, you can use the Windows native option. On other operating systems, you require a **computer** service account in the Active Directory used by SSO Plugin to validate the NTLM tokens.

You are required to enter the following information:

1. Fully qualified DC hostname: This fully qualified hostname of the Active Directory/Domain Controller. You can provide a comma separated list of AD hostnames for failover support.
2. Windows DNS Domain: This is the fully qualified name of the Windows domain.
3. Computer account name and password of the computer created from the script.

There are a number of advanced options when enabling NTLM:

- NETBIOS Domain name: Some Windows networks are configured with NETBIOS Domain names that are not related to the Windows Domain name, ie your Windows Domain may be called ADMIT and the DNS domain may be devit.mycompany.com. If this is the case, set the NETBIOS Domain name in this field.

  This is an unusual configuration because most networks are configured with a NETBIOS Domain name that is a subset of the Windows Domain name, ie NETBIOS Domain of ADMIT and a Windows Domain name of admit.mycompany.com.
- Fixed DC IP: In the unlikely event that the hostname of the Domain Controller does not resolve to an IP address, enter the IP address.
- DC fail timeout (seconds): If there is a connection failure to the DC, it is marked as failed for the period of time defined in this field. The default, 30 seconds, should suffice but the option is configurable.

Discovering the hostname of your Domain Controller

Open a command prompt and type:

```
echo %LOGONSERVER%
```

It will report the name with two backslashes, ie \\server2k3. You do not need the backslashes. The **fully qualified** hostname of the DC must be added, so look it up using nslookup:

```
nslookup adhostname
```

Discovering the Windows DNS Domain

Open a command prompt and type:

```
net config workstation
```

The fully qualified domain name is printed by Windows Domain DNS Name.

In some cases, the fully qualified domain name is not the same as the Windows Domain DNS name. The value should start with the Windows Domain name, and if it does not, remove the string before this value.

For example, if the Windows Domain DNS Name is ds.jss.com, and the Windows Domain is called JSS, the correct value for SSO Plugin is jss.com.

Creating service accounts using a script

The script is only required if your vendor product is **not** installed on a Windows server and is not in the same domain your users will be authenticating against. Therefore the script is needed if your vendor product is installed on Linux or UNIX. It is called set-service-account.cmd and is included in the installation files. Copy it to your Active Directory, run it, and you can almost certainly accept the default options. It will create a computer called JSS-SSO-SERVICE – **note down the password it generates!** Accept the default option (no) for "Do not require Kerberos pre-authentication" unless otherwise instructed by JSS.

The script also asks you for the hostnames on which a user will browse to the product (typically when running behind a load balancer) – please provide both the hostname and the fully qualified hostname, ie jss-sso-dev and jss-sso-dev.mycorp.com.

If you do not wish to run our script, refer to the manually creating a computer account section.

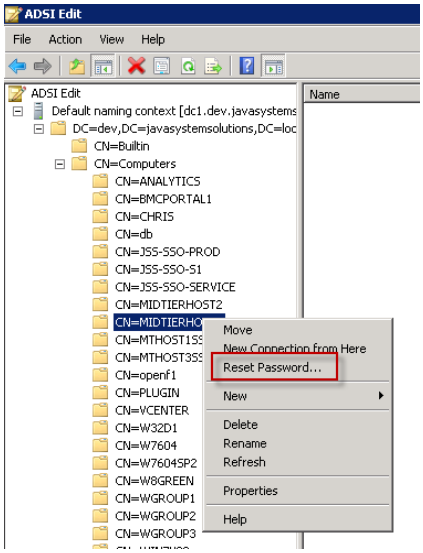
You must create a separate computer service account for each deployment of SSO Plugin - one for each JVM if multiple Java web servers are configured on a single server. Name them appropriately, ie JSS-SSO-DEV, JSS-SSO-UAT, JSS-SSO-PRD1, JSS-SSO-PRD2, etc.

NTLM will not work if the same computer account is configured on two different JVMs, because it will cause SSO Plugin to fail when there are many concurrent users trying to access the product.

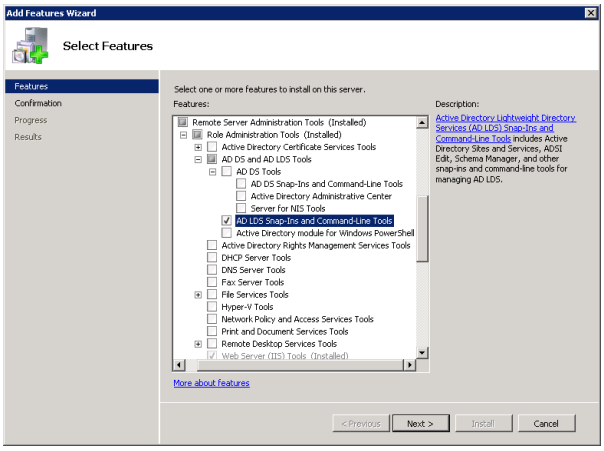
Manually creating a computer account

If you don't want to run our set-service-account.cmd script then you will need to configure the Active Directory service account manually. Please follow these steps:

1. Using the Active Directory Users and Computes tool, create a new computer object called JSS-SSO-SERVICE. If you choose any other name, it must be in upper case and no more than 15 characters in length.
2. The Active Directory Users and Computes tool provides no way to set the password on the computer account.
    1. Use the ADSI Edit tool to locate the Computer and set the password as shown in this screenshot:



If ADSI Edit is not installed, the following screenshot shows how to add the 'Windows feature' that provides it:



2. A small script can be used to set the password. Please edit the LDAP path appropriately and run from a command prompt as an administrator on the domain controller:

```
GetObject('LDAP://CN=JSS-SSO-
SERVICE,CN=Computers,DC=development,DC=jss,DC=com').setPassword('new-
password'); > temp.js
cscript //E:jscript temp.js
del temp.js
```

If you do not know the LDAP path of the computer account, the dsquery command can be used to discover it:

```
dsquery computer -name JSS-SSO-SERVICE
```

Supporting multiple Windows domains

If your Windows domains are in a trusted relationship then you only need configure the SSO Plugin to authenticate against one of the domains. The Domain Controller should be able to authenticate users connecting from any other domain where that other domain is trusted.

If the domains are in an untrusted relationship then we recommend you configure Kerberos and contact JSS support for advice.

Configuring Kerberos

There are a number of ways in which Kerberos can be configured. The recommended method is acceptor mode, where you are required to provide the username and password of the service account configured with the Service Principal Names.

To create a service account, a script is provided to do this for you or you can manually create the account.

The more complicated initiator configuration requires two separate elements to the configuration, each of which can be configured in two ways, providing four possible ways to configure the product:

1. A mechanism of authenticating with the KDC. This is configured by providing service account credentials, or a keytab file created using the ktpass program.
2. The location of the Kerberos Domain Controller (KDC). This is configured by providing the hostname of the KDC and a Kerberos realm, or by configuring a krb5.conf file
    1. When using an IBM JDK, you must set up the krb5.conf file.
    2. An advantages of using a krb5.conf file is the encryption types can be specifically set, removing the potential for the KDC and SSO Plugin using differing types.

Manually creating a service account

If you are creating a user account for (only) Kerberos authentication, the account does not need to be trusted for delegation when using the SSO Plugin out of the box 'acceptor' configuration. The 'initiator' configuration does require delegation but this should be not be used unless advice is sought from JSS support.

For SSO Plugin to be able to authenticate clients using Kerberos, an SPN must be configured on the Domain Controller. The setspn.exe tool is used by the administrators to create an SPN which maps the Java web server host(s) to a service account in the Active Directory.

To find out the fully qualified hostname of the Active Directory, ping it from the command prompt (you will see the hostname and fully qualified hostname).

We assume that:

- The Windows domain is called BLUE.
- The domain's fully qualified name is blue.jss.com.
- The Java web server is running on a machine with the hostname itsm.jss.com.
- The service account username is JSS-SSO-KRB.

Here is an example of how to use setspn – you must add both the hostname and the fully qualified hostname of the Java web server:

```
setspn.exe -s HTTP/itsm.jss.com BLUE\JSS-SSO-KRB
setspn.exe -s HTTP/itsm BLUE\JSS-SSO-KRB
```

You can check to see if the SPN has been added by using the -L option, which lists the SPNs for a computer or user account:

```
setspn.exe -L BLUE\JSS-SSO-KRB
```

Please note, a hostname should only ever be declared against one user account – to declare it against multiple users will confuse Active Directory.

A recent version of the setpn command includes an option to list duplicate SPNs, which can be useful in resolving issues when browsers will not authenticate using Kerberos:

```
setspn.exe -X
```

Supporting multiple domains

If the Kerberos Domain Controllers are in a trusted relationship then the KDC for domain A should be able to authenticate users for domain B, and vice versa, so the krb5.conf isn't required unless there's a need to make use of advanced Kerberos configuration options.

If the KDCs are in an untrusted relationship then there are two ways to ways to configure SSO Plugin:

1. Create a service account in each domain with the **same username and password** and configure SSO Plugin in acceptor mode with the credentials. This is the recommended and easiest configuration.
2. Use a krb5.conf file to configure each domain, and use a keytab so SSO Plugin can authenticate with each domain.

An example krb5.conf file (krb5.conf.example) has been provided in the WEB-INF/classes directory of the files copied to the Java web server. Make a copy of this file, call it krb5.conf, and modify the copy so you do not lose the file during an upgrade.

Creating a keytab

Organisations that do not wish to store service account credentials with the SSO Plugin configuration can use a keytab. Keytabs are created with the ktpass program, and plenty of examples are available on the Internet, however it is briefly covered below.

Following on from the SPN example above, a keytab can then be created as follows:

```
ktpass -princ HTTP/itsm.jss.com@BLUE.JSS.COM  -out JSS-SSO-KRB.keytab -
mapuser BLUE\JSS-SSO-KRB -pass service_account_password -ptype
KRB5_NT_PRINCIPAL -crypto RC4-HMAC-NT
```

(Note, the realm – BLUE.JSS.COM - has to be in upper case.)

Using the above configuration, you would store the keytab in the product (we recommend under WEB-INF) and configure the SSO Plugin by providing:

- The full path to the keytab.
- The service principal name, which is HTTP/itsm.jss.com@BLUE.JSS.COM

Enabling Windows Authentication in IE and Firefox

In order to use SSO, your browser must support SSO and you must be logged into the domain. If a logon box, or unauthorised, is presented, please review the following:

1. The client must be using a Windows operating system that is a member of the Windows Domain to which SSO Plugin is configured. To ensure this is the case, press ctrl+alt+del and look at the 'You are logged in as' dialog - the Windows Domain shown must be the target domain and not the local machine.
2. If using IE, check the following:
   a. The Java web server hostname must be listed in the 'Local Intranet zone'. To check this, go to Internet Explorer -> Tools -> Internet Options -> Security -> Local Intranet and make sure that the hostname is present in the list.
   b. Automatic login must be enabled. To check this, go to Internet Explorer -> Tools -> Internet Options -> Security -> Custom Level, scroll all the way to the bottom and make sure 'Automatic logon only in Intranet zone' is selected.
3. If using Firefox, additional configuration is required:
   a. Type about:config in the URL bar and 'trusted-uri' in the search field.
   b. You will be presented with network.automatic-ntlm-auth.trusted-uris and network.negotiate-auth.trusted-uris. Type the hostname from the URL into these fields.

c. Type 'delegated' into the search field, locate network.negotiate-auth.delegation-uris and enter the hostname from the URL into this field.

d. Multiple hostnames can be comma separated.

4. If your browser is configured to use a proxy server, the target website may need to be added to the proxy exceptions list as SSO is known to be problematic through some proxies.
5. Ensure the clocks on the workstation and the AD are set correctly. Kerberos authentication can fail if the clocks are skewed.

Allowing large Kerberos tokens in Tomcat

SSO Plugin should provide a warning on the status page if any action is required with respect to Kerberos tokens.

Kerberos tokens are sent by the browser. By default, Tomcat has a hard coded limit of 4Kb for an HTTP header, and if the Kerberos token exceeds 4Kb then Tomcat returns status code 400 without passing the request to the product. The standard BMC Tomcat distribution has been known to have 8Kb set, which is inadequate.

Open the Tomcat server.xml file (in the conf directory) and look for the HTTP connector:

```
<Connector port="8080" protocol="HTTP/1.1"
```

and add a maxHttpHeaderSize attribute, which is given a value in bytes (6500 is almost 64Kb):

```
<Connector port="8080" protocol="HTTP/1.1" maxHttpHeaderSize="65000"
```

Restart Tomcat and check the product still works as expected.

Windows and AES 256 bit encryption

SSO Plugin should provide a warning on the status page if any action is required with respect to encryption.

Without a patch from Oracle, due to US export rules, the standard Java Virtual Machine does not support 256bit encryption and cannot decode AES 256bit tokens. AES 256bit tokens are often generated by IE when using a Windows 2008 Domain Controller and a Windows 7+ client.

To enable AES 256bit support, you need to download and install the Oracle "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files", currently available from http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html.

Installing the patch is easy: You unzip two jar files and place them in the JRE lib/security directory.

For your information, the Java Security documentation includes the following explanation:

"The JCE framework within JDK includes an ability to enforce restrictions regarding the cryptographic algorithms and maximum cryptographic strengths available to applications. Such restrictions are specified in "jurisdiction policy files". The jurisdiction policy files bundled in Java SE limits the maximum key length. Hence, in order to use AES256 encryption type, you will need to install the JCE crypto policy with the unlimited version to allow AES with 256-bit key."

Ports and a firewall

If a firewall is sitting between the Java web server and the Active Directory and/or Kerberos Domain Controller, the following TCP ports should be allowed:

- 445 for NTLM authentication to the Active Directory.
- 88 and 464 for Kerberos authentication to a KDC

Kerberos with load balancing / F5 / VIP / Netscaler

When using a load balancer / F5 / VIP with a group of web servers, there are extra steps to configure Kerberos through the load balancer hostname.

The set-service-account.cmd script will set up the accounts required for this configuration and we recommend it is used if possible.

Kerberos relies on a service principal name (SPN) being present in the Active Directory, mapping a hostname to a service account. Setting up SPNs has been documented above but the golden rule is as follows: An SPN for a hostname can only exist once; if it exists more than once, it is ignored.

In a situation where there are multiple web servers, each with a computer service account (for the purposes of NTLM), it is impossible to set up an SPN for a load balancer hostname against each different service account. For example, the following is **not** valid:

```
setspn -A HTTP/itsm.jss.com JSS-SSO-P1
setspn -A HTTP/itsm.jss.com JSS-SSO-P2
```

The solution is to create a separate service account for Kerberos only, and configure Kerberos independently of NTLM on each web server.

The Kerberos account can be a normal user account and assuming it is called JSS-SSO-KRB, the Active Directory administrator can enable Kerberos as follows:

```
setspn -A HTTP/itsm.jss.com JSS-SSO-KRB
setspn -A HTTP/itsm JSS-SSO-KRB
```

Please note, both the short hostname and fully qualified hostnames are set up to ensure that it works whether a user types http://itsm or http://itsm.jss.com into a browser.

To re-configure each web server, select Windows Authentication, leave the NTLM configuration as is (ie JSS-SSO-P1 on one web server, and JSS-SSO-P2 on the other) and configure both servers with the JSS-SSO-KRB service account in the Kerberos setup.

HTTPS URLs

When setting up an SPN for the URL https://itsm.jss.com, the syntax for setspn is still HTTP/itsm.jss.com, ie the HTTP/ part of the setspn command does not change.

Example configuration

If you have two Java web servers behind a load balancer:

1. Create two computer accounts, JSS-SSO-P1 and JSS-SSO-P2.
2. Create a third user account, JSS-SSO-KRB and set the load balancer hostname as an SPN against this account.
3. Go to the SSO Plugin configuration page on the first Java web server and configure Windows Authentication custom settings.
4. Check Permit Kerberos, set the client type to acceptor and enter the JSS-SSO-KRB account details.
5. Check Permit NTLM and enter the JSS-SSO-P1 account details.
6. Go to the SSO Plugin configuration page on the second Java web server and configure Windows Authentication custom settings.
7. Repeat step 4.
8. Repeat step 5 but use the JSS-SSO-P2 account.

Mapping domain names to alternative values

Please Note: This functionality does not let two untrusted domains work with Integrated Windows Authentication. Once the user has been authenticated within the domain,

details from the domain can be used to uniquely identify the user within the application. This is accomplished by submitting queries to the application e.g. ITSM. This functionality allows our customers to substitute values in that query. Complex Windows networks can contain many different domains, and functionality exists to quickly map domains (whether DNS or NETBIOS) to alternative values.

The functionality can be used with both Kerberos and NTLM tokens that are successfully authenticated by SSO Plugin.

Consider a scenario where a company has two Windows domains below - the value in brackets is the NETBIOS domain:

- blue.abc.corp.com (BLUE).
- red.def.corp.com (RED).

Yet the company stores usernames in BMC or HP in the format, user@corp.com. There are now a number of scenarios in which an SSO username could be passed to SSO Plugin:

1. user@blue.abc.corp.com
2. BLUE\user
3. user@red.def.corp.com
4. RED\user

Whilst SSO Plugin provides user aliasing functionality, the domain mapping functionality can solve this problem.

By creating a file called domainmap.properties and placing it in the classpath (ie tomcat/webapps/application/WEB-INF/classes), domain values (whether DNS or NETBIOS) can be mapped to alternative values.

In the above scenario, the file will contain the following entries (it is case insensitive):

```
blue.abc.corp.com=corp.com
red.abc.corp.com=corp.com
BLUE=corp.com
RED=corp.com
```

This mapping will enable users being assigned an SSO username that is either user@corp.com or corp.com\user. Whilst the latter isn't a valid NTLM formatted token, it does allow the user aliasing feature to consistently use the SSO_DOMAIN variable with corp.com in both cases.

Finally, the user can implement user aliasing to query for the correct user, ie. '101'="SSO_USER@SSO_DOMAIN" on BMC AR System.

Using IIS and Windows Authentication

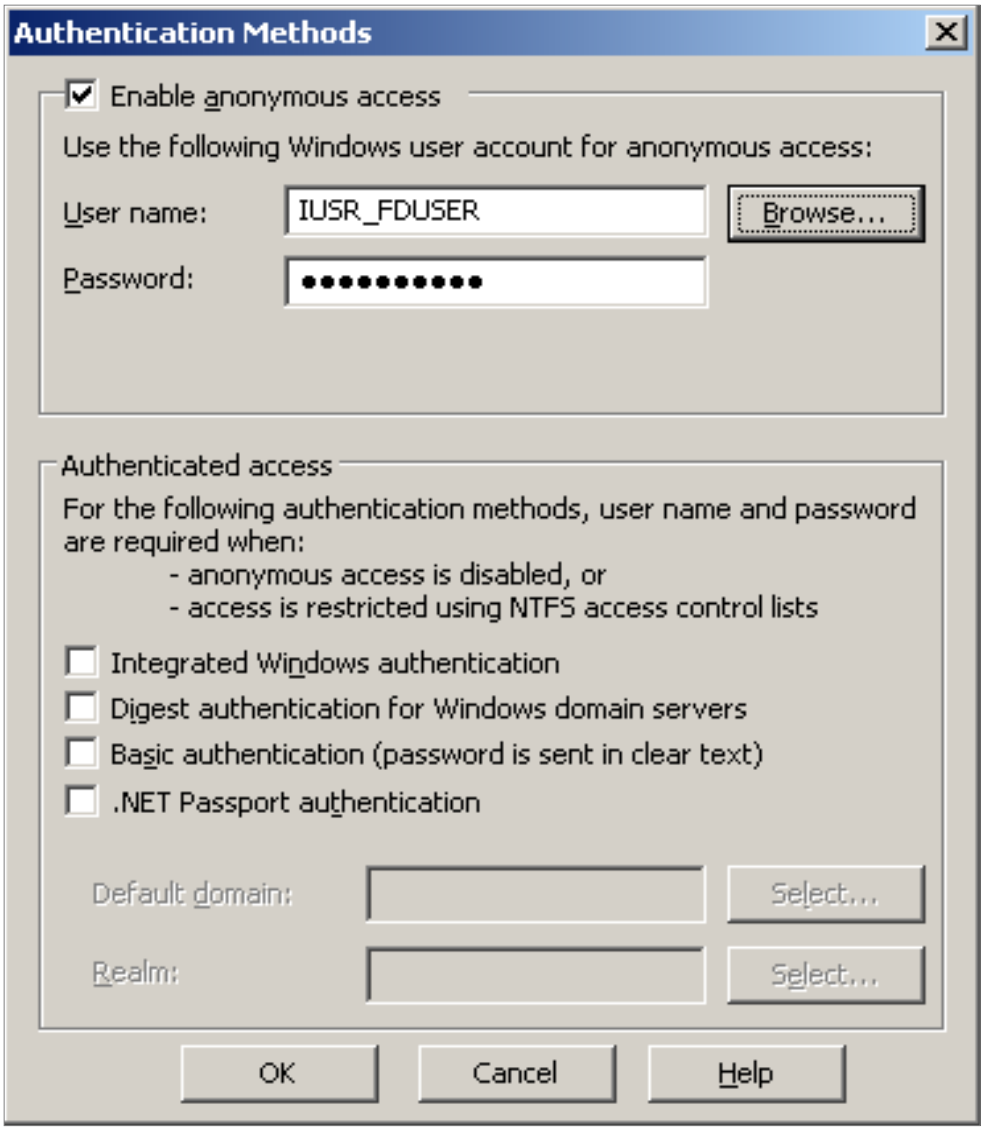
We do not recommend this configuration unless there's a good reason for IIS to be in place, ie it is used to serve other applications.

If using a single Java web server instance, and are not engaging in software load balancing, then there is little value in an IIS front end with Windows Authentication.

The BMC Mid Tier installer will configure IIS if it's present, and while we do not recommend this configuration, it is possible to use Internal Windows Authentication with IIS.

In order to do this, you must ensure IIS is not configured to perform any authentication. This is done by configuring the IIS website authentication to anonymous only:

1. Open the Windows Control Panel.
2. Open Administrative Tools.
3. Open the IIS management console.
4. Locate Websites / Default website / jakarta, right click and select Properties.



5. Locate the Directory Security tab and click Edit in 'Authentication and Access Control'.
6. Ensure 'Enable anonymous access' is checked, and the 'Authenticated access' check boxes are unchecked. The following dialog box shows the configuration:

Windows Authentication performed by IIS

Please do not run the set-service-account.cmd script if using an IIS front end – it's not required and may lead to IIS being unable to authenticate users.

While an IIS front end is fully supported, there is often little value in this configuration if IIS is doing nothing more than sending requests to the Java web server instance because SSO Plugin has built-in Active Directory integration, performing the authentication task that IIS performs.

If you use the Windows Authentication integration instead of an IIS front end, the IIS component can be removed, simplifying the architecture.

Configuring IIS

IIS must be configured to perform Windows Authentication (IWA) and anonymous. The anonymous access is required for WUT Data Visualisation fields which are displayed in an IE component that can not perform IWA.

Make the changes to IIS as follows:

1. Open the Windows Control Panel.
2. Open Administrative Tools.
3. Open the IIS management console.
4. Locate Websites / Default website / jakarta, right click and select Properties.
5. Locate the Directory Security tab and click Edit in 'Authentication and Access Control'.
6. Ensure 'Enable Anonymous Authentication' and 'Windows Authentication' are checked.

Service principal names

To make IIS perform Windows Authentication (IWA), service principal names (SPNs) must be configured. It is common to find these have already been setup on a corporate installation of IIS but in the event Windows Authentication fails, ask the Active Directory administrators to ensure they are configured for the hostnames on which IIS is running.

There are a number of examples of setting up SPNs in this document, and the task of setting up a corporate IIS instance falls within the bounds of the Active Directory / network administrators.

If you require assistance, JSS are happy to help.

Configuring Tomcat

To tell Tomcat that IIS is performing Windows Authentication, locate the Tomcat server.xml file, which will be in the Tomcat conf directory. Locate the ajp/13 connector, which looks like this:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

and add the following attribute:

```
tomcatAuthentication="false"
```

In this example, it would now look like this:

```
<Connector port="8009" tomcatAuthentication="false" ...
```

Large Kerberos tokens

IE clients can send very large Kerberos tokens which can be too big to be passed between IIS and Tomcat through the mod_jk connector (this is the software that connects the two systems). This will cause browser issues and often only on some machines (as Kerberos tokens contain group information, so if a user is in many groups, the token is likely to be larger than a user who is not).

To rectify this, two files must be modified:

1. The mod_jk workers.properties, also called workers.properties.minimal in some deployments. You will need to search for this file as it could be in many locations, but is often found near the Apache Tomcat installation. Open the file and add the line following line, where X is the name of the worker - you will see many other similar lines from which to copy and edit:

```
worker.X.max_packet_size=65000
```

2. Locate the Tomcat server.xml file, which will be in the Tomcat conf directory. Locate the ajp/13 'Connector, which looks like this:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

and add the following attribute:

```
packetSize="65000"
```

In this example, it would now look like this:

```
<Connector port="8009" packetSize="65000" ...
```

Configuring SSO Plugin

Select the authentication method Windows authentication performed by IIS.

This operation will result in a patch being applied to the product web component web.xml file (as is the case with Windows Authentication). If the web.xml file is patched, a warning message will be displayed when you submit the setup form and you **must restart the Java web server (ie Tomcat).**

Finally, it is essential to secure the link between IIS and Tomcat.

Securing IIS or Apache and Tomcat

It is important to secure Tomcat once IIS or Apache integration has been completed. This is achieved by turning off the Tomcat HTTP Connector (so all requests must go through IIS or Apache) and setting a secret on the Tomcat's AJP connector so it is only accessible by the IIS or Apache instance configured.

Turning off Tomcat's AJP connector

Open the server.xml file, look for the following and comment it out by surrounding with <!-- and --> (it is usually located directly above the AJP connector modifed above):

```
<Connector port="8080" ...
```

Setting a secret on the AJP connector

Open the workers.properties file located with the mod_jk installation on IIS or Apache and add a secret to the worker:

```
worker.X.secret=mysecretkey
```

Open the Tomcat server.xml file and add this attribute to the AJP connector:

```
<Connector port="8009" packetSize="16000" request.secret="mysecretkey"
```

Restart both IIS or Apache and Tomcat when complete.

OAuth2 & OpenID Connect

The OAuth2 protocol allows SSO Plugin to authenticate with an OAuth2 identity provider (such as Google or Facebook). The implementation is as follows:

1. If a user is not authenticated, SSO Plugin will redirect the browser to the OAuth2 login/authorisation URL.
2. The OAuth2 identity provider should authenticate the user and redirect the browser back to SSO Plugin with the HTTP parameter 'code'.
3. SSO Plugin verifies the code parameter using the token URL, which returns an 'access_token' parameter.
4. SSO Plugin retrieves the user details by passing the access_token to the User Info URL, which returns a JSON object that is queried for the return attribute (ie email).

Whilst the Google and Facebook providers have been implemented out of the box, any OAuth2 provider can be configured using the Custom option. The product has also been tested with Ping Federate.

Some OAuth2 providers (ie Facebook) do not implement the OAuth2 specification correctly, and where there is demand, JSS will expand the list of out of the box providers to work around non-compliance.

Configuring an integration

The user interface requires values that should be provided by the OAuth2 service administrator. The provider type selects well known configurations (ie Facebook & Google) that reduces the required values.

There are a few values that may not be familiar to the OAuth2 administrator:

1. JSON attribute: The username from which to extract a value in the JSON structure returned by the IDP from the user info URL.
2. Metadata URL: Some IDPs publish a JSON object with a list of configuration options, ie the login/authorisation, token and user info URLs. Entering the URL and pressing fetch will cause SSO Plugin to retrieve these values and populate the configuration.

   The PingFederate product publishes metadata at the following URL:
   https://pfhostname:9031/.well-known/openid-configuration

If the user info URL is SSL, the JVM must be configured with the OAuth2's SSL certificate in order for SSO Plugin to query the user info URL to retrieve an SSO username. This is a typical problem when deploying a PingFederate instance with a default self-signed certificate, that PingFederate generates itself.

If you see an error such as, 'Unable to valid certificate' after configuring an OAuth2 integration, this is the likely problem and the solution is documented here.

CA SiteMinder / RSA Access Manager (ClearTrust)

There are two ways to integrate with these products: using a native library to connect directly to the authentication service, or by placing a webserver (ie Apache) in front of the Java web server and letting the webserver perform the authentication.

This section covers the native integration. Please consult using HTTP header and cookie values if you have a web server front end providing the integration to CA SiteMinder (CA SM) or RSA Access Manager (RSA AM).

The two integrations require the native libraries to be installed in the Java webserver:

- CA SM: the SDK jar files (ie cryptoj.jar, imsjavasdk.jar, smagentapi.jar, smjavaagentapi.jar, SmJavaApi.jar, smjavasdk2.jar) must be present in the classpath of the Java web server and the platform native libraries that accompany it must be set in the directory pointed to by the LD_LIBRARY_PATH environment variable.
- RSA AM: the ct_runtime_api.jar file must be present in the classpath of the Java web server.

When SSO Plugin detects the relevant API files, the authentication options are enabled in the web interface. Both integrations require a set of values that can be provided by the team managing CA SM or RSA AM. These values are:

- CA SM:
  - Required: Trusted hostname and either:
    - Shared secret and policy server IP/hostname.
    - An SmHost.conf file as configured for the SM API version 5.
      - Note, the policy server login URL and SM cookie name variables are still configured through the SSO Plugin user interface.
  - Optional (defaults):
    - API return attribute, username, userdn and universal ID.
    - Login URL ie http://policyserver.corp.com/login?goto= (see below).
    - Cookie name (SMSESSION).
    - Authorisation port (44443).

        - Authentication port (44442).
        - Accounting port (44441).
        - Connections min (2).
        - Connections max (25).
        - Connection pool increment step (2).
        - Connection timeout (60).
- RSA AM:
    - Required: Comma separated list of dispatcher hostnames. If you wish to specify a port then use the format hostname:port. The default port is 5608.
    - Optional (defaults):
        - Login URL ie http://policyserver.corp.com/login?ct_orig_uri= (see below),
        - Cookie name (CTSESSION).

CA SiteMinder policy server configuration

SSO Plugin not only authenticates the SiteMinder cookie, to extract a username, but it also checks to see if the user is authorised to request the protected URLs. Therefore, the policy server must be set up to authorise users to the product entry points, ie.

All products, where /context is /arsys, /webtier, etc:

- /context/jss-sso/testsso.jsp,
- /context/jss-sso/debug.jsp

BMC Mid Tier:

- /arsys/home
- /arsys/forms/*
- /arsys/apps/*

HP Web Tier:

- /webtier/index.do
- /webtier/ess.do

Keep in mind that these URLs are only examples, because if the product root (ie /arsys) is different then the URLs need modifying accordingly.

When SSO Plugin has patched the product web.xml file, the URLs mapped to the ssoproxyfilter are a good guide to the ones that require authorisation.

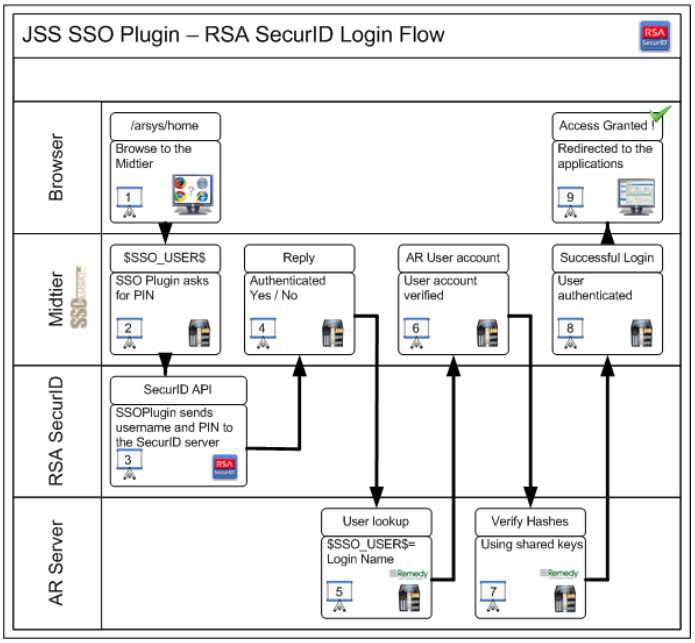
Login URLs

When using RSA AM, the login URL must end with the HTTP parameter used to specify the return URL to the login server, ie ct_orig_uri is common for RSA Access Manager.

When using CA SiteMinder, specify just the URL and SSO Plugin will add the return URL parameter. Please note, SSO Plugin does not support the SMAGENTNAME HTTP parameter for the SiteMinder login portal. This is because there is no known way to create the value for this variable when not using the proprietary SiteMinder login module within Apache/IIS.

RSA SecurID

The RSA SecurID integration requires the path to an SD configuration file. The process flow is as follows:



Using HTTP headers or cookies to retrieve SSO username

This integration can be used for third party SSO products where the SSO username is placed in an HTTP header or cookie, such as:

- CA SiteMinder.
- RSA Access Manager.
- Novell Access Manager,
- IBM Tivoli Access Manager.
- Oracle Identity Manager.

Typically, an SSO module is installed in a web server (ie Apache) front end to the Java web server and the correct header/cookie can be derived from the SSO Plugin debug page located in the web interface.

The typical HTTP header name for RSA Access Manager (ClearTrust) is CT_REMOTE_USER.

If you wish to use an Apache front end to Tomcat, configured to protect required URLs (search the web.xml.patch file with the string <url-patthern> for a full list), it is essential to secure the link between IIS and Tomcat.

IBM Tivoli Access Manager (TAM)

IBM provide a plugin for Tomcat that provides a mechanism of verifying a request and setting a username for SSO Plugin (set the 'Generic' authentication type in SSO Plugin's configuration).

At the time of writing, the plugin is being maintained and ships with reasonable documentation. The plugin can be found here:

https://www-304.ibm.com/support/docview.wss?uid=swg24021393

JSS are happy to assist with deployment of this plugin.

OpenSSO (aka OpenAM)

The product is supplied with a patch for integrating with OpenSSO. In order to do this, the OpenSSO management team must set up the Java web server so that the OpenSSO JEE filter can be inserted into the web application without any further configuration.

The patch can be found in the web.xml.patch.opensso file within the installation files and must be manually applied to the web application web.xml file. Apply the patch before the first <filter> tag.

The OpenSSO filter must be configured to read the username from a header set by the OpenSSO agent – set it as the parameter value in this extract:

```
<init-param>
  <param-name>header</param-name>
  <param-value>header on which OpenSSO places username</param-value>
</init-param>
```

It is important to note that if SSO Plugin automatically patches the file with the standard SSO Plugin patch (ie to set up all the standard filters for features such as supporting SSO for Business Objects), the manual patch must be applied before the automatically applied patch. The automatically applied patch can be easily located by looking for the string JSS and is contained within comments, ie. <!-- JSS Start --> and <!-- JSS end -->.

On Tomcat, the OpenSSO patch can be applied before the first <filter> tag to ensure the automatic patching doesn't disrupt/remove it. On other Java web servers, please consult JSS.

X509 client certificates (DoD CAC, FIPS 140-2)

The SSO Plugin supports X509 client certificates if your webserver has been configured with an SSL connector configured to use a server side certificate. This subject can be challenging and there are many online tutorials, so this document provides a brief guide. JSS support would be happy to help you configure this integration.

To make matters more complicated, Tomcat has two separate methods of configuring SSL:

- The JSSE implementation provided as part of the Java Virtual Machine.
- The APR implementation, which uses the OpenSSL engine by default.

The implementation depends on the Tomcat installation: if the APR native library has been installed, then the OpenSSL engine is configured. The documentation for SSL configuration can be found on the Tomcat website. This document provides a brief overview of both implementations.

Please note, X509 is also used by DoD Common Access Cards (CAC) and Federal Information Processing Standard (FIPS) 140-2.

Example SSL configuration using JSSE

For the purposes of demonstrating the functionality, you can generate your own and we've provided a script called generate-example-client-cert.bat to do this for you. It will create a fake key store and client side certificate. **This script is only for the purposes of demonstration – if your business wishes to use SSL, you should not have to generate these files.**

If you run the script and pass a username, a certificate will be created with the subject:

```
cn=[USER],ou=yourlocation,o=jss,l=location,s=state,c=uk
```

You can change this subject by editing the script.

The script generates two files:

1. user.p12, which should be loaded into your browser.
    1. On Firefox, go to Tools / Options / Advanced / Encryption / View Certificates / Your certificates / press import. Select the file and when prompted for a password, enter password.
    2. On IE, go to Tools / Internet Options / Content / Certificates / Personal / press import, browse to the file / select .p12 in the file types drop down, enter password and press next, next, finish.
2. server.jks, which is placed in the Tomcat conf directory.

To complete the setup, you must enable Tomcat's SSL port by opening the server.xml and adding this connector:

```
<Connector
  clientAuth="true" port="8443" minSpareThreads="5"
  maxSpareThreads="75"
  enableLookups="true" disableUploadTimeout="true"
  acceptCount="100" maxThreads="200"
  scheme="https" secure="true" SSLEnabled="true"
  keystoreFile="${catalina.home}/conf/server.jks"
  keystoreType="JKS" keystorePass="password"
  truststoreFile="${catalina.home}/conf/server.jks"
  truststoreType="JKS" truststorePass="password"
  SSLVerifyClient="require" SSLEngine="on" SSLVerifyDepth="2"
  sslProtocol="TLS"
/>
```

Example DoD CAC SSL configuration using APR

Whilst each Department Of Defence site may have a different set of documents to configure SSL to work with the Common Access Cards, the following high level guide covers a common scenario.

To start the process, generate a public and private key using the keytool command (found in the Java installation bin directory):

```
keytool.exe -genkeypair -alias servername -keystore keystore.pfx -storepass
Pa33word -validity 365 -keyalg RSA -keysize  2048 -storetype pkcs12
```

This will ask for a number of details and the Java web server full hostname must be entered under 'First and last name'.

Next, export the Certificate Signing Request public key:

```
keytool.exe -certreq -alias servername -file csr.cer -keystore keystore.pfx
-storetype pkcs12
```

Typically, a DoD website will be available to sign the csr.cer file, generating a signed public key. Download this and call it public.pem.

Next, extract the private key from the keystore.pfx file generated in the first step:

```
openssl pkcs12 -in keystore.pfx -out private.pem -nodes
```

The DoD website that signed the Certificate Signing Request used a Certificate Authority (CA) and this is required to configure Tomcat, so Tomcat can trust the client certificate sent by the browser (stored on the CAC).

The easiest way to obtain this is using your IE instance: go to Tools -> Options -> Content -> Certificates -> Trusted root certificate authorities and locate the DoD Root CA 2, a common root level CA. You can verify this is the correct root CA by viewing your certificates under the Personal tab and looking at the top most entry in the Certificate Path tab.

When the root CA is highlighted within the Trusted root certificates tab, click Export and select Base 64 (.CER). Save the certificate to a file called rootca.cer.

Finally, copy public.pem, private.pem and rootca.cer into the Tomcat conf directory and add the following Connector to the Tomcat server.xml file:

```
<Connector port="8443" enableLookups="true" disableUploadTimeout="true"
  acceptCount="100" scheme="https" secure="true" SSLEnabled="true"
  SSLCertificateFile="${catalina.home}/conf/public.pem"
  SSLCertificateKeyFile="${catalina.home}/conf/private.pem"
```

```
  SSLCACertificateFile="${catalina.home}/conf/rootca.cer"
  SSLVerifyClient="require"
/>
```

It is important to remember that SSL client certificates will only work correctly when you access the Java web server using the hostname entered when generating the private key in step 1.

Configuring SSO Plugin

To look for an X509 client certificate, select the appropriate integration type (generic REMOTE_USER, JAAS and X509). You can now choose a number of ways to map a subject back to the User form.

You could choose to implement user aliasing: insert a new field on the User form, enter the certificate subject against a user and let the SSO Plugin match the record when that user accesses Mid Tier.

You could use the 'strip domain' feature which, in this case, extracts the cn or uid value (ie X from "cn=X, ou=Y, o=Z.." or "uid=X, ou=Y, o=Z..") and matches that against a User form Login Name, or invoke user aliasing.

SAML (version 2)

The process of configuring SAML will require input from the SAML Identity Provider team, who will invariably need to configure the IDP to accept authentication requests from SSO Plugin. SSO Plugin supports the POST (using an HTTP POST), Redirect (using an HTTP GET) and Artifact profiles, and has been validated against two common integrations detailed below.

JSS clients have integrated SSO Plugin with a number of Identity Providers including Microsoft ADFS, Ping Federate, Carbon, Symphony Identity Federation Manager v5.7.1 and the Juniper SA-4500 7.2R1.1 Networks appliance.

Assertion Consumer URLs

The SSO Plugin SAML interface provides two ways to define end points (ie protected URLs) in the IDP:

1. Defining a single entry point /jss-sso/saml/authenticate. For each application, the context path needs to be included, ie /arsys/jss-sso/saml/authenticate (Mid Tier), /webtier/jss-sso/saml/authenticate (Web Tier), etc.
2. Defining each application entry point, ie /arsys/home, /arsys/forms, /arsys/jss-sso/testsso, /webtier/index.do, /webtier/testsso.

Some IDPs (such as the Symphony Identity Federation Manager and Juniper [SA-4500] network appliances) do not support multiple SP URLs in a single configuration (2 above). This is technically referred to as the SP only supporting a single Assertion Consumer URL (1 above).

A single entry point is easier to configure, because Plugin takes care of redirecting the browser to the actual application URL.

IDP Metadata

The IDP Metadata functionality allows SSO Plugin to perform some configuration from an XML file published by the IDP. Its use is not mandated and the worked examples below assume it has not been used. However, the use of IDP Metadata drastically reduces the amount of SSO Plugin configuration.

The easiest way to use this functionality is to download the metadata from the IDP, store it in a local file, enter the path into the configuration page and press the 'retrieve' button, which will extract the relevant configuration and auto populate most of the SAML configuration.

You can enter http, https and file URLs into the metadata configuration field, ie.

- https://host/FederationMetadata/2007-06/FederationMetadata.xml (ADFS)
- https://host:9031/pf/metadata.ping (Ping Federate)
- file:///path/to/metadata.xml

SSO Plugin will currently use the metadata to configure the following items:

1. The Identity URL, preferring the POST binding but otherwise picking the first available binding.
2. The IDP Issuer ID.
3. The public certificate required for verifying messages from the IDP.
4. The artifact resolution URLs.

With this information in place, the only item required for an end to end integration is the Service Provider Issuer ID which must be entered into the user interface (those managing the IDP can supply this information).

SP Metadata

The SP Metadata functionality provides an XML document that can be used to speed up the process of configuring the SSO Plugin instance within the IDP. The XML document reflects the current SSO Plugin configuration.

The SP Metadata includes a URL generated from the URL used to retrieve the data, so if you connected via http://localhost/arsys/jss-sso/api/saml/metadata then the URLs in the metadata contain will be as follows (these URLs are for BMC Mid Tier, but relevant URLs for Web Tier and other products are generated):

```
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://localhost/arsys/jss-sso/saml/authenticate"
isDefault="true" index="0"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://localhost/arsys/home" index="1"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://localhost/arsys/jss-sso/testsso" index="2"/>
```

Some IDPs, such as ADFS, will ignore non-SSL URLs because they only support SSL URLs as endpoints. Therefore, ensure you connect to SSO Plugin using the end user facing URL when generating metadata, to ensure the URLs within it contain the correct protocol and hostname.

Importing Metadata into ADFS

The following items should be addressed when the metadata has been imported:

1. The Claim Rules still need to be configured as documented.
2. Double click the Relaying Party Trust, go to the Advanced tab and select SHA-1 as the Secure Hash Algorithm. ADFS 2.0 does not pass an assertion [username] in the SAML response when this is set to SHA-256.

SAML Artifacts

The SAML Artifact profile is the more complicated than the POST and redirect profiles because SSO Plugin has to connect back to the Identity Provider to verify a SAML Artifact. The process flow is as follows:

1. User requests protected resource.

2. SSO Plugin redirects browser to IDP.
3. IDP authenticates the user and if successful, redirects the browser to SSO Plugin with a SAMLart HTTP parameter.
4. SSO Plugin connects to the IDP and passes the SAMLart value.
5. IDP validates the SAMLart value and returns a SAML assertion, which SSO Plugin decodes to extract a user.

This process requires that the web server running SSO Plugin has network access to the IDP, unlike the POST and redirect profiles that are entirely driven through the browser.

SSO Plugin provides the ability to set the Artifact Resolution URL (step 4) because it is often different to the Identity Provider URL used to initiate the SAML authentication (step 2).

For example, Microsoft ADFS2 typically uses the following values:

- IDP URL: https://host/adfs/ls/.
- Artifact Resolution URL: https://host/adfs/services/trust/artifactresolution (this must be explicitly enabled in the ADFS console).

And Ping Federate uses these values:

- IDP URL: https://host:9031/idp/SSO.saml2.
- Artifact Resolution URL: https://host:9031/idp/ARS.ssaml2 (the double ss in saml2 is not a mistake!).

The problem is further complicated by the use of signing/verification. Some IDPs will require the request to resolve an artifact (step 4) is signed, and it will in turn sign the SAML assertion (step 5). Therefore, the sign and verify certificate aliases are used for these steps if set.

These integrations are complicated so please do not hesitate to contact JSS for advice.

SSL Artifact Resolution URLs

Most IDPs require an SSL connection to resolve an artifact. This presents a challenge with respect to configuring the JVM to trust the SSL certificate. To achieve this, you must configure the JVM to trust the certificate.

SAML Encryption

An Identity Provider can encrypt a SAML response using a public key supplied by the Service Provider. This also replaces the requirement for verifying the SAML response, because it's encrypted and is validated by virtue of requiring decryption.

The product supports the decryption of a SAML response through re-using the signing private key. To configure the IDP, the signing public key can be exported from the keystore and configured with the IDP (in ADFS, it is set in the Encryption tab of the Relaying Party Trust configuration).

To configure SSO Plugin, select 'to IDP' in the message security selector, configure the keystore path and password, and the name of the signing and decryption private key within the keystore. When SSO Plugin receives an encrypted SAML response from the IDP, it will decrypt it using the private key.

Given this configuration makes use of the message signing functionality, using the same private key to sign messages sent to the IDP, you may wish to configure the IDP to verify messages sent by SSO Plugin.

Given this configuration makes use of the message signing private key, you may wish to configure the IDP to verify messages sent from SSO Plugin. This is discussed in the signing messages to the IDP section, which shows you how to export the public key required for the IDP encryption configuration.

IDP initiated SSO

The product supports IDP initiated SSO when the RelayState HTTP parameter is set to the target URL. No additional configuration of the product is required beyond ensuring SP initiated SSO works correctly.

The following URLs are entry points for IDP initiated SAML:

- ADFS: https://adfshost/adfs/ls/IdpInitiatedSignon.aspx

Integrating with ADFS 2.0

The Microsoft ADFS 2.0 (not 1.0, as installed with Windows 2008R2) integration involves a pre-requisite of ensuring the Java web server can be accessed via SSL. ADFS only allows endpoints of Service Providers (ie Mid Tier or Web Tier) to be entered as https URLs. Therefore, before proceeding to integrate with ADFS, ensure the Java web server (ie Tomcat) is SSL enabled.

Once ensuring your Java web server supports SSL, you can configure ADFS. The first step is to configure a Relaying Party Trust for SSO Plugin. It is performed as follows:

1. Right click on Relaying Party Trust and click Add Relaying Party Trust.
2. Press start, select 'Enter data about the relaying party manually' and press next.
3. Enter a display name, such as JSS SSO Plugin and press next.
4. Select AD FS 2.0 profile and press next.
5. The next stage allows a certificate to be supplied that's used to sign outgoing requests (from ADFS to SSO Plugin). For now, skip this step and press next.
6. Select Enable support for the SAML 2.0 WebSSO protocol and enter the following URL: https://yourmidtier:8443/arsys/jss-sso/saml/authenticate, or /webtier/jss-sso/saml/authenticate, or /path-to-the-web/jss-sso-saml/authenticate depending on implementation,
7. Enter a relying party trust identifier, ie ssoplugin, and press next. Keep a note of the identifier as it will be required when configuring SSO Plugin (which will be urn:ssoplugin). Please note non-ASCII characters, including spaces, are not permitted.
8. Select permit all users, press next, press next again (leaving open claim rules ticked) and press finish.
9. Export the SAML metadata from ADFS, so you can use this to import into SSO Plugin.

Claim rules

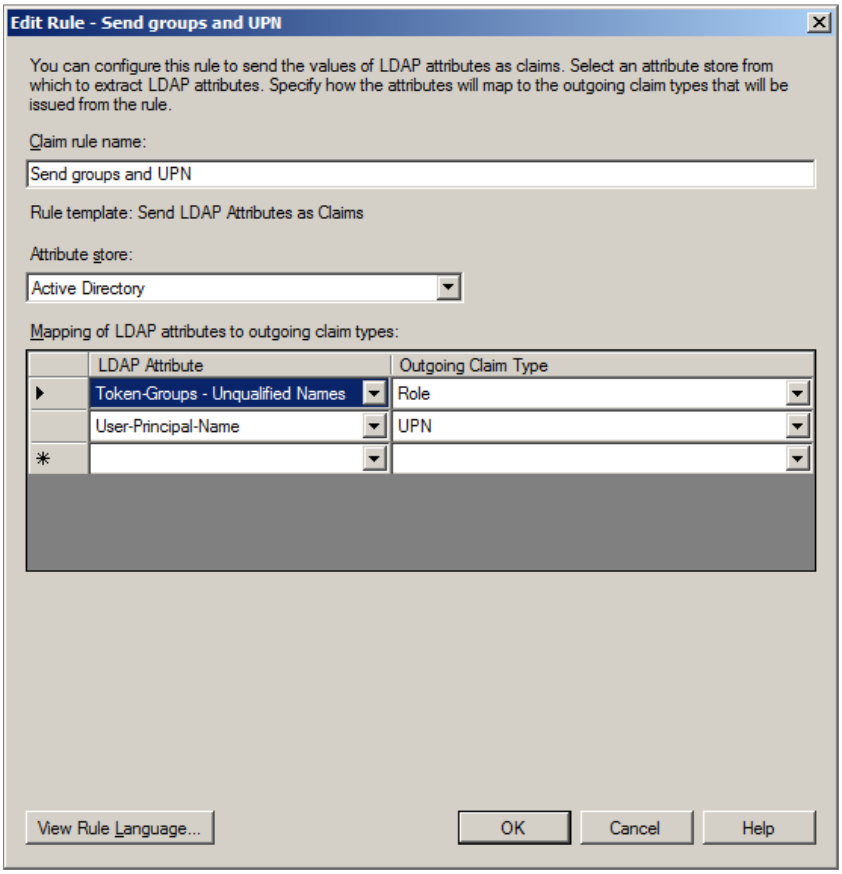
With the claim rules dialog open, two rules are created to issue the Windows user name and DNS domain as a Universal Principal Name (UPN), and the Active Directory groups.

The UPN format is user@dns.domain, allowing the most flexible integration with AR System or Service Manager where full DNS domains can be mapped to User form or Operator table entries. This is important for multi service providers who may have two end clients with a domain called EUROPE, where mapping the full DNS domain to the user record is vital (ie europe.company1.com and europe.company2.com).

Rule 1: Send the Universal Principal Name (UPN) and groups

In order to send the UPN to the client, it must be mapped from the Active Directory attribute. And whilst the information is currently not used by SSO Plugin, it is prudent to map the group information too in the event a future version of SSO Plugin makes use of this information.

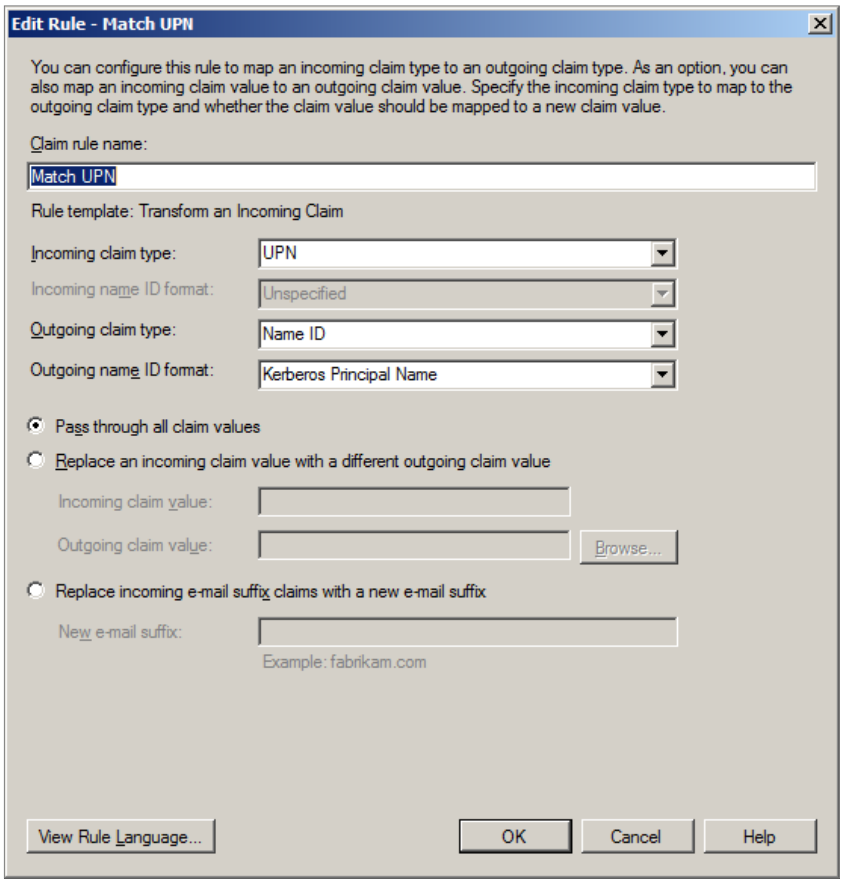
Click add rule, select "Send LDAP attributes as claims" and press next. Edit the dialog as per the image:



Rule 2: Match Kerberos NameID and send the UPN

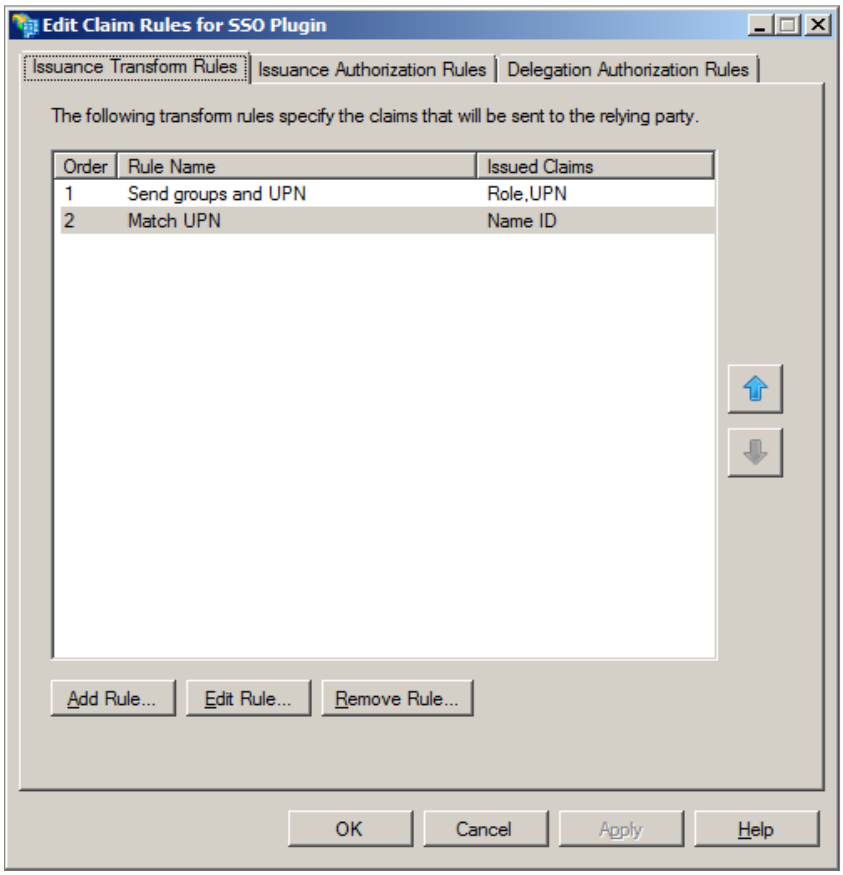
This rule matches the claim for the Kerberos NameID and sends the UPN mapped in rule 1.

Click add rule, select "Transform incoming claim" and press next. Edit the dialog as per the image:



The result

The Issuance Transform Rules should now look as follows:



Configuring SSO Plugin

The next step is to configure SSO Plugin.

1. Go to the SSO Plugin setup interface.
2. Select SAML in the list of integration types.
3. Select the SAML metadata exported from ADFS and import it using the integrations UI.
4. Set the SP Issuer ID to urn:ssoplugin.
5. Set the SAML Binding to POST.
6. Press set configuration.

At this point, the easiest configuration is complete and you should be able to press Test SSO. **Remember, you must test via SSL as ADFS only allows SSL endpoint URLs.** This means the application being secured with SSO Plugin must be exposed via SSL.

The next step is to configure message verification.

Integrating with Ping Federate 6.5+

Ping Federate 6.5+ is provided with a quick start application that includes an Identity Provider and Service Provider configuration and sample applications. If you do not have an IDP installed, this is the quickest route to a successful integration.

To configure the integration with SSO Plugin, a Service Provider must be added.

1. Select manage Service Providers and select create connection.
2. Connection template: Do not use a template.
3. Connection type: Browser SSO profile.
4. Connection options: Browser SSO.
5. Import metadata: Skip step.
6. General info: Most options are optional but the following are required.
    1. Partner's Entity ID: urn:sp. This must be a URI, ie urn:identifier. SSO Plugin verifies the format (as per the SAML specification).
    2. Connection name: JSS SSO Plugin
7. Browser SSO.
    1. SAML Profiles: Click configure browser SSO and select both IDP and SP initiated SSO.
    2. Assertion lifetime: Skip.
    3. Assertion creation.
        1. Identity mapping: Standard.
        2. Attribute contract: Accept default, unspecified mapped to SAML_SUBJECT.
        3. IDP Adapter mapping.
            1. Authentication type: Single two factor.
            2. Adapter: IDP Adapter.
            3. Assertion mapping: Use only the Adapter Contract values in the SAML assertion.
            4. Attribute contract fulfilment: SAML_SUBJECT / Adapter (source) / username (value).
    4. Protocol settings: Select configure.
        1. Add SSO Plugin protected URLs, ie http://host:8080/arsys/jss-sso/testsso.jsp and http://host:8080/arsys/home, mapped to POST.

2. Allowable SAML Bindings: Select only POST.
3. Signature policy: Select always sign the assertion.
4. Encryption: None.

8. Credentials: Select configure and the certificate used for signing. This is the certificate you will need to export when setting up messaging verification.

Configuring SSO Plugin

The next step is to configure SSO Plugin.

1. Go to the SSO Plugin setup interface.
2. Select SAML in the list of integration types.
3. Set the SAML Binding to POST.
4. Select Unspecified in the list of Claim NameID formats, or whatever was mapped when configuring assertion creation in the Ping Federate interface
5. Set the identity URL to point at Ping Federate, ie. https://host:9031/idp/SSO.saml2.
6. Enter the service provider issuer which is the partner entity ID you entered (ie urn:ssoplugin) under general info in the Ping Federate interface.
7. Optionally, set the identity provider issuer to the one set in the Ping Federate interface.
8. Set messaging signing to none and press set configuration.

When complete, test SSO Plugin without signing or message verification to ensure the integration functions. After this test, the next step is to configure message verification.

Message security: signing and verification of messages

There are two ways in which certificate are used when communicating with the IDP: for signing the request to the IDP, and verifying the response from the IDP. On the SSO Plugin setup page, the select field titled message security can be used to select message signing to the IDP, message verification from the IDP or both signing and verification.

It is recommended to test signing and verification individually before selecting both, however in practice, signing messages sent to the IDP has little value and only the message verification feature is required for a secure solution.

Signing messages to the IDP

In order to sign messages to the IDP, a private/public key pair is required. This can be generated automatically by SSO Plugin (recommended) or by using the Java keytool command.

Generate automatically (recommended)

If you wish to generate automatically, simply select the 'Generate automatically' radio button under the 'Signing/decryption' controls. You will be prompted to enter a Distinguished Name for the generated certificate and it should be entered in the format cn=yourhost,ou=yourlocation,o=yourorganisation,l=location,s=state,c=uk.

The certificate with the private key is stored in the SSO Plugin configuration file and the public key is exposed by accessing the Service Provider Metadata link.

Use keystore

The keytool command creates a keystore that contains the private key. The keystore is designed to hold a collection of keys, including the IDP key to sign messages sent to the IDP.

To generate a new private key, run this command from the Java bin directory. The command will ask for a keystore password, and will also prompt for a password to secure the key. When prompted for a password for "the alias", press enter, which tells keytool that you don't want an additional password.

The keystore can quickly become confusing if individual passwords are assigned to certificates - it is not a great piece of design.

```
keytool -genkeypair -keyalg rsa -sigalg SHA256withRSA -keysize 2048 -
keystore keystore.jks -alias sp
```

To verify the key is in the keystore:

```
keytool -list -keystore keystore.jks
```

Finally, to export a certificate for the IDP (to verify messages):

```
keytool -exportcert -keystore keystore.jks -alias sp -file sp.cer
```

When exporting a certificate for Ping Federate, add the -rfc option to export a Base64 version (to create a PEM file).

This certificate can also be used to configure the encryption functionality on the IDP.

To configure SSO Plugin, enter the alias (sp above) into the 'Keystore alias' field and configure SSO Plugin to read the keystore.

Configuring Microsoft ADFS 2.0

Load the exported certificate using the Signature tab on the Relaying Party Trust. Also set the secure hash algorithm to SHA-1 if the keytool example above was used to generate the certificate.

If you wish ADFS to encrypt SAML responses, configure the certificate with under the Encryption tab.

Verifying messages from the IDP

This functionality allows SSO Plugin, the Service Provider, to be sure that messages sent from the Identity Provider are authentic by signing them with a certificate. This can be configured by entering a PEM encoded certificate from the IDP or exporting a certificate from the IDP and importing it into a keystore.

To export the certificate from ADFS, follow this example, and then import it to the keystore following the steps below.

Entering a PEM encoded certificate (recommended)

Select the 'Use PEM' radio button and copy the certificate into the text field. The certificate should be in the format:

```
-----BEGIN CERTIFICATE-----
XXXXX
-----END CERTIFICATE-----
```

Where XXX is a base64 encoded certificate.

If IDP metadata has been retrieved, the IDP's certificate will be configured in this field.

Use keystore

Once you have obtained the IDP's public key, it must be imported into a keystore. If you followed the previous example (verifying messages from the IDP) then you can re-use this keystore, otherwise a new one will be created. In both cases, run this command:

```
keytool -keystore keystore.jks -import -alias idp -file idp.cer -noprompt
```

If you need to re-import a certificate, the previous one must be deleted:

```
keytool -keystore keystore.jks -delete -alias idp
```

To configure SSO Plugin, enter the alias (idpp above) into the 'Keystore alias' field and configure SSO Plugin to read the keystore.

Configuring SSO Plugin to read a keystore

SSO Plugin can load certificates from a keystore when signing messages to, or verifying messages from, an IDP. In both cases, follow these steps to configure the keystore with SSO Plugin:

1. Select the appropriate message security option and enable use keystore in signing/validation. This will result in the keystore name and password fields being enabled.
2. Enter the keystore filename into the keystore name field and place the keystore file into the Java web server classpath, ie place into the Mid Tier or Web Tier WEB-INF/classes directory. You can also enter a pathname, ie /path/to/keystore.jks or c:\\path\\to\\keystore.jks.
3. Enter the keystore password.
4. Press set configuration and test.

Exporting the certificate from Microsoft ADFS 2.0

The certificate is exported from the ADFS management interface.

1. Navigate to services / certificates.
2. Right click and view the certificate under token signing.
3. Select the details tab and click copy to file.
4. Click next, select the first radio option (DER encoded binary), click next and select a file – call the file idp.cer. Click next and finish.

Exporting the certificate from Ping Federate 6.5+

The certificate is exported from the administration interface:

1. Go to the management interface overview page.
2. Click Digital Signing & XML Decryption Keys & Certificates.
3. Export the certificate for the IDP, ie for the quick start application the DN is CN=localhost, O=Quick Start App, C=US.

Single Log Out (SLO)

SSO Plugin will respond to Single Log Out (SLO) requests, whilst accessing a URL ending in /jss-sso/saml/authenticate?logout=true will also cause the session to be killed, ie. http://midtier:8080/arsys/jss-sso/saml/authenticate?logout=true.

LDAP

The LDAP authentication type provides a traditional integration with an LDAP repository. This does not provide a seamless sign-on, ie open a web browser, navigate to a web application and login, but it does provide a single point of sign on between different applications deployed through SSO Plugin (ie BMC Mid Tier and SAP Business Objects).

To configure this authentication type, three pieces of information are required:

1. The LDAP authentication URL. This contains the hostname, basedn, attribute, scope and filter in the format ldap://host:port/basedn?attribute?scope?filter. The values are configured as follows:
    a. ldap://host:port: The LDAP URL used to locate the LDAP.
    b. attribute: The attribute on which to search for a username, default uid.
    c. scope: One level (one) or sub-tree (sub), default is sub.
    d. filter: The filter that will be combined with the attribute=username, default (objectClass=*).

    For example, if the filter is set to a=b then the search will use the combined filter: (&(a=b)(attribute=username))
2. The bind DN: This must be the distinguished name (DN) of a user that has access to read the users that will be connecting, ie those defined by the LDAP authentication URL.

3. The password for the bind DN.

The SSO username is set to a value with this format "attribute=ssousername,parentdn" but when 'Remove domain part' is enabled, ssousername is returned to the application. The non-stripped username provides a unique value for the purposes of user aliasing.

The parentdn value is set as the SSO_DNS_DOMAIN user aliasing variable.

Failover

Multiple URLs can be configured by separating with a space, however the attribute, filter and scope settings must be configured against the first URL.

For example, this string configures two LDAP servers:

ldap://host:389/cn=users,dc=corp,dc=com?samAccountName?sub
ldap://failoverhost:389/cn=users,dc=corp,dc=com

Using SSL with LDAP

To use an SSL connection with LDAP, the ldaps URL can be used, ie ldaps://host:port/basedn?attribute?scope?filter.

If the LDAP server uses a self-signed SSL certificate, you need to configure the JVM to trust the certificate.

Central Authorisation Services (CAS)

The product can be configured to integrate with a Central Authorisation Service. The integration will obtain a ticket from the CAS via browser redirection and validate the ticket, before passing the authenticated username to the application.

To configure, select the CAS integration option and provide the URL to the CAS service, ie http://host.edu/cas, and the base URL of the SSO Plugin webserver, ie http://yourhost:8080/ (do not add /arsys, /webtier, /authentication-service etc).

Local logout invoking CAS logout

To enable a local logout to cause a CAS logout, replace the application's logout JSP file with a new file containing the following:

```
<%
  session.invalidate();
  response.sendRedirect("http://host.edu/cas/logout");
%>
```

- In BMC Mid Tier, the logout page to replace is shared/logout.jsp.
- In HP Web Tier, the logout page to replace is goodbye.jsp.

Supporting the CAS global logout

SSO Plugin supports a CAS 2.0 global logout. If you have manually patched the web.xml file (which would be unusual because SSO Plugin does this automatically), edit the web.xml file with your favourite text editor, locate an existing <listener> and insert the following before the listener:

```
<listener>
  <listener-
class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listen
er-class>
</listener>
```

Restricting SSO access by client IP address or hostname

The Windows Authentication and IIS integration options reveal configuration to restrict client access. The restriction can be performed in two ways, by client IP address or client domain.

It should be noted that obtaining client IP address and hostname information is not always reliable if a web proxy or firewall is hiding the information.

By IP address

One or more IP address or address range can be configured. If set, the product only challenges the client for single-sign on if the client's IP address falls within one or more IP ranges, or is a specific IP address.

For example, "192.168.0.1-192.168.1.200,192.168.2.100" will deny access to a client with IP address 192.168.0.3 but allow access to a client with IP address 192.168.2.101.

By domain

One or more host domain names can be configured. If set, the product only challenges the client for single-sign on if the client's IP address resolves to a hostname that is matched by one of the defined suffixes.

For example, if the domains ".europe.corp.com,.america.corp.com" were set, and a user's IP address resolved to ldn102.europe.corp.com then they would not be challenged for SSO because it matches .europe.corp.com.

SSO user matching

The integration and configuration interface allows for actions to perform when a user has been authenticated with the SSO system. It is common for the SSO username to differ in some way from the product username, and these features allow the two to be mapped.

This may involve simply changing the case of the username, or with more advanced features such as querying an LDAP, querying the BMC AR System or HP Web Tier database, or running a Javascript function. These advanced features fall under the heading 'User aliasing' in the SSO Plugin integrations user interface.

Default configuration

The default SSO Plugin configuration is suited to most organisations. It assumes the product user names do not contain Windows domain names and is as follows:

1. Strip domain names, ie removing the Windows domain from the SSO username.
2. Match the user in the product case insensitively, to avoid database case sensitive login name issues.
3. When no account is matched in the product, redirect the user to the product login page.

BMC AR System

If no user is matched through the configured user matching rules, the request can be passed to the ITSM user provisioning or raise incident modules.

Query an LDAP

Some deployments require an identifier held in an LDAP for login purposes, yet the SSO system returns a different username.

For example, an Windows Authentication implementation returns a Windows username yet the desired login name is an email address, which is held in the Active Directory.

The product can be configured to look up a value from an LDAP using the username returned from the SSO system. To do this, enable the "Alias by LDAP" option in the integration interface and enter the LDAP connection details:

1. The LDAP authentication URL. This contains the hostname, basedn, attribute, scope and filter in the format ldap://host:port/basedn?attribute?scope?filter. The values are configured as follows:
    a. ldap://host:port: The LDAP URL used to locate the LDAP.
    b. attribute: The attribute on which to search for a username, default uid.
    c. scope: One level (one) or sub-tree (sub), default is sub.
    d. filter: The filter that will be combined with the attribute=username, default (objectClass=*).

    For example, if the filter is set to a=b then the search will use the combined filter: (&(a=b)(attribute=username))
2. The bind DN: This must be the distinguished name (DN) of a user that has access to read the users that will be connecting, ie those defined by the LDAP authentication URL.
3. The password for the bind DN.
4. The return attribute for which a value will be retrieved from the matched LDAP entry.

If no value is returned from the LDAP, the SSO login request will fail.

Querying the database

This allows you to run a query against the product user database to return a product login name using the SSO username (and optionally domain name) as part of the query. You can use any field in the product user database as part of the query.

When writing the query, you can use the following place holders which will be replaced with real values.

1. $SSO_USERNAME$:
    a. The SSO username.
    b. The X509/LDAP username value of the matched user, ie if the user DN is cn=X,ou=Y, ou=Z then SSO_USER is set to X.
2. SSO_DNS_DOMAIN:
    a. The Windows DNS domain name if using Windows Authentication integration. This will always contain the DNS domain name except when a client authenticates via NTLM from a domain not configured with SSO Plugin, ie through Active Directory trust relationship. This is because the DNS domain name isn't available in this scenario.
    b. The Windows DNS domain name if using an IIS front end and a DNS name was sent.
3. SSO_NB_DOMAIN:
    a. The Windows NetBIOS domain name when using Windows Authentication.
    b. The Windows NetBIOS domain name if using an IIS front end and a NETBIOS name was sent.
4. $SSO_USERNAME_DOMAIN$: The domain part of the username. This could be different from SSO_NB_DOMAIN or SSO_DNS_DOMAIN in the event SSO Plugin is configured to authenticate against domain A (and hence, NB and DNS domain return values for domain A), but the user has authenticate from domain B and hence their username could be B\user. This variable returns B in this instance.
5. SSO_PARENT_DN: The parent DN when authenticating with LDAP or X509, ie if the user DN is cn=X,ou=Y,ou=Z then the parent DN is ou=Y,ou=Z.
6. $SSO_ORIGINAL_USER$: The original SSO username before any translation, ie LOCALDOAMAIN\user, user@localdomain.local, etc.
7. $HTTP_HEADER_X$: Where X is an HTTP header value passed from a front end to the Java web server, such as Apache HTTPd running the SiteMinder agent, that passes a number of useful values as SM HTTP headers.
8. $SYSTEM_X$: Where X is an environment variable passed to the JVM, retrieve the value, ie if the JVM is started with -DX=abc then $SYSTEM_X$ will return abc.

For example, if user dkellett is logged into the Windows domain DEVELOPMENT, which has a DNS domain name development.jss.com, then the following variables are set:

1. SSO_USER: dkellett
2. SSO_DNS_DOMAIN: development.jss.com
3. SSO_NB_DOMAIN: DEVELOPMENT
4. $SSO_ORIGINAL_USER$: DEVELOPMENT\dkellett or dkellett@development.jss.com, depending on which Windows Authentication protocol was used during the SSO process.

If you want to pass the value returned from the Windows authentication system (i.e. user@domain or DOMAIN\user) to SSO_USER, do not set user domain to strip domain.

Convert to upper or lower case

The username case selection field in the integrations user interface can be used to change the case of some variables. If it is set to "To upper" or "To lower", the following variables will be modified: SSO_USER, SSO_DNS_DOMAIN, SSO_NB_DOMAIN and SSO_PARENT_DN.

For example, consider the SSO username is DKellett@localdomain.local and the query '8'="SSO_USER@$SSO_DNS_DOMAIN". If the username case field is set to "To upper", the query '8'="DKELLETT@LOCALDOMAIN.LOCAL" will be created.

Typical use cases

BMC AR System

Case 1

If the User form holds the SSO usernames in field 536870912 and the SSO user is held in this field, set the alias query to '536870912' = "SSO_USER".

When this query is executed against the User form, the SSO_USER string is replaced with the username, and the value for the Login Name field is returned. This value is then used to connect to AR System.

Case 2

If you have a policy of storing all SSO accounts in the format user@dns.domain within field 536870912 on the User form, use the following query:

'536870912' = "SSO_USER@$SSO_DNS_DOMAIN".

Please note

Do not use fields 117 (or 118), known as the "authentication fields". These have been reserved by BMC and AR System behaves differently when passing information to AREA plugin when these fields are populated.

HP Service Manager

Case 1

If the Operators table has login names that do not match the SSO usernames, a new field called SSOid can be added and the SSO usernames placed in this field.

The alias query is set to SSOid="SSO_USER".

When this query is executed against the Operators table, the SSO_USER string is replaced with the username, and the value for the Login Name field is returned.

This value is then used to connect to Service Manager.

For example, if the SSO username is bob and the Operator entry with Login Name john has bob in the SSOid field, a user with the SSO username bob will be logged into Service Manager as user john.

Case 2

Consider a deployment where the SSO username, in the format user@dns.domain, is stored in the field SSOid. However, there are existing users with the SSO username stored in the Login Name field.

The following query will search for an Operator record with SSOid set to user@dns.domain and if one is not found, a record with the Login Name set to user@dns.domain.

SSOid="SSO_USER@SSO_DNS_DOMAIN"|name=" SSO_USER@SSO_DNS_DOMAIN"

Note, the Login Name field is referenced by the field name when using a query, because name is defined out of the box in the Operator webservice, updated during the installation process.

Run Javascript query

This feature is for advanced use cases where the SSO username requires considerable modification in order to create the desired username. The Javascript (executed by the nashorn engine) must return either:

- A single string that represents the translated username.
- An array of two strings, the first representing the domain and the second a username (at present, only the username is retained).

This feature is incredibly useful if the SSO username is not in a format where by SSO Plugin can split it into domain & username. For example, if the SSO username is domain_username, then the following script would extract the two parts:

```
ssoUsername.split('_');
```

Here is an example of how to write a function and return its value:

```
var f= function() { var v= ssoUsername.toLowerCase(); return "mycorp"+v; }
f();
```

The following variables are set as global variables when executing the script:

- ssoUsername: The SSO username provided by the SSO integration.
- userAliasingVariables: A dictionary of user aliasing variables, ie username [without domain], originalUsername [same as ssoUsername above], dnsDomain, nbDomain. The configuration document details all variables for specific integrations.
- userFormQuery: When running against BMC AR System, this variable allows you to query the User form, ie userFormQuery.loginName("'101=\"xyz\"").

Contact JSS support if you require assistance in writing Javascript functions.

Detailed overview of the username matching process

The following illustrates the process of deciding how to match an SSO user to a product user:

1. If User domain is set to 'strip domain' then a Windows domain name will be stripped from the SSO username. If the username a distinguished name (DN), as it would be if using X509/CAC then this feature extracts the cn or uid value (ie X from "cn=X, ou=Y, ou=Z.." or "uid=X, ou=Y, ou=Z..").
2. If case sensitivity is set to convert to upper or lower then username is modified.
3. Perform one of the following:
    1. If user aliasing is enabled, execute query against the product user database and if a user is returned, login with that user.
    2. If match case insensitively is selected then search for a user.
    3. If user domain is set to try matching either way, search for a user entry with or without the Windows domain name.
    4. Query the product user database to see if a user exists matching the SSO username when 'strip domain' set, or the domain name and username when 'use domain and username' set.

If there is no match for any of the username matching options then run the action selected for when a user has no SSO account.

Automation

When using BMC AR System or HP Service Manager, SSO Plugin will attempt to ensure an account is SSO enabled during the login process. The following actions are performed automatically:

- For BMC AR System, disable password management is set to true, and user must change password is set to false.
- For HP Service Manager, expire password is set to false.

These actions are performed because an SSO user should not be presented with a user interface to change their password when they login via SSO.

Actions available when a user has no SSO account

The SSO Plugin provides a range of options to deal with the scenario where an SSO user has no valid SSO account in the product. This forms an important part of integrating the SSO Plugin into a corporate environment where there are thousands of users, with many leaving and joining on a daily basis.

The configuration page contains a section marked 'when SSO user has no valid User form entry' and the options are described below.

Redirect user to login page

Users who do not have an SSO enabled account in the product will be redirected to the product login form.

For BMC AR System & HP Service Manager, the browser redirects to the JSS login page and if SSO Plugin has been configured to integrate with Windows Authentication that includes NTLM, an option to login with Windows credentials is provided. A Windows login requires an SSO enabled account in the User form.

If a user login is successful, the credentials are passed through the matching process as if the user had passed through SSO. This means that user aliasing etc. will be actioned when retrieving the user from the product for which to login.

Simplifying product configuration

This functionality allows you to remove the product LDAP integration configuration (ie BMC AREA LDAP plugin(s) on BMC AR System) because by doing so, you reduce the number of components to configure in product.

For BMC AR System and HP Service Manager, the Windows client tools requires the product LDAP configuration so if users need to login to the client with their Windows credentials, the plugin must still be configured. However, for many implementations, users only require browser access and hence this functionality will be beneficial.

Dynamically creating a user account in BMC AR System

The feature is specific to BMC AR System but will be ported to HP Service Manager in the future.

When a user has no account in the User form, a new entry can be created from a template entry. When this option is selected, the name of an existing User form entry will be required to be used as a template for new entries. The new entry will be created by copying all fields from the template entry, replacing the login name with the SSO user name from the request.

If a user has an existing entry but it's not correctly configured then they will be redirected to the login page.

Redirect to the ITSM user registration page

The feature is specific to BMC AR System but will be ported to HP Service Manager in the future.

If the user doesn't have an SSO enabled account, they will be redirected to a page that allows them to enter some information on themselves (ie first name, last name, phone number, email address) and a new ITSM People record will be created.

Assuming this process is completed successfully, the user will have access to BMC ITSM without having to call the service desk and ask someone to manually create their account.

Raise an incident in BMC ITSM

The feature is specific to BMC AR System but will be ported to HP Service Manager in the future.

If the user doesn't have an SSO enabled account, or an error occurs creating one using the create User form or ITSM user registration functionality, an incident will be raised and the user will be redirected to a page with the incident number.

To configure this feature, use the SSO Plugin Administration Console within ITSM to map values into the new incident. This feature is covered in more detail in the installation guide for BMC AR System.

Authenticate unregistered users

This feature is specific to BMC AR System. The AR System server has a feature to authenticate unregistered users, where an unregistered user is defined as not having an entry in the User form.

SSO Plugin looks for the SSO user in the User form before passing back the user to Mid Tier, and allows a login page to be displayed if there is no match in the User form. However, if authenticate unregistered users is enabled, the administrator may wish SSO Plugin to pass the SSO user back to Mid Tier even when there is no entry in the User form, so the SSO user is logged in as an unregistered user.

To enable 'authenticate unregistered users' in AR System, log in as an administrator and go to AR System Administration / Server Information and locate the EA tab..

If AR System isn't correctly configured, the default BMC Mid Tier behaviour (an ARERR623 error page) will be exhibited after SSO Plugin has passed the SSO user into Mid Tier.

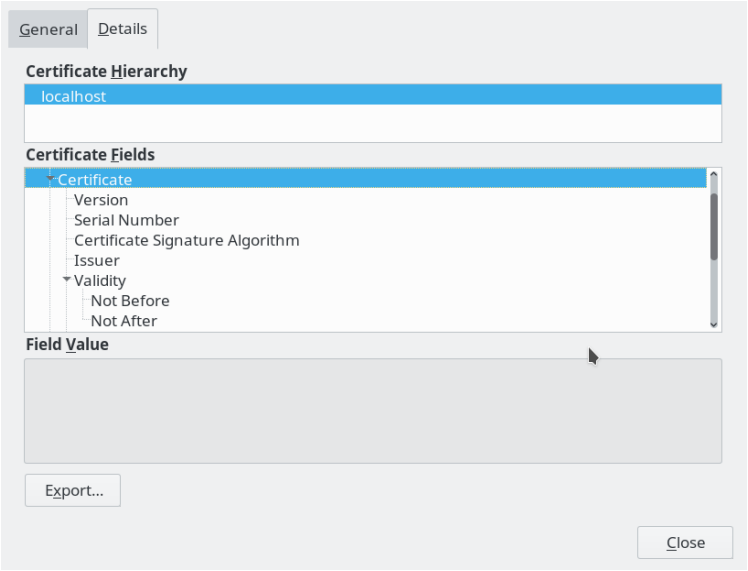
Configure the JVM with an SSL certificate

Follow these steps if you see an error such as "Unable to find valid certificate".

If the target server (ie LDAP, OAuth2 IDP or SAML IDP) makes use of a self-signed SSL certificate then this will need to be imported into a trust store or the JVM's cacerts file (located in the JRE installation directory).

To export and import the certifcate:

1. Navigate to the OAuth2 or SAML IDP in your browser (if LDAP, you will need to ask an administrator for ti), click on the padlock and select export certificate, saving to a file called oauth2.cert. This process will vary depending on browser and the following is a screenshot from Firefox:



2. Use the keytool command, part of the JRE installation, to import the certificate into the JRE cacerts file as follows:

```
keytool -keystore /local/jvm/java-8-sun/jre/lib/security/cacerts -
importcert -alias pf-localhost -file oauth2.crt
```

It is possible to import the certificate into a different keystore and configure the JVM to use it instead of the default cacerts file.

If a separate trust store (ie not using cacerts) is created, the path and password (to the trust store) need to be configured as JVM options:

```
-Djavax.net.ssl.trustStore=/path/to/truststore
-Djavax.net.ssl.trustStorePassword=truststore-password
```

There are many online examples of importing self-signed certificates into a trust store and JSS are happy to help with this often complicated subject.

The SSO Plugin jss.sso.ignore.ssl.host.verification environment variable can be set to trust any remote host. It is set as an option to the JVM as follows, but this should only be used in non-production:

```
-Djss.sso.ignore.ssl.host.verification=true
```

We recommend that the JVM is correctly configured to trust the SSL certificate in a production environment.

Obtaining SSL certificate from Microsoft IIS

Open the IIS Manager, click on the server home, select 'Server Certificates', right click on the relevant certificate and export as a PFX file. This contains both the public and private certificates, and the following command can be used to export the public certificate:

```
openssl pkcs12 -in exported-from-iis.pfx -nokeys -out cert.pem
```

The public certificate can then be imported into a keystore as follows:

```
keytool -importcert -file cert.pem -keystore cacerts -alias server.hostname
-storepass password -noprompt
```

SAML/OAuth2 and reverse proxies (such as an F5)

If you have deployed one or more web fsapplications behind a reverse proxy, typically for load balancing purposes, and you're using SAML or OAuth2, a further configuration step is required in order to tell SSO Plugin about the user facing URL.

This is required because the SAML and OAuth2 protocols require the URL on which the user accessed the application to be sent in the request for authentication, and whilst the user may have entered https://myitsm.mycompany.com into a browser, where myitsm.mycompany.com is the load balancing hostname, this is not the URL provided to SSO Plugin by the webserver.

Consider the following infrastructure:

1. Two Tomcat servers with URLs http://tomcatfarm1.mycompany.com:8080 and http://tomcatfarm2.mycompany.com:8080 are configured with the SSO enabled web application.
2. An F5 load balancer acting as a reverse proxy is configured with the URL https://myitsm.mycompany.com, which balances traffic across the two Tomcat instances.

When a user accesses https://myitsm.mycompany.com and is routed to one of the Tomcat servers, ie tomcatfarm1, SSO Plugin intercepts the request and if unauthenticated, asks Tomcat for the URL that the user accessed. The Tomcat server does not return https://myitsm.mycompany.com but instead its own URL, ie http://tomcatfarm1.mycompany.com:8080.

SSO Plugin proceeds to create a URL with http://tomcatfarm1.mycompany.com:8080 and sends it to the SAML IDP, which after authenticating the user, returns the browser to http://tomcatfarm1.mycompany.com:8080 not https://myitsm.mycompany.com. At this point the process fails.

To overcome this limitation in infrastructure design, SSO Plugin should be configured with the user facing URL by entering it into the Reverse Proxy URL field on the integrations page. You need only enter the URL's protocol, hostname and port, ie https://myitsm.mycompany.com.

Multiple SSO integrations

The product can provide multiple SSO integrations through the use of differing hostnames within the URL on which a user access the web application or an encrypted value stored in a cookie called jss.sso.configuration.

This is configured through the Integrations page. Simply type in a hostname or cookie value on which a user can connect to the product, click add, and configure the settings for that hostname. The drop down box can be used to select the integration to edit.

Use cases

There are two common use cases for this functionality:

1. If a Mid Tier serves both internal and external users, different hostnames can be used to provide different SSO services, such as Windows Authentication for internal users and SAML for external users.
2. If a Multi-Service Provider is supporting many clients on a group of Mid Tiers, they can use different hostnames to support different SSO integrations for each client.
3. If a Multi-Service Provider is supporting many clients on a group of Mid Tiers, they can use the encrypted cookie to hold the ITSM company name and set up integrations for each ITSM company name. This provides a single URL for Mid Tier access, with the cookie allowing SSO Plugin to know which SSO integration should be presented.

BMC ITSM Multi Service Provider registration

If an MSP wishes to use a cookie based approach to distinguishing users, a page has been provided within the SSO Plugin product to allow a user to select a value, encrypting the value and storing in a cookie. The values from which the user can select are populated by querying a form and is configured through the configuration interface.

Once the user has registered, the company's SSO integration will be presented when they access ITSM (ie /arsys/home).

The MSP registration page can be found at http://host/arsys/jss-sso/msp and can be easily customised by the MSP.

Customising user facing messages

You can customise user facing error messages, such as "The user aliasing query failed to return a user" in the the following:

The messages are stored in a file called messages.properties that is contained within the jss-sso.jar file. Follow these steps:

1. Open the jss-sso.jar file in your favourite zip file viewer.
2. Extract the messages.properties file and place it in the application classpath, ie the WEB-INF/classes directory.
3. Alternatively, run these two commands if you have zip and unzip installed:

```
unzip jss-sso.jar messages.properties
zip -d jss-sso.jar messages.properties
```

4. You can now edit the messages.properties file in your favourite text editor, for example:

```
TestUserAccountResults.6=Please contact the service desk at
help@mycompany.com.
```

Weblogic and SAML

Weblogic uses its own XML libraries that are not compatible with the Oracle Java (default) XML libraries invoked by SSO Plugin. Therefore, if using SAML with Weblogic, the following changes (taken from Weblogic documentation) must be made to the Weblogic server:

1. Extract all the files from <Weblogic Path>/modules/com.bea.core.weblogic.stax_1.10.0.0.jar

```
cp -rp com.bea.core.weblogic.stax_1.10.0.0.jar <tmpdir>
cd <tmpdir>
jar -xvf com.bea.core.weblogic.stax_1.10.0.0.jar
```

2. Change directory to the META-INF/services.
3. Modify the contents of the following files:

File	Original text	New text
javax.xml.stream.XMLInputFactory	weblogic.xml.stax.XMLStreamInputFactory	com.sun.xml.internal.stream.XMLInputFactoryImpl
javax.xml.stream.XMLOutputFactory	weblogic.xml.stax.XMLStreamOutputFactory	com.sun.xml.internal.stream.XMLOutputFactoryImpl
javax.xml.stream.XMLEventFactory	weblogic.xml.stax.EventFactory	com.sun.xml.internal.stream.events.XMLEventFactoryImpl

4. Repack the three files into a jar file.

```
jar -cvf jss_xmlparser.jar javax.xml.stream.XMLInputFactory
javax.xml.stream.XMLOutputFactory javax.xml.stream.XMLEventFactory
```

5. Copy the jar file into the WEB-INF/lib directory of your application.
6. Modify the application deployment descriptor (the weblogic.xml file in the application WEB-INF directory) to use filtering classloader to load the three files from the application:

```
<container-descriptor>
  <prefer-web-inf-classes>false</prefer-web-inf-classes>
  <prefer-application-resources>
    <resource-name>META-INF/services/*</resource-name>
  </prefer-application-resources>
 </container-descriptor>
```

7. Restart the application.

Redistribution

Like most modern commercial products, SSO Plugin makes use of a number of open source third party libraries. SSO Plugin is distributed with, and gives thanks to, the following open source products that are redistributed: Apache commons codec, Apache commons HTTP client, Apache commons logging, jcifs, Java Native Access, log4j, slf4j, Picketlink, xmlsec, Google OAuth and Bouncy Castle. More information on these products can be found on their relevant websites.