Setting up Single Sign-on in Service Manager

SSL Setup and Single Sign-on in Service Manager using Windows or Third Party Authentication

HP® Service Management

Introduction

Service Manager single sign-on functionality addresses the complexity of maintaining duplicate user accounts, multiple passwords, and separate logins across applications. By replacing the need to log into multiple applications using the same login and password with a single, secure login process, you can ensure that information is both secure and easily accessed. This single sign-on solution provides security and convenience while greatly reducing operational expenses.

This document is intended to help Hewlett Packard customers, consultants, and partners implement Service Manager single sign-on functionality. It provides the steps required to set up a basic single sign-on implementation. Custom environments with extensive tailoring may require more detail than what is provided in this document.

Overview of trusted sign-on

Activating single sign-on requires that you either create or purchase Secure Socket Layer (SSL) certificates for the Service Manager server, Service Manager Web Tier, and Service Manager Windows® clients. You can purchase SSL certificates from a *certificate authority* (CA), which is a trusted third party that issues root digital certificates and confirms certificate authenticity. You use these certificates to create a secure network connection between the Service Manager Windows-client and the Service Manager server, or between the Service Manager Web Tier and the Service Manager server. The connection between the user's Web browser and the Web Tier remains unchanged and requires no additional configuration in terms of importing certificates.

Note: When using Internet Explorer, only versions 5 and greater can handle single sign-on authentication without prompting for a username and a password.

Single sign-on is an optional Service Manager configuration that relies on a working SSL configuration, and integration with a trusted authentication source such as Integrated Windows Authentication or a network security management tool. This document addresses both options.

Prerequisites

The following software products must be available before you implement single sign-on for Service Manager using a private certificate authority:

- Service Manager client and server
- A fully implemented Service Manager Web Tier environment, if Web Clients are used
- Sun Microsystems™ JDK™ version 1.5 or greater
- The CertCreator.zip file, unzipped to a new directory. This zip file includes:
    - The `keytool` executable file

      For more information about `keytool` go to: http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html.
    - OpenSSL

      Note: If you purchase certificates from a certificate authority, you do not need to use OpenSSL to create certificates. The examples in this document use OpenSSL to create your own certificates.

For more information about downloading and installing OpenSSL go to www.openssl.org.

Other prerequisites

SSL must be enabled for the Service Manager server to support trusted sign-on. This capability allows both Windows and Web clients to bypass the Service Manager login screen after they have been pre-authenticated by a trusted source.

In a single sign-on scenario the Service Manager server grants access to clients only if *all* the following conditions are met:

- The signature on the client's signed certificate matches the certificate issued by the selected certificate authority.
- The client's signed certificate is on the list of trusted certificates in the Service Manager server's trusted client keystore (required when ssl_reqClientAuth is set to 2 for trusted sign-on).
- When prompted, always use the fully qualified name (*computer.domain.com*). keystool prompts for this information with `"What is your first and last name?"` whereas openssl prompts with `"Common Name"`
- The user's logon credentials match an existing operator record in Service Manager or a valid LDAP source that Service Manager recognizes. The user information must be from a domain user, and not a local user.
- A trusted authentication authority, such as the operating system, validates the user's logon credentials.
- All settings in the Web application server, Web server, and Web browser are set correctly. (Refer to the section *Configuration of the Web server and Web application server* for more information.)

The SSL Handshake

This section describes the SSL v3 handshake using RSA for key exchange.

SSL is not an encryption algorithm, but a protocol that allows two parties – the client and the server – to communicate using encryption by negotiating which encryption algorithm and key to use. Encryption provides confidentiality and ensures that the conversation between both parties is private. The purpose of the SSL handshake is to authenticate one or both parties of the connection as well as to negotiate which encryption algorithm and keys to use. The key in an SSL session is generated during the handshake and is known as the "session key". With a few exceptions, the session key is used only for the session it was created in.

The steps to the handshake are:

- ClientHello (C(lient) -> S(erver))
- ServerHello (C <- S)
- Certificate (C <- S)
- *CertificateRequest (C <- S)*
- ServerHelloDone (C <- S)
- *Certificate (C -> S)*
- ClientKeyExchange (C -> S)
- *CertificateVerify (C -> S)*
- ChangeCipherSpec (C -> S)
- Finished (C -> S)
- ChangeCipherSpec (C <- S)
- Finished (C <- S)

The steps in *italics* are only taken when client-side authentication is performed. Typically, only the server proves its identity during the handshake. "Client-side authentication" indicates the client will also be required to prove its identity during the handshake. Client-side authentication is not usually

done when an average user is browsing a web site using HTTPS (HTTP over SSL), but is more common with SSL connections between businesses.

The parts of the handshake

ClientHello (C -> S)

The handshake begins with a message from the client to the server called *ClientHello*. The ClientHello contains basic information indicating of what the client is capable. This message contains the client version, a random value, a session ID, a list of cipher suites, and a list of compression methods.

The client version indicates the highest version of SSL the client can support. It is assumed the client is backwards compatible. A client using SSL version 3 can also support version 2, which the server can negotiate down to if it chooses to do so.

The random value, known as a *nonce*, serves two purposes. The nonce is one of the variables used in generating the session key as discussed below, and also prevents replay attacks which are discussed in the *Finished* message.

The session ID can be used to indicate the client wants to resume a previously negotiated session. This saves time by not having to negotiate a new session key. The client will send a session ID of zero to indicate that a new session must be negotiated.

The cipher suites are a list of encryption algorithms the client supports, such as RSA with 3DES or RSA with IDEA. The client provides to the server a complete list of the ciphers it is able to support allowing the server to choose one.

The list of compression algorithms functions much like the list of cipher suites: the client provides a list of what it can do and the server can pick one. No compression algorithms are officially defined for SSL, so the only valid value is NULL. Some implementations of SSL support various compression methods although none have officially been defined.

ServerHello (C <- S)

The ServerHello message indicates which of the client options the server has chosen. The ServerHello message includes the SSL version the server will support for this connection, a nonce, a session ID, a cipher suite, and a compression method.

The nonce is a random value generated by the server that is used in the same fashion as the client's nonce.

The session ID, cipher suite, and compression method are all values chosen by the server and imposed onto the client. The client sent the values it can support, and the server makes the decision. If the server is not willing to support the client for any reason, the server aborts the handshake and closes the connection. This could happen if the server considers none of the client's cipher suites to be secure.

Certificate (C <- S)

As the first step in proving the servers identity, the server sends a copy of its digital certificate to the client. In some cases, such as the use of intermediate CAs, the server will also send the certificate of the issuer.

CertificateRequest (C <- S) (client-side authentication only)

For client-side authentication, the server can request the client's certificate. This message is issued when the server wants to verify the identity of the client.

ServerHelloDone (C <- S)

The server sends this message to the client to indicate it is done sending messages at this point.

After the client has received the server's certificate or certificate chain, it will validate the certificate. The client will check the subject name on the certificate and compare it to the domain name used to connect to the server. If the names do not match, the client may abort the handshake. On many popular web browsers though, the user is prompted to make a decision on whether to continue with the handshake or abort.

The client checks the valid dates on the certificate to make sure the certificate is not expired or is not being used before it was allegedly issued. The client also attempts to validate the digital signature on the server's certificate, assuming it trusts the issuer. That is, if the server sent a VeriSign certificate, the client would have to trust VeriSign before the client accepted the certificate. If the client cannot validate the certificate, the client will abort the handshake. Again, on many popular web browsers, the user is prompted to make a decision on whether to continue with the handshake or abort it.

Certificate (C -> S) (client-side authentication only)

If the server requested client certification with the CertificateRequest message, the client will send its own certificate.

ClientKeyExchange (C -> S)

The client uses the nonce values from the ClientHello and ServerHello messages and, using additional sources of randomization, creates the PreMasterSecret which is used by the client and server to derive the session key.

The integrity and confidentiality of the PreMasterSecret is important. For these reasons, the client encrypts the PreMasterSecret with the public key in the server's certificate, and sends it to the server as part of the ClientKeyExchange message. As long as the server is the legitimate owner of the certificate, it will have the private key necessary to decrypt the PreMasterSecret. If the server is actually an attacker posing as the owner of the certificate, it will be unable to decrypt the PreMasterSecret, which means it will be unable to derive the session key. Without the session key, the server is unable to complete the handshake.

CertificateVerify (C -> S) (client-side authentication only)

If the server requested the client's certificate with the CertificateRequest message, the client needs to prove it has the private key that corresponds to the public key in the certificate it sent to the server. The client accomplishes this by digitally signing the handshake messages sent up to this point using the client's private key and sends the result to the server.

The server attempts to validate the digital signature using the public key in the certificate provided by the client. If the signature fails validation, the server aborts the handshake and closes the connection.

ChangeCipherSpec (C -> S)

The client sends the ChangeCipherSpec message to the server to indicate it is switching to the negotiated encryption algorithm. Every message the client sends during the session from this point on is encrypted with the session key.

Finished (C -> S)

The client sends the Finished message to the server to indicate it is finished with the handshake. This message is encrypted with the session key, and contains a digital signature of the session key and the handshake messages up to this point.

The nonce values sent in the ClientHello and ServerHello messages help to ensure that the handshake messages from different SSL sessions are different, even if the sessions are between the same client and server. Without the nonce values, it may be possible under certain circumstances for an attacker to capture the handshake messages between the client and server and replay them later in an attempt to impersonate one side.

ChangeCipherSpec (C <- S)

The server sends the ChangeCipherSpec message to the client to indicate it is switching to the negotiated encryption algorithm. All messages from the server during this session will now be encrypted with the session key.

Finished (C <- S)

The server concludes the handshake by sending the Finished message, which is encrypted with the session key. This message contains the digital signature of the session key and all handshake messages in this session.

Connection scenario between a Windows client and Service Manager application server

The following figure depicts the connection process between a Service Manager Windows client and the Service Manager application server:

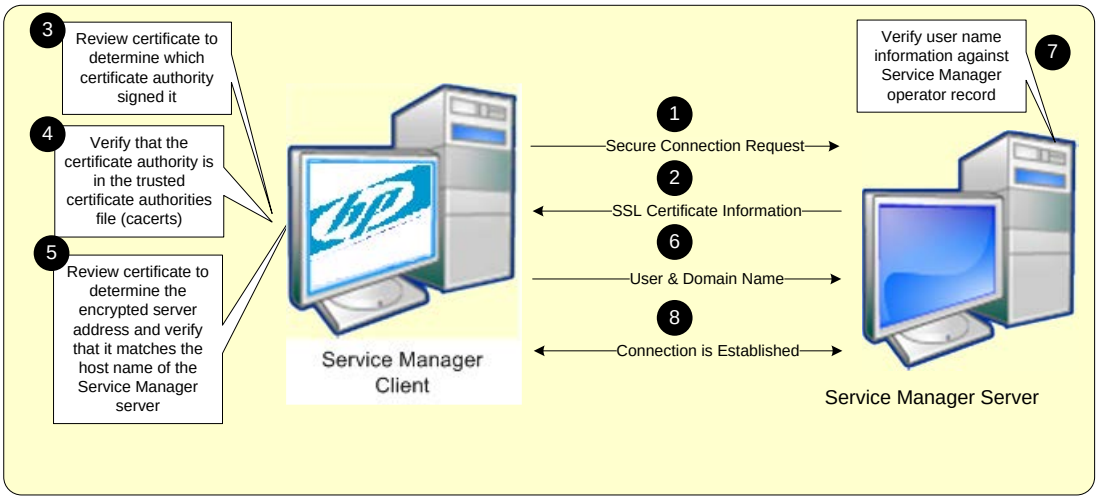


Figure 1: Service Manager client/server SSL handshake process

During the client/server handshake process, the client reads the server certificate, determines which certificate authority signed the certificate, and compares the certificate signature to a list of trusted certificate authorities that are identified in the `cacerts` file.

Note: Service Manager includes a sample server certificate signed by a fictitious certificate authority, and a modified `cacerts` file that includes the certificate for the fictitious certificate authority.

The client also compares the IP address or host name of the server to the address encrypted in the server certificate. If they do not match, Service Manager displays an alert and denies the connection. The Service Manager application server also checks whether the user was authenticated by a valid domain. Local machine authentication is not accepted by the Service Manager server.

Connection scenario between a Web server, a Web application server, and the Service Manager application server

The following figure depicts this connection process:

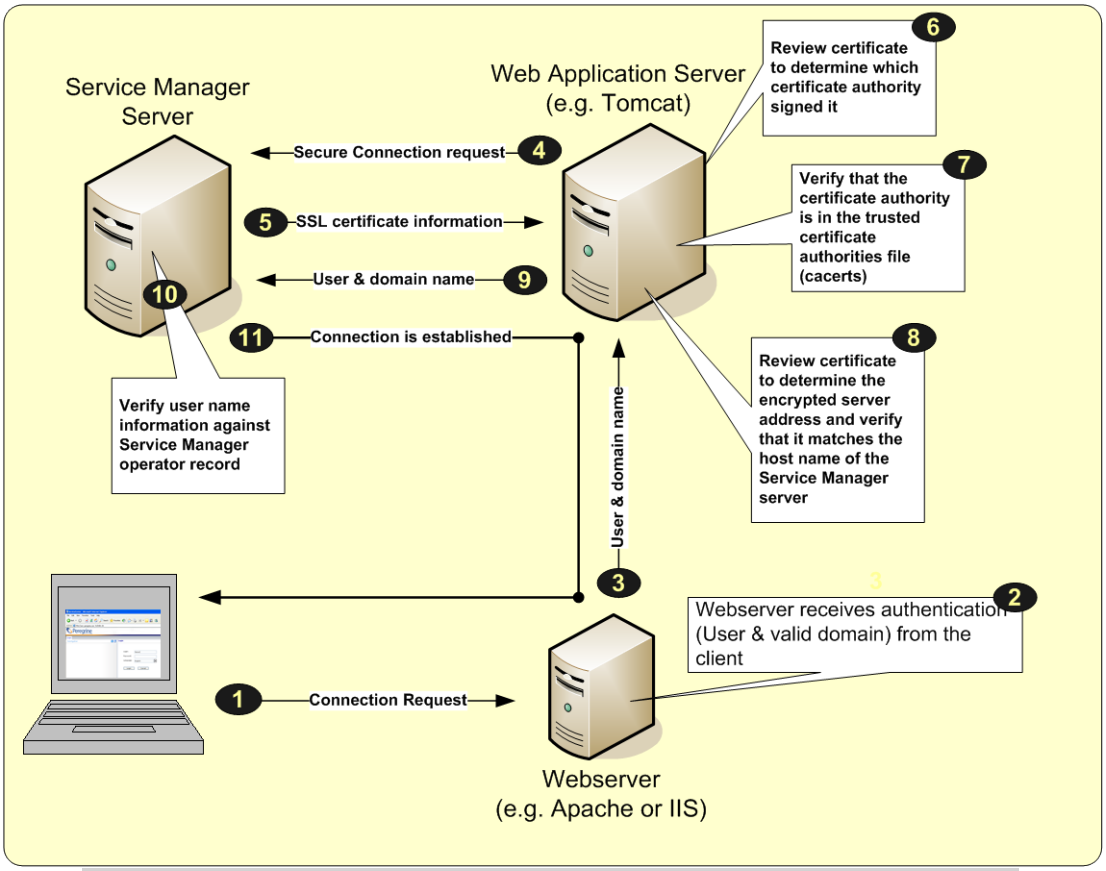


Figure 2: Service Manager Web Tier/server SSL handshake process

The Web server receives the user information from the client via the browser, and passes the user name and domain name to the Web application server.

Note: The Web application server (such as Tomcat, WebSphere®, or WebLogic Server®) acts as a client, and communicates with the Service Manager application server. All Web clients will authenticate with the Service Manager server using the certificates on the Web application server that they connect through.

The Service Manager application server also checks whether the user was authenticated by a valid domain. Local machine authentication is not accepted; if attempted the Service Manager server will reject such a request.

Server configuration

The SSL integration is performed in two parts: First on the server and then on the client. If you bought the certificates from a certificate authority, you may have to create the trusted clients keystore as shown in section "Import the signed certificate into the keystore" and you will have to edit the sm.ini as shown in section "Adding Service Manager SSL/Single sign-on parameters".

A configuration file (openssl.conf) is necessary to perform the following steps. You can open this file with any text editor (for example Notepad) and modify the `[ req_distinguished_name ]` section only to fit your needs, such as in the following example:

```
[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_default             = US
countryName_min                 = 2
countryName_max                 = 2
```

```
stateOrProvinceName      = State or Province Name (full name)
stateOrProvinceName_default  = CA
```

```
localityName          = Locality Name (eg, city)
localityName_default  = San Diego
```

```
0.organizationName       = Organization Name (eg, company)
0.organizationName_default   = Hewlett Packard
```

Note: You can modify the openssl.conf file, but do not delete any sections from it. Modifications to other sections are not recommended.

The OpenSSL configuration file has to be stored in the directory from which the openssl.exe command is called, usually the Service Manager RUN directory. To use a modified file called openssl.conf, append **–config ./openssl.conf** to each of the openssl req commands. For example:

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

Creating the server certificates using the batch file

The following batch file is provided with this paper. Each command used in the batch file will be described in more detail in this paper, but does not have to be performed manually if the batch file is used.

Note: Although the certificates will be created on windows using a batch file, they can be used on any supported platform.

You need to modify the following line in the batch file before running it:

```
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
```

Set the JAVA_HOME variable to the location of your 1.5.x JRE.

The batch file will be called as follows from a DOS command prompt in the directory that contains the content of the zip file provided with this document:

tso_srv_svlt.bat

You will be prompted for the following information. You can use the defaults by pressing enter or modify them according to your information. Please enter the proper fully qualified common name and the correct email address.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State [CA]:
Locality Name (eg, city) [San Diego]:
Organizational Name [HPSW]:
Organizational Unit Name (eg, section) [BTO]:
Common Name (eg, computer hostname) [server.domain.com]:server.domain.com
Email Address [user@domain.com]:first.last@company.com
```

When prompted if you trust this certificate, enter **yes**.

```
Importing the certificate into the System-wide keystore (cacerts)
Owner: EMAILADDRESS=first.last@company.com, CN=server.domain.com, OU=BTO,
O=HPSW, L=San Diego, ST=CA, C=US
Issuer: EMAILADDRESS=first.last@company.com, CN=server.domain.com,
OU=BTO, O=HPSW, L=San Diego, ST=CA, C=US
Serial number: f0ccd9267cf3c330
Valid from: Tue Jun 17 14:15:16 PDT 2008 until: Fri Jun 17 14:15:16 PDT
2011
Certificate fingerprints:
         MD5:  1F:AF:6D:EF:AF:50:D6:2B:66:C2:C5:6E:9E:42:7D:9D
         SHA1:
9C:E0:7E:D7:2C:F7:1E:47:D1:2E:D1:F3:D6:AB:83:A3:8A:2D:E9:35
Trust this certificate? [no]:  yes
```

When prompted enter the fully qualified host name (first and last name) as well as organizational information. When prompted for the password, press enter if it is the same as the keystore password.

```
Creating the Server keystore (server.keystore)

What is your first and last name?
  [Unknown]:  server.domain.com
What is the name of your organizational unit?
  [Unknown]:  HPSW
What is the name of your organization?
  [Unknown]:  HP
What is the name of your City or Locality?
  [Unknown]:  San Diego
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=server.domain.com, OU=HPSW, O=HP, L=San Diego, ST=CA, C=US correct?
  [no]:  yes

Enter key password for <smserver>
        (RETURN if same as keystore password):
```

After the batch file is finished, copy the cacerts file from the /certs subdirectory into the Service Manager server RUN directory and copy the server.keystore file from the key directory into the Service Manager RUN directory.

Vertically scaled systems

Once the batch file was run, copy the cacerts file from the /certs subdirectory into the Service Manager server RUN directory and copy the server.keystore file from the key directory into the Service Manager RUN directory.

Important: If the system is **horizontally scaled**, do not copy any files until the other server certificates are created and appended. Refer to the horizontally scaled section to continue.

View the contents of the signed certificate (optional)

Type the following command to view the signed certificate:

openssl x509 -in smservercert.pem -text –noout

A message such as the following is displayed:

```
Certificate:
    Data:
```

```
        Version: 1 (0x0)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, ST=CA, L=San Diego, O=Private HP CA, OU=HP CA,
        CN=server.domain.com/emailAddress=falcon@hp.com
        Validity
            Not Before: Mar 22 17:13:21 2006 GMT
            Not After : Mar 21 17:13:21 2009 GMT
        Subject: /C=US/ST=California/L=San
        Diego/O=HP/OU=TSG/CN=server.domain.com
        Subject Public Key Info:
            Public Key Algorithm: dsaEncryption
            DSA Public Key:
                 pub:
                    3b:ae:62:00:9c:d9:74:1f:6f:9a:60:f4:ea:30:72:
                    4b:2c:a0:2c:68:16:c2:c9:8a:a9:81:57:82:81:52:
                    b1:21:e1:02:74:bd:96:3c:75:ab:4f:0b:42:02:00:
                    4a:4e:1a:91:00:51:0f:d8:08:83:8a:ce:88:24:15:
                    05:5e:9e:4b:27:cc:86:fa:c1:67:ce:1a:1f:40:b5:
                    ad:0e:31:3b:76:b4:33:52:19:20:74:49:91:3e:cb:
                    68:30:18:92:e7:60:bf:ab:34:5c:c2:b1:ae:ec:9f:
                    84:64:f1:5f:7e:58:28:94:be:dc:63:5b:e2:e8:dc:
                    6a:cb:7d:78:0e:d3:84:59
                P:
                    00:fd:7f:53:81:1d:75:12:29:52:df:4a:9c:2e:ec:
                    e4:e7:f6:11:b7:52:3c:ef:44:00:c3:1e:3f:80:b6:
                    51:26:69:45:5d:40:22:51:fb:59:3d:8d:58:fa:bf:
                    c5:f5:ba:30:f6:cb:9b:55:6c:d7:81:3b:80:1d:34:
                    6f:f2:66:60:b7:6b:99:50:a5:a4:9f:9f:e8:04:7b:
                    10:22:c2:4f:bb:a9:d7:fe:b7:c6:1b:f8:3b:57:e7:
                    c6:a8:a6:15:0f:04:fb:83:f6:d3:c5:1e:c3:02:35:
                    54:13:5a:16:91:32:f6:75:f3:ae:2b:61:d7:2a:ef:
                    f2:22:03:19:9d:d1:48:01:c7
                Q:
                    00:97:60:50:8f:15:23:0b:cc:b2:92:b9:82:a2:eb:
                    84:0b:f0:58:1c:f5
                G:
                    00:f7:e1:a0:85:d6:9b:3d:de:cb:bc:ab:5c:36:b8:
                    57:b9:79:94:af:bb:fa:3a:ea:82:f9:57:4c:0b:3d:
                    07:82:67:51:59:57:8e:ba:d4:59:4f:e6:71:07:10:
                    81:80:b4:49:16:71:23:e8:4c:28:16:13:b7:cf:09:
                    32:8c:c8:a6:e1:3c:16:7a:8b:54:7c:8d:28:e0:a3:
                    ae:1e:2b:b3:a6:75:91:6e:a3:7f:0b:fa:21:35:62:
                    f1:fb:62:7a:01:24:3b:cc:a4:f1:be:a8:51:90:89:
                    a8:83:df:e1:5a:e5:9f:06:92:8b:66:5e:80:7b:55:
                    25:64:01:4c:3b:fe:cf:49:2a
    Signature Algorithm: md5WithRSAEncryption
        92:d4:33:b8:39:60:0f:72:c9:43:ee:64:1c:b6:48:85:11:bf:
        b9:d0:51:7d:da:70:1b:f0:7b:02:fc:af:06:07:a6:33:72:c2:
        ca:c7:c8:bf:e1:0a:90:00:a1:6f:cb:05:c0:c2:3e:7c:54:25:
        54:d0:e0:f0:8d:57:ba:7c:6d:0c:84:06:35:a1:f5:78:52:47:
        bb:ed:b4:ca:70:45:32:48:f9:bf:90:9d:30:a0:91:6d:5a:98:
        85:60:82:9b:f8:af:3b:63:4f:39:d0:b6:5f:70:6c:8c:44:83:
        4b:06:9a:3e:85:8b:ab:68:50:ee:0a:ef:d2:83:eb:ff:6f:d6:
        30:0b:ad:04:a8:b4:f5:77:7b:54:8c:21:83:6e:78:02:d8:95:
        0a:0a:e0:43:73:9b:f9:9d:ea:ab:a2:40:40:20:65:82:b1:90:
        b2:49:d6:a0:a0:d9:df:20:b3:50:23:61:c6:8b:7d:b1:1d:82:
        a8:32:f6:29:f2:f1:7f:1f:95:d8:39:89:0c:90:b8:64:e8:d9:
        df:70:87:d6:69:46:03:52:e4:63:d8:8a:0d:33:3d:b1:c5:07:
        68:39:00:ff:95:e1:f3:60:a4:60:ea:73:0a:70:a1:b2:71:9a:
```

```
        ec:cb:f9:33:e2:65:36:0c:5c:f4:0d:aa:13:3b:bd:e1:65:24:
        b2:df:29:6c
```

Print the keystore file (optional)

Enter the following command:

keytool -list -v -keystore server.keystore

A message such as the following is displayed:

```
Enter keystore password:  serverkeypwd

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: tomcat
Creation date: Mar 22, 2006
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=host.domain.com, OU=Server RTE, O=HP, L=San Diego,
ST=California, C=US
Issuer: EMAILADDRESS=name@domain.com, CN=host.domain.com, OU=HP  CA,
O=Private HP  CA, L=San Diego, ST=CA, C=US
Serial number: 2
Valid from: Wed Mar 22 09:13:21 PST 2006 until: Sat Mar 21 10:13:21 PDT
2009
Certificate fingerprints:
         MD5:  65:7C:11:5E:B6:D9:A4:CD:18:00:E4:82:20:54:95:CE
         SHA1: DB:9C:FD:7D:71:1A:A7:C3:04:DA:8B:24:26:33:47:11:56:81:
         70:61
Certificate[2]:
Owner: EMAILADDRESS=name@hp.com, CN=host.domain.com, OU=HP  CA, O=Private
HP  CA, L=San Diego, ST=CA, C=US
Issuer: EMAILADDRESS=name@hp.com, CN=host.domain.com, OU=HP  CA,
O=Private HP  CA, L=San Diego, ST=CA, C=US
Serial number: 0
Valid from: Tue Mar 21 14:38:00 PST 2006 until: Fri Mar 20 15:38:00 PDT
2009
Certificate fingerprints:
         MD5:  C1:52:73:03:30:69:21:33:CB:89:14:06:2F:3F:E2:1B
         SHA1: 53:D0:61:87:AB:5E:E8:E5:67:23:7E:A9:77:C7:EC:F0:99:6D:
         F9:00
```

To test that the server certificates are set up correctly, you can force SSL encryption by adding the ssl:1 parameter into the `sm.ini` file of the server and restarting the server. In the `sm.log` file, the line shown in italics below verifies that the startup was successful:

```
2328(   312) 10/06/2006 11:35:36 JRTE I SC servlet initialization
2328(   312) 10/06/2006 11:35:36 Initializing Coyote HTTP/1.1 on
http-13080
2328(   312) 10/06/2006 11:35:36 Starting Coyote HTTP/1.1 on http-13080
2328(   312) 10/06/2006 11:35:37 Initializing Coyote HTTP/1.1 on
http-13084
2328(   312) 10/06/2006 11:35:37 Starting Coyote HTTP/1.1 on http-13084
2328(   312) 10/06/2006 11:35:37 JRTE I Started Tomcat - HTTP port
is 13080
```

```
2328(   312) 10/06/2006 11:35:37 JRTE I Started Tomcat - HTTPS port
is 13084
```

Horizontally scaled systems

Important Requirements

On a horizontally scaled system, it is very important that all certificates are created from the same machine.- Ensure to not copy any files to their target directories until all certificates for all server machines in the horizontally scaled environment are created.

Running the batch file to create slave/secondary server certificates:

For each server node, create a server keystore. Append the same cacerts certificate to each newly created server keystore.

Once the tso_srv_svlt.bat batch job has finished, keep all files it created in the certs, crs and key directories.

tso_2nd_srvs_svlt.bat

This batch file is distributed in a zip file with this document. It is called from a DOS command prompt from the directory containing the unzipped content of the zip file.

Adjust the following line in the batch file to point to the correct location of your installed JRE prior to executing the batch file:

```
set JAVA_HOME ="INSERT PATH HERE"
```

Then call the batch file to create each of the slave server certificates as follows:

tso_2nd_srvs_svlt.bat <slave server machine name>

When prompted for the first and last name, enter the fully qualified computer name. When prompted for the password, press enter for a password that is the same as the keystore password.

```
Creating the Server keystore (<slavename.keystore)
What is your first and last name?
[Unknown]: fully qualified server.domain.co
What is the name of your organizational unit?
[Unknown]: HPSW
What is the name of your organization?
[Unknown]: HP
What is the name of your City or Locality?
[Unknown]: San Diego
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=client.domain.com, OU=HPSW, O=HP, L=San Diego, ST=CA, C=US
correct?
[no]: yes
Enter key password for <Server>
(RETURN if same as keystore password):
```

When prompted enter yes for trusting the certificate.

```
Importing Server public certificate into slavename.keystore
Owner: CN=server.domain.com, OU=HPSW, O=HP, L=San Diego, ST=CA, C=US
Issuer: EMAILADDRESS=first.last@company.com, CN=client.domain.com,
OU=BTO, O=HPSW, L=San Diego, ST=CA, C=US
Serial number: b6b6c8903ea66438
Valid from: Tue Jun 17 14:30:04 PDT 2008 until: Fri Jun 17 14:30:04 PDT
2011
Certificate fingerprints:
MD5: 76:4F:D9:2D:D4:4E:80:C0:F9:5A:1B:67:C0:D0:50:A1
```

```
SHA1:
F0:A4:36:F0:BD:CC:D5:BB:9D:35:8F:AD:BE:AB:EE:A7:F5:2B:9D:E9
Trust this certificate? [no]: yes
```

Repeat running the tso_2nd_srvs_svlt.bat <slave server machine name> batch file for every server node in the horizontally scaled environment.

Once all the server keystores are created, copy the cacerts from the certs folder to the Service Manager server\RUN directory on each server in the horizontally scaled environment. Then copy the server.keystore file from the key directory that was created in section Creating the server certificates using the batch file to the primary server's – the Loadbalancer server's – RUN directory.

Finally copy the <slavename_server>.keystore files from the key directory to each of the perspective machine's Service Manager server\RUN directories. The sm.ini on each server will reference the <slavename_server>.keystore using the keystoreFile parameter. Only the LoadBalancer servers's sm.ini will point to the server.keystore created in section Creating the server certificates using the batch file.

```
ssl:1
#ssl_reqClientAuth:2
trustedsignon:1
keystoreFile:slavename.keystore
keystorePass:serverkeystore
ssl_trustedClientsJKS:trustedclients.keystore
ssl_trustedClientsPwd:trustedclients
truststoreFile:cacerts
truststorePass:changeit
```

Windows Client or Web Tier configuration

This section shows how you configure the Windows client or the Web Tier.

Notes:

- For each Windows client you need a unique client certificate.
- For each Service Manager Web application server you need a unique client certificate.
- If a Windows client and the Web application server are on the same physical machine, it is possible to use the same `cacerts` and `clientcerts` files for both, rather than creating two sets of nearly identical keystores. In such a case, copy the files created for the Windows or Web client – whichever was created first– into either the `<Service Manager Client>/plugins/com.hp.ov.sm.client.common_x.xx` directory or the `Service Manager/WEB-INF` folder of the Web application server.

Best Practice recommendation: When you configure the Web tier, type the word **web** in front 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.

Running the batch file to create client certificates

The following batch file will be distributed with this document. It will be called as follows from a DOS command prompt in the directory that contains the content of the zip file provided with this document.

You need to modify the following line in the batch file before running it:

```
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
```

Set the JAVA_HOME variable to the location of your 1.5.x JRE.

Call the batch file to create the client certificates as follows:

tso_cln_svlt.bat <client machine name>

When prompted, enter the following information. First and last name are the fully qualified computer name. When prompted for the password, press enter if the password is the same as the keystore password.

```
Creating the Client keystore (client.keystore)

What is your first and last name?
  [Unknown]:  client.domain.com
What is the name of your organizational unit?
  [Unknown]:  HPSW
What is the name of your organization?
  [Unknown]:  HP
What is the name of your City or Locality?
  [Unknown]:  San Diego
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=client.domain.com, OU=HPSW, O=HP, L=San Diego, ST=CA, C=US correct?
  [no]:  yes
Enter key password for <client>
        (RETURN if same as keystore password):
```

When prompted enter yes for trusting the certificate.

```
Importing Client public certificate into Trustedclients keystore
(trustedclients.keystore)

Owner: CN=client.domain.com, OU=HPSW, O=HP, L=San Diego, ST=CA, C=US
Issuer: EMAILADDRESS=first.last@company.com, CN=client.domain.com,
OU=BTO, O=HPSW, L=San Diego, ST=CA, C=US
Serial number: b6b6c8903ea66438
Valid from: Tue Jun 17 14:30:04 PDT 2008 until: Fri Jun 17 14:30:04 PDT
2011
Certificate fingerprints:
         MD5:  76:4F:D9:2D:D4:4E:80:C0:F9:5A:1B:67:C0:D0:50:A1
         SHA1:
F0:A4:36:F0:BD:CC:D5:BB:9D:35:8F:AD:BE:AB:EE:A7:F5:2B:9D:E9
Trust this certificate? [no]:  yes
```

Copy the trustedclients.keystore from the certs directory to the Service Manager server RUN directory. Copy the smclient.keystore and the certs/cacerts to the <*Service Manager Client*>/plugins/com.hp.ov.sm.client.common_x.xx directory for windows clients or the WEB-INF folder of the Service Manager Web Application Server for web clients.

Note: smclient.keystore in this paper is a placeholder for <clientname>.keystore, since every client's keystore file will be named differently.

To import more client public keys into the `trustedclients.keystore` file, repeat these steps below for each client certificate:

Import this certificate into the truststore

```
@echo Importing Client public certificate into Trustedclients keystore
(trustedclients.keystore)
@echo.
%KEYTOOL% -import -alias %1 -file certs/clientpubkey.cert -keystore
certs/trustedclients.keystore -storepass %TRUSTEDCLIENTS_KEYSTORE_PASSWD%
@echo.
```

Enter the following command:

keytool -import -alias <host name> -file clientpubkey.crt -keystore trustedclients.keystore -storepass <TRUSTEDCLIENTS_KEYSTORE_PASSWD>

A message such as the following is displayed:

```
Owner: CN=server.domain.com, OU=Client, O=HP, L=SD, ST=CA, C=US
Issuer: EMAILADDRESS=falcon@hp.com, CN=server.domain.com, OU=HP CA,
O=Private HP CA, L=San Diego, ST=CA, C=US
Serial number: 3
Valid from: Thu Mar 30 16:38:57 PST 2006 until: Sun Mar 29 16:38:57 PST
2009
Certificate fingerprints:
         MD5:  8B:F4:57:C4:BD:C6:92:8A:CB:3B:F2:4E:44:3A:75:EE
         SHA1:
46:3C:6E:A8:B3:1D:0B:D3:33:C2:A0:B8:C0:98:90:28:38:C7:3E:FD
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Configuring Service Manager to use SSL

Setting the security preferences for the clients

Whether you are using the Web client or the Windows client, you need to copy its signed client certificate to the Service Manager RUN directory and set SSL preferences in Service Manager. This task varies based on the client type.

For Service Manager Windows clients

1. Copy the `cacerts` and `smclient.keystore` files to the `<Service Manager Client>/plugins/com.hp.ov.sm.client.common_x.xx` directory.
2. Open the Service Manager Windows client.
3. Click **Windows** -> **Preferences** -> **Service Manager** -> **Security** to open the Security Preferences window.
4. Set the CA certificates file to the fully qualified path of the `cacerts` file that you copied to the `com.hp.ov.sm.client.common_x.xx` directory.
5. Set the client `keystore` file to the fully qualified path of the `smclient.keystore` file that you copied to the `com.hp.ov.sm.client.common_x.xx` directory.
6. Enter the password of the client's `keystore` (**CLIENT_KEYSTORE_PASSWD** from the batch file) into the client Keystore Password field.
7. Click **OK** to save the settings and close any open connections.
8. Once the server is configured for SSL authentication and ssl:1 is activated in the sm.ini, click **File** -> **Connect** -> **Connections** to open the Service Manager Connections window. Click the **Advanced** tab and click to select the **Use SSL Encryption** option.
9. Under the connections, make sure to use the host name that was used in the section *Error! Reference source not found.*. Do *not* use localhost if server and client reside on the same machine.
10. Click **Apply** -> **Close** to apply and save the options.

For Service Manager Web clients

1. Copy the `cacerts` and *`smclient.keystore`* files to the `WEB-INF` folder of the Service Manager Web Application Server.

   Note: These files can also be copied to a shared network drive that is accessible to the Service Manager Web Tier client.

2. Stop the Web application server running the Service Manager Web Tier.
3. Open the Web configuration file, `web.xml`, in a text editor.
4. Make sure that the serverHost parameter contains the fully qualified name of the Service Manager Server, as follows:

```
<init-param>
<param-name>serverHost</param-name>
<param-value>servername.domainname.com</param-value>
</init-param>
```

5. Control the encryption of network communication between the application server and the Service Manager server, using the following entries to the `web.xml` file:

```
<init-param>
<param-name>ssl</param-name>
<param-value>true</param-value>
</init-param>
```

6. Set the cacerts parameter to the keystore file that contains your server's certificate authority (for example `cacerts`). This is the keystore file that you copied into the `WEB-INF` folder in step 1.
7. Specify the client's private keystore to use in encrypted communication. If this is a relative path, it will be relative to the Web application's deploy directory, but still needs a leading backslash (for example **/WEB-INF/smclient.keystore**)

```
<init-param>
   <param-name>keystore</param-name>
   <param-value>enter path to smclient.keystore here</param-value>
</init-param>
```

8. Specify the password for the client's private keystore

```
<init-param>
   <param-name> keystorePassword</param-name>
   <param-value>enter keystore password here</param-value>
</init-param>
```

9. 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 login to the system.

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

Adding Service Manager SSL/Single sign-on parameters

This section provides information about parameters that you add to enable SSL/Trusted sign-on.

1. Using a text editor, open the `sm.ini` file located in your Service Manager `RUN` directory.
2. Set the `sslConnector` parameter to 1 if it is 0.
3. Add the following parameters to the `sm.ini` file.

Mandatory parameters for all single sign-on implementations

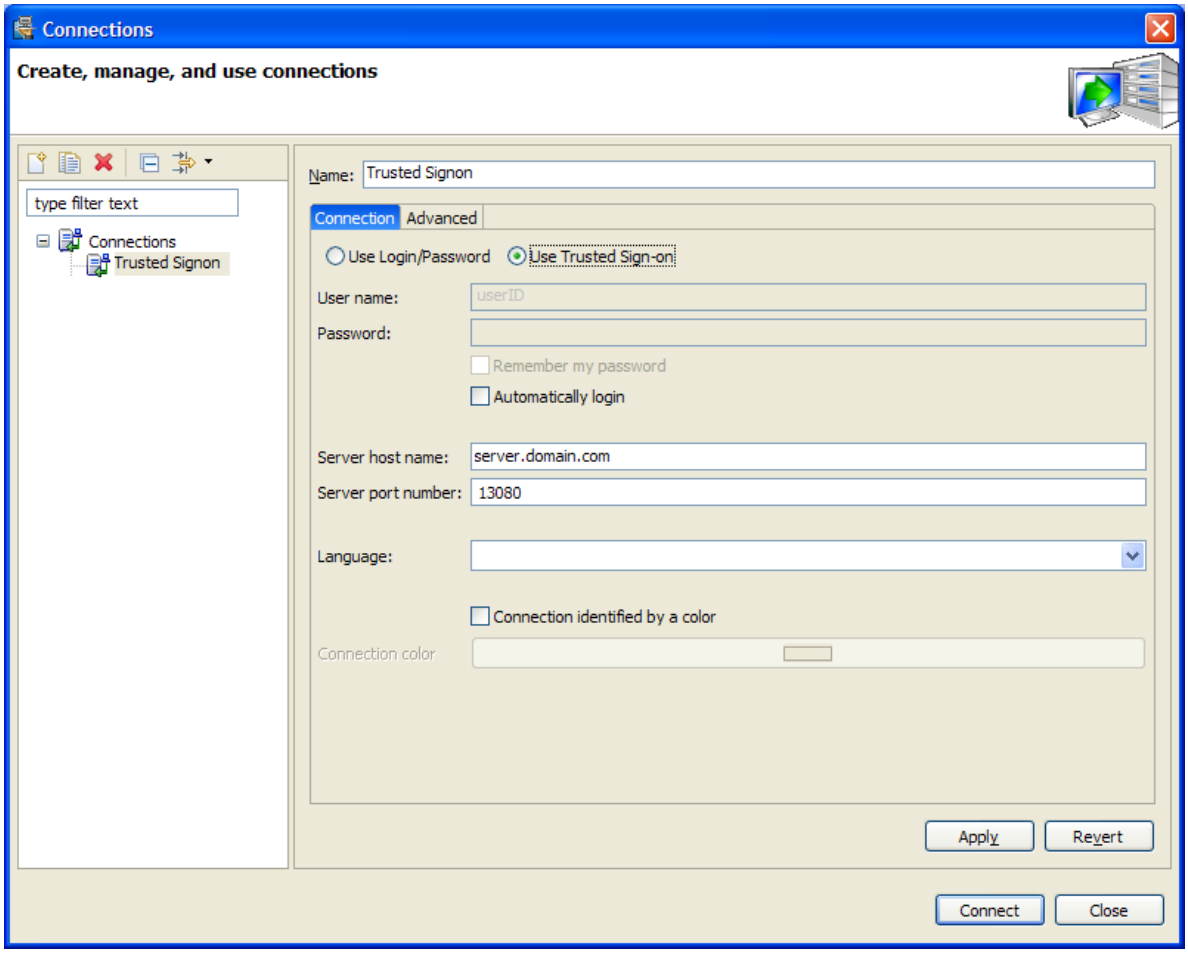
Parameter	Description

ssl:1	Enables SSL encryption requiring all clients to use SSL connections.
ssl_reqClientAuth:2	Clients are required to present signed certificates to the server and need to be on the list of trusted clients.
trustedsignon:1	Single sign-on capability is enabled.
keystoreFile:*server.keystore*	keystore file containing the Service Manager server certificate and private key
keystorePass:<*ServerKeyPwd*>	Password to the Service Manager server keystore
ssl_trustedClientsJKS:*trustedClients.jks*	keystore file containing the signed certificates of trusted SC clients
ssl_trustedClientsPwd:<*TrustedClientsPwd*>	Password to the trusted client keystore
truststoreFile:*cacerts*	keystore containing the certificate authority's certificate
truststorePass:<*changeit*>	Password to the CA keystore

4. Save the `sm.ini` file and close the text editor.
5. All these parameters will apply to each new client connection that is established; however HP recommends restarting the Service Manager server to ensure that all clients are using SSL connections.

Test the configuration

You should now be able to start the Service Manager server, and then log on to the server through your Windows client using SSL. Test the SSL connection first before enabling trusted sign-on. To use trusted sign-on, ensure that you have an operator record with the same username and password as those you use to log on to the network. No third-party product is necessary to set up single sign-on with the Windows client. On the connections screen, use the following setup:



Set up your Web application server (such as Apache or IIS) to allow for trusted sign-on using the web client.

Configure Service Manager Web clients to validate the Service Manager server's signed certificate, present signed client certificates, and identify the trusted authentication source as described in the section *Setting the security preferences - For Service Manager Web clients*.

Configuration of the Web server and Web application server

This section provides some examples for configuring the Web server and the Web application server to enable trusted sign-on. The Web server must be compatible with the Web tier application server.

Note: The following sections assume that the Web server and Web application server configurations are already established, and that the only necessary changes to the configurations of these servers are those described in this document.

Tomcat with Apache/Internet Information Server

Tomcat configuration changes

1.If you are using Tomcat 5.0.x, enter

request.tomcatAuthentication=false

at the end of the `/tomcat/conf/jk2.properties` file.

2. As of Tomcat 5.5.x, it no longer uses jk2.properties by default. For Tomcat 5.5.x, include the tomcatAuthentication="false" parameter in the jk2 worker port definition.
3. Open the `Tomcat/conf/server.xml` file in the same directory and search for the following line:

   <!- Define a Coyote/JK2 AJP 1.3 Connector on port 8009 ->

   Change the parameters in this section from

   <Connector port="8009"
   enableLookups="false" redirectPort="8443" debug="0" protocol="AJP/1.3" />

   to

   <Connector port="8009"
   enableLookups="false" tomcatAuthentication="false" redirectPort="8443"
   debug="0" protocol="AJP/1.3" />

4. Save the file.
5. Restart Tomcat for the changes to take effect.

Apache configuration changes

Note: The mod_auth_sspi.so module is available only for Windows. If Apache is installed on a UNIX® operating system, it is necessary to create a custom class to perform trusted sign-on.

6. Add the mod_auth_sspi.so module to the `/modules` directory in the Apache installation.
7. Add the following lines to the bottom of the `http.conf` file to allow for trusted sign-on:

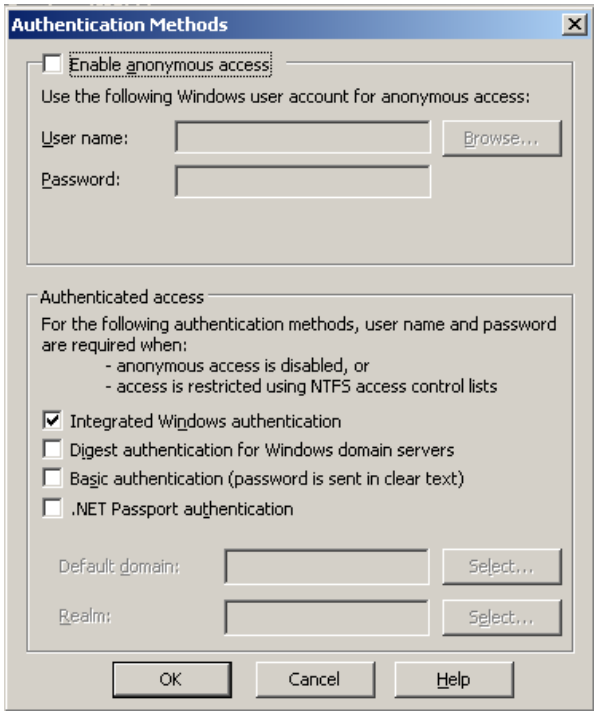
```
#SspiAuth Module
LoadModule sspi_auth_module modules/mod_auth_sspi.so

<Location "/sm">
   AllowOverride None
   Options None
   Order allow,deny
   Allow from all
   AuthType SSPI
   SSPIAuth On
   SSPIDomain MYDOMAIN
   SSPIAuthoritative On
   SSPIOfferBasic Off
   SSPIPerRequestAuth On
   require valid-user
</Location>
```

   The name within the Location tag needs to be the path the user enters to open the Service Manager Web Client Web site; it is usually `/sm`, since the name is taken from the `sm.war` file. In a configuration with multiple domains, comment out the SSPIDomain parameter by adding a crosshatch character (#) in front of the line.

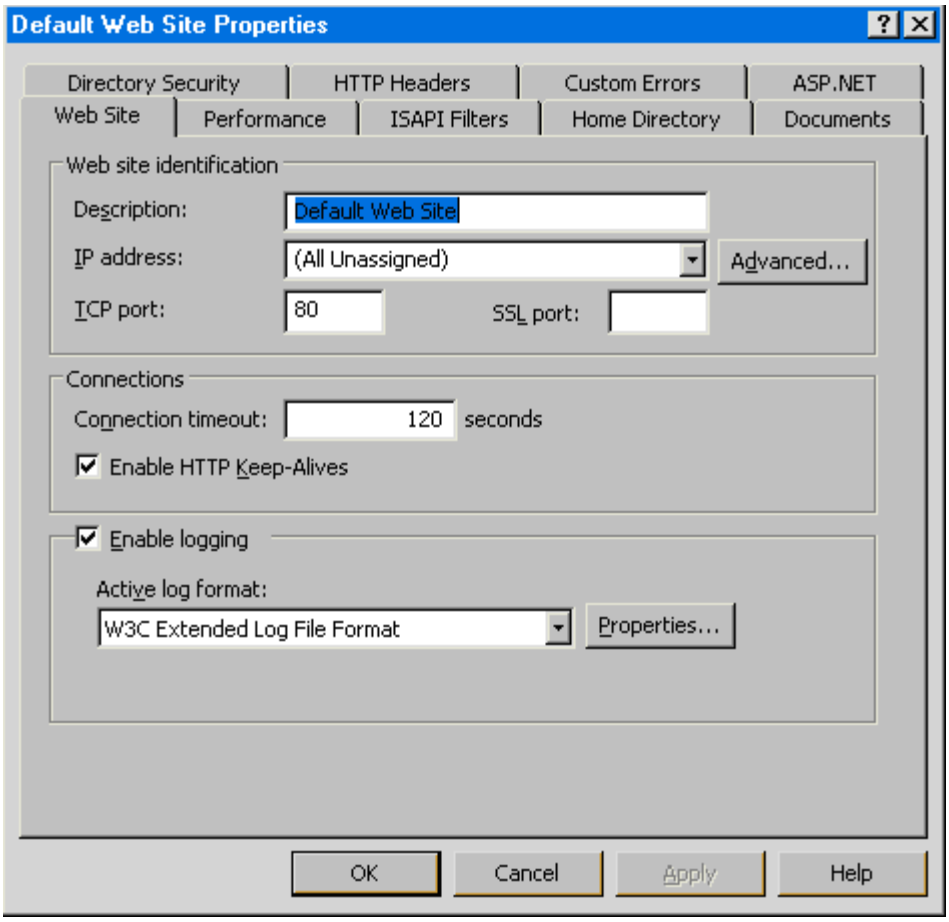
Internet Information Server

In the Properties window for the Service Manager virtual directory, click the **Directory Security** tab and enter information as shown below:

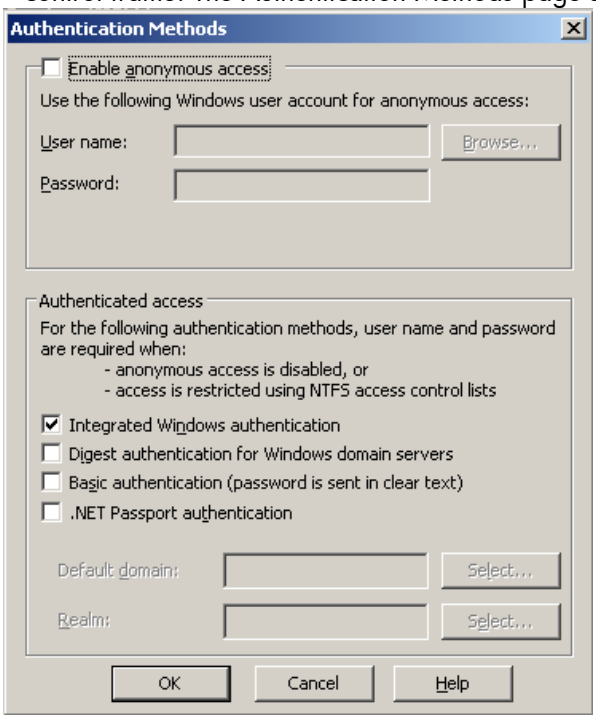


Configuring Internet Information Server version 6

1. Open the IIS Manager (Start – Administrative Tools – Internet Information Services (IIS) Manager)
2. Click on Web Service Extensions
3. Set extension status to Allowed for All Unknown ISAPI Extensions
4. Optionally set Active Server Pages on Allowed.
5. Check the properties of the Default Web Site.



6. Go to the ISAPI Filters tab and check if the green upward arrow is in the Status column points
7. Go to the Directory Security tab and click on the Edit button in the Authentication and access control frame. The Authentication Methods page should have the following settings:



Ensure to disable “Enable anonymous access” and enable “Integrated Windows authentication”

Optionally, Advanced Digest Authentication can be enabled.

Additional Information: Advanced Digest Authentication is an extension of Digest security. Digest security uses MD5 hashing to encrypt user credentials (user name, password and user roles).

Basic authentication sends the user name and password details over the network in base64 encoded format. These details can be easily "sniffed" (captured with a protocol analyzer) and decoded by an intruder, who could then use the credentials for nefarious purposes. Digest security's MD5 hash enhances security by applying cipher algorithms that are more sophisticated and more difficult to crack. An MD5 hash is binary data consisting of the encrypted user name, password and realm. The 'realm' is the name of the domain that authenticates the user.

The MD5 hash is embedded into an HTTP 1.1 header thus is only supported by HTTP 1.1-enabled browsers. Digest or Advanced Digest authentication mechanisms can not be enabled if the target browsers do not support HTTP 1.1.

Advanced Digest Security takes the Digest authentication model a bit further by storing the user credentials on a domain controller as an MD5 hash in the Active Directory database. Intruders would need to get access to the Active Directory to steal the credentials. This adds another layer of security to protect access to Windows 2003 Web sites.

Both Digest and Advanced Digest Authentication only work on Web Distributed Authoring and Versioning (WebDAV) enabled directories. WebDAV (formerly called Web Folders) is a secure file transfer protocol that lets people download, upload, and manage files on remote computers across the internet and intranets WebDAV is similar to the File Transfer Protocol (FTP) except that WebDAV always uses password security and data encryption on file transfers, whereas FTP doesn't support those features.

When you enable this feature, you'll get the message: "Digest authentication only works with Active Directory domain accounts. For more Information about configuring Active Directory domain accounts to allow digest authentication click Help. Are you sure you want to continue (Yes, No, Help).

Clicking on Help gives the following information:

Digest Authentication Warning

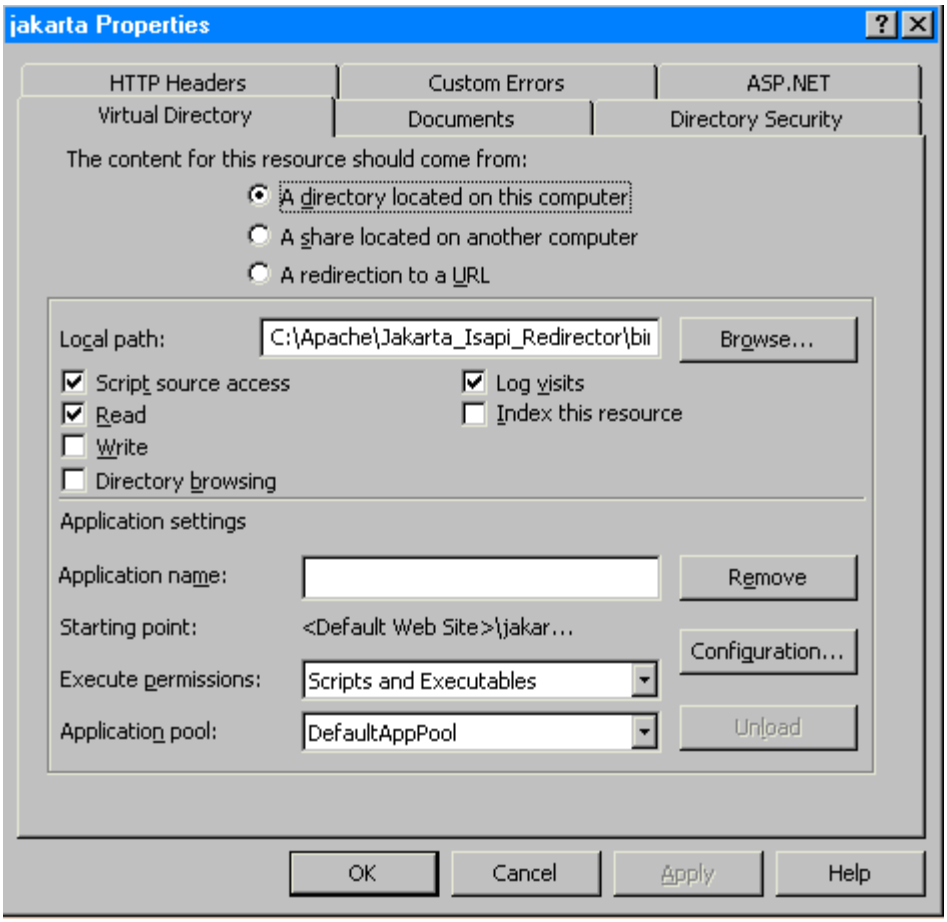
The authenticated access method, Digest authentication, applies only to domain accounts on servers running Microsoft® Windows® Server 2003 and requires the accounts to store passwords using reversible encryption. Internet Information Services (IIS) sends a hash value rather than the password over the network, working across proxy servers and other firewalls.

Requirements for Digest Authentication

Before enabling Digest authentication on your server running IIS, ensure that all of the following minimum requirements are met. Only domain administrators can verify that the domain controller requirements are met. Check with your domain administrator if you are unsure about whether your domain controller meets the following requirements:

- *· All clients that access a resource that is secured with Digest authentication are using Microsoft Internet Explorer 5.0 or later.*
- *· The user and the server running IIS must be members of, or be trusted by, the same domain.*
- *· Users must have a valid Windows user account stored in Active Directory® on the domain controller.*
- *· The domain must have a Windows 2000 or later domain controller.*
- *· The IIS server must be running a member of the Windows Server 2003 family or later.*

8. In the Default Web Site Folder, right click on Jakarta and select Properties
Ensure to have the following settings on the Virtual Directory tab:



9. On the Directory Security Tab, click on Edit in the Authentication and access control frame.
10. Ensure to have Enable anonymous access disabled and Integrated Windows authentication enabled.
11. Restart the Internet Information Server service.

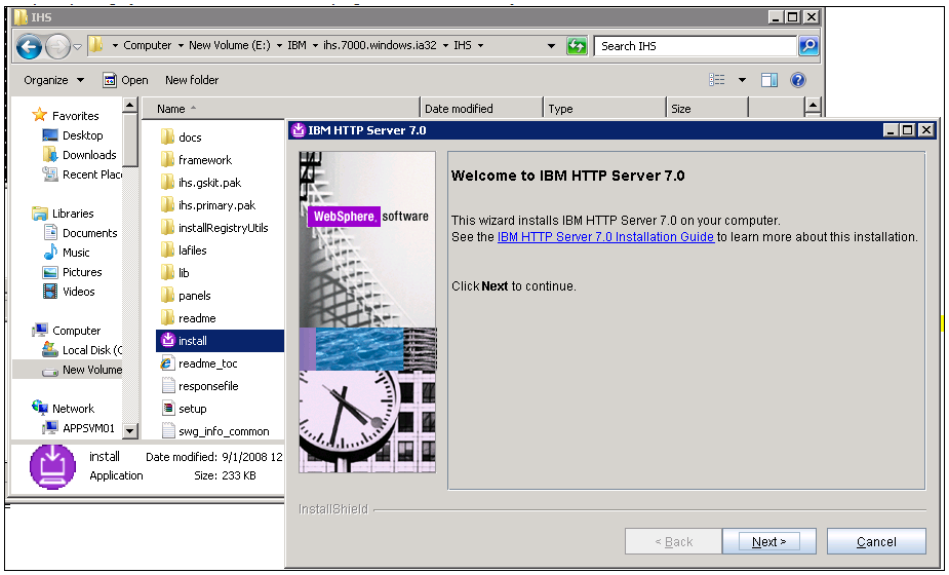
WebSphere with IBM HTTP Server

This section provides an example for how to enable trusted sign-on using IBM WebSphere Application Server 7.0 and IBM HTTP Server 7.0.

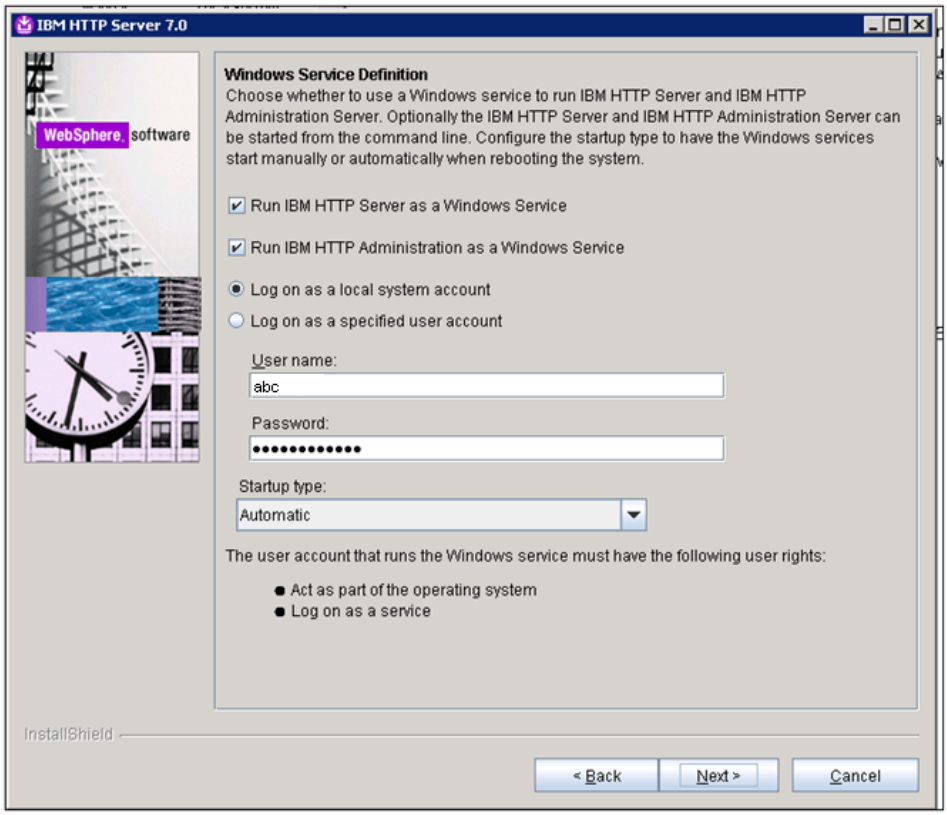
Note: IBM HTTP Server is not officially supported by Service Manager as of the writing of this section. Therefore, the configurations provided in this section just serve for your reference only. For more information about the officially supported Web tier application servers and Web servers, see document *Compatibility Matrix* available on HP Software Product Manuals web site.

Installing IBM HTTP Server

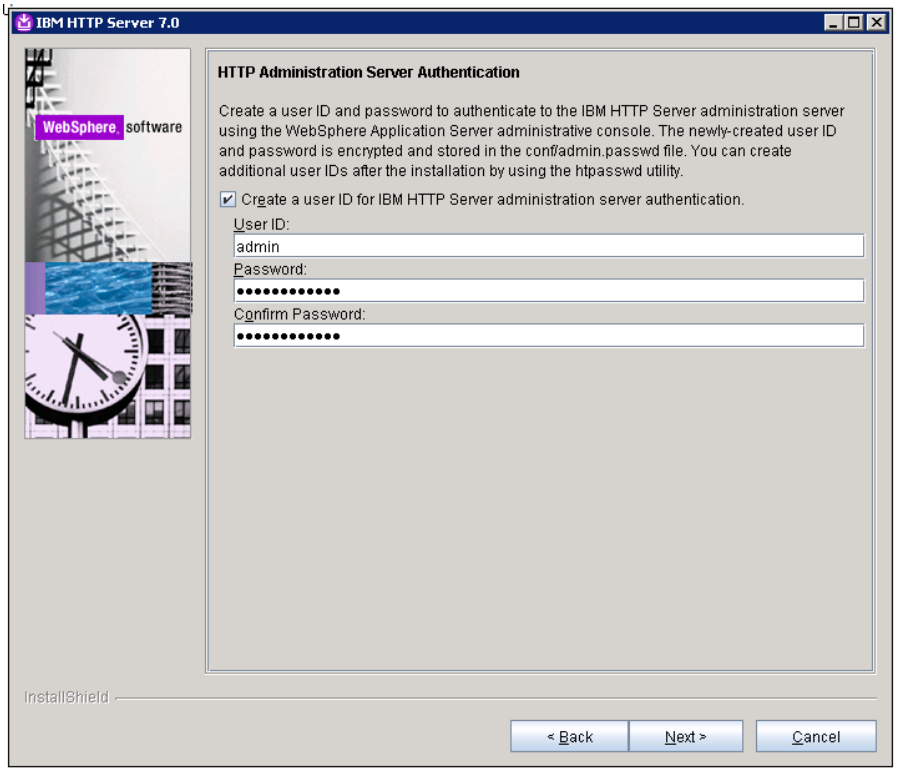
1. Download IBM HTTP Server 7.0 from IBM web site.
2. Unzip the installation package you downloaded, and then run the install program under the `IHS` directory.



3.Define Windows Service.



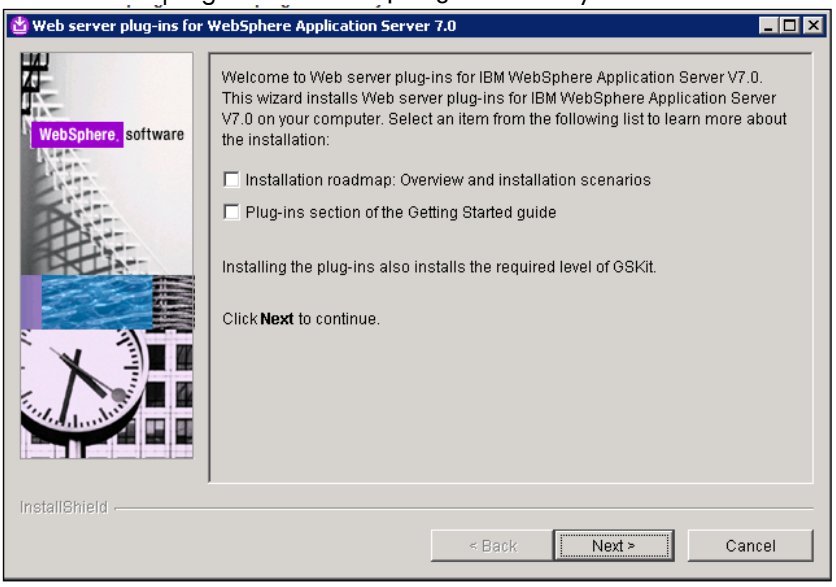
4.Create a user ID and password for HTTP administration server authentication.



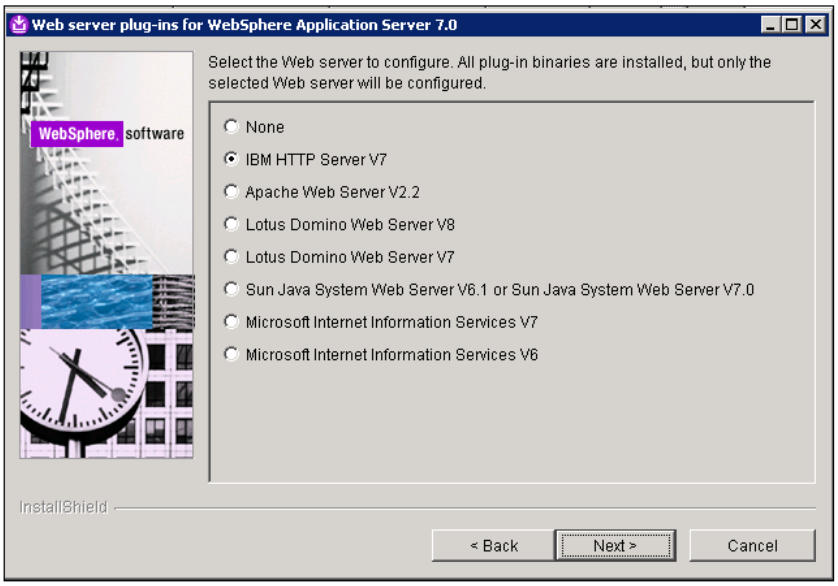
5.Follow the on-screen instruction to complete the installation of IBM HTTP Server.

Installing Web server plug-ins for WebSphere Application Server

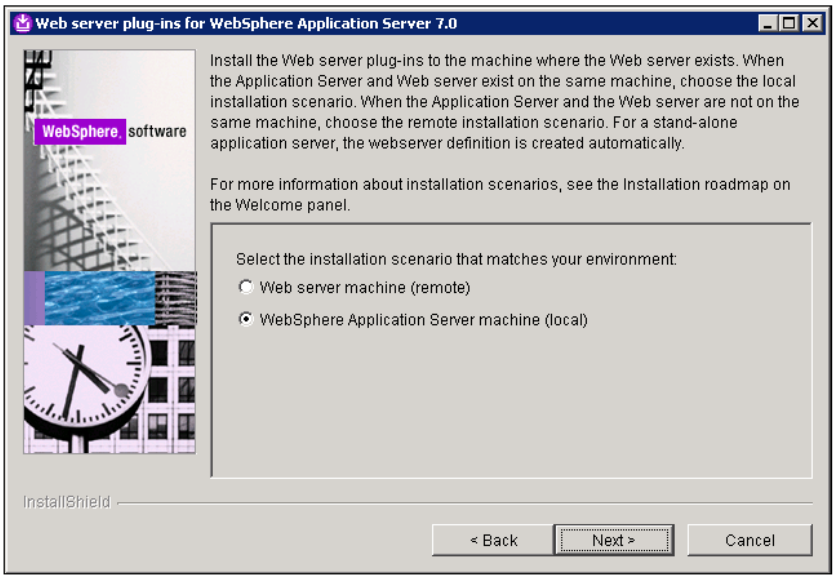
1.Run the install program under the `plugin` directory.



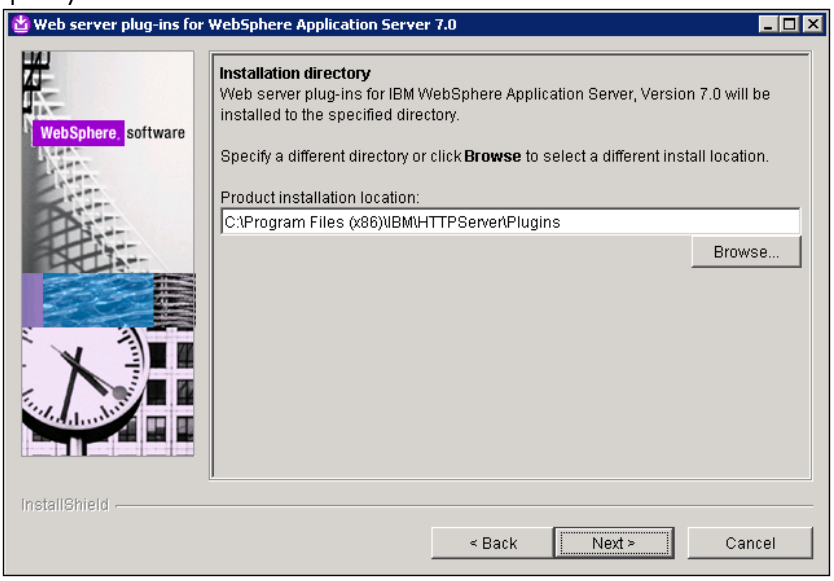
2.Select **IBM HTTP Server v7** as the Web server to configure.



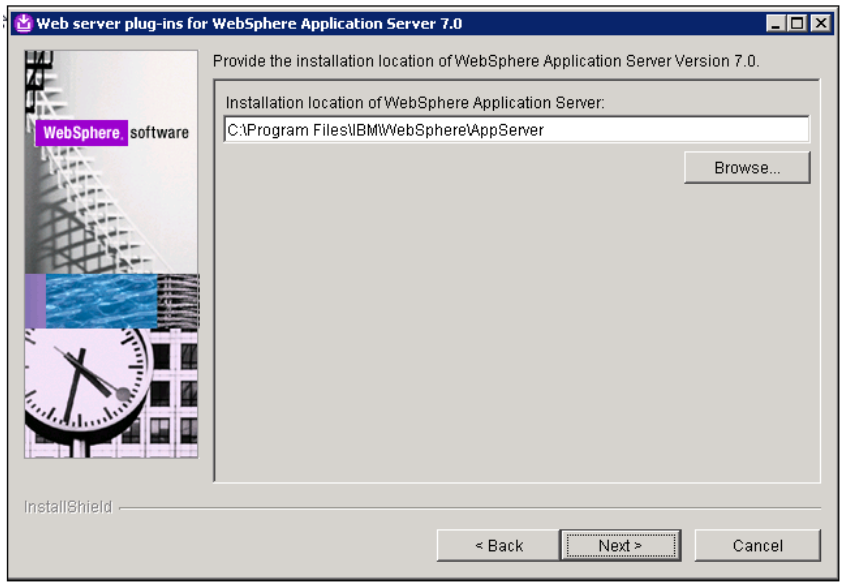
3.Select the installation scenario.



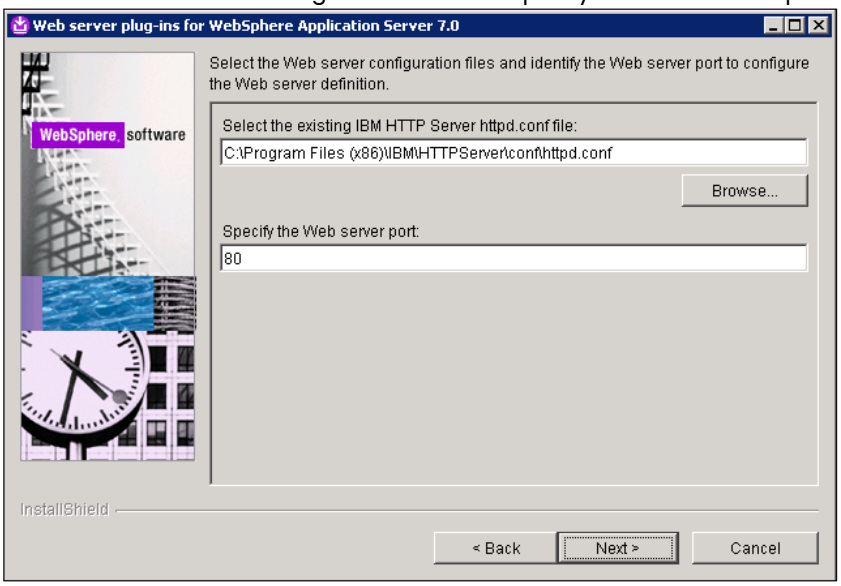
4.Specify the installation location.



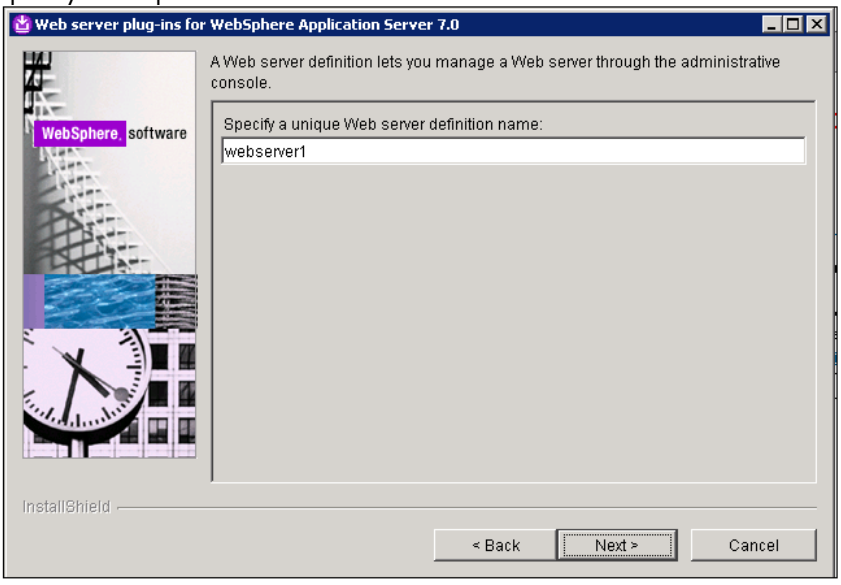
5.Choose the installation location of WebSphere Application Server Version 7.0.



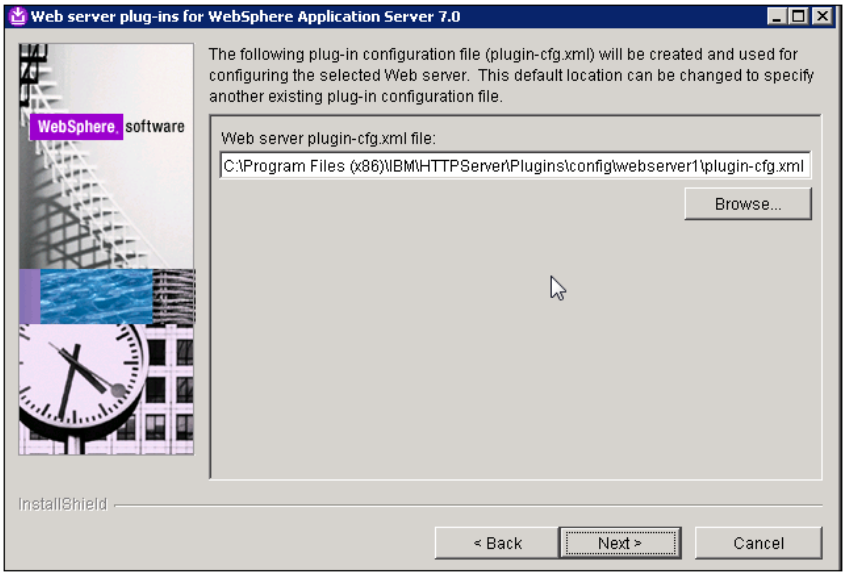
6. Select the Web server configuration file and specify the Web server port.



7. Specify a unique Web server name.



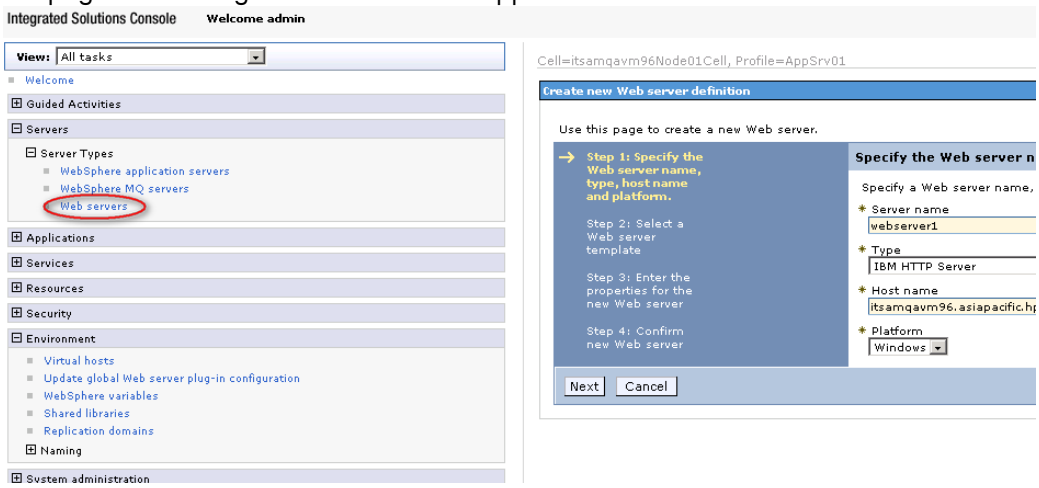
8. Create the Web server plug-in configuration file (`plugin-cfg.xml`).



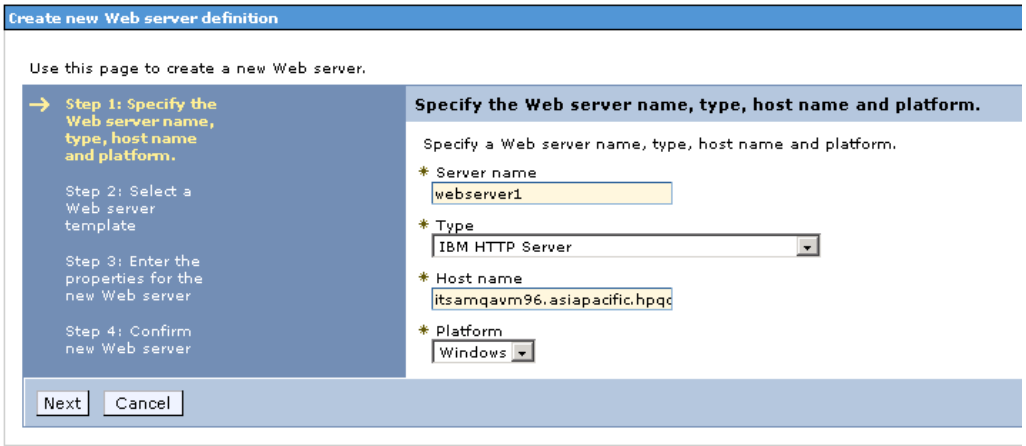
9. Click **Finish** to complete the installation.

Creating a Web server on Websphere

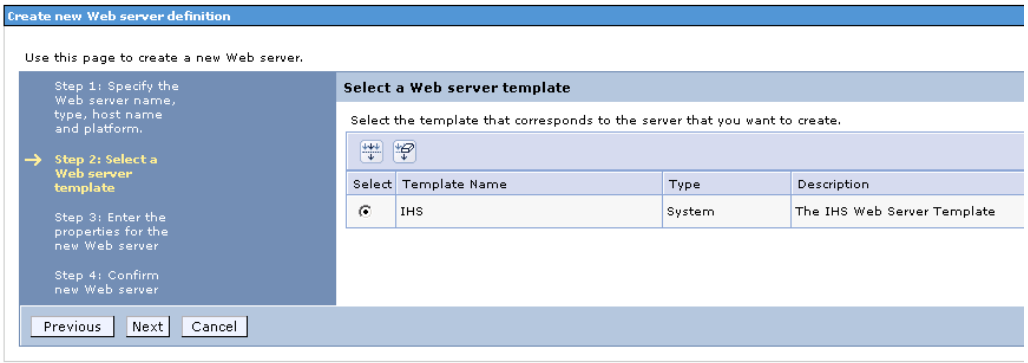
1. From the Windows **Start** menu, click **All Programs** -> **IBM Websphere** -> **Application Server7.0** -> **Profiles** -> **AppSvr01** -> **Administrative Console**.
2. Enter the admin user name and password to log on to the Integrated Solution Console.
3. From the left panel, select **Servers** -> **Server Types** -> **Web servers**.
   The page for creating a new Web server appears.



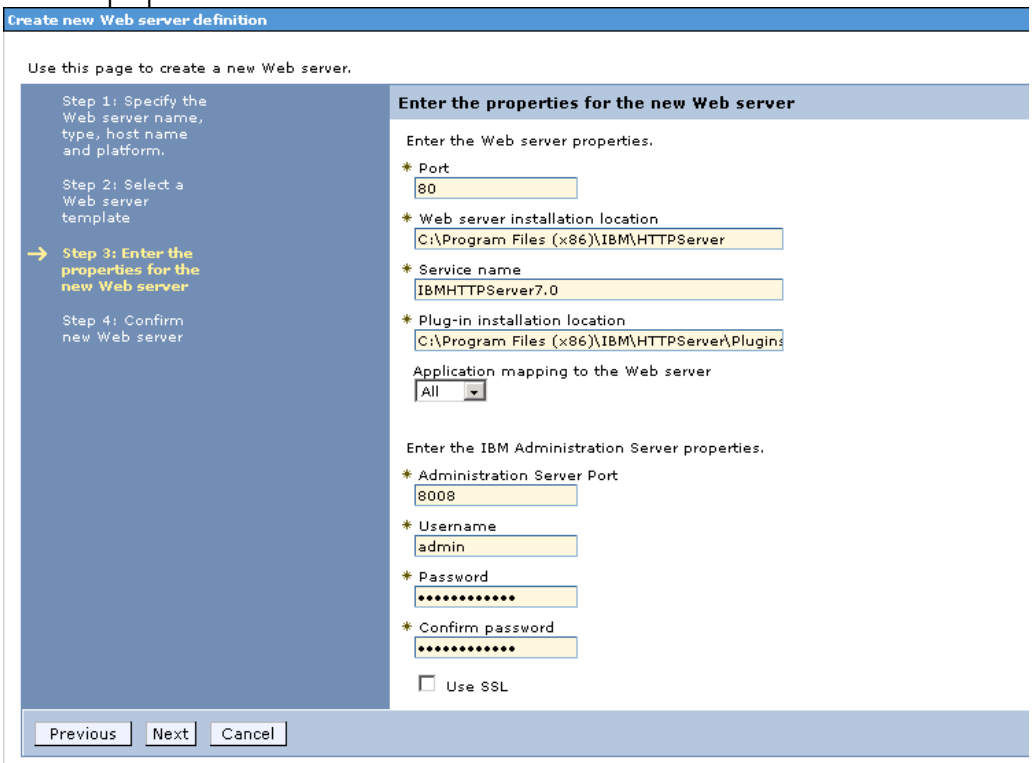
4. Specify the Web server name, type, host name and platform.



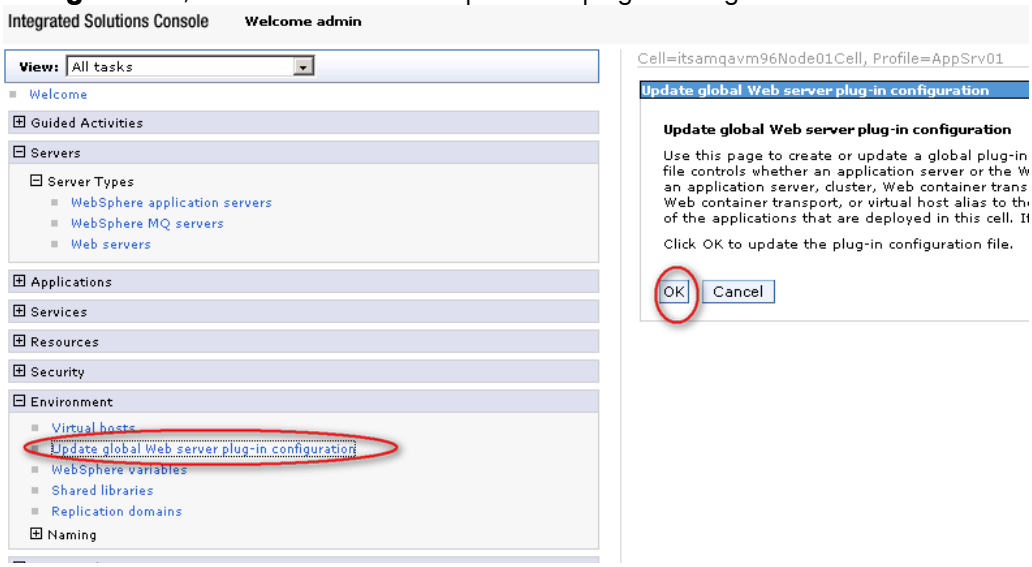
5. Select **IHS** as the Web server template.



6. Enter the properties for the new Web server.



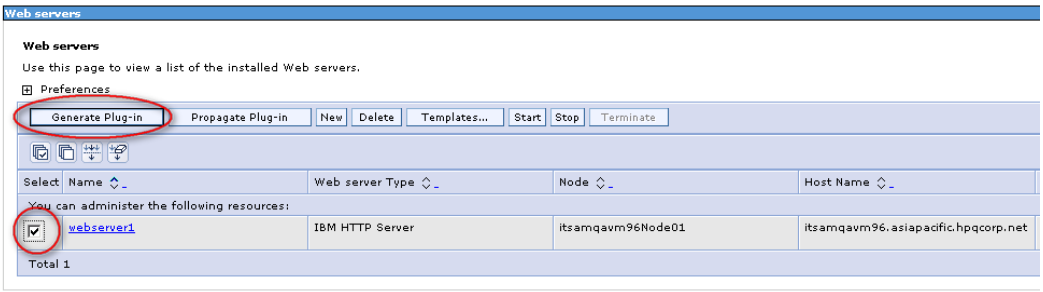
7. Click **Finish** to confirm the new Web server.
8. From the left panel, click **Environment** -> **Update global Web server plug-in configuration**, and then click **OK** to update the plug-in configuration file.



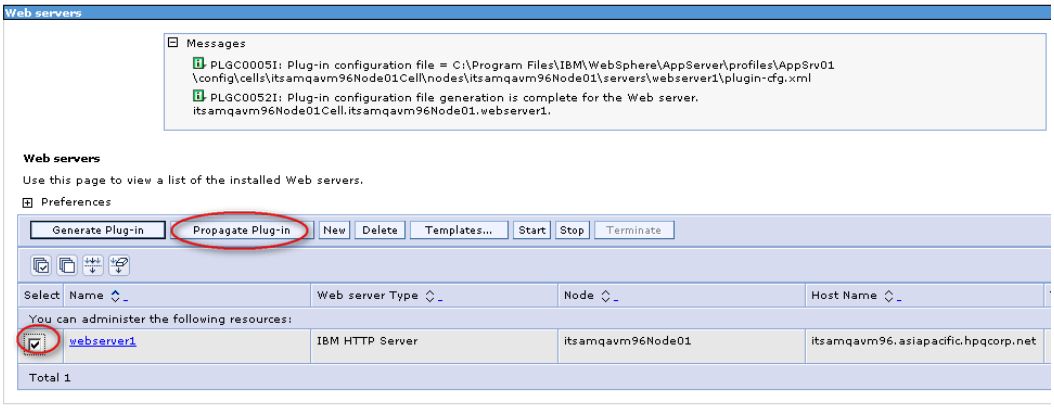
9. Generate and propagate plug-in.
    a. From the left panel, select **Servers** -> **Server Types** -> **Web servers.**

A list of installed Web servers appears.

b. Select the Web server you created, and then click **Generate Plug-in**.



c. Click **Propagate Plug-in**.



IBM HTTP Server configuration changes

Note: The mod_auth_sspi.so module is available only for Windows. If IBM HTTP Server is installed on a UNIX® operating system, it is necessary to create a custom class to perform trusted sign-on.

1. Add the mod_auth_sspi.so module to the `<IHS Install Dir>/modules`.
2. Create a file named as mod_auth_sspi.conf under directory `<IHS Install Dir>/conf`.

```
LoadModule sspi_auth_module modules/mod_auth_sspi.so

<Location "/sm">
   AllowOverride None
   Options None
   Order allow,deny
   Allow from all
   AuthType SSPI
   SSPIAuth On
   SSPIDomain MYDOMAIN
   SSPIAuthoritative On
   SSPIOfferBasic Off
   #SSPIPerRequestAuth On
   require valid-user
</Location>
```

Note: The name within the Location tag needs to be the path the user enters to open the Service Manager Web Client Web site; it is usually `/sm`, since the name is taken from the `sm.war` file.

Make sure that the SSPIPerRequestAuth On is commented out with a crosshatch character (#).

3. Add the following lines to the bottom of the `http.conf` file:

```
include conf/mod_auth_sspi.conf
LoadModule was_ap22_module "<IHS Install
Dir>\Plugins\bin\mod_was_ap22_http.dll"
```

```
WebSpherePluginConfig "<IHS Install
Dir>\Plugins\config\webserver1\plugin-cfg.xml"
```

Browser security settings

Internet Explorer

1. In Internet Explorer, select **Internet Options** on the **Tools** menu.
2. On the **Security** tab, select **Local intranet**, and then click the **Sites** button.
3. Add the following address to the list of trusted web sites: http://<Fully Qualified Domain Name of Service Manager Web Sever>. Make sure that the "Require server verification (https:) for all site in this zone" option is not selected.
4. On the **Security** tab, select **Local intranet**, and then click the **Custom Level...** button.
5. Under Logon in User Authentication section, select **Automatic logon with current username and password**.
6. Click **OK** to save the settings.

Firefox

1. Open Firefox and then type **about:config** in the address bar.
2. In the **Filter** field, type **network.automatic-ntlm-auth.trusted-uris**.
3. Double click the preference name as in Step 2.
4. Enter the URL of Service Manager Web server in the form of: http://<Fully Qualified Domain Name of Service Manager Web Sever>. You can use a comma separated list in this field.

Troubleshooting

Errors can occur both during the setup of the certificates and keystore files as well as when you start Service Manager. This section shows error messages that you may encounter and then describes the cause of each.

General Troubleshooting

If errors occur when you attempt to connect to the Service Manager server, add the debugstartup and debughttp parameters to your `sm.ini` file. Restart the Service Manager server and attempt to log on with the client again. Check your `sm.log` file for details about any errors you may receive. Using Service Manager, it is possible that no errors are written to the `sm.log` file on startup, but that the SSL connection is still not successful. If that is the case, you can read the error message on failed startup by stopping the Service Manager server and then entering the following command from the DOS command prompt:

```
sm –servletcontainer –httpPort:13080 –httpsPort:13081
```

Then wait for the error message to be displayed in the DOS command prompt.

The following errors are commonly seen in the `sm.log` when first setting up the SSL connection, after successful startup of the server:

Error message:

```
keytool error: Failed to establish chain from reply
```

Cause:

This message is issued during the import of a certificate when the `cacerts` file in the `<JAVA_HOME>/lib/security` folder is not the same as the `cacerts` file used to create the certificates. To fix this issue, copy the `cacerts` file used when building the certificates to the `<JAVA_HOME>/lib/security` folder.

Error Message:

Not a trusted client. IP/host name: *<IP Address of Client>/<Hostname of Client>*

Cause:

The hostname of the client that sent the request was different from the DN in the client's certificate. To fix this, recreate the client certificate correctly.

Error Message:

No SSL certificate was presented by the peer!

Cause:

The request was an HTTPS request, but no client certificate is available. Ensure that the web.xml or the windows client preferences point to the correct client certificate.

Error Message:

SSL debug: Could not load trusted client file.

Cause:

Service Manager could not find the trusted client JKS file to which the ssl_trustedClientsJKS parameter points. Verify that the parameter points to the correct location.

Error Message:

Client *<DN in the client's certificate>* is not in the trusted list file.

Cause:

The client's certificate is not in the trusted list file. To fix this issue, follow the steps described in section "**Error! Reference source not found.**".

Error message:

```
SOAP FAULT: SOAP-ENV:Server
"SSL_ERROR_SSL"
Detail: SSL_accept() failed in soap_ssl_accept()
```

Cause:

Under the Windows Preferences in the Service Manager security section, check if the client `keystore` file and client keystore password are entered correctly. If not, correct the values and restart the client. If the issue still occurs, check the path to the `cacerts` file as well. If necessary, correct this information and restart the client. If the error message is still issued, perform the following steps:

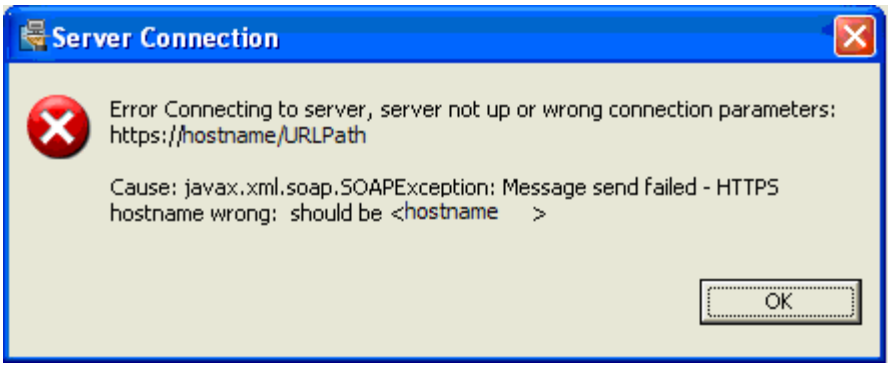
1.Enter the following commands

keytool –list –keystore ./cacerts
keytool –list –keystore *./smclient.keystore*

2.Check if the imports into the private certificates were done correctly. You should see an output such as:

```
Keystore type: jks
Keystore provider: SUN
Your keystore contains 1 entry
test, May 3, 2006, keyEntry,
Certificate fingerprint (MD5):
ED:BF:05:81:0D:BF:CD:53:33:6F:39:07:14:60:87:B3
```

Error message:



Cause:

The Server Host Name in the client connections window or the `web.xml` file is not the same as that used in the server key file. Make the Server Host Name the client tries to connect to the same name as that used when creating the server certificates. Generally, this is the fully qualified server name.

Error Message:

In the `sm.log` file:

```
SOAP FAULT: SOAP-ENV:Client
"Peer is not a trusted client"
Detail: SSL_accept() failed in soap_ssl_accept()
```

Cause:

Either the ssl_trustedClientsJKS parameter is missing in the server's `sm.ini` file, or the contents of the client keystore file (*`smclient.keystore`*) were not imported into the `trusted JKS` correctly, and thus a trusted connection could not be established.

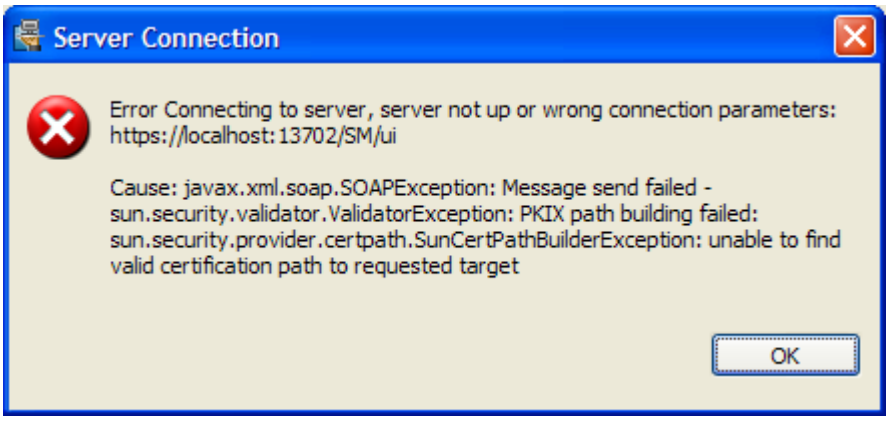
Error message:

Failed to verify user name for trusted sign-on

Cause:

A local user name was entered to connect to Service Manager instead of a domain user name. Only domain user names are accepted to log on to the Service Manager server.

Error message:



With debugging in sm.log:

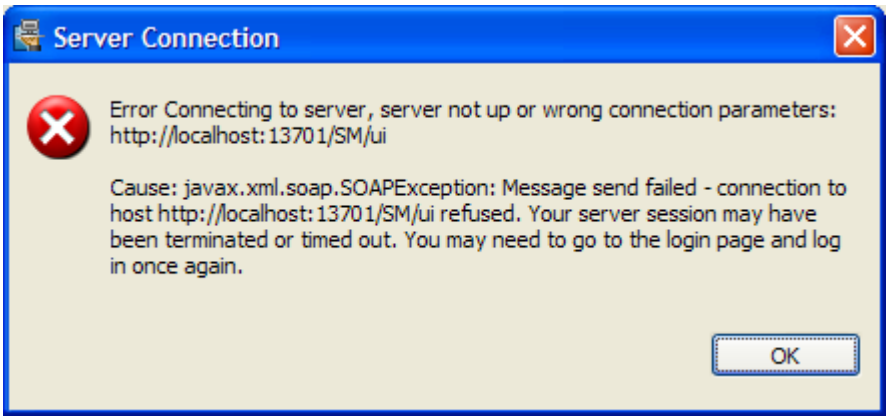
```
  7768(  8052) 06/17/2008 15:06:34 Error initializing endpoint
java.io.IOException: Keystore was tampered with, or password was
incorrect
   at
sun.security.provider.JavaKeyStore.engineLoad(JavaKeyStore.java:768)
```

```
   at java.security.KeyStore.load(KeyStore.java:1150)
   at
org.apache.tomcat.util.net.jsse.JSSESocketFactory.getStore(JSSESocketFact
ory.java:282)
   at
org.apache.tomcat.util.net.jsse.JSSESocketFactory.getKeystore(JSSESocketF
actory.java:222)
   at
org.apache.tomcat.util.net.jsse.JSSE14SocketFactory.getKeyManagers(JSSE14
SocketFactory.java:141)
   at
org.apache.tomcat.util.net.jsse.JSSE14SocketFactory.init(JSSE14SocketFact
ory.java:109)
   at
org.apache.tomcat.util.net.jsse.JSSESocketFactory.createSocket(JSSESocket
Factory.java:88)
   at
org.apache.tomcat.util.net.PoolTcpEndpoint.initEndpoint(PoolTcpEndpoint.j
ava:292)
   at
org.apache.coyote.http11.Http11BaseProtocol.init(Http11BaseProtocol.java:
138)
   at
org.apache.catalina.connector.Connector.initialize(Connector.java:1016)
   at org.apache.catalina.startup.Embedded.start(Embedded.java:826)
   at com.hp.ov.sm.tomcat.EmbeddedTomcat.main(EmbeddedTomcat.java:267)
```

Cause:

The server Keystore password is incorrect. Check the sm.ini to make sure the password for the server keystore is entered exactly as in the batch file, or as entered manually. Restart the server after correcting the issue.

Error message:



Cause:

The machine name entered in the client does not match the machine name of the certificate. Change the machine name to the one used for certificate creation and reconnect.

Error:

Incorrect user is passed through to Service Manager when using IIS for the Web Tier

Cause:

Sometimes, the user name / password from the Service Manager server is passed through instead of the domain user logged in to the client. To troubleshoot this issue, set the Internet Explorer security settings to: prompt for user name and password. This allows the user to see that the user name / password provided by Windows is actually the correct one.

Appendix A – Explanation of the Batch Files

Explanation of the steps required to create the server certificates

The following is a copy of the batch file. The document sections following this batch file describe in detail what each section of the file is doing as well as provide information on how to create the certificates and keystores manually. If you have used the batch file to create the certificates, you do not have to perform the steps in this section.

```
REM #
REM # SM SSL Certificates Creator (server component)
REM #
REM # This batch file facilitates the creation of the SSL certificates
REM # that are needed to setup SSL encryption for Service Manager 7.0x.
REM #
REM # Run this batch file only once to create the certificates for the
REM # Service Manager server.
REM #
REM #-----------------------------------------------------------------
cls

@echo off

REM # OpenSSL settings
REM #
REM # This batch file uses the openssl.conf file as input for the
REM # OpenSSL program. All _default values can be set according to your
REM # organization.
REM #
REM # Only one openssl.conf is needed.
REM #
REM #-----------------------------------------------------------------
set OPENSSL=openssl

REM # Java Settings
REM #
REM # set the JAVA_HOME variable to the installation path of the JRE you
REM # want to use.
REM #
REM #-----------------------------------------------------------------
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
set KEYTOOL=%JAVA_HOME%\bin\keytool

REM # Password settings
REM #
REM # These are the default password settings used by the OpenSSL and
REM # keytool programs. All passwords can be changed, EXCEPT the
REM # CACERT_PASSWD, as this is the default password that the SUN cacert
REM # from the JRE uses!
REM #
REM #-----------------------------------------------------------------
set CAROOT_PASSWD=caroot
set CACERT_PASSWD=changeit
set SERVER_KEYSTORE_PASSWD=serverkeystore
set CLIENT_KEYSTORE_PASSWD=clientkeystore
set TRUSTEDCLIENTS_KEYSTORE_PASSWD=trustedclients
```

```
@del /q key
@del /q certs
@del /q crs

@mkdir key
@mkdir certs
@mkdir crs

copy %JAVA_HOME%\lib\security\cacerts
%JAVA_HOME%\lib\security\cacerts.orig
copy %JAVA_HOME%\lib\security\cacerts certs\cacerts

REM #-------------------------------------------------------------------
REM # Private Key & Root Certificate generation
REM #-------------------------------------------------------------------

REM create the private key for your private CA
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Creating a Self-Signed Certificate (cakey.pem)
@echo.
%OPENSSL% genrsa -des3 -passout pass:%CAROOT_PASSWD% -out key/cakey.pem
2048
@echo.
@echo
_____________________________________________________________________________
@echo.

REM create the root CA cert
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Creating the root ca certificate (mycacert.pem)
@echo.
%OPENSSL% req -new -key key/cakey.pem -x509 -days 1095 -out
certs\mycacert.pem -config ./openssl.conf -passin pass:%CAROOT_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM import the certificate into the System-wide keystore
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Importing the certificate into the System-wide keystore (cacerts)
@echo.
%KEYTOOL% -import -keystore certs/cacerts -trustcacerts -alias
servicemanager -file certs/mycacert.pem -storepass %CACERT_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

copy certs\cacerts %JAVA_HOME%\lib\security
```

```
REM #-----------------------------------------------------------------
REM # Server Key & Certificate generation
REM #-----------------------------------------------------------------

REM generate private server key and keystore
@echo.
@echo
_____________________________________________________________________
@echo.
@echo Creating the Server keystore (server.keystore)
@echo.
%KEYTOOL% -genkey -alias smserver -keystore key/server.keystore -
storepass %SERVER_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________
@echo.

REM generate the server request certificate to be signed using our CA key
& cert
@echo.
@echo
_____________________________________________________________________
@echo.
@echo Generating the Server request certificate (servercert_request.crs)
@echo.
%KEYTOOL% -certreq -alias smserver -keystore key/server.keystore -file
crs/servercert_request.crs -storepass %SERVER_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________
@echo.

REM sign the server request certificate using our CA
@echo.
@echo
_____________________________________________________________________
@echo.
@echo Signing the Server request certificate (smservercert.pem)
@echo.
%OPENSSL% x509 -req -days 1095 -in crs/servercert_request.crs -CA
certs/mycacert.pem -CAkey key/cakey.pem -CAcreateserial -out
certs/smservercert.pem -passin pass:%CAROOT_PASSWD%
@echo.
@echo
_____________________________________________________________________
@echo.

REM import the server certificate into the keystore
@echo.
@echo
_____________________________________________________________________
@echo.
@echo Importing Server certificate into Server keystore
@echo.
%KEYTOOL% -import -trustcacerts -alias smserver -keystore
key/server.keystore -file certs/smservercert.pem -storepass
%SERVER_KEYSTORE_PASSWD%
```

```
@echo.
@echo
_____________________________________________________________________
@echo.
```

Setting the environment variables and passwords

```
set OPENSSL=openssl

REM # Java Settings
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
set KEYTOOL=%JAVA_HOME%\bin\keytool

REM # Password settings
set CAROOT_PASSWD=caroot
set CACERT_PASSWD=changeit
set SERVER_KEYSTORE_PASSWD=serverkeystore
set CLIENT_KEYSTORE_PASSWD=clientkeystore
set TRUSTEDCLIENTS_KEYSTORE_PASSWD=trustedclients
```

First the OpenSSL (if running the commands from the same directory as the batch file, just set OPENSSL to the name of the executable: openssl. If running it from a different directory, include the path to the executable.), Java, and password settings have to be provided. If you choose to create the certificates manually instead of using the batch file, set the JAVA_HOME environment variable and write all passwords down for later reference.

Generating the private key and root certificate

Copying the cacerts file

```
@del /q key
@del /q certs
@del /q crs

@mkdir key
@mkdir certs
@mkdir crs

copy %JAVA_HOME%\lib\security\cacerts
%JAVA_HOME%\lib\security\cacerts.orig
copy %JAVA_HOME%\lib\security\cacerts certs\cacerts
```

It is very important that the cacerts file in the JAVA_HOME\lib\security folder is updated to include the root CA information. Because of this requirement, copy the original cacerts file to a new name and then copy the original cacerts file to the certificates working directory that was created earlier during this step.

Generate an RSA® private key

```
REM #-----------------------------------------------------------------
REM # Private Key & Root Certificate generation
REM #-----------------------------------------------------------------

REM create the private key for your private CA
@echo Creating a Self-Signed Certificate (cakey.pem)
@echo.
%OPENSSL% genrsa -des3 -passout pass:%CAROOT_PASSWD% -out key/cakey.pem
2048
@echo.
```

The following steps are required to create your Service Manager private key:

3. Open a command window from the Service Manager RUN directory and execute the following command to create the private key (cakey.pem):

openssl genrsa -des3 –passout pass:<*password*> -out key/cakey.pem 2048

4. Enter a password for your CA (Certificate Authority) and verify it when prompted.

Important: Write down this password, because you will need it later to sign certificates.

Create a self-signed root certificate for the Certificate Authority (CA)

A *root certificate* is a trusted copy of the CA's certificate that is used to sign other certificates. Its private key is highly sensitive and should never be compromised by removing the password that protects it. Before SSL communications can succeed, connections need to trust the self-signed root certificate by downloading and registering it.

```
REM create the root CA cert
@echo Creating the root ca certificate (mycacert.pem)
@echo.
%OPENSSL% req -new -key key/cakey.pem -x509 -days 1095 -out
certs\mycacert.pem -config ./openssl.conf -passin pass:%CAROOT_PASSWD%
@echo.
```

When prompted for locale information and common name, enter the information as prompted. The common name is the fully qualified machine name for which the certificate is created.

If creating the certificates manually, follow these steps.

1. From the Service Manager `RUN` directory or the OpenSSL directory command window, execute the following to create the self-signed root certificate (`mycacert.pem`):

openssl req -new -key cakey.pem -x509 -days 1095 -out mycacert.pem -config ./openssl.conf –passin pass:<*password*>

Enter the CA password that you created in the section **Generate an RSA® private key** when prompted to do so.

2. When prompted for certificate details such as country and state, you can enter either the appropriate data, choose the system defaults, or leave the items blank. When prompted for the Common Name, ensure to enter the fully qualified name for your machine.

Importing the self-signed root certificate (mycacert.pem) into the system-wide keystore (cacerts)

In the JRE, SSL certificates are stored in two files: a *keystore* file and a *truststore* file. The keystore file holds key entries, each of which is an entity's identity and private key that is used to identify oneself to a server as a trusted client. The truststore file holds trusted certificate entries, each of which contains the identity and public key of an entity (usually a CA), which are used to identify trusted servers.

For normal SSL communications, where the only requirement is that the client trusts the server, the self-signed root certificate (`mycacert.pem`) must be imported into the system-wide keystore (`cacerts`).

```
@echo Importing the certificate into the System-wide keystore (cacerts)
@echo.
%KEYTOOL% -import -keystore certs/cacerts -trustcacerts -alias
servicemanager -file certs/mycacert.pem -storepass %CACERT_PASSWD%
@echo.
```

Respond to the prompt “Trust this certificate” by typing **y**.

Follow these steps to import the signed CA certificate into the keystore file:

1. Open a command window from the <JDK|JRE>/bin directory (see KEYTOOL environment variable defined above) and execute the following:

   keytool -import -keystore cacerts -trustcacerts -alias servicemanager -file mycacert.pem -storepass <*password*>

2. When a dialog box prompts "Trust this certificate?" type **y**.
3. Verify that a message is received stating that the certificate was added to the keystore file.
4. You can run the following command to ensure that the import was successful:

   keytool -list -keystore ./cacerts

   and enter the **<*password*>** created above when prompted.

Note: Although it is possible to create your own cacerts file, HP recommends copying the JAVA_HOME/lib/security/cacerts file, import the new certificate and then copy the modified cacerts file back to JAVA_HOME/lib/security.

Move the trusted certificates file and the Certificate Authority (CA)

The keytool utility allows an administrator to manage keystore files and certificates from trusted entities. The utility can be found in the <JRE|JDK>/bin directory.

The system-wide keystore that holds CA certificates, the cacerts file, and the self-signed root certificate, mycacert.pem created in the section *Error! Reference source not found.*, must be moved so that the keytool utility in the Java™ Runtime Environment (JRE™) can access them.

```
@echo.

copy certs\cacerts %JAVA_HOME%\lib\security
```

Copy the cacerts file from the directory where it was updated in the previous step to the <JRE|JDK>/lib/security directory.

Creating the server keystore

The following steps apply to setting up single sign-on and required SSL communication in Service Manager.

Note: It is very important to always enter the fully qualified machine name when prompted for the name (*machine.domain.com*).

Create a private key and keystore for the Service Manager server

```
@echo Creating the Server keystore (server.keystore)
@echo.
%KEYTOOL% -genkey -alias smserver -keystore key/server.keystore -
storepass %SERVER_KEYSTORE_PASSWD%
@echo.
```

Enter all other input when prompted.

In this step, you create the server.keystore file. To do so, enter the following command and fill in the requested information as shown below:

keytool -genkey -alias smserver -keystore server.keystore –storepass <*password*>

```
What is your first and last name?
  [Unknown]:  machinename.domain.com
What is the name of your organizational unit?
  [Unknown]:  TSG
```

```
What is the name of your organization?
  [Unknown]:  HP
What is the name of your City or Locality?
  [Unknown]:  San Diego
What is the name of your State or Province?
  [Unknown]:  California
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=server.domain.com, OU=TSG, O=HP, L=San Diego, ST=California, C=US
correct?
  [no]:  yes

Enter key password for <smserver>
        (RETURN if same as keystore password):
```

Create a certificate request for the Service Manager server

```
@echo Generating the Server request certificate (servercert_request.crs)
@echo.
%KEYTOOL% -certreq -alias smserver -keystore key/server.keystore -file
crs/servercert_request.crs -storepass %SERVER_KEYSTORE_PASSWD%
@echo.
```

You can enter the password when prompted or via the **–storepass <*password*>** parameter.

From the keystore file created in the previous section, you create a certificate request for the Service Manager server, using the following command:

keytool -certreq -alias smserver -keystore server.keystore -file servercert_req.crs –storepass <*password*>

Sign the Service Manager server's certificate request with your private certificate authority

```
@echo Signing the Server request certificate (smservercert.pem)
@echo.
%OPENSSL% x509 -req -days 1095 -in crs/servercert_request.crs -CA
certs/mycacert.pem -CAkey key/cakey.pem -CAcreateserial -out
certs/smservercert.pem -passin pass:%CAROOT_PASSWD%
@echo.
```

The following command will sign the server's certificate request (servercert_req.crs) with the private certificate authority (`mycacert.pem`) and create the `smservercert.pem` file as a result:

openssl x509 -req -days 1095 -in servercert_req.crs -CA mycacert.pem -CAkey cakey.pem -CAcreateserial -out smservercert.pem -passin pass:<*CAROOT_PASSWD*>
You can either enter the pass phrase when prompted or use the **-passin pass:<*password*>** parameter.

The following message is displayed:

```
Loading 'screen' into random state - done
Signature ok
subject=/C=US/ST=California/L=San Diego/O=HP/OU=TSG/CN=server.domain.com
Getting CA Private Key
```

Import the signed certificate into the keystore

After the certificate request is signed, you import it into the server's keystore with the following command:

```
@echo Importing Server certificate into Server keystore
@echo.
```

```
%KEYTOOL% -import -trustcacerts -alias smserver -keystore
key/server.keystore -file certs/smservercert.pem -storepass
%SERVER_KEYSTORE_PASSWD%
@echo.
```

keytool -import -trustcacerts -alias smserver -keystore server.keystore -file smservercert.pem -storepass <*SERVER_KEYSTORE_PASSWORD*>

A message such as the following is displayed:

```
Certificate reply was installed in keystore
```

Explanation of the steps needed to create client certificates

The following section describes the batch file and each of its sections in more detail.

```
REM #
REM # SC-SM SSL Certificates Creator (client component)
REM #
REM # This batch file facilitates the creation of the SSL certificates
that are needed to setup SSL encryption for Service Manager 7.0x.
REM #
REM # Run this batch file with the fully-qualified domain name of the
client machine as the first argument (%1), from the command line :
REM #
REM # \prompt>tso_cln_svlt <fully-qualified domain name>
REM #
REM # Rerun this batch file for each client machine to create a unique
REM # set of certificates for the Service Manager Eclipse or Web client.
REM #
REM #-------------------------------------------------------------------
cls

@echo off


REM # OpenSSL settings
REM #
REM # This batch file uses the openssl.conf file as input for the the
OpenSSL program. All _default values can be set according to your
REM # organization.
REM #-------------------------------------------------------------------
set OPENSSL=openssl


REM # Java Settings
REM #
REM # set the JAVA_HOME variable to the installation path of the JRE you
REM # want to use.
REM #
REM #-------------------------------------------------------------------
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
set KEYTOOL=%JAVA_HOME%\bin\keytool


REM # Password settings
REM #
REM # These are the default password settings used by the openssl and
REM # keytool programs. All passwords can be changed, EXCEPT the
REM # CACERT_PASSWD, as this is the default password that the SUN
```

```
REM # cacert from the JRE uses..!!
REM #
REM #-------------------------------------------------------------------
set CAROOT_PASSWD=caroot
set CACERT_PASSWD=changeit
set SERVER_KEYSTORE_PASSWD=serverkeystore
set CLIENT_KEYSTORE_PASSWD=clientkeystore
set TRUSTEDCLIENTS_KEYSTORE_PASSWD=trustedclients


#########################################################################
# Only do this step if run from a different machine than the one that
# created the server certs
#########################################################################
# copy %JAVA_HOME%\lib\security\cacerts
%JAVA_HOME%\lib\security\cacerts.origcopy
copy %SSL_CERT_HOME%\certs\cacerts %JAVA_HOME%\lib\security

echo Client Key and Certificate creation

REM #-------------------------------------------------------------------
REM # Client Key & Certficate generation
REM #-------------------------------------------------------------------

REM generate private client key and keystore
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Creating the Client keystore (%1.keystore)
@echo.
%KEYTOOL% -genkey -alias %1 -keystore key/%1.keystore -storepass
%CLIENT_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM generate the Client request certificate to be signed using our CA key
REM & cert
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Generating the Client request certificate (clientcert_request.crs)
@echo.
%KEYTOOL% -certreq -alias %1 -keystore key/%1.keystore -file
crs/clientcert_request.crs -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM sign the Client certificate using our CA
@echo.
@echo ------------------------------------
@echo.
@echo Signing the Client request certificate (scclientcert.pem)
@echo.
```

```
%OPENSSL% x509 -req -days 1095 -in crs/clientcert_request.crs -CA
certs/mycacert.pem -CAkey key/cakey.pem -CAcreateserial -out
certs/scclientcert.pem -passin pass:%CAROOT_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM import the client certificate into the keystore
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Importing Client certificate into Client keystore
@echo.
%KEYTOOL% -import -trustcacerts -alias %1 -keystore key/%1.keystore -file
certs/scclientcert.pem -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM #-----------------------------------------------------------------
REM # Adding the client Certificate to Trusted Keystore
REM #-----------------------------------------------------------------

REM export client public key/certificate
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Exporting Client public certificate from Client keystore
(clientpubkey.cert)
@echo.
%KEYTOOL% -export -alias %1 -keystore key/%1.keystore -file
certs/clientpubkey.cert -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.

REM import public key/certificate into the keystore
@echo.
@echo
_____________________________________________________________________________
@echo.
@echo Importing Client public certificate into Trustedclients keystore
(trustedclients.keystore)
@echo.
%KEYTOOL% -import -alias %1 -file certs/clientpubkey.cert -keystore
certs/trustedclients.keystore -storepass %TRUSTEDCLIENTS_KEYSTORE_PASSWD%
@echo.
@echo
_____________________________________________________________________________
@echo.
```

Setting the environment variables and passwords

```
set OPENSSL=openssl
```

```
REM # Java Settings
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_12"
set KEYTOOL=%JAVA_HOME%\bin\keytool

REM # Password settings
set CAROOT_PASSWD=caroot
set CACERT_PASSWD=changeit
set SERVER_KEYSTORE_PASSWD=serverkeystore
set CLIENT_KEYSTORE_PASSWD=clientkeystore
set TRUSTEDCLIENTS_KEYSTORE_PASSWD=trustedclients
```

First the OpenSSL (if running the commands from the same directory as the batch file, just set OPENSSL to the name of the executable: openssl. If running it from a different directory, include the path to the executable.), Java, and password settings have to be provided. If you choose to create the certificates manually instead of using the batch file, set the JAVA_HOME environment variable and write all passwords down for later reference.

Copying the cacerts file

```
@del /q key
@del /q certs
@del /q crs

@mkdir key
@mkdir certs
@mkdir crs

########################################################################
# Only do this step if run from a different machine than the one that
# created the server certs
########################################################################
# copy %JAVA_HOME%\lib\security\cacerts
%JAVA_HOME%\lib\security\cacerts.origcopy
%JAVA_HOME%\lib\security\cacerts certs\cacerts
copy %SSL_CERT_HOME%\certs\cacerts %JAVA_HOME%\lib\security
```

It is very important that the cacerts file in the JAVA_HOME\lib\security folder is updated to include the root CA information. To ensure that the original cacerts is not accidentally overwritten, copy it to a different name before proceeding.

Create the client's keystore

Certificate requests for both Service Manager Windows and Web clients can be generated using the same client `keystore` file. The steps in this section apply to both client types. The entries in *italics* will have to be different for each client.

```
@echo Creating the Client keystore (%1.keystore)
@echo.
%KEYTOOL% -genkey -alias %1 -keystore key/%1.keystore -storepass
%CLIENT_KEYSTORE_PASSWD%
@echo.
```

1. From the `<JDK|JRE>/bin` directory command window, execute the following command:
   keytool -genkey -alias ***sm client*** **-keystore** ***sm client.keystore*** **-storepass <CLIENT_KEYSTORE_PASSWD>**
2. Select a password value that is easily remembered. The password passed in the storepass parameter will be the password for your client's keystore.
3. When prompted for first and last name, enter the fully qualified domain name of the Service Manager client or the Service Manager Web server. For example, enter:

smclient01.HP.com or **webtier02.HP.com**.

- When prompted for certificate details such as country, state, and locality, you can enter the appropriate data, choose the system defaults, or leave the items blank.

4. When a dialog box prompts "Is this correct?" type **y**.

Create the client's certificate request

```
@echo.
@echo Generating the Client request certificate (clientcert_request.crs)
@echo.
%KEYTOOL% -certreq -alias %1 -keystore key/%1.keystore -file
crs/clientcert_request.crs -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
```

1. To generate the client certificate request, execute the following command (where entries in *italics* will have to be different for each client):
   keytool -certreq -alias ***smclient*** **-keystore** ***smclient.keystore*** **-file** ***smclientcert_request.*****crs -storepass <CLIENT_KEYSTORE_PASSWD>**

2. The password needs to be the same as the one used for creating the client's keystore in step 2 in the section above..
3. To verify that this worked, you can enter the following command:

   keytool -list -keystore ***./smclient.keystore***

4. and enter the password when prompted.

Sign the client certificate request using the root certificate and private key

```
@echo Signing the Client request certificate (smclientcert.pem)
@echo.
%OPENSSL% x509 -req -days 1095 -in crs/clientcert_request.crs -CA
certs/mycacert.pem -CAkey key/cakey.pem -CAcreateserial -out
certs/smclientcert.pem -passin pass:%CAROOT_PASSWD%
@echo.
```

1. Copy the client certificate request file, `smclientcert_request.crs`, to the Service Manager `RUN` directory.
2. Execute the following command from the Service Manager `RUN` directory command window:
   openssl x509 -req -days 1095 -in ***smclientcert_request*****.crs -CA mycacert.pem -CAkey cakey.pem -CAcreateserial -out** ***smclientcert*****.pem-passin pass:<CAROOT_PASSWD>**

3. You can check if this was successful by entering the following command:

   openssl x509 -in ***smclientcert*****.pem -text –noout**

Import the client certificate into the clients keystore

```
@echo Importing Client certificate into Client keystore
@echo.
%KEYTOOL% -import -trustcacerts -alias %1 -keystore key/%1.keystore -file
certs/smclientcert.pem -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
```

1. Enter the following command to import the client certificate into the client keystore:
   keytool -import -trustcacerts -alias smclient -keystore ./smclient.keystore -file smclientcert.pem -storepass <CLIENT_KEYSTORE_PASSWD>

A message such as the following is displayed:

```
Certificate reply was installed in keystore
```

Creating the trusted certificates file

In Service Manager you first export the public key/certificate, and then import the public key/certificate into the `truststore`.

Export the public key / certificate

```
@echo Exporting Client public certificate from Client keystore
(clientpubkey.cert)
@echo.
%KEYTOOL% -export -alias %1 -keystore key/%1.keystore -file
certs/clientpubkey.cert -storepass %CLIENT_KEYSTORE_PASSWD%
@echo.
```

Enter the following command in the directory where keytool is installed, usually `jre/bin`:

keytool -export -alias smclient -keystore smclient.keystore -file clientpubkey.cert -storepass <CLIENT_KEYSTORE_PASSWD>

A message such as the following is displayed:

```
Certificate stored in file <clientpubkey.crt>
```

Import this certificate into the truststore

```
@echo Importing Client public certificate into Trustedclients keystore
(trustedclients.keystore)
@echo.
%KEYTOOL% -import -alias %1 -file certs/clientpubkey.cert -keystore
certs/trustedclients.keystore -storepass %TRUSTEDCLIENTS_KEYSTORE_PASSWD%
@echo.
```

Enter the following command:

keytool -import -alias <host name> -file clientpubkey.crt -keystore trustedclients.keystore -storepass <TRUSTEDCLIENTS_KEYSTORE_PASSWD>

A message such as the following is displayed:

```
Owner: CN=server.domain.com, OU=Client, O=HP, L=SD, ST=CA, C=US
Issuer: EMAILADDRESS=falcon@hp.com, CN=server.domain.com, OU=HP CA,
O=Private HP CA, L=San Diego, ST=CA, C=US
Serial number: 3
Valid from: Thu Mar 30 16:38:57 PST 2006 until: Sun Mar 29 16:38:57 PST
2009
Certificate fingerprints:
         MD5:  8B:F4:57:C4:BD:C6:92:8A:CB:3B:F2:4E:44:3A:75:EE
         SHA1:
46:3C:6E:A8:B3:1D:0B:D3:33:C2:A0:B8:C0:98:90:28:38:C7:3E:FD
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Appendix B - Setting up Single Sign-on with third party authentication on the Web Tier

Configuring the Web Client for third-party authentication

1.To activate single sign-on in Service Manager's Web Tier navigate to the *<path to Web application>*/**WEB-INF/classes/** directory.

In the application-context.xml file modify the following line. Change

```
/**=httpSessionContextIntegrationFilter,anonymousProcessingFilter
```

To

```
/**=httpSessionContextIntegrationFilter,preAuthenticationFilter,
anonymousProcessingFilter
```

as described in the comment in the `application-context.xml` file.

2.Save your changes and restart the Web server on the Service Manager Web Tier.

3.When using IIS you need to configure an ISAPI connector for your Web application server, and you need to modify the virtual directory to use Integrated Windows Authentication.

See the HP Customer Support Web site for Knowledge Base articles about configuring Integrated Windows Authentication on common Web application servers or refer to the section *Configuration of the Web server and Web application server* in this document.

Defining a JavaBean® to handle authentication

All modifications for the single or trusted sign-on settings in the Service Manager Web Tier are done in the *<path to web application>*`/WEB-INF/classes` directory in the

`application-context.xml` file.

For Windows-based authentication, out-of-box Service Manager ships with a JavaBean called preAuthenticationFilter, which is defined as:

```
<bean id="preAuthenticationFilter"
 class="com.hp.ov.cwc.security.acegi.PreAuthenticationFilter">
   <property name="authenticationManager">
      <ref bean="authenticationManager"/>
   </property>
   <property name="defaultRole">
      <value>ROLE_PRE</value>
   </property>
</bean>
```

To use another authentication source that passes the username in the HTTP header, you add or replace the preAuthenticationFilter block with code such as the following.

The following example shows how to add a new filter. To replace the filter, set the bean ID to **preAuthenticationFilter** (rather than **SiteminderPreAuthenticationFilter** as shown below).

```
<bean id="SingleSignOnpreAuthenticationFilter"
  class="com.hp.ov.cwc.security.acegi.SiteminderPreAuthenticationFilter">
   <property name="authenticationManager">
      <ref bean="authenticationManager"/>
   </property>
   <property name="defaultRole">
      <value>ROLE_PRE</value>
   </property>
   <property name="keepDomain">
      <value>true</value>
```

```
   </property>
   <property name="siteminderUsernameHeaderKey">
   <value>USERNAME</value>
   </property>
   <property name="siteminderPasswordHeaderKey">
   <value>PASSWORD</value>
   </property>
</bean>
```

In this definition the siteminderUsernameHeaderKey property passes the user name value from the single sign-on tool to Service Manager. To successfully pass the user name value, set the <value> element to the name of the property in the HTTP header that stores the user name in the third-party authentication tool's configuration.

If the HTTP header was…

```
POST /sc61server/ws HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "Retrieve"
Content-Length: 1474
Expect: 100-continue
[…]
REMOTE_USER: falcon
Cookie: $Version=1; SessionId=127.0.0.1:4819
```

Then replace the *<USERNAME>* with **REMOTE_USER**.

To activate the sign-on filters, modify the following line in the `application-context.xml` file.

```
/**=httpSessionContextIntegrationFilter,
SingleSignOnpreAuthenticationFilter,preAuthenticationFilter,
anonymousProcessingFilter
```

The SingleSignOnpreAuthenticationFilter has to match the bean ID from above. If you want multiple sign-on filters, add them in order to the filter chain. If you do this, the first to succeed will provide the authenticated username.

Note: The httpSessionContextIntegrationFilter always needs to be in first position within the chain.

Integrating Custom Java Classes into the Java Bean

In Service Manager, it is possible to develop and use a custom Java class for third party authentication as well. You can either write a new custom Java class or modify the existing Java class to fit the new requirements for Service Manager. The requirements for the new Java class are described on the example of integrating an old Java class into the Service Manager system. The modified Java class then needs to be put into the `web-inf/classes` directory, or as a jar file into `web-inf/lib`.

The following changes are necessary to go from a ServiceCenter 6.1 Java class to the Service Manager Java class:

- Instead of implementing the com.HP.shared.cwc.security.AuthenticationHandler interface, change it to extend the com.HP.shared.security.acegi.PreAuthenticationFilter class.
- Change the `authenticate(HttpServletRequest, User)` method to `getAuthenticatedUsername(HttpServletRequest)`
- Instead of calling `user.setUserName()` and `user.setAuthenticated()`, the `getAuthenticatedUsername` method returns the user name as a String if the user is authenticated, or null if not. This user name must match the operator record in Service Manager, so include the domain in the String if the operator names in Service Manager include the domain.

In the <path to web application>/WEB-INF/classes directory in the application-context.xml file, your bean will then look as follows:

```
<bean id="preAuthenticationFilter"
   class="com.hp.ov.cwc.security.acegi.<custom java class>">
   <property name="authenticationManager">
      <ref bean="authenticationManager"/>
   </property>
   <property name="defaultRole">
      <value>ROLE_PRE</value>
   </property>
</bean>
```

In short examples, here are the changes you need to implement going from ServiceCenter version 6.1 to Service Manager. Instead of:

```
public class <custom class> implements AuthenthicationHandler
```

You now do:

```
public class <custom class> extends PreAuthenticationFilter
```

…where "extends" means that the custom public class following will replace a method in the PreAuthenticationFilter java class.

Within the public class, you define which part of the PreAuthenticationFilter you want to replace and how to replace it by defining the method:

```
protected String getAuthenticatedUsername(HttpServletRequest request)
```

to your specification. If this is a new implementation, you will do your custom programming here.

The main modification you have to do to the old Java code is to remove all mention of the user object, which is not passed into the class in Service Manager any longer. Instead we return the user name, if authentication succeeded, or return null if it failed. This username has to be returned after a successful run of the class.

Example: Creating a custom Java Class for Single-Sign-On using LDAP

Note: This is an example on how to create a custom Java class for Single Sign-On using LDAP authentication. Actual implementation will vary from system to system, depending on the LDAP server, setup and configuration.

The following classes are required to compile this example:

class jar file	comments
jasypt-1.4.1.jar*	Part of the jasypt package. Copy the jar to <web-inf>\lib
commons-lang-2.2.jar	Part of SM7 web client
commons-codec-1.3.jar	Part of SM7 web client
com.ibm.icu_3.4.5.20061213.jar	From the full client plugins folder
security-4.0.jar	Part of SM7 web client
spring-2.0.5.jar	Part of SM7 web client
servlet-api.jar*	Copied from Tomcat, only required for compilation. Same classes are provided by WebSphere at runtime.

```
//Provided by B.Hartley
package com.hp.ov.cwc.security.acegi;
```

```
// servlet-api.jar
import javax.servlet.http.HttpServletRequest;

import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.Attributes;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.ldap.LdapContext;


// defined in jasypt-1.4.1.jar
// should copy to app-lib folder
import org.jasypt.util.text.*;

// require spring-2.0.5.jar in classpath

public class LDAPPreAuthenticationFilter  extends PreAuthenticationFilter
{
   public LDAPPreAuthenticationFilter ()
   {
      keepDomain = false;
      credentialProvider = null;
      //implements upn lookup if set to true can bbe overridden in
      application-context.xml
      useUPN = true;
      ldapBindDn = "";
      //for example CN=adm2hartley,OU=Admins,OU=Users and Groups,OU=GC
      CTR,OU=CTR,OU=Organizations,DC=company,DC=com";
      ldapBindPassword = "";
      ldapBaseDn= ""; //OU=Admins,OU=Users and Groups,OU=GC
      CTR,OU=CTR,OU=Organizations,DC=company,DC=com";
      ldapServerURL= ""; //ldap://ldap.company.com:389";
      //to implement - the filter would be iserted in an AND clause
      //ldapExcludeFilter = (!(dn=*Contractors*))

   }

   protected String getAuthenticatedUsername(HttpServletRequest request)
   {
      String username = null;
      if(credentialProvider == null ||
      credentialProvider.getUserName(request) != null &&
      credentialProvider.getUserName(request).equals(""))
      {
         username = request.getRemoteUser();
         System.out.println("Default remote user : " + username);
         if(username != null)
         if(username.length() == 0)
            username = null;
         else
         if(!keepDomain)
         {
            int i = username.indexOf('\\');
            username = username.substring(i + 1);
         }
      }
```

```
      else
      {
         username = credentialProvider.getUserName(request);
      }
      if(useUPN)
      {
         System.out.println("Using UPN. SamAcctName : " + username);
         String userUPNname = lookupUPN(username);
         System.out.println("Found this UPN : " + userUPNname );
         return userUPNname ;
      }
      else
      {
         System.out.println("Not using UPN. Username : " + username);
         return username ;
      }
   }

   public boolean useUPN(){return useUPN;}
   public void setuseUPN(boolean useUPN){this.useUPN = useUPN;}

   public void setCredentialProvider(CredentialProvider
   credentialProvider)
   {
      this.credentialProvider = credentialProvider;
   }

   public String getldapBindDn(){return ldapBindDn;}
   public void setldapBindDn(String ldapBindDn){this.ldapBindDn =
   ldapBindDn;}
   public String getldapBindPassword(){return ldapBindPassword;}
   public void setldapBindPassword(String
   ldapBindPassword){this.ldapBindPassword = ldapBindPassword;}
   public String getldapBaseDn(){return ldapBaseDn;}
   public void setldapBaseDn(String ldapBaseDn){this.ldapBaseDn =
   ldapBaseDn;}
   public String getldapServerURL(){return ldapServerURL;}
   public void setldapServerURL(String ldapServerURL)
   {
      this.ldapServerURL = ldapServerURL;
   }
   protected String lookupUPN(String samUserName )
   String myUPNuser="";
   {
      try
      {
         Hashtable env = new Hashtable();
         String adminName = ldapBindDn;
         String adminPassword = ldapBindPassword;
         String ldapURL = ldapServerURL;
         env.put(Context.INITIAL_CONTEXT_FACTORY,"com.sun.jndi.ldap.
         LdapCtxFactory");
         //set security credentials, note using simple cleartext
         authentication
         env.put(Context.SECURITY_AUTHENTICATION,"simple");
         env.put(Context.SECURITY_PRINCIPAL,adminName);

         //decrypt the password if it starts with *** else it is clear
         text, use as is.
         if (adminPassword.indexOf("***")==0 )
```

```
         {
            adminPassword = adminPassword.substring(3);
            BasicTextEncryptor textEncryptor = new BasicTextEncryptor();
            // this password the same as that used to encrypt the
            adminpassword with FilterEncrypt
            textEncryptor.setPassword("PssWrd");

            adminPassword = textEncryptor.decrypt(adminPassword);
         }
         env.put(Context.SECURITY_CREDENTIALS,adminPassword);
         //connect to my domain controller
         env.put(Context.PROVIDER_URL,ldapURL);

         //Create the initial directory context
         LdapContext ctx = new InitialLdapContext(env,null);
         //Create the search controls
         SearchControls userSearchCtls = new SearchControls();
         //Specify the search scope
         userSearchCtls.setSearchScope(SearchControls.SUBTREE_SCOPE);

         //specify the LDAP search filter to find the user in question
         String userSearchFilter = "(&(objectClass=user)(sAMAccountName="
         + samUserName +"))";

         //Specify the Base for the search
         String userSearchBase = ldapBaseDn;

         //Specify the attributes to return
         String userReturnedAtts[]={"userPrincipalName"};
         userSearchCtls.setReturningAttributes(userReturnedAtts);

         //Search for objects using the filter
         NamingEnumeration userAnswer = ctx.search(userSearchBase,
         userSearchFilter, userSearchCtls);

         //Loop through the search results
         while (userAnswer.hasMoreElements())
         {
            SearchResult sr = (SearchResult)userAnswer.next();
            Attributes attrs = sr.getAttributes();

            if (attrs != null)
            {
               myUPNuser =
               attrs.get("userPrincipalName").get().toString();
            }
         }

      }

      catch (Exception e)
      {
         System.err.println("Problem searching directory: " + e);
         System.out.println("Problem searching directory: " + e);
      }
      return myUPNuser;
   }
// added for UPNLookup
boolean useUPN;
String ldapBindDn;
```

```
String ldapBindPassword;
String ldapBaseDn;
String ldapServerURL;

}
```

This class contains a reference to FilterEncrypt, which is shown below:

```
import org.jasypt.util.text.*;
public class FilterEncrypt
{
   public static void main(String arg[])
   {
      try
      {
         String GivenPwd = arg[0];
         BasicTextEncryptor textEncryptor = new BasicTextEncryptor();
         textEncryptor.setPassword("PssWrd");
         String myEncryptedText = "***" + textEncryptor.encrypt(GivenPwd
         );
         System.out.println("here is the encrypted : " +
         myEncryptedText);
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

For more information

Please visit the HP OpenView support web site at:
http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track progress on support cases
- Submit enhancement requests online
- Download software patches
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Note: Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract.

To find more information about support access levels, go to the following URL:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to the following URL:

http://www.managementsoftware.hp.com/passport-registration.html

01/2012