Configuring HP Service Manager to Use the SSL-based Trusted Sign-On and LW-SSO

Document Release Date: July 2011

Modified Date: March 2016

Software Release Date: July 2011

Table of Contents

Introduction

In Service Manager earlier than version 9.30, SSL-based Trusted sign-on is required to enable Lightweight Single Sign-On (LW-SSO) for integrations with other HP applications that need to connect to Service Manager via LW-SSO by calling Service Manager web services or launching the Service Manager web UI.

In version 9.30, the Service Manager server provides enhanced LW-SSO support, which no longer requires SSL-based trusted sign-on. This enhancement makes it much easier to enable LW-SSO for integrations with applications like RC, BSM OMi, etc. For information about how to enable LW-SSO for integrations by using the enhanced LW-SSO support, see the Service Manager 9.30 help.

HP recommends that you use the new LW-SSO configuration approach. However, for backward compatibility, Service Manager 9.30 still supports enabling LW-SSO through SSL-based trusted sign-on.

This document describes the configuration steps on the Service Manager 9.30 side to enable LW-SSO for integrations through SSL-based trusted sign-on.

Using Trusted Sign-On to enable LW-SSO for integrations

The Service Manager server uses a proprietary Single Sign-On (SSO) protocol that is based on mutually-authenticated Secure Sockets Layer (SSL).

When an HP application (for example, HP BSM Operations Manager i) needs to access the Service Manager web tier using LW-SSO, you can configure Service Manager to use SSL-based trusted sign-on and to support LW-SSO.

When an HP application (for example, HP Release Control) needs to access Service Manager web services using LW-SSO, an extra step is required to install and configure the Symphony Adapter (for web services), which cannot work without SSL.

The following table lists the configuration procedures required in Service Manager for HP applications that need to access Service Manager through the web tier or web services using LW-SSO.

Example Application	**Need to access Service Manager through**	**Configurations required in Service Manager**
HP BSM Operations Manager i	Web tier	• Configure Service Manager to use SSL-based trusted sign-on • Configure Service Manager for LW-SSO support
HP Release Control	Web Services	• Configure Service Manager to use SSL-based trusted sign-on • Install and configure Symphony Adapter • Configure Service Manager for LW-SSO support

Configuring Service Manager to Use SSL-based Trusted Sign-On

As an example, the following describes how to create signed server and client certificates using the OpenSSL toolkit as a private certificate authority. This example also uses the keytool utility available with the Sun Microsystems™ standard Java Development Kit (version 1.4 or later).

Prerequisites:

- You must have the following software installed on a machine (on which you will create signed certificates for the Service Manager server and clients):
    - OpenSSL: can be downloaded from http://www.slproweb.com/products/Win32OpenSSL.html
    - JDK 1.4 or later: Can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/index.html
- You need to add the bin folders of your JDK and OpenSSL to the PATH environment variable definition of the machine, so that you do not need to change directories to the bin folders before running openssl or keytool commands:
    - OpenSSL bin folder: <InstallDir>\OpenSSL-Win32\bin
    - JDK bin folder: <InstallDir>\Java\jdk1.x.x_xx\bin

NOTES:

- The following procedures will prompt you to enter several passwords multiple times. Using the same password over and over is not best practice in production, however if you are performing the procedures for test purposes, you are recommended to enter the same password at each prompt to avoid any confusion about which password you are being asked for. Also be aware that nothing displays on the screen when you are entering pass phrases.
- Whenever asked to confirm whether to trust the current certificate, type **y** and press ENTER (the default response is no). If you just press ENTER, the certificate will not be trusted and you will have to start over.

Task 1: Create a root CA

Note: The following steps use the JDK bin folder as the working directory, in which you will create the certificates and keystore files. If you wish, you can create your own working directory and run the commands from there.

1. Open the operating system's command prompt, and change directory to the JDK bin folder.
2. Create the private key for your private certificate authority by running the command:

   ```
   openssl genrsa -des3 -out cakey.pem 2048
   ```

   When prompted, enter a pass phrase you want to use to protect your certificate authority's private key file (cakey.pem). For example, CAKeyPassword.You must use the same password phrase each time you sign a certificate request with your private certificate authority. You will be asked to enter this pass phrase later many times again.
3. Export the public key as the self-signed root CA certificate by running the command:

   ```
   openssl req -new -key cakey.pem -x509 -days 1095 -out mycacert.pem
   ```

   a. When prompted, enter the pass phrase you selected for cakey.pem.

b. Enter other required information. When asked for a Common Name, enter the fully-qualified domain name of the machine on which you are creating the root CA.

4. Import your private certificate authority's certificate into the Java `cacerts` file that you will publish to the rest of your network. It is very important that the `cacerts` file in the `<JAVA_HOME>\lib\security` folder is updated to include the root CA information.

   a. Make a backup copy of the cacerts file in the `<JAVA_HOME>\lib\security\` folder, and copy this file to the `<JDK_home>\bin` folder.

   b. Run the following command:

   ```
   keytool -import -keystore ./cacerts -trustcacerts -file mycacert.pem -
   storepass changeit
   ```

   When prompted, type: y to trust the root CA's certificate.

   The root CA certificate is added to the Java `cacerts` file.

   c. Copy the updated Java `cacerts` file to the `<JAVA_HOME>\lib\security\` folder.

Task 2: Set up the Service Manager server

Note: Change directory to the `<JDK_home>\bin` folder before proceeding.

Subtask 1: Create a keystore for the Service Manager server

1. Generate a private/public key pair by running the command:

   ```
   keytool -genkey -alias myserver -keystore servercert.keystore
   ```

   NOTE: If you would like SHA2-based certificates, run the below command:

   ```
   keytool -genkeypair -alias myserver –sigalg SHA256withRSA -
   keystore servercert.keystore
   ```

   For detailed information on the Java *keytool* command please see the official documentation:

   https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

   a. Enter a password for the server keystore (<Service Manager server keystore password>). Write down this password, which will be added to the sm.ini file later.

   b. When prompted to enter your first and last name, enter the fully-qualified domain name (computer.domain.com) of the Service Manager server host.

   c. Enter other identification information.

   d. When prompted for a key password for <smserver>, press ENTER to use the same password as the server keystore password.

   Note: Do not use the same password as your root CA key password.

2. Generate a request file by running the command:

   ```
   keytool -certreq -alias myserver -keystore servercert.keystore -file
   smserver_certrequest.crs
   ```

   Enter the password you selected for the server keystore in step 1 of this subtask.

3. Self-sign the request by running the command:

   ```
   openssl x509 -req -days 365 -in smserver_certrequest.crs -CA mycacert.pem -
   ```

```
CAkey cakey.pem -CAcreateserial -out smserver_cert.pem
```

If everything goes well, a message "Signature ok" displays. When prompted, enter the pass phrase for the root CA's private key.

4. Import the signed certificate into the server keystore by running the command:

```
keytool -import -trustcacerts -alias myserver -
keystore ./servercert.keystore -file smserver_cert.pem
```

Enter the password you selected for the server keystore in step 1 of this subtask.

The signed certificate is installed in the server keystore.

Subtask 2: Create a trusted client keystore

This task will create a client keystore that contain the signed certificates of your Service Manager server's trusted clients. You need to repeat this task for each trusted client, including the web tier host and Windows client hosts.

Best Practice recommendation: When you configure the web tier, type the word *web* in front of the keystore, certificate request and certificate name. For Windows client certificates, enter the name of the machine in front of all names to make them unique and easier to distinguish.

1. Generate a private/public key pair for your client by running the command:

   ```
   keytool -genkey -alias client -keystore clientcerts.keystore
   ```

   NOTE: If you would like SHA2-based certificates, run the below command:

   ```
   keytool -genkey -alias client –sigalg SHA256withRSA -keystore
   clientcerts.keystore
   ```

   For detailed information on the Java *keytool* command please see the official documentation:
   https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

   a. Enter a password for the client keystore.

   b. When prompted to enter your first and last name, always enter the fully-qualified domain name (computer.domain.com) of the web tier or Windows client host.

   c. When asked for a password for the client private key (<client>), press ENTER to use the same password as the client keystore password.

2. Generate a request file:

   ```
   keytool -certreq -alias client -keystore clientcerts.keystore -
   file client_certrequest.crs
   ```

   When prompted, enter the client keystore password.

3. Self-sign the request:

   ```
   openssl x509 -req -days 365 -in client_certrequest.crs -CA mycacert.pem -
   CAkey cakey.pem -CAcreateserial -out client_cert.pem
   ```

   If everything goes well, a message "Signature ok" displays.

   When prompted, enter the pass phrase of the root CA's private key.

4. Import the signed certificate into the client keystore:

   ```
   keytool -import -trustcacerts -alias client -keystore ./clientcerts.keystore
   -file client_cert.pem
   ```

   When prompted, enter the client keystore password.

   A message displays: "Certificate reply was installed in keystore".

Subtask 3: Generate a trust-list keystore for the Service Manager server

This task applies for both the Windows and web clients. You need to import all client certificates to a JKS list of trusted clients.

Note: Repeat this task for each trusted client certificate.

1. Import the client certificate you created in subtask 2 into a jks file:

   ```
   keytool -import -alias client1 -file client_cert.pem -keystore
   trustedclients.keystore
   ```

2. When prompted, enter a password for the trusted keystore (<trusted client keystore password>. **NOTE**: Write down this password, which will be added to the sm.ini file later.

3. When asked, type **y** to confirm that you want to trust the certificate.

   A message displays: "Certificate was added to keystore".

Subtask 4: Modify the Service Manager configuration

1. Go to the <JDK_home>\bin folder, and copy the generated files `cacerts`, `servercert.keystore`, and `trustedclients.keystore` into the <Service Manager installation path>\Server\RUN\ folder.
2. In the `sm.ini` file, set the `sslConnector` parameter to 1 if it is 0.
3. Add the following entries to the sm.ini file:

```
keystoreFile:servercert.keystore
keystorePass:<Service Manager server keystore password>
ssl:1
ssl_reqClientAuth:2
ssl_trustedClientsJKS:trustedclients.keystore
ssl_trustedClientsPwd:<trusted client keystore password>
trustedsignon:1
truststoreFile:cacerts
truststorePass:changeit
```

4. Restart the Service Manager server.

Task 3: Set up the Service Manger Windows client to enable SSL connection

After SSL-based trusted sign-on is enabled for the Service Manager server, you can enable an SSL connection for each Windows client so that Windows client users can connect to the server.

Note: To enable Windows client users to use trusted sign-on, you must first create an operator record for each Windows client user you want to log on to Service Manager. The operator's login name should match the NT account user name, but does not need a password.

1. Go to the <JDK_home>\bin folder, copy the CA keystore file `cacerts` and client keystore file `clientcerts.keystore` to the default certificate path of the client:

   <Windows client installation path>\plugins\com.hp.ov.sm.client.common_x.xx.
2. Start the Service Manager Windows client.
3. Set the Windows client security preferences.
   a. From the menu bar, select **Window** > **Preferences...** to open the Preferences dialog.
   b. Expand the **HP Service Manager** node in the menu tree, and select **Security** to open the client security dialog.
   c. In the **CA certificates file** field, browse to the ca.keystore file: <Windows client installation path>\plugins\com.hp.ov.sm.client.common_x.xx\cacerts
   d. In the **Client keystore file** field, browse to the client.keystore file: <Windows client installation path>\plugins\com.hp.ov.sm.client.common_x.xx\clientcerts.keystore.
   e. In the **Client keystore password** field, enter the password you specified for the Windows client's keystore file.
4. Update the server connections.
   a. Go to **File** > **Connect** > **Connections**, and select a connection to the Service Manager server.

b On the Connection tab, make sure that the **Server host name** field value is the FQDN of the Service Manager server host. This should be the same as mentioned while creating the server keystore in Task 2.

c On the **Advanced** tab, select **Use SSL Encryption**.

d Click **Apply**.

5. Restart the Windows client for the new security configuration to take effect.
6. Log on to the Windows client with your NT account user name, without using trusted sign-on.

   If the log on is successful, it confirms that SSL is set up correctly.
7. Log on to the Windows client using trusted sign-on.

   Service Manager should automatically log you on with your NT account user name.

Task 4: Set up the Service Manager web tier

This task is to configure the web tier to connect to the Service Manager server using the Service Manager trusted sign-on protocol.

The following steps assume that your Service Manager web tier is deployed on Tomcat.

1. Stop the web application server running the Service Manager web tier.
2. Go to the `<JDK_home>\bin` folder, copy the CA keystore file `cacerts` and web tier client keystore file `clientcerts.keystore` to the webapps\<Service Manager web tier>\WEB-INF folder of the Tomcat installation.
3. Edit <Tomcat>\webapps\<Service Manager web tier>\WEB-INF\web.xml.

   a Configure the Service Manager web tier to use the client certificate you have just created.

   You have to specify three parameters as shown below:

   - the location of the CA keystore file created previously
   - the location of the web tier client keystore just created
   - the password for the web tier client keystore

```
<!-- Specify the CA certificate store to use in encrypted communication -
->
<init-param>
<!-- If this value is empty, the JDK's default
jre/lib/security/cacerts file is used -->
<!-- If this is a relative path, it will be relative to the web
application's deploy directory but still needs a leading slash -->
  <param-name>cacerts</param-name>
  <param-value>/WEB-INF/cacerts</param-value>
</init-param>


<!-- Specify the client's private keystore to use in encrypted
communication. This is necessary for client authentication when
using single sign-on, but not for a standard SSL connection. -->
<!-- If this is a relative path, it will be relative to the web
application's deploy directory but still needs a leading slash -->
<init-param>
  <param-name>keystore</param-name>
  <param-value>/WEB-INF/clientcerts.keystore</param-value>
```

```
</init-param>

<!-- Specify the password for the client's private keystore -->
<init-param>
  <param-name>keystorePassword</param-name>
  <param-value><client keystore password></param-value>
</init-param>
```

b Make sure the server FQDN name is placed instead of 'localhost'.

```
<!-- Specify the HP Service Manager server host and port location -->
<init-param>
    <param-name>serverHost</param-name>
    <param-value><Service Manager server host name (FQDN)></param-value>
 </init-param>
```

c Make sure ssl is set to true.

```
<!-- Control the encryption of network communication between
the application server and the HP Service Manager server -->
    <init-param>
      <param-name>ssl</param-name>
      <param-value>true</param-value>
    </init-param>
```

d For using trusted sign on, set the value of the `isCustomAuthenticationUsed` parameter to false, in order for Service Manager to send the current user name in the HTTP header. If set to false without trusted sign-on, web client users will not be able to log in to the system.

```
<context-param>
  <param-name>isCustomAuthenticationUsed</param-name>
  <param-value>false</param-value>
</context-param>
```

e Enable drill from BAC to Service Manager.

Set the following in case an error message (cracking attempt) comes up while drilling to Service Manager from the EMS monitor.

```
<init-param>
    <param-name>querySecurity</param-name>
    <param-value>false</param-value>
</init-param>
```

4. Restart Tomcat.

When the configuration is complete and the Tomcat container has been restarted, the Service Manager web tier is enabled to use trusted sign-on when communicating with the Service Manager server.

Note: If you start a web browser from your desktop and start the web client, it will still display the log-in panel.

Installing and configuring Symphony Adapter

This section describes installation and configuration of the latest version of Symphony Adapter for Service Manager.

Symphony Adapter resides between another HP application and Service Manager to convert LW-SSO to Service Manager Trusted Sign-on SSO protocol.

IMPORTANT:

- If you are setting up your environment for an application to access the Service Manager Web Services using LW-SSO, you must install and configure Symphony Adapter.
- If you are setting up your environment for an application to access the Service Manager Web tier using LW-SSO, you do NOT need to install and configure Symphony Adapter. Symphony Adapter is bypassed in this case.

Installation

The adapter is provided as a .war file, intended to be deployed in a suitable web container. It is recommended that the SymphonyAdapter.war file be deployed in the same Tomcat container in which the Service Manager Web tier has been deployed. This allows the web tier and Symphony Adapter to share the same client certificate.

NOTE: Before proceeding to the following steps, make sure that the Service Manager Windows and Web clients are still working properly following the configuration steps described above.

1. Copy and unzip the latest adapter file SymphonyAdapter.zip from the Service Manager DVD to a local drive on your Service Manager web tier host.
2. Deploy the SymphonyAdapter file in the Tomcat container, by copying and pasting the SymphonyAdapter.war file into the Tomcat webapps directory.

Configuration

1. Copy the CA keystore file "cacerts" and web client keystore file "clientcerts.keystore" you created into the <symphonyadapter>\WEB-INF\classes folder of the SymphonyAdapter webapp.
2. Edit the SymphonyAdapter.properties file to correct the settings for your installation as described in the following table.

Setting	**Description**
servicecenter.ws.targetLocation URL	This property setting can be left alone if the Service Manager server is running on the same host. Otherwise, edit the hostname. Leave the port and path alone. For example: servicecenter.ws.targetLocationURL= https://<Service Manager server host name (FQDN)>:13443/sc62server/ws
servicecenter.webtier.URL	Update this property to make the hostname and port correct for the current Tomcat container. **NOTE**: Do NOT Specify LOCALHOST. You must provide the full host name. For example: http://<Service Manager web tier host name (FQDN)>:8080/sm930/index.do

clientcerts.keystore	Update this parameter to point to the Symphony Adapter client keystore you created. **NOTE**: For this parameter, you must use a full path name starting from C: and using double slashes, for example: C:\\Program Files\\Apache Software Foundation\\Tomcat 5.5.23\\webapps\\SymphonyAdapter\\WEB-INF\\classes\\ clientcerts.keystore **NOTE**: Do not put any newline characters in the path. The example above is just wrapped in this document due to the length of the line.
clientcerts.keystore.password	Specify the correct pass-phrase for the client keystore specified above. For example: !qaz2wsx3edc
truststore	Specify the full path to the CA keystore file you created. For example: C:\\Program Files\\Apache Software Foundation\\Tomcat 5.5.23\\webapps\\SymphonyAdapter\\WEB-INF\\classes\\ cacerts
truststore.password	Specify the pass-phrase for the cacerts file: changeit.
debug_ssl	Keep the default (false).

3. Copy and rename the file lwssofmconf.xml.sample (under <symphonyadapter>\WEB-INF\classes) to lwssofmconf.xml.
4. Edit the parameter values in bold in lwssofmconf.xml as shown below.

```
<lwsso-config
  xmlns="http://www.hp.com/astsecurity/idmenablmentfw/lwsso/1.0">
  <webui enabled="true">
       <web-lwsso>
         <lwsso startLWSSO="enabled">
            <domain><Domain name of the Symphony Adapter server
host></domain>
            <crypto cipherType="symmetricBlockCipher"
              engineName="AES" paddingModeName="CBC" keySize="256"
              encodingMode="Base64Url" initString="This is a shared secret
passphrase"></crypto>
            <expirationPeriod>50</expirationPeriod>
         </lwsso>

         <protectedDomains>
            <url><Domain name of the Symphony Adapter server host></url>
            <url><Domain name of another application (for example,
Release Control)’s server host></url>
          </protectedDomains>

         <roleSecurityFrameworkIntegration
```

```
            rolePrefix="ROLE_"
            fromLWSSOToSecurityFramework="both"
            fromSecurityFrameworkToLWSSO="enabled"
            caseConversion="upperCase"/>

          <groupSecurityFrameworkIntegration
            fromLWSSOToSecurityFramework="both"
            fromSecurityFrameworkToLWSSO="enabled"
            caseConversion="upperCase"/>
        </web-lwsso>
    </webui>
  </lwsso-config>
```

5. Restart Tomcat.

Configuring the Service Manager web tier for LW-SSO support

To configure the Service Manager web tier for LW-SSO support, you must first configure the Service Manager web client for trusted sign-on and SSL support with the Service Manager server. This involves generating and deploying certificates and modifying the sm.ini file on the Service Manager server and web.xml on the web client.

To configure the Service Manager web tier for LW-SSO support:

1. Uncomment the following filter elements in the web tier's web.xml file to enable LW-SSO as shown below.

```
<!-- LWSSO filter for integrations using HP lightweight single sign-on
       PLEASE NOTE: Uncomment this filter and the associated filter-
       mapping,
and see application-context.xml for additional configuration needed for LWSSO.
-->
<filter>
    <filter-name>LWSSO</filter-name>
    <filter-class>com.hp.sw.bto.ast.security.lwsso.LWSSOFilter</filter-class>
</filter>
...

<!-- LWSSO filter-mapping, please read description for LWSSO filter above
before uncommenting this. -->

  <filter-mapping>
    <filter-name>LWSSO</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

2. Locate the isCustomAuthenticationUsed context-param element in the web tier web.xml. Make sure the param-value element is set to false. It should look like the following:

```
<context-param>
     <param-name>isCustomAuthenticationUsed</param-name>
     <param-value>false</param-value>
 </context-param>
```

3. Modify the application-context.xml file located in the WEB-INF\classes folder of the Service Manager web tier deployment.

   a. Locate the filterChainProxy bean element. Add the lwSsoFilter to the value element.

```
 <bean id="filterChainProxy"
class="org.acegisecurity.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
...

/**=httpSessionContextIntegrationFilter,lwSsoFilter,anonymousProcessi
n gFilter
      </value>
    </property>
 </bean>
```

   b. Uncomment the lwSsoFilter bean, as shown below.

```
<!-- This bean is used for HP Lightweight Single Sign-on, to
integrate with other Hewlett-Packard software products. Uncomment it
here and reference it in the filterChainProxy as commented above. -->
 <bean id="lwSsoFilter"
class="com.hp.ov.sm.client.webtier.lwsso.LwSsoPreAuthenticationFilter
"
>
    <property name="authenticationManager">
      <ref bean="authenticationManager"/>
    </property>
    <property name="defaultRole">
      <value>ROLE_PRE</value>
    </property>
 </bean>
```

4. In the lwssofmconf.xml file located in the WEB-INF\classes folder of the Service Manager Web client deployment, set the following parameters.

   - Set the value of enableLWSSOFramework to **true** (default is false).
   - <domain>: Domain name of the server where you deploy your Web tier. For example, if your Web tier's fully qualified domain name is mywebtier.example.com, then the domain portion is example.com.
   - <initString>: Password used to connect HP products (minimum length: 12 characters). For example, smintegrationlwsso. Make sure that this value is the same as those used in the LW-SSO configurations of the other HP applications (such as Operations Orchestration, and Business Service Management), which you want to connect via LW-SSO.
   - <multiDomain>: The <multiDomain> element should include the domain names (DNSDomain), server names (NetBiosName), IP addresses (IP), fully-qualified domain names (FQDN) of the Service Manager web tier server and other product servers (for example, the Release Control server). **Note**: The multi-domain functionality is relevant only for UI LW-SSO (not for web services LW-SSO). In addition, you must set the multiDomain element in each product for which you want to support LW-SSO.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<lwsso-config
xmlns="http://www.hp.com/astsecurity/idmenablmentfw/lwsso/2.0">
  <enableLWSSO
enableLWSSOFramework="true"
enableCookieCreation="true"
cookieCreationType="LWSSO"/>

  <webui>
    <validation>
      <in-ui-lwsso>
        <lwssoValidation id="ID000001">
         <domain>example.com</domain>
          <crypto cipherType="symmetricBlockCipher"
            engineName="AES" paddingModeName="CBC" keySize="256"
            encodingMode="Base64Url"
            initString="This is a shared secret passphrase"/>
        </lwssoValidation>
      </in-ui-lwsso>

      <validationPoint
        enabled="false"
        refid="ID000001"
        authenicationPointServer="http://server1.example.com:8080/bsf"/>

    </validation>

    <creation>
      <lwssoCreationRef>
        <lwssoValidationRef refid="ID000001"/>
        <expirationPeriod>50</expirationPeriod>
      </lwssoCreationRef>
    </creation>

    <logoutURLs>
      <url>.*/goodbye.jsp.*</url>
    </logoutURLs>

    <nonsecureURLs>
      <url>.*/sso_timeout.jsp.*</url>
    </nonsecureURLs>

<multiDomain>
  <trustedHosts>
        <DNSDomain>example.com</DNSDomain>
<DNSDomain>example1.com</DNSDomain>
<NetBiosName>myserver</NetBiosName>
<NetBiosName>myserver1</NetBiosName>
<IP>xxx.xxx.xxx.xxx</IP>
<IP>xxx.xxx.xxx.xxx</IP>
<FQDN>myserver.example.com</FQDN>
<FQDN>myserver1.example1.com</FQDN>
  </trustedHosts>
```

```
</multiDomain>

  </webui>

  ……

</lwsso-config>
```

5. Restart your Tomcat server.

July 2011