Configuring the JBoss Application Server for Secure Sockets Layer and Client-Certificate Authentication on SAS® 9.3 Enterprise BI Server Web Applications

Table of Contents

Overview

This document explains how to configure one-way Secure Sockets Layer (SSL), two-way SSL, and client-certificate authentication with the JBoss application server for use with SAS® 9.3 Enterprise BI Server Web applications. The document explains the following steps in detail:

- configuring the system for one-way SSL
- configuring your system for Web authentication (optional step for one-way SSL only)
- configuring your system for two-way SSL
- configuring client-certificate authentication

Note: To configure client-certificate authentication, your system must first be configured for basic Web authentication. The procedure for configuring Web authentication is covered in *Configuring JBoss Application Server 4.3 and 5.1 for Web Authentication with SAS® 9.3 Web Applications*. For more details, see the section "Configuring JBoss for Web Authentication," later in this document. If you are configuring one-way SSL only, Web authentication is optional. You can use the default SAS host authentication instead.

Configuring One-Way SSL Authentication

One-way SSL authentication requires that a server machine provide a valid certificate to a client machine so that the client can verify the identity of the server. However, because the authentication is only one way, the client is not required to provide a certificate to the server.

In this document, the *client* is the Web browser on which the Web application is displayed. The *server* is JBoss, from which the application actually runs.

The basic steps for setting up one-way SSL are as follows:

1. Create a server-identify certificate, sign it with a certificate-authority (CA) certificate, and add it to a Java keystore.
2. Add the CA root certificate (used to sign the server-identity certificate) to a keystore.
3. Import the server's CA certificate into your browser's trusted signer's area.
4. Configure JBoss so that it can locate the keystore that contains the server-identity certificate.
5. Configure SAS Remote Services so that it can locate the keystore that contains the CA root certificate.
6. Use SAS® Management Console to change the connection information for the SAS Logon Manager and the set of SAS Web applications that you want to access via Hypertext Transfer Protocol Secure (HTTPS). The Web applications should include the SAS® Content Server.

These topics are addressed in the following sections.

Creating the Certificates and the Keystore That Are Required for One-Way SSL

This section explains how to create the certificates and keystores that you need in order to configure one-way SSL. The certificates are created with the OpenSSL toolkit. The keystores are managed with the Java keytool utility, which is shipped with the Oracle Java Development Kit.

Installing and Configuring the OpenSSL Toolkit

You can obtain OpenSSL from the Internet. If you plan to install the program in a Windows operating environment, you can find and use a binary distribution. Otherwise, you can download the source and build the program.

Before you create a self-signed certificate in the sections that follow, you need to perform the following steps:

1. Create a directory (**C:\*directory***), in which to store all of the certificates and keystores. (**Example: C:\SSL**)
2. In **C:\*directory***, create the following subdirectories:
   - **C:\*directory*\demoCA**
   - **C:\*directory*\keys**
   - **C:\*directory*\certs**
   - **C:\*directory*\requests**
3. In **C:\directory\demoCA**, create the following subdirectory:

   C:\*directory*\demoCA\newcerts
4. Copy ***OPENSSL-directory*\bin\PEM\demoCA\serial** file to **C:\directory\demoCA**, as shown in the following example:

   ```
   "C:\SSL\demoCA>copy c:\OpenSSL\bin\PEM\demoCA\serial"
   ```
5. Create an empty text file called index.txt in the **demoCA** subdirectory:

   C:\*directory*\demoCA\index.txt

Creating the Certificates and Keystores

This section provides step-by-step instructions for creating the required certificates and keystore that are necessary for one-way SSL. In general, the example does the following:

- creates a CA certificate that you can use to sign a server-identity certificate
- creates a server-identity certificate and a keystore to hold it
- imports the CA certificate to a trusted signer's keystore

The following example assumes that certificate-related files and keystores are placed in a location (for example, ***JBoss-home-directory*\server\SASServer1\conf**) that is relative to the JBoss server. However, that does not necessarily have to be the case. In addition, the commands that are used in this example represent one of many ways to accomplish the tasks. For example, rather than submitting each entry at a prompt, OpenSSL enables you to specify a distinguished name (DN) on the command line. For more details, see “OpenSSL Cryptography and SSL/TLS Toolkit” (**www.openssl.org/**) and “keytool – Key and Certificate Management Tool” (**docs.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html**).

For simplicity, the examples use the same password for all certificates and keystores, but they can be defined as unique passwords as well.

To create the certificates and the keystore:

1. Generate the CA private key by using the following command syntax at a system prompt:

   ```
   openssl genrsa -des3 -out keys/CA-key-name.pem 1024
   ```

   In this syntax, *CA-key-name* specifies the name of the CA private key.

   Note: The private key should be stored in a secure location in a production environment.

   The following example illustrates using this syntax with an actual value for the placeholder value. Examples are provided throughout this document for all command syntax that is shown.

   Example:

   ```
   openssl genrsa –des3 –out keys/myCAK.pem 1024
   ```

   After you submit this command, the system prompts you for a passphrase that is required in order to access the private key. Enter the passphrase when you are prompted. In subsequent steps, the placeholder *passphrase.*is used in places where you need to supply your passphrase.

2. Generate the CA root certificate by using the following command syntax:

   ```
   openssl req -new -key keys/CA-key-name.pem -x509 -days 1024
               -out certs/CA-certificate-name.crt
   ```

   In this command, *CA-certificate-name* specifies the name of the CA root certificate.

   For testing purposes, this example creates a CA certificate. However, you also can use a self-signed certificate. Fill in the fields when you are prompted, and provide the same passphrase that you specified in step 1.

   Note: For the fields in the distinguished name, you can use any values except when you are prompted for a value for **`Common Name`**. For this field, enter the name that you want to appear in user's list of trusted root certificates when the user imports the certificate into a browser.

   You can also just add all of the information in the command at one time, as follows, thereby avoiding the prompts:

   ```
   openssl req -new -key keys/CA-key-name.pem -x509 -days 1024
               -subj /C=US/OU=TSD/O=SAS/ST=NC/L=Cary/CN=CA-name
               -out certs/CA-certificate-name.crt
   ```

   In this instance, *`CA-name`* specifies a name that you want to use to reference the CA.

   Example:

   ```
   openssl req –new –key keys/myCAkey.pem –x509 –days 1024 –subj
        /C=US/OU=TSD/O=SAS/ST=NC/L=Cary/CN=sasbi -out certs/myCA.crt
   ```

3. Generate the server-identity keystore, as follows:

   ```
   keytool -genkey -alias alias-name -keyalg RSA -keystore serverids.jks
           -keypass key-password -storepass store-password
   ```

   In this syntax, *`key-password`* and *store-password* specify, respectively, the passwords for the keystore and the trusted store. Fill in the appropriate fields when you are prompted, including the passphrase that you specified in step 1.

(list continued)

Notes:

- You can use the short name of the server for the alias value rather than the full address; for example, you can use **`S1234`** rather than **`S1234.`*`domain.name`*`.com.`**
- The values for *`key-password`* and *`store-password`* **must** be the same.
- In the example below, the value for **`CN=`** must be the host name.

Example:

```
keytool -genkey -alias JBossOneWayCert -keyalg RSA
        -keystore serverids.jks -keypass Jboss4 -storepass Jboss4
        -dname "CN=sasbi.demo.sas.com,OU=blades,O=SAS Institute,
        L=Cary,S=NC,C=US"
```

4. Generate the server-identity request file, as follows:

```
keytool -certreq -alias alias-name -keystore serverids.jks
        –keypass key-password -storepass store-password
        -file requests/certificate-signing-request-filename.csr
```

The server-identity request file contains the server public key and the server identity.

The value for *certificate-signing-request-filename* can be whatever name you want to use. For example, you can use your host name as the name of the generated .csr file.

Example:

```
keytool -certreq -alias JBossOneWayCert -keystore serverids.jks
        -keypass Jboss4 -storepass Jboss4 -file requests/JBoss.csr
```

5. Generate the server-identity certificate by submitting a command with the following syntax:

```
openssl x509 -req -days 365 -in requests/certificate-signing-request-
        filename.csr -CA certs/CA-certificate-name.crt -CAkey keys/CA-key-
        name.pem -CAcreateserial -out certs/server-certificate-name.crt
```

This certificate contains the server public key, the name of the machine on which JBoss runs, and the CA identity. It is also signed with the CA private key.

Example:

```
openssl x509 -req -days 365 -in requests/JBoss.csr -CA certs/myCA.crt
             -CAkey keys/myCAkey.pem -CAcreateserial -out certs/JBoss.crt
```

6. Import the CA root certificate into the server-identity keystore.

```
keytool -import -alias alias-name –file certs/CA-certificate-name.crt
        –keystore serverids.jks
```

Example:

```
keytool -import -alias JBossRootCert -file certs/myCA.crt

        -keystore serverids.jks
```

7. Import the server-identity certificate into the server-identity keystore, using the short host name for the alias value, as you did in step 3.

```
keytool -import -alias alias-name -file certs/server-certificate-name.crt

        -keystore serverids.jks
```

Example:

```
keytool -import -alias JBossOneWayCert -file certs/JBoss.crt
        -keystore serverids.jks
```

8. Generate the Trusted CA Certificates keystore and import the CA root certificate into the trust store.

```
keytool –import –alias alias-name –file certs/CA-certificate-name.crt
        –keystore trusted-certificate-keystore.jks
```

Example:

```
keytool -import -alias JBossRootCert -file certs/myCA.crt
        -keystore trustedca.jks
```

If you use the examples given after each of the previous steps, you will create the following files (shown in red) in the SSL directories that are listed here:

- ***SSL_directory*\keys\myCAkey.pem**: The private key for the CA certificate that is created in step 1.

  (*CA-key-name*.pem)
- **SSL_directory\certs\myCA.crt**: The CA certificate that is created in step 2. (*CA-certificate-name*.crt)
- **SSL_directory\serverids.jks**: The keystore that contains the server certificates. This keystore is created in step 3. (serverids.jks)
- **SSL_directory\requests\JBoss.csr**: The certificate signing request file that is created in step 4. (*certificate-signing-request-filename*.csr)
- **SSL_directory\certs\JBoss.crt**: The server certificate that is created in step 5. (*server-certificate-name*.crt)
- **SSL_directory\trustedca.jks**: The keystore that contains the CA certificates. This keystore is created in step 8. (*trusted-certificate-keystore*.jks)

You now have all of the certificates that you need to configure one-way SSL.

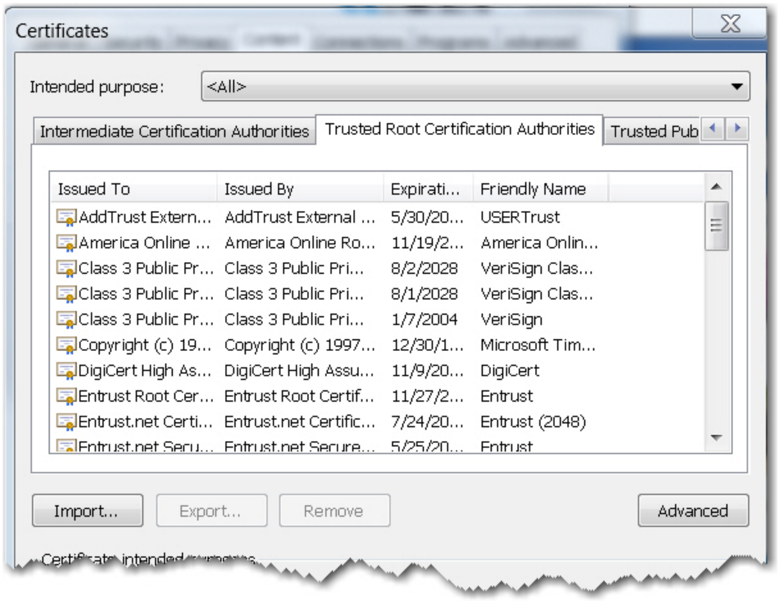
Importing the CA Root Certificate into Your Browser

You need to import the CA root certificate named *CA-certificate-name*.crt (for example, myCA.crt) that you created in the last section into your browser so that the browser trusts the server's certificate. Copy the certificate to the machine where you access the browser.

To import this certificate into Microsoft Internet Explorer:

1. Select **Tools** ► **Internet Options** ► **Content** ► **Certificates** ► **Trusted Root Certification Authorities**.

   Note: Depending on what version of the browser you have, this path might be different.


Certificates
Intended purpose:
<All>
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Pub
Import...
Export...
Remove
Advanced
Certificate intended purposes

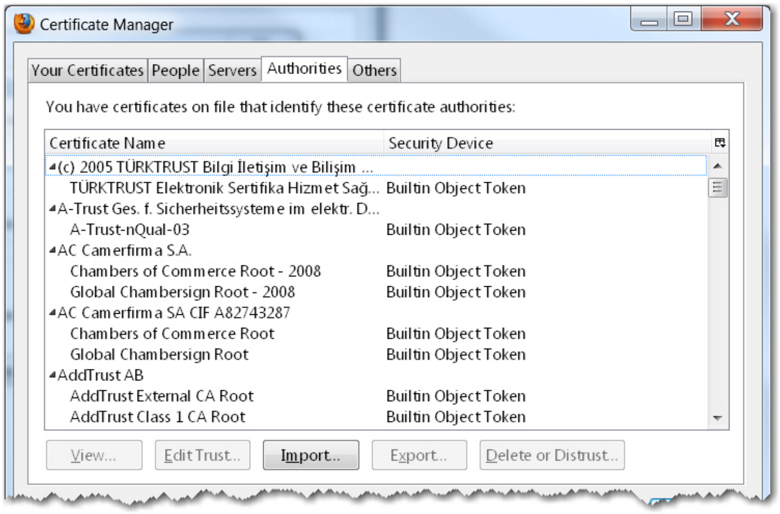
Issued To	Issued By	Expirati...	Friendly Name
AddTrust Extern...	AddTrust External ...	5/30/20...	USERTrust
America Online ...	America Online Ro...	11/19/2...	America Onlin...
Class 3 Public Pr...	Class 3 Public Pri...	8/2/2028	VeriSign Clas...
Class 3 Public Pr...	Class 3 Public Pri...	8/1/2028	VeriSign Clas...
Class 3 Public Pr...	Class 3 Public Pri...	1/7/2004	VeriSign
Copyright (c) 19...	Copyright (c) 1997...	12/30/1...	Microsoft Tim...
DigiCert High As...	DigiCert High Assu...	11/9/20...	DigiCert
Entrust Root Cer...	Entrust Root Certif...	11/27/2...	Entrust
Entrust.net Certi...	Entrust.net Certific...	7/24/20...	Entrust (2048)
Entrust.net Secu...	Entrust.net Secure...	5/25/20...	Entrust

2. Click the **`Import`** button to start the Certificate Import Wizard, and follow the wizard's instructions. You need to supply the path to the certificate file. For all other values, accept the default values.

To import this certificate into Mozilla Firefox:

1. Select **Tools** ► **Options** ► **Advanced** ► **View Certificates** ► **Authorities**.

   Note: Depending on what version of the browser you have, this path might be different.



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
(c) 2005 TÜRKTRUST Bilgi İletişim ve Bilişim ...	
TÜRKTRUST Elektronik Sertifika Hizmet Sağ...	Builtin Object Token
A-Trust Ges. f. Sicherheitssysteme im elektr. D...	
A-Trust-nQual-03	Builtin Object Token
AC Camerfirma S.A.	
Chambers of Commerce Root - 2008	Builtin Object Token
Global Chambersign Root - 2008	Builtin Object Token
AC Camerfirma SA CIF A82743287	
Chambers of Commerce Root	Builtin Object Token
Global Chambersign Root	Builtin Object Token
AddTrust AB	
AddTrust External CA Root	Builtin Object Token
AddTrust Class 1 CA Root	Builtin Object Token

2. Click the **`Import`** button to start the Certificate Import Wizard, and follow the wizard's instructions. You need to supply the path to the certificate file. For all other values, accept the default values.

Configuring the JBoss Application Server for One-Way SSL

To configure JBoss for one-way SSL:

1. Open the server.xml file for your JBoss server. For the SASServer1 server in JBoss Application Server 4.3, server.xml is located in ***JBoss-home-directory*\server\SASServer1\deploy\jboss-web.deployer\server.xml**. For JBoss Application Server 5.1, server.xml is located in ***JBoss-home-directory*\server\SASServer1\deploy\jbossweb.sar\server.xml**

2. Remove the comment delimiters around the **<Connector>** element and add the **keystoreFile, keystorePass**, and **emptySessionPath** parameters, as shown below.

   ```
   <Connector port="port-value" protocol="HTTP/1.1" SSLEnabled="true"
              maxThreads="150" scheme="https" secure="true"
              clientAuth="false"
              address="${jboss.bind.address}"
              strategy="ms"
              useBodyEncodingForURI="true"
              keystoreFile="${jboss.server.home.directory}/conf/serverids.jks"
              keystorePass="password"
              emptySessionPath="true"
              sslProtocol="TLS" />
   ```

   The password that you specify here should be the same keystore password that you specified in step 8 of the section "Creating the Certificates and Keystores."

   Note: The **emptySessionPath** parameter is required for SAS BI Portlets when you use SSL. For more information, see "Enabling SAS BI Portlets for SAS Information Delivery Portal 4.31 on JBoss" in the *SAS® 9.3 Intelligence Platform: Web Application Administration Guide, Third Edition*. (**support.sas.com/documentation/cdl/en/biwaag/65230/PDF/default/biwaag.pdf**)

3. Specify the path to the trust keystore in the Java Virtual Machine (JVM) parameters for each JBoss server instance where the SAS Web applications are deployed. To do that, you need to add the following parameters:

   - If JBoss is installed as a service, add the following parameters in the wrapper.conf file that is located in the server directory for each SASServer*x* instance.

     ```
     wrapper.java.additional.nn=-Djavax.net.ssl.trustStore="path-to-
                                                   keystore\trustedca.jks"
     wrapper.java.additional.nn=-Djavax.net.ssl.trustStorePassword=password
     wrapper.java.additional.nn=Dsas.auto.publish.protocol=https
     ```

     Note: The number specified as *nn* in **wrapper.java.additional.*nn*=** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.

   - If JBoss is started manually, modify the SASServer*x*.bat or SASServer*x*.sh for each SAS server instance and add the following parameters:

     ```
     -Djavax.net.ssl.trustStore="path-to-keystore\trustedca.jks"
     -Djavax.net.ssl.trustStorePassword=password
     ```

(list continued)

4. Change the port number in the following parameter from **`8080`** to the port value that you use in step 2 of this section for **`<Connector port="`*`port-value`*`". . .>`**. (**Note:** The default HTTPS/SSL port for JBoss is **`8443`**, and that value is used in this example.)

```
wrapper.java.additional.36=Dsas.auto.publish.port=8443
```

Configuring SAS Remote Services for One-Way SSL

You need to specify the path to the Trusted CA keystore and its associated password in the wrapper.conf file for SAS Remote Services. You specify the location of the Trusted CA keystore and its password in the following JVM parameters, which you must add to wrapper.conf:

- If SAS Remote Services is installed as a service, add the following parameters to the wrapper.conf file, which is located in ***`SAS-configuration-directory`*`\ Lev1\Web\Applications\RemoteServices`**.

  ```
  wrapper.java.additional.nn=-Djavax.net.ssl.trustStore="path-to-keystore\
                                                           trustedca.jks"

  wrapper.java.additional.nn=-Djavax.net.ssl.trustStorePassword=password
  ```

  Notes:

  - In the second parameter above, the value for *password i*s the same password that you use in step 8 of the section "Creating the Certificates and Keystores."
  - The number specified as *nn* in **`wrapper.java.additional.`*`nn`*`=`** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.

- If SAS Remote Services is started from a batch file, add the following parameters to both the **`start2`** and **`start3`** sections in the SASServer*x*.bat file or the SASServer*x*.sh file:

  ```
  -Djavax.net.ssl.trustStore="path-to-keystore\trustedca.jks"

  -Djavax.net.ssl.trustStorePassword=password
  ```

  Note: In the second parameter above, the value for *password i*s the same password that you use in step 8 of the section "Creating the Certificates and Keystores."

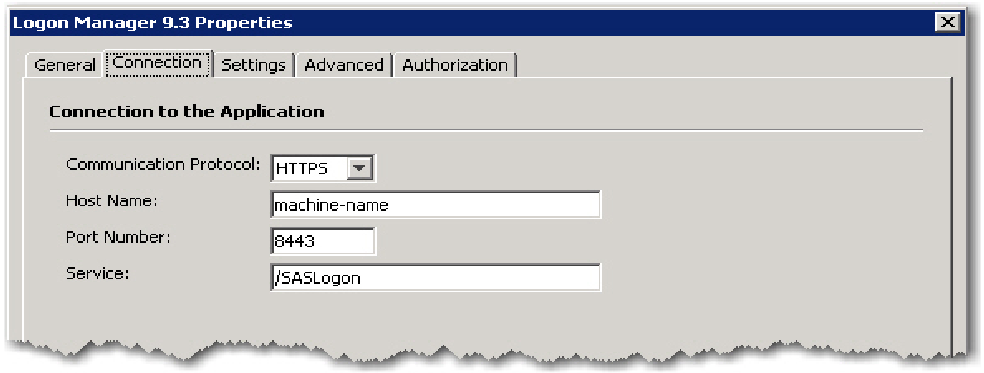
Updating the SAS Metadata

Configuring the Connection Metadata for SAS Web Applications

You communicate with your SAS Web applications via HTTPS. However, first you must use the Configuration Manager plug-in in SAS Management Console to change the connection information for the SAS Logon Manager and the other SAS Web applications (for example, SAS® Information Delivery Portal) to which you want to connect via HTTPS. You need to change the connection information because in SAS 9.3 the SAS Logon Manager does not direct you to applications by using information from a URL that you enter in the browser (other than the application name). Instead, the logon manager uses information that is stored in the SAS® Metadata Repository. Each application is represented by a metadata object that has properties such as a communication protocol, a host name, a port number, and a service name (context root).

To configure the connection information for SAS Web applications:

1. Open SAS Management Console.
2. On the **`Plug-ins`** tab in the left pane, select **Application Management** ► **Configuration Manager** ► **SAS Application Infrastructure**.
3. In the right pane, right-click **`Logon Manager 9.3`** and select **`Properties`** from the pop-up menu.
4. In the Logon Manager 9.3 Properties dialog box, click the **`Connection`** tab.
5. On the **`Connection`** tab, set **`Communication Protocol`** to **HTTPS** and set **`Port Number`** to the port number that you used previously for **`<Connector port="`*`port-value`*`". . .>`** in step 2 of the section "Configuring the JBoss Application Server for One-Way SSL." (**Note:** The default HTTPS/SSL port for JBoss is **`8443`** is used as the value in the following display.)



Connection to the Application

Communication Protocol: HTTPS
Host Name: machine-name
Port Number: 8443
Service: /SASLogon

6. After you enter the port number, click **`OK`**.
7. Repeat steps 3 through 6 for the SAS Web applications that you want to access to via HTTPS.

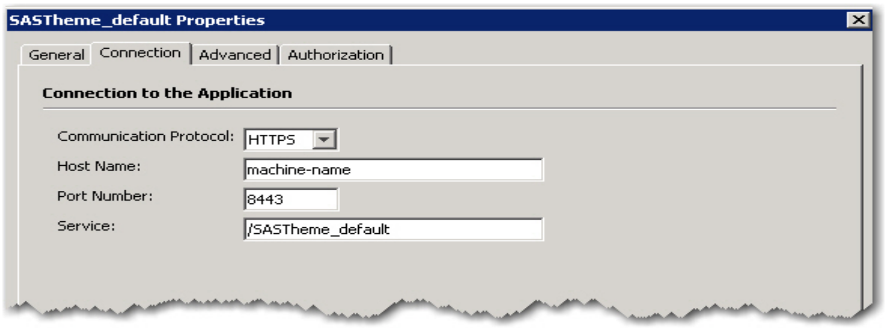
SAS Themes and SAS Help both consist of static content. You might want to configure these applications for HTTPS, depending on your performance and security requirements. If you configure these applications for HTTP only, you will receive mixed content warnings from the browser. These warnings might not be acceptable in your environment.

To configure SAS Themes and SAS Help Viewer for SSL:

1. In SAS Management Console, select **Plug-ins** ► **Application Management** ► **Configuration Manager**.
2. In the right pane, right-click the entry that you want (either **`SASTheme_default`** or **`Help Viewer for Midtier App 9.3`**) and select **`Properties`** from the pop-up menu.

   Note: For this example, **`SASTheme_default`** is selected.
3. In the properties dialog box, click the **`Connection`** tab.
4. On the **`Connection`** tab, set **`Communication Protocol`** to **`HTTPS`** and set **`Port Number`** to the port number that you used previously for **`<Connector port="`*`port-value`*`". . .>`** in step 2 of the section

"Configuring the JBoss Application Server for One-Way SSL." (**Note:** The default HTTPS/SSL port for JBoss is **8443**, which is used as the value in the following display.)



Connection to the Application

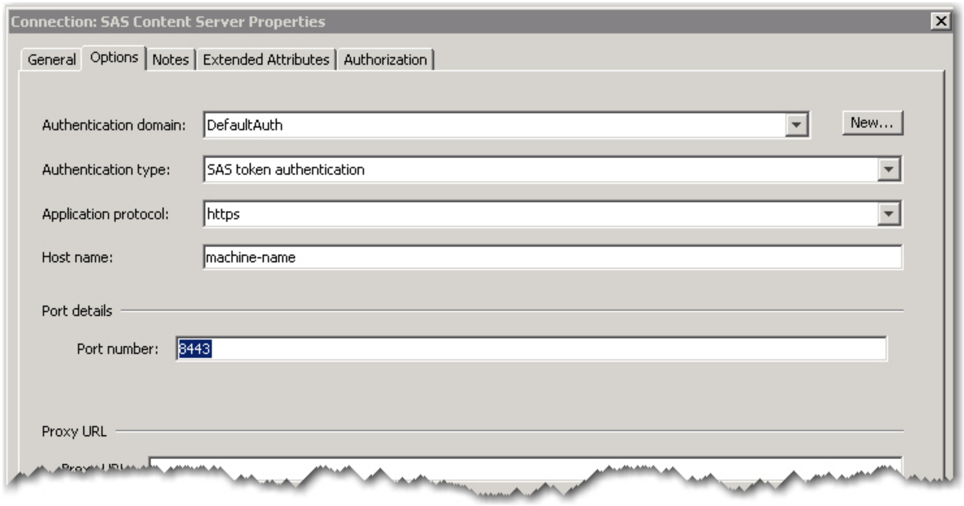
Communication Protocol: HTTPS
Host Name: machine-name
Port Number: 8443
Service: /SASTheme_default

5. After you enter the port number, click **OK**.

Configuring the Connection Metadata for the SAS® Content Server

To configure the connection metadata for the SAS Content Server:

1. In SAS Management Console, select **Plug-ins ► Environment Manager ►Server Manager ► SAS Content Server**. Expand **`SAS Content Server`** so that **`Connection: SAS Content Server`** appears in the right pane of SAS Management Console.
2. Right-click **`Connection: SAS Content Server`** entry and select **`Properties`**.
3. On the **`Options`** tab in the Connection: SAS Content Server Properties dialog box, select **`https`** from the **`Application protocol`** list.
4. For **`Port number`**, enter the port number value that you used previously for **`<Connector port="`*`port-value`*`". . .>`** in step 2 of the section "Configuring the JBoss Application Server for One-Way SSL." (**Note:** The default HTTPS/SSL port for JBoss is **8443**, which is used as the value in the following display.



Authentication domain: DefaultAuth
Authentication type: SAS token authentication
Application protocol: https
Host name: machine-name
Port number: 8443

5. Click **OK** to save your settings and exit the Connection: SAS Content Server Properties dialog box.
6. In the SAS Management Console, select **Folders ► SAS Folders**.
7. Right-click **`SAS Folders`** and select **`Properties`**.

8. In the SAS Folders Properties dialog box, click the **`Content Mapping`** tab.
9. In the **`Server`** list, select **`SAS Content Server`**.
10. Click **`OK`** to exit the dialog box.

Note: The changes you make in this section become effective after you restart the JBoss server instance.

SAS Management Console enables you to validate your connection to the SAS Content Server. However, if SAS Management Console cannot locate the keystore that contains the certificate, you might see the following exception message when you right-click **`SAS Content Server`** and select **`Validate`** from the menu:

```
(sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target)
```

If SAS Management Console cannot locate the trusted certificate, you need to add the following two JVM parameters to the smc.ini file. (These are the same parameters that you added previously in the section “Configuring SAS Remote Services for One-Way SSL.”) This file is located in the installation directory for SAS Management Console.

```
JavaArgs_nn=-Djavax.net.ssl.trustStore="path-to-keystore\trustedca.jks"
JavaArgs_nn=-Djavax.net.ssl.trustStorePassword=password
```

After you add these parameters, the SAS Content Server connection should validate as expected.

Notes:

- In the second parameter above, the value for *password i*s the same password that you use in step 8 of the section “Creating the Certificates and Keystores.”
- The number specified as *nn* in **`JavaAgs_nn=`** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.
- If SAS Management Console is located on another machine (or machines), you need to ensure that the trustedca.jks file is accessible from a shared location. As an alternative, you can import the *server-certificate-name*.crt file into the ***`JRE-directory`*`/lib/security/cacerts`** directory on each machine where SAS Management Console is installed. To do that, use the following command syntax:

  ```
  keytool -import -keystore cacerts -storepass password
          -file server-certificate-name.crt -alias alias-name
  ```

  Example:

  ```
  keytool -import -keystore cacerts -storepass changeit -file JBoss.crt
          -alias jbossonewaycert
  ```

Configuring the URL for the WebDAV Repository

You need to configure the WebDAV URL for HTTPS in SAS Management Console for the following application services:

- **`Remote Services`**
- **`SASBIPortlet4.3 Local Services`**
- **`SASJSR168RemotePortlet4.3 Local Services`**

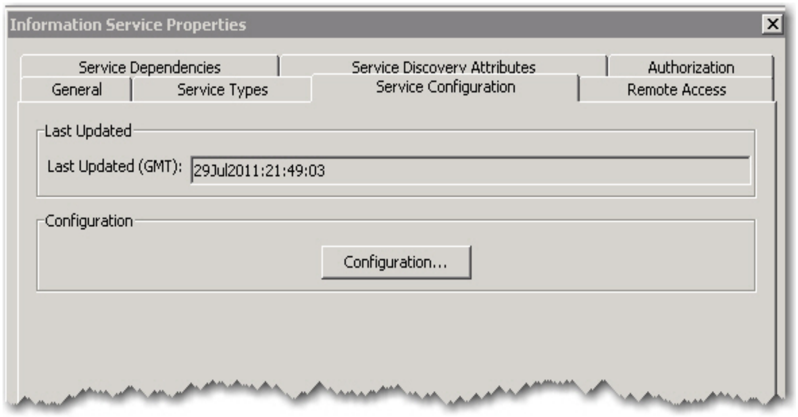
(list continued)

- **`SASPackageViewer4.3 Local Services`**
- **`SASPortal4.3 Local Services`**
- **`SASStoredProcess9.3 Local Services`**
- **`SASWebReportStudio4.3 Local Services`**

For each set of services:

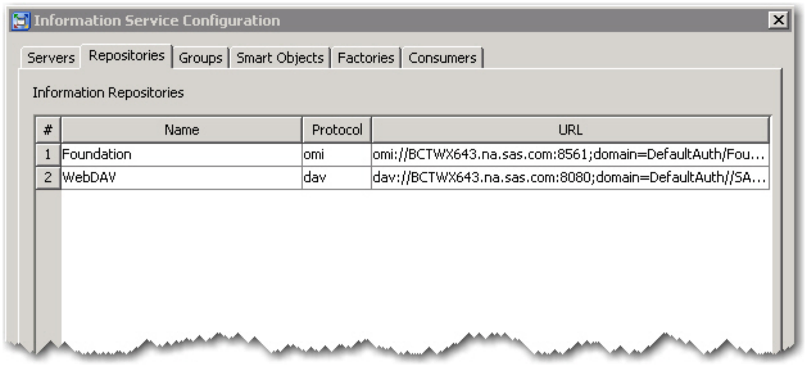
1. In SAS Management Console, select **Plug-ins ► Environment Management ► Foundation Services Manager ►*application-service-name* ►Core ► Information Service**.

   Note: For this example, **`Remote Services`** is selected *for application-server-name*.
2. Right-click **`Information Service`** and select **`Properties`**.
3. In the Information Service Properties dialog box, click the **`Service Configuration`** tab.



Last Updated (GMT): 29Jul2011:21:49:03

4. On the **`Service Configuration`** tab, click the **`Configuration`** button to open the Information Service Configuration dialog box.
5. In the dialog box, click the **`Repositories`** tab.



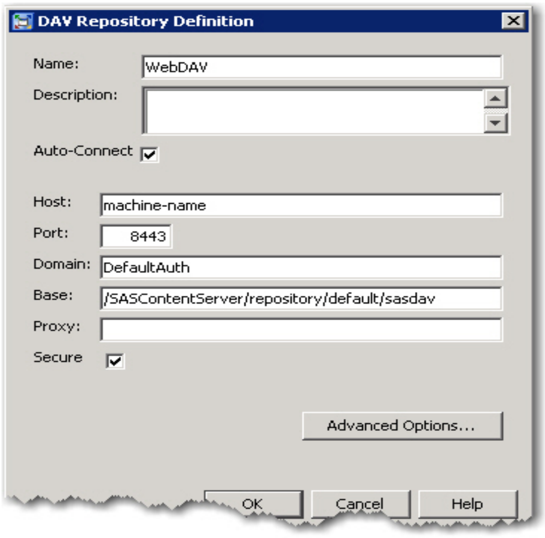
Information Repositories

#	Name	Protocol	URL
1	Foundation	omi	omi://BCTWX643.na.sas.com:8561;domain=DefaultAuth/Fou...
2	WebDAV	dav	dav://BCTWX643.na.sas.com:8080;domain=DefaultAuth//SA...

6. Select **`WebDAV`** and then click **`Edit`**, which opens the DAV Repository Definition dialog box.
7. In the dialog box, change the **`Port`** value to the SSL port that you have configured for your application server. (Similar to the examples in previous steps, the value that is shown the following step is **`8443`**, which is the default HTTPS/SSL port value for JBoss. However, you should use the same value that you use for **`<Connector port="`*`port-value`*`". . .>`** in step 2 of the section "Configuring the JBoss Application Server for One-Way SSL.")

The host name should already be set to the Web application server that is hosting the SAS Content server; that is, the host on which the sas.wip.scs9.3.ear file is deployed.

8. Select the **`Secure`** check box.



Name: WebDAV
Description:
Auto-Connect ✓
Host: machine-name
Port: 8443
Domain: DefaultAuth
Base: /SASContentServer/repository/default/sasdav
Proxy:
Secure ✓

9. Click **OK** to close the Information Service Configuration dialog box.

10. Click **OK** to close the Information Service Properties dialog box.

Configuring the sas-environment.xml File

The sas-environment.xml file is located in the ***`SAS-configuration-directory/`*****`Lev1/web/commons`** directory by default. However, it is possible that this file might be moved to another directory in order to be accessible to your browser. As shown in the following example, you need to change the URL that is listed in the **`<service-registry>`** tags so that the URL reflects the HTTPS changes that you have made previously.

The code in the file is as follows:

```
<?xml version"1.0" encoding="UTF-8"?>
   <environments xmlns=http://www.sas.com/xml/schema/sas-environments-9.2
                 xmlsn:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.sas.com/xml/schema/sas-
                 environments-9.2 http://www.sas.com/xml/schema/sas-environments-
                 9.2/sas-environments-9.2 xsd">
      <environment name="default" default="true">
         <desc>Default SAS Environment</desc>
         <service-
         registry>http://sasbi.demo.sas.com:8080/SASWIPClientAccess/remote
                        /ServiceRegistry</service-registry>
         <service-registry interface-
         type="soap">http://sasbi.demo.sas.com:8080/SASWIPSoapServices
                           /services /ServiceRegistry</service-registry>
      </environment>
   </environments>
```

Change the code as follows. **Note:** This example uses the default JBoss port value (**8443**) that is used for other steps previously in this document. Your port value might be different.

```
<?xml version"1.0" encoding="UTF-8"?>
   <environments xmlns=http://www.sas.com/xml/schema/sas-environments-9.2
                 xmlsn:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.sas.com/xml/schema/sas-
                 environments-9.2 http://www.sas.com/xml/schema/sas-environments-
                 9.2/sas-environments-9.2 xsd">

      <environment name="default" default="true">
         <desc>Default SAS Environment</desc>
         <service-
         registry>https://sasbi.demo.sas.com:8443/SASWIPClientAccess/remote
                       /ServiceRegistry</service-registry>
         <service-registry interface-
         type="soap">https://sasbi.demo.sas.com:8443/SASWIPSoapServices
                          /services /ServiceRegistry</service-registry>
      </environment>
   </environments>
```

Verifying the SSL Connection

If you are using this document to configure one-way SSL with host authentication, that process is complete after you verify your connection. To verify your SSL connection, stop and then restart SAS Remote Services and the JBoss SAS domain servers on the server where you made the updates for SSL (for example, on SASServer1).

At this point, you should be able to log on to your SAS application by entering **`https://`*`fully-qualified-hostname:port-number/SAS-application-name`*** in the browser. (An example of the value for ***`SAS-application-name`*** is **`SASPortal`**; an example for ***`port-number`*** is **8443**. As noted in previous sections, you should use the same value for **`port-number`** that you use for **`<Connector port="`*`port-value`*`". . .>`** in step 2 of the section "Configuring the JBoss Application Server for One-Way SSL.")

The SAS Logon Manager is still configured to use host authentication, so you are prompted for a user name and password. In addition, you should be able to validate the SAS® Content Server and save a project from SAS® Enterprise Guide® 5.*x*.

The next section explains how to configure your environment with two-way SSL and client-certificate authentication.

Configuring Two-Way SSL Authentication

Deploying SAS Web applications to use client-certificate Web server authentication requires configuration of the Web server to support two-way SSL. With *one-way SSL* only the identity of the server is verified; while in *two-way SSL*, the identities of both the server and the client are verified.

The following sections cover these topics related to the steps for configuring two-way SSL:

- creating and importing personal certificates to your browser
- configuring JBoss for two-way SSL
- configuring SAS Remote Services for two-way SSL
- configuring client-certificate authentication for JBoss

Creating Personal Client Certificates

You must create personal certificates for each client person or identity that needs to be verified.

To create these personal certificates:

1. Create a personal private key for the client identity.

   ```
   openssl genrsa –des3 -out keys/client-identity.pem 1024
   ```

   Example:

   ```
   openssl genrsa -des3 -out keys/sasdemo.pem 1024
   ```

2. Generate the personal private-key *x*.509 request file (.csr) with a DN.

   ```
   openssl req -new -key keys/client-identity.pem -days 365
               –out requests/client-identity.csr
   ```

   Example:

   ```
    openssl req -new -key keys/sasdemo.pem -days 365
                -out requests/sasdemo-req.csr
   ```

   a. When you are prompted for the CN, enter the name of the user, as follows:

      ```
      prompt: Common Name (eg., server FQDN or YOUR name)[]: sasdemo
      ```

   b. When you are asked for an extra challenge passphrase, enter a new passphrase.

3. Sign the certificate with the CA certificate, as follows:

   ```
   openssl ca -in requests/client-identity.csr
              -keyfile keys/CA-key-name.pem
               -out certs/client-identity.crt –notext
               -cert certs/CA-certificatename.crt
   ```

   Example:

   ```
   openssl ca –in requests/sasdemo-req.csr –keyfile keys/myCAkey.pem
              –out certs/sasdemo.crt –notext –cert certs/myCA.crt
   ```

4. Combine the public and private keys into PKCS12 format.

   ```
   openssl pkcs12 –export –in certs/CA-certificate-name.pem
                  –inkey keys/client-identity.pem
                  –out certs/client-certificate-name.p12
   ```

(list continued)

Example:

```
openssl pkcs12 -export -in certs/sasdemo.crt -inkey keys/sasdemo.pem
               -out certs/sasdemo.p12
```

You will be prompted to provide an export passphrase.

5. Import the client certificate into the browser in the **`Personal`** area for Internet Explorer and the **`Your Certificates`** area for Firefox as described in the next section.

You now have the certificates that you need for two-way SSL.

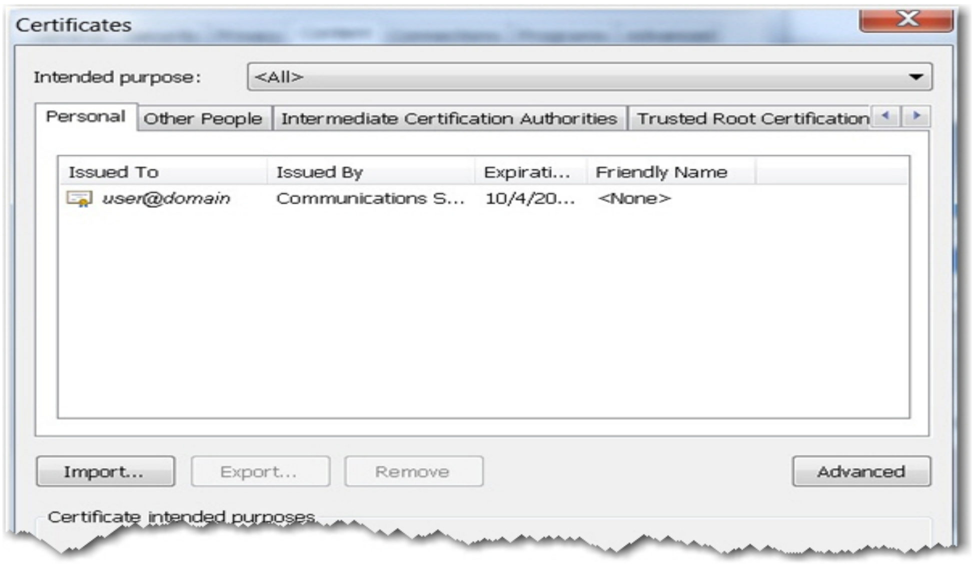
Importing the Personal Certificate into Your Browser

During one-way SSL configuration, you imported the CA certificate into the **`Trusted Certificate`** area of your browser. For two-way SSL, you also need to import the certificate that you just created to identify the client verification during the SSL message exchange (also known as a *handshake*). First, copy the certificate to the machine where you access the browser. Then perform the steps that follow for your particular browser.

To import this certificate into Microsoft Internet Explorer:

1. Select **Tools** ► **Internet Options** ► **Content** ► **Certificates** ► **Personal**.

   Note: Depending on what version of the browser you have, this path might be different.


Certificates
Intended purpose: <All>
Personal
Other People
Intermediate Certification Authorities
Trusted Root Certification
Import...
Export...
Remove
Advanced
Certificate intended purposes

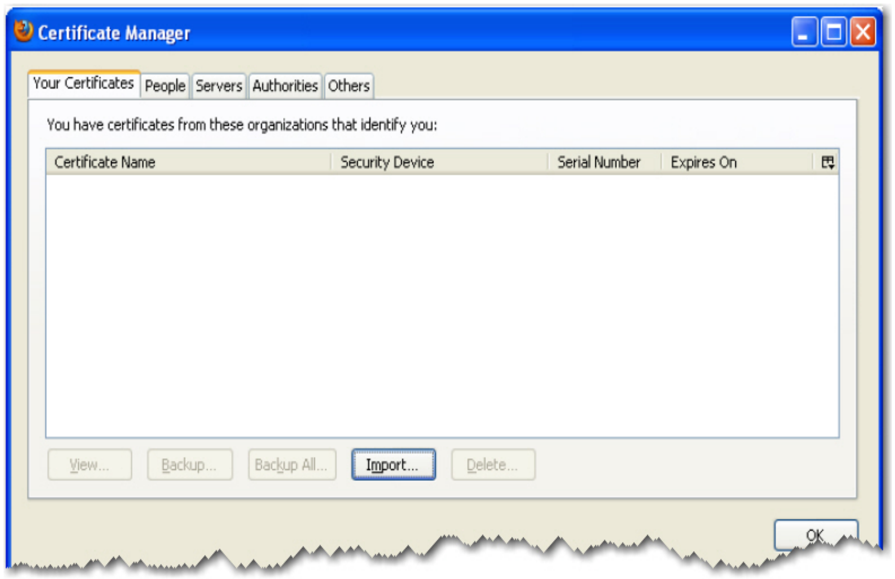
Issued To	Issued By	Expirati...	Friendly Name
user@domain	Communications S...	10/4/20...	<None>

2. Click the **`Import`** button to start the Certificate Import Wizard. Follow the wizard's instructions. You need to supply the path to the certificate file (*client-certificate-name*.p12) and the password. (In step 4 of the previous section, sasdemo.p12 is the example value for *client-certificate-name*.p12.) For any other values, accept the defaults.

To import this certificate into Mozilla Firefox:

1. Select **Tools** ► **Options** ► **View Certificates** ► **Your Certificates**.

   Note: Depending on what version of the browser you have, this path might be different.

   You have certificates from these organizations that identify you:

2. Click the **`Import`** button to start the Certificate Import Wizard. Follow the wizard's instructions. You need to supply the path to the certificate file (*client-certificate-name*.p12) and the password. (In step 4 of the previous section, sasdemo.p12 is the example value for *client-certificate-name*.p12.) For any other values, accept the defaults.

You now have the certificates that you need for two-way SSL.

Configuring JBoss for Two-Way SSL

To configure JBoss for two-way SSL:

1. Open the server.xml file for your SAS server instance and add the **`truststoreFile`** and **`truststorePass`** parameters to the connector. (The **`keystoreFile`** and **`keystorePass`** parameters were added previously during one-way SSL configuration process.) For the SASServer1 server in JBoss Application Server 4.3, server.xml is located in ***`JBoss-home-directory`*`\server\SASServer1\deploy\jboss-web.deployer\server.xml`**. For JBoss Application Server 5.1, server.xml is located in ***`JBoss-home-directory`*`\server\SASServer1\deploy\jbossweb.sar\server.xml`**

   In the server.xml file, you add the **`truststoreFile`** and **`truststorePass`** parameters to the **`<Connector>`** element, as follows:

```
<Connector port="port-value" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="true"
           address="${jboss.bind.address}"
           strategy="ms"
           useBodyEncodingForURI="true"
```

(code continued)

```
keystoreFile="${jboss.server.home.directory}/conf/serverids.jks"
keystorePass="password"
emptySessionPath="true"
truststoreFile="${jboss.server.home.directory}/conf/trustedca.jks"
truststorePass="password"
sslProtocol="TLS" />
```

The values presented above for the **`truststoreFile`** and **`truststorePass`** parameters are taken from the steps performed for one-way SSL. The keystore password should be the same password that you use in step 4 of the section “Creating the Certificates and Keystores.” The trust-store password should be the same as the password that you use in step 8 of that same section.

2. Specify the path to the keystore in the JVM parameters for each JBoss server instance where the SAS Web applications are deployed, as explained below. You can add these parameters directly after the **`trustStore`** and **`trustStorePassword`** parameters that were included for the one-way SSL configuration.

    - If JBoss is installed as a service, add the following parameters to the wrapper.conf file that is located in the server directory for each SASServer instance:

      ```
      wrapper.java.additional.nn=-Djavax.net.ssl.keyStore="path-to-
                                               keystore\serverids.jks"
      wrapper.java.additional.nn=-Djavax.net.ssl.keyStorePassword=password
      ```

      Notes:

      - The numbers specified as *nn* in **`wrapper.java.additional.`*`nn`*`=`** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.
      - In the second argument above, the value for password is the same value that you use in step 4 of the section "Creating the Certificates and Keystores.

    - If JBoss is started manually, add the following parameters to either the RemoteServices.bat file or the RemoteServices.sh file for each SAS server instance:

      ```
      -Djavax.net.ssl.keyStore="path-to-keystore\serverids.jks"
      -Djavax.net.ssl.keyStorePassword=password
      ```

      Note: In the second parameter above, the value for *password i*s the same password that you use in step 8 of the section “Creating the Certificates and Keystores.”

Configuring SAS Remote Services for Two-Way SSL

You need to specify the path to the server-identity keystore and its associated password in the wrapper.conf file for SAS Remote Services. You specify the location and name of the server-identity keystore and its password in JVM parameters that you add to wrapper.conf, as follows:

- If SAS Remote Services is installed as a service, add the following parameters to the wrapper.conf file, which is located in ***`SAS-configuration-directory`*`\ Lev1\Web\Applications\RemoteServices`**:

  ```
  wrapper.java.additional.nn=-Djavax.net.ssl.keyStore="path-to-keystore\
                                                       serverids.jks"
  wrapper.java.additional.nn=-Djavax.net.ssl.keyStorePassword=password
  ```

Notes:

- The number specified as *nn* in **`wrapper.java.additional.nn=`** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.
- In the second argument above, the value for password is the same value that you use in step 4 of the section "Creating the Certificates and Keystores.

- If you start SAS Remote Services from a batch file, add the following parameters both to the **`start2`** and **`start3`** sections in the SASServerx.bat or SASServerx.sh files:

```
-Djavax.net.ssl.keyStore="path-to-keystore\serverids.jks"
-Djavax.net.ssl.keyStorePassword=password
```

  Note: In the second argument above, the value for password is the same value that you use in step 4 of the section "Creating the Certificates and Keystores.

In either of these cases, you can add these parameters directly after the **`trustStore`** and **`trustStorePassword`** parameters that were added for the one-way SSL configuration.

For SAS Management Console, add the following parameters:

```
JavaArgs_nn=-Djavax.net.ssl.keyStore="path-to-keystore\serverids.jks"
JavaArgs_nn=-Djavax.net.ssl.keyStorePassword=password
```

Notes:

- The number specified as *nn* in **`wrapper.java.additional.nn=`** are sequential numbers, so the numbers you see will vary from instance to instance. The value for *nn* will be the last number used plus 1.
- In the second argument above, the value for password is the same value that you use in step 4 of the section "Creating the Certificates and Keystores.

Verifying the SSL Connection

If you are using this document to configure two-way SSL with host authentication, that process is complete after you verify your connection. To verify your SSL connection, stop and then restart SAS Remote Services and the JBoss SAS domain servers on the server where you made the updates for SSL (for example, on SASServer1). At this point, you should be able to log on to your SAS application by entering **`https://fully-qualified-hostname:8443/SAS-application-name`** in the browser. (An example of the value for ***`SAS-application-name`*** is **`SASPortal`**). The SAS Logon Manager is still configured to use host authentication, so you are prompted for a user name and password

You should be prompted to select the client certificate. In some browser versions, if there is only one certificate in the **`Personal`** section, that certificate is used by default and you are not prompted.

Configuring JBoss Application Server for Web Authentication with SAS 9.3® Web Applications

The default security mechanism for SAS Web applications is to authenticate against the authentication provider of the SAS® Metadata Server. As an alternative authentication mechanism, you can use Web authentication. With this method, you configure JBoss to authenticate against a user registry such as a Lightweight Directory Access Protocol (LDAP) server, and you configure SAS Web applications to trust the authentication that JBoss performs. For client-certificate authentication, the environment must be configured for Web authentication. To configure your system for basic Web authentication, see the steps in *Configuring JBoss Application Server 4.3 and 5.1 for Web Authentication with SAS® 9.3 Web Applications*. (**`support.sas.com/resources/thirdpartysupport/v93/appservers/ConfiguringJBossWebAuth.pdf`**)

Configuring Client-Certificate Authentication for JBoss

To configure client-certificate authentication for JBoss:

1. Create a new Java keystore that contains the client certificate that you created previously.

```
keytool -import -alias sasdemo -keystore clientStore.jks
        -storepass Jboss4 -file certs/sasdemo.crt
```

2. Add the following **`<mbean>`** element before the closing **`<server>`** tag in the jboss-service.xml file, which resides in ***`JBoss-server-home-directory`*`}/conf`**. This **`<mbean>`** element specifies the path to the keystore and instructions about how to access it.

```
<mbean code="org.jboss.security.plugins.JaasSecurityDomain"
       name="jboss.security:service=SecurityDomain">
   <constructor>
      <arg type="java.lang.String" value="SASApplicationLogin"/>
   </constructor>
   <attribute name="KeyStoreURL">resource:clientStore.jks</attribute>
   <attribute name="KeyStorePass">password</attribute>
   <depends>jboss.security:service=JaasSecurityManager</depends>
</mbean>
```

Notes:

- The value that you use for **`value=`** in the **`<arg>`** attribute might differ from what is shown in the previous example. This value is defined when you set up Web authentication.
- The value for password is the same password you create in step 4 of the section "Creating the Certificates and Keystores.
- To locate the serverids.jks file without any path information, make a copy of the file and place it in the ***`JBoss-server-home-directory`*`}/conf`** directory**.**

3. In the login-config.xml file that resides in ***`JBoss-server-home-directory`*`/conf`**, add the following login module to the SAS application policy. **Note:** The SAS application policy name is defined during the setup for Web authentication, so the name might be different. For example, it might be similar to **`SASApplicationLogon`** or **`SASLogonManager`**.)

```
<login-module code="org.jboss.security.auth.spi.BaseCertLoginModule"
                  flag="required">
    <module-option name="password-stacking">useFirstPass</module-option>
    <module-option
    name="securityDomain">java:/jaas/SASApplicationLogin</module-option>
</login-module>
```

Note: For the **`<module-option name="securityDomain">`** attribute above, the value that is specified as **`SASApplicationLogin`** might be different in your case. This value is defined during the setup for Web authentication.

You should add this block after the application tag, which looks similar to the following:

```
<application-policy name="SASApplicationLogon">
   <authentication>
```

4. In the server.xml file (in ***`JBoss-server-home-directory`*****`/server/SASServer1/deploy/jboss-web.deployer`**), modify the value of the **`certificatePrincipal`** parameter to use the common name as the principal. In the following example, the principal has been changed to the common name **`SubjectCNMapping`**:

```
<Realm allRolesMode="authOnly"
certificatePrincipal="org.jboss.security.auth.certs.SubjectCNMapping"
className="org.jboss.web.tomcat.security.JBossSecurityMgrRealm"/>
```

5. In the same file, under the HTTPS connector, change the value of the **`clientAuth`** parameter to **`true`**.

```
<Connector port="port-value" protocol="HTTP/1.1" SSLEnabled="true"
   maxThreads="150" scheme="https" secure="true"
   clientAuth="true"
   address="${jboss.bind.address}"
   strategy="ms"
   useBodyEncodingForURI="true"
   keystoreFile="${jboss.server.home.directory}/conf/serverids.jks"
   keystorePass="password"
   emptySessionPath="true"
   truststoreFile="${jboss.server.home.directory}/conf/trustedca.jks"
   truststorePass="password"
   sslProtocol="TLS" />
```

Notes:

- The keystore password should be the same password that you use in step 4 of the section “Creating the Certificates and Keystores.”
- The trust-store password should be the same as the password that you use in step 8 of that same section.

Configuring SAS Logon Manager for Client-Certificate Authentication

To configure the SAS Logon Manager:

1. Change the SAS Logon Manager application’s web.xml file so that it contains the **`CLIENT-CERT`** authentication method. The web.xml file resides in the following location:

   `JBoss-home-directory`**`\SASServer1\deploy_sas\sas.wip.apps9.x.ear\sas.svcs.logon.war\WEB-INF`**

(list continued)

You can find these files in the directory from which the SAS applications are deployed to your application server (for example, in the **deploy_sas** directory). If your applications are not packaged in an .ear and a .war file and the directories for the .ear and .war files are already exploded, you need to modify only the file. There is no unpacking or repacking involved.

To modify web.xml, add the following method prior to the closing **</webapp>** tag:

```
<security-constraint>
    <web-resource-collection>
       <web-resource-name>All resources</web-resource-name>
       <url-pattern>/*</url-pattern>
       <http-method>GET</http-method>
       <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
       <role-name> SASWebUser </role-name>
    </auth-constraint>
</security-constraint>

<login-config>
   <auth-method>CLIENT-CERT</auth-method>
   <realm-name>default</realm-name>
</login-config>

<security-role>
   <role-name>SASWebUser</role-name>
</security-role>
```

2. Rebuild the sas.svcs.logon.war and sas.wip.apps9.3.ear files. Then redeploy the .ear file.
3. Test the client-certificate authentication, as follows:
   a. Restart SAS Remote Services and the JBoss SAS server.
   b. Verify that you can access the SAS applications through the URL without having to provide logon credentials.

Resources

Open SSL Software Foundation (2010). "OpenSSL Cryptography and SSL/TLS Toolkit." Available at **www.openssl.org/**. Accessed May 30, 2012.

Oracle Technology Network (1993). "keytool – Key and Certificate Management Tool." Available at **docs.oracle.com/javase/6/docs/technotes/tools/solaris/keytool.html**. Accessed May 30, 2012.

SAS Institute Inc. (2011). *Configuring JBoss Application Server 4.3 and 5.1 for Web Authentication with SAS® 9.3 Web Applications*. Available at **support.sas.com/resources/thirdpartysupport/v93/appservers/ConfiguringJBossWebAuth.pdf**.

SAS Institute Inc. (2012). *SAS® 9.3 Intelligence Platform: Web Application Administration Guide, Third Edition*. Available at **support.sas.com/documentation/cdl/en/biwaag/65230/PDF/default/biwaag.pdf.**

Glossary

certificate

A digitally signed statement from one entity (a company or person) that says that the public key and other information from another entity have a certain value. When data is digitally signed, its authenticity and integrity can be validated by the signature.

keystore

A repository that holds digital certificates and private keys that are used to identify client or server machines. Because of the security information held in a keystore, it is password protected.

keytool

A tool that stores keys and certificates in a keystore and which you can use to manage keystores.

one-way SSL

A Secure Socket Layer (SSL) protocol that is used to encrypt information that passes between a client application and a server. One-way SSL requires that the server provide a certificate to the client. However, the client is not required to reciprocate with a certificate to the server.

OpenSSL tool

An open-source toolkit that implements both the Secure Socket Layer (SSL) and the Transport Layer Security (TLS) protocols. The tool also contains a robust cryptography library for security purposes.

private key

A number that is known only to its owner. The owner uses the private key to read (decrypt) an encrypted message.

Secure Socket Layer (SSL)

A protocol developed by Netscape Communications that provides network security and privacy. SSL, which provides a high level of security, uses encryption algorithms RC2, RC4, DES, TripleDES, and AES.

two-way SSL

A Secure Socket Layer (SSL) configuration that provides the encryption capabilities of one-way SSL and enables authentication of a user based on the contents of the user's digital certificate. In two-way SSL, the server is required to present a certificate to the client, and the client is required to reciprocate with a certificate.