Oracle ® Enterprise Data Quality

SSL Configuration

Version 9.0

January 2012

ORACLE

Table of Contents

1 Introduction

From version 9.0.1 onwards, the OEDQ application enforces the use of SSL when using the web pages on the Launchpad. Any HTTP requests are automatically redirected to the HTTPS port. Therefore, SSL must be set up on the application server in order to use these web pages.

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

If OEDQ is installed using the provided Windows Installer, SSL is automatically enabled. However, if OEDQ is installed manually, SSL must be enabled separately, if this has not been done already.

This document provides instructions for setting up SSL on an OEDQ Application Server running on Tomcat. For other application servers (such as Oracle Weblogic Server or IBM Websphere), consult the standard server documentation.

> **WARNING**: A specific issue with the Weblogic internal implementation in some versions is that the emailAddress component (OID 1.2.840.113549.1.9.1) of a X.500 distinguished name is not recognized. This means that a client certificate containing an emailAddress value in the subject name will not be accepted.

2 Configuring SSL During Installation

The Windows EDQ installer automatically configures SSL using an automatically generated certificate containing the fully qualified name of the host.

However, if Tomcat is manually installed on Windows or any other platform, the HTTPS connector must be configured using the following procedure:

1. Find the server.xml file for the Tomcat installation. Typically it contains the following lines:

```
<!-- 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=”pathtokeyfkle” />
```

See the Tomcat and mod_ssl documentation for further information.[1]

3 Configuring SSL Client Authentication

EDQ can support SSL client authentication using SSL client certificates.

There are three stages to configuring SSL client certificates:

---

[1]Tomcat documentation: http://tomcat.apache.org/tomcat-6.0-doc/config/http.html

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

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

As mentioned previously, this document focuses on environments using the Tomcat server. For information on configuring another server with SSL certificates, refer to the server documentation.

3.1 Configuring Tomcat to support client certificates

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

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

   - 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).
   - Add the location of the trust file containing the certificate issuers for trusted client certificates.
   - Set truststoreType to JKS or PKCS12.

2. If the Tomcat installation includes 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=”pathtokeyfkle”
      SSLCACertificateFile="pathtocabundlefile"
      SSLVerifyClient="require” />
```

3. Locate the OEDQ login.properties file in the Configuration folder, and add the following line 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
```

4. 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
   ```

3.2 Assigning Personal Certificates/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 PC. 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' such as the SafeNet eToken.

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

3.3 Associating certificates with a user

Once a user has a certificate, it must be associated with an EDQ user account to enable automatic authentication (an alternative to Single Sign-On).

3.3.1 Internal users

A 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
```

Where:

- SERVER is the host name of the EDQ server and JMXPORT is the port used with jconsole.
- The 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.
- The username supplied with -for is the internal user being updated.

3.3.2 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.

4 SSL with JMX

EDQ supports the use of SSL to connect to JMX.

The connector for JMX is created within EDQ so the configuration of SSL is done in director.properties, not the application server.

4.1 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 PEM or 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.

4.2 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 JRE.

If the issuer is not currently trusted, the 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:[1]

```
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 via 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 above).

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/or 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 property file above would be specified as:

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

[1]Note 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 which is trusted by the server trust store (or certificate bundle).

4.2.1 Command examples

NOTE: In all the examples below, line breaks in a command are added for clarity; the command should be entered on a single line.

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

Specify SSL trust and key information in a properties file, using the command line option.

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

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 inline properties:

```
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
```