Oracle® Enterprise Data Quality

SSL Configuration

Release 11g R1 (11.1.1.7)

E40048-02

October 2013

This document provides instructions for setting up Secure Sockets Layer (SSL) on an Oracle Enterprise Data Quality (EDQ) running on Tomcat application server. For other application servers (such as Oracle WebLogic Server or IBM Websphere), consult the standard server documentation.

> **Note:** EDQ user applications (such as Director) always encrypt user passwords, so SSL is not required.

This document is intended for system administrators responsible for setting up SSL on an EDQ installation running on Tomcat application server.

1 Configuring SSL During Installation

When installing Tomcat on Windows or any other platform, the HTTPS connector must be configured using the following procedure:

1. Locate the `server.xml` file for the Tomcat installation. Typically it contains the following:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
-->
```

2. Enable the Connector element by removing the Comment characters around it.
3. Set the port value for HTTPS. The default is 8443, so if a different value is used also change the `redirectPort` value in the HTTP connector to match.
4. Generate the server certificate.

> **Note:** The certificate is supplied in a Java keystore, either in the default JKS format or as a PKCS#12 file. The latter may be preferred in certain instances, as there are many tools available for working with PKCS#12 files.

5. Update the connector element as follows:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
sslProtocol="TLS"
keystoreFile="pathtokeystorefile"
keystorePass="keystorepassword"
keystoreType="keystoretype"
/>
```

6. Set the `keystoreType` value to `JKS` or `PKCS12` as required. If the key store contains multiple certificates, use the `keyAlias` attribute to set the alias.
7. Some Tomcat distributions include the Apache Portable Runtime (APR) native library. If this is the case, the certificate must be configured using `mod_ssl` style attributes. For example:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
SSLCertificateFile="pathtocrtfile"
SSLCertificateKeyFile="pathtokeyfile" />
```

For additional Tomcat information, see *Apache Tomcat Configuration Reference* at

http://tomcat.apache.org/tomcat-7.0-doc/config/http.html

For additional `mod_ssl` information, see *Apache Module mod_ssl* at

http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

2 Configuring SSL Client Authentication

EDQ can support authentication using SSL client certificates.

There are three stages to configuring SSL client certificates:

1. Configure the server to request client certificates.
2. Assign a personal certificate and associated private key to each user.
3. Associate each certificate with an internal EDQ user or an entry in an external LDAP server.

2.1 Configuring Tomcat to Support Client Certificates

1. Locate the HTTPS connector and add the following settings:

```
clientauth="true"
truststoreFile="pathtotruststore"
truststorePass="truststorepassword"
truststoreType="truststoretype"
```

   a. Set the `clientauth` attribute to `true` (valid client certificate required for a connection to succeed) or want (use a certificate if available, but still connect if no certificate is available).

   b. Add the location of the trust file containing the certificate issuers for trusted client certificates.

   c. Set `truststoreType` to `JKS` or `PKCS12`.

2. If the Tomcat installation includes Apache Portable Runtime (APR), then the equivalent `mod_ssl` settings are used:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
SSLCertificateFile="pathtocrtfile"
SSLCertificateKeyFile="pathtokeyfile"
SSLCACertificateFile="pathtocabundlefile"
SSLVerifyClient="require" />
```

3. Locate the EDQ `login.properties` file in the `security` subdirectory of the `configuration` directory. If the file does not exist, create it by copying the `login.properties` file from oedg.home.

4. Add the following line to the `login.properties` file to enable authentication for all realms using X.509 certificates:

```
x509 = true
```

> **Note:** To enable X.509 certificate for specific realms, add a line of code for each realm, including the realm name as a prefix. For example, for the dn realm, add the following line:
>
> ```
> dn.x509 = true
> ```

5. If required, enable certificate authentication for web pages in all contexts by adding the following setting:

```
http.x509 = true
```

   Alternatively, to enable selectively for different contexts, add the context as a suffix to the setting; for example:

```
http.x509.admin = true
http.x509.formws = true
http.x509.ws = true
http.x509.dashboard = true
```

2.2 Assigning Personal Certificates and Key Combinations

When SSL client authentication is enabled on a server, each user must have a certificate and associated private key available on the client. The certificate is not sensitive and can be distributed freely, but the private key must be stored and distributed securely.

Each certificate/key combination user can be stored in a number of ways:

- In the operating system certificate store; for example, Internet Options/Content/Certificates on a Windows platform.
- A smart card.
- A USB dongle.

Certificate/key combinations can either be generated and distributed to users or created by a certificate authority website, allowing users to apply for one as required. The latter approach is preferable, because the private key is generated on the system of the user and therefore is not transmitted.

Note that on Windows platforms, Java Web Start uses the operating system certificate store in addition to the internal Java store. If a certificate has been created in Internet Explorer (or Google Chrome), it is stored in the system store and will be used by Web Start. Mozilla Firefox has an internal certificate store; certificates generated in Firefox must be added manually to the system or Java store before use with Java Web Start.

2.3 Associating Certificates With a User

Once a user has a certificate, it must be associated with an EDQ user account to enable automatic authentication. This association provides an alternative way of performing pass-through authentication, or single sign-on (SSO), whereby users do not need to log in to EDQ applications.

2.3.1 Internal Users

A Java Management Extensions (JMX) interface and associated script can be used to store the certificate in the user record:

```
java -jar jshell.jar setcert.groovy -server SERVER:JMXPORT
[ -username adminusername -pw password ]
[ -cert certfile ] -for username
```

- *SERVER* is the host name of the EDQ server and *JMXPORT* is the port used with jconsole.
- *adminusername* and *password* are for an EDQ administrator (they can be omitted if SSO is enabled and jshell is run from the tools directory).
- *certfile* is the file containing the certificate, in PEM or DER format. If `-cert` is omitted, any existing certificate is removed from the user record.
- *username* is the internal user being updated.

2.3.2 External Users

The certificate is stored in an attribute for the user in the external LDAP server. For example, in Active Directory the `userCertificate` attribute is used.

3 SSL With JMX

EDQ supports the use of SSL to connect to JMX.

Because the connector for JMX is created within EDQ, the configuration of SSL is performed by editing the `director.properties` file and not by altering the configuration of the application server.

3.1 Settings

1. To enable SSL for JMX add the following line to the `director.properties` file:

   ```
   management.ssl.port = portnumber
   ```

   - Replace *portnumber* with the required port number.
   - The key and certificate for the connection can be specified in separate files or in a Java keystore.

2. To use separate crt and key files in Privacy-enhanced Electronic Mail (PEM) or Distinguished Encoding Rules (DER) format, add the settings:

```
management.ssl.km.crt = crtfile
management.ssl.km.key = keyfile
```

3. If the key is encrypted, add the following setting to set the key password:

   ```
   management.ssl.km.keypw = password
   ```

   However, if the certificate is in a Java keystore, use these settings:

   ```
   management.ssl.km.keystore = keystorefile
   management.ssl.km.storetype = storetype
   management.ssl.km.storepw = storepassword
   management.ssl.km.alias = alias
   management.ssl.km.keypw = keypassword
   ```

   - The alias property can be omitted if the key store contains a single entry; otherwise it is the alias of the certificate/key entry in the store.
   - The store type defaults to `JKS`.
   - The keypw can be omitted if the password for the key is the same as the password for the store (this is usually the case).

4. To enable SSL client authentication for JMX, add the following setting:

   ```
   management.ssl.clientauth = required
   ```

   Replace required with optional to use a certificate if available, but to successfully connect if it is not.

5. To configure the issuer certificates for valid client certificates, add the following setting to accept any client certificate:

   ```
   management.ssl.tm.any = true
   ```

   Alternatively, add the following setting to specify a certificate bundle file (containing a series of concatenated PEM certificates):

   ```
   management.ssl.tm.bundle = bundlefile
   ```

   Or add the following settings to specify a keystore:

   ```
   management.ssl.tm.keystore = keystorefile
   management.ssl.tm.storetype = keystoretype
   management.ssl.tm.storepw = keystorepassword
   ```

Note: The default keystore type is JKS, which does not require the keystorepassword setting.

3.2 Using SSL with JMX Clients

If SSL has been enabled for JMX and SSL client authentication is not enabled, no special configuration is required if the server certificate is issued by an authority trusted by the Java Runtime Environment (JRE).

If the issuer is not currently trusted, the Certification Authority (CA) certificate can be added to the JRE cacerts store using the keytool command supplied with the JRE. Alternatively a keystore containing the CA certificate can be supplied using the standard Java SSL properties.

For example:

```
jconsole -J-Djavax.net.ssl.trustStore=trustkeystorefile
```

If SSL client authentication has been enabled, key store properties are also required:

Notice the parallel with server configuration. The client trust store is used to trust the certificate in the server key store; the client key store contains the certificate that is trusted by the server trust store (or certificate bundle).

```
jconsole -J-Djavax.net.ssl.trustStore=trustkeystorefile
-J-Djavax.net.ssl.keyStore=keystorefile
-J-Djavax.net.ssl.keyStoreType=keystoretype
-J-Djavax.net.ssl.keyStorePassword=keystorepassword
```

See the JRE documentation for details of the `java.net.ssl` property set.

The JMX command line tools (and the jshell script interpreter) also support setting SSL configuration with environment variables and/or command line arguments. Not all JMX scripts support SSL options; if not available, use the environment variables.

If SSL client authentication is not enabled, use the `EDQ_SSL_TRUST` environment variable or `-ssltrust` command line option to specify a Java keystore containing the CA certificate for the server (this is analogous to the first jconsole example).

If SSL client authentication has been enabled, use the `EDQ_SSL_PROPS` environment variable or `-sslprops` command line argument to specify a properties file containing key and trust store settings. The property format is identical to the server configuration in director.properties except that the `management.prefix` is not used.

For example, to specify trust for the server certificate using a Java keystore, and the client certificate and key as separate crt and key files the properties file would contain:

```
tm.keystore = trustkeystorefile
km.crt = crtfile
km.key = keyfile
```

Key and store password, etc, properties can be added as necessary.

The property file contents can be specified directly in the environment or on the command line by enclosing the property settings in `{..}` and separating property values with commas. For example, the preceding property file would be specified as:

```
{tm.keystore=trustkeystorefile, km.crt=crtfile, km.key=keyfile}
```

3.2.1 Command Examples

```
java -jar jmxtools.jar runjob -job x -project z -sslprops c:\tmp\ssl.properties
localhost:9005
```

To specify SSL trust and key information in a properties file, using the command line option, you could use the following:

```
set EDQ_SSL_TRUST=c:\tmp\trust.jks
java -jar jshell.jar scripts\system\sysreport.groovy -user dnadmin -pw password
-server localhost:9005
```

To run a system report specifying a trust store using an environment variable. In UNIX, for example, the command might be:

```
EDQ_SSL_TRUST=/tmp/trust.jks
export EDQ_SSL_TRUST
```

```
java -jar jshell.jar scripts/system/sysreport.groovy -user dnadmin -pw password
-server localhost:9005
```

To use a client certificate with in line properties, the command might be:

```
EDQ_SSL_PROPS="{tm.keystore=/tmp/trust.jks,km.crt=/tmp/me.crt,km.key=/tmp/me.key}"
export EDQ_SSL_PROPS
java -jar jshell.jar scripts/system/sysreport.groovy -server localhost:9005
```

4 Related Documents

For more information, see the following documents in the documentation set:

- *Oracle Enterprise Data Quality Installation Guide*
- *Oracle Enterprise Data Quality LDAP Integration Guide*

See the latest version of these and all documents in the Oracle Enterprise Data Quality Documentation website at:

http://download.oracle.com/docs/cd/E48549_01/index.htm

5 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Oracle Enterprise Data Quality SSL Configuration, Release 11g R1 (11.1.1.7)
E40048-02