Setting up SSL configuration in WebSphere Message Broker

Gautam K. K. Bhat

May 09, 2012

This article shows you how to set up SSL communication in WebSphere Message Broker on AIX. It includes an example of an important factor to be considered during SSL troubleshooting and problem determination.

Introduction

In many environments, Secured Socket Layer (SSL) configuration is challenging because of the number of components involved in the configuration and setup. SSL configuration and usage in IBM® WebSphere® MQ is altogether different from SSL usage in WebSphere Message Broker, including differences in terminology. Implementing WebSphere Message Broker SSL requires a good understanding of WebSphere Message Broker nodes for developers, as well as a good understanding of WebSphere Message Broker Infrastructure for infrastructure support teams.

WebSphere Message Broker is a convenient central point for web services brokering and transformation of Web Services Definition Language (WSDL) definitions. A message flow can either be a requester (client) that calls out to a web service, or it can be a service provider that its web service clients invoke. The most commonly nodes used for this purpose are HTTPInput node, HTTPReply node, HTTPRequest node, and the corresponding HTTPS nodes.

This article show you how to implementing SSL on WebSphere Message Broker and configure HTTP to use SSL (HTTPS) communication.

Terminology

Certificate authority (CA)

A trusted third-party that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate.

Certificate signing request (CSR)

A message sent from an applicant to a certificate authority in order to apply for a digital identity certificate.

Keystore

A repository that stores the key entries and security certificates used for instance in SSL encryption.

Nodes In WebSphere Message Broker

Nodes are entities that you can use to define and create message flows. Of the many nodes available in WebSphere Message Broker, the following ones are used to with SSL: HTTPInput, HTTPReply, HTTPRequest, SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest.

Truststore

If a keystore that is used to contain trusted certificates.

Truststore directory structure

The Trust store cacerts file in a Java keystore (JKS) format is stored in the following default locations on AIX:

- WebSphere Message Broker V6: /opt/IBM/mqsi/610/jre15/ppc64/lib/security
- WebSphere Message Broker V7: /opt/IBM/mqsi/7.0/jre16/lib/security

The keystore file can be stored in any location as long as it is specified in the broker registry, as described below.

SSL configuration steps

As in WebSphere MQ, SSL configuration in WebSphere Message Broker requires a key repository, referred to as a keystore. SSL is used to enhance the security of the WebSphere Message Broker infrastructure. Here are the high-level SSL configuration steps:

1. Generate a keystore -- There are several ways to create a keystore file such as using `gsk7cmd/gsk6cmd`, which comes as part of the Global Secure Toolkit (GSK) graphical tool called ikeyman. This article uses a command-line tool called keytool.
2. Generate a certificate signing request (CSR) for the existing keystore.
3. Import a root or intermediate Certificate Authority (CA) certificate to the existing keystore.
4. Import a signed certificate to the existing keystore.
5. Validate the certificate details, including:
    - List all certificates.
    - List a specific certificate.
    - List trusted CA certificates.

1. Generate a keystore

```
keytool -genkey -alias <broker name> -keystore <broker name>.jks -keysize 2048
```

The keytool command will be in path of the Broker service id. Here, <broker name> indicates the broker instance running on your server; having broker name as alias differentiates between separate entries for every broker.

As a best practice, the keystore file name (keystore.jks) should have <broker name> in it, such as <Broker name>.jks. For simplicity, we will use BROKER1 as the name of the broker. The above command generates the private key along with the keystore file. After you enter the above command, you will be prompted with these questions:

```
What is your first and last name?
    [Unknown]:
What is the name of your organizational unit?
    [Unknown]:
What is the name of your organization?
    [Unknown]:
What is the name of your City or Locality?
    [Unknown]:
What is the name of your State or Province?
    [Unknown]:
What is the two-letter country code for this unit?
    [Unknown]:
```

After you provide answers to the above questions, you will be prompted to verify that all are correct, as shown below. If all are correct, enter `Yes`:

```
Is CN=, OU=, O=, L=, ST=, C= correct? (type "yes" or "no")
    [no]:  yes

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

A sample entry looks like this,

```
What is your first and last name?
    [Unknown]:  BROKER1
What is the name of your organizational unit?
    [Unknown]:  ZONE1
What is the name of your organization?
    [Unknown]:  IBM
What is the name of your City or Locality?
    [Unknown]:  US
What is the name of your State or Province?
    [Unknown]:  Washington
What is the two-letter country code for this unit?
    [Unknown]:  US
Is CN=BROKER1, OU=ZONE1, O=IBM, L=US, ST=Washington, C=US correct? (type "yes" or "no")
    [no]:  yes
Enter key password for <bonca60>:
    (RETURN if same as keystore password):  ********
$
```

2. Generate a certificate signing request (CSR) for the existing keystore

```
keytool -certreq -alias BROKER1 -keystore BROKER1.jks -file BROKER1.csr
```

Here you create a private key. Send the CSR file to the CA team to get the certificates generated. The procedure for passing the CSR to the CA depends on the CA -- the most popular way to transfer the certificate details is via a web link. After you receive the signed certificate (named certificate.der below) from the CA, proceed with the following steps:

3. Import a root or intermediate CA certificate to the existing keystore

```
keytool -import -trustcacerts -alias root -file Thawte.crt -keystore BROKER1.jks
```

The keystore file name is BROKER1.jks and the intermediate CA cert is Thawte.crt.

You must import the root and/or the intermediate CA certificates before importing the signed certificate, because the certificates work in sequence. The root certificate must be present in the key file so that the signed certificate has a platform to fit into. The most commonly used CA's are GlobalSign and VeriSign.

4. Import a signed certificate into the existing keystore

```
keytool -import -trustcacerts -alias BROKER1 -file certificate.der -keystore BROKER1.jks
```

This certificate is the one that you received from the CA above. The signed certificate file name is certificate.der.

5. Validate the certificate details

To ensure that the above steps have been performed correctly, it is important to do the following validation and verification checks:

List all certificates available in the keystore

```
keytool -list -keystore BROKER1.jks

/home/brkr>keytool -list -keystore BROKER1.jks
IBMJSSEProvider2 Build-Level: -20100325
Enter keystore password:

Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 11 entries

verisign class 1 public primary certification authority - g3,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
verisign class 1 public primary certification authority - g2,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
verisign class 4 public primary certification authority - g3,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): DB:C8:F2:27:2E:B1:EA:6A:29:23:5D:FE:56:3E:33:DF
verisign class 4 public primary certification authority - g2,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 26:6D:2C:19:98:B6:70:68:38:50:54:19:EC:90:34:60
verisign class 2 public primary certification authority,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): B3:9C:25:B1:C3:2E:32:53:80:15:30:9D:4D:02:77:3E
entrust.net global client certification authority,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 9A:77:19:18:ED:96:CF:DF:1B:B7:0E:F5:8D:B9:88:2E
thawte_dv_ssl_ca_3, Oct 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): A5:97:C7:3F:D2:0D:F6:0C:10:D5:4D:31:49:D6:CA:9D
verisign class 2 public primary certification authority - g3,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
verisign class 2 public primary certification authority - g2,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
verisign class 3 secure server ca, Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 2A:C8:48:C0:85:F3:27:DE:32:29:44:BB:B0:2C:79:F8
verisign class 3 public primary certification authority,
    Sep 14, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 10:FC:63:5D:F6:26:3E:0D:F3:25:BE:5F:79:CD:67:67
```

List a specific certificate

```
keytool -list –v –keystore BROKER1.jks –alias <alias name>
/home/brkr>keytool -list -v -keystore BROKER1.jks -alias broker1
IBMJSSEProvider2 Build-Level: -20100325
Enter keystore password:
Alias name: broker1
Creation date: Sep 14, 2011
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=xxxx.xxx.xxxxxxxx.com, OU=Zone1, O=IBM, L=India, ST=Chennai, C=IN
Issuer: CN=M-PKI-TER-CA, O=IBM, C=IN
Serial number: 13fd3b
Valid from: 9/8/11 1:51 PM until: 10/12/12 1:51 PM
Certificate fingerprints:
    MD5:  21:6B:F8:B8:31:3B:CA:5A:6D:92:86:80:B6:24:70:C1
    SHA1: DC:88:DA:49:72:4E:53:F5:74:6D:7C:82:A8:18:7C:7F:A3:E1:FA:BD
```

List trusted CA certificates

This command shows CA authority certificate details:

```
keytool -list –v -keystore /opt/IBM/mqsi/7.0/jre16/lib/security/cacerts
```

Configuring Message Broker to serve HTTP/HTTPS requests

In WebSphere Message Broker, HTTPInput, HTTPReply, HTTPRequest, SOAPInput, SOAPReply, SOAPRequest, and SOAPAsyncRequest nodes are used to facilitate HTTP/HTTS requests to and from the web service. For more information on these nodes, see Built-in nodes in the WebSphere Message Broker information center.

As part of broker infrastructure changes, you must tell the broker where to look for keystore and truststore files:

1. List the broker registry

```
mqsireportproperties BROKER1 -o BrokerRegistry -r
BrokerRegistry
    uuid='BrokerRegistry'
    brokerKeystoreType='JKS'
    brokerKeystoreFile=' /home/brkr/BROKER1.jks’
    brokerKeystorePass='brokerKeystore::password'
    brokerTruststoreType='JKS'
    brokerTruststoreFile=' /opt/IBM/mqsi/7.0/jre16/lib/security/cacerts'
    brokerTruststorePass='brokerTruststore::password'
    httpConnectorPortRange=''
    httpsConnectorPortRange=''
    modeExtensions=''
    operationMode='enterprise'
    shortDesc=''
    longDesc=''
BIP8071I: Successful command completion.
```

For more information, see mqsireportproperties command in the WebSphere Message Broker information center.

2. Import root certificates and server certificates to the broker truststore

Traverse to `CODE/opt/IBM/mqsi/7.0/jre16/lib/security` and proceed with the following steps:

```
keytool -import -trustcacerts –alias root.Cert -file /home/brkr/ Thawte.crt
   -keypass <password> -keystore cacerts –storepass changeit

keytool -import -alias BROKER1 -file /home/brkr/certificate.der
   -keystore cacerts –storepass changeit
```

The default password of trustore (cacerts) is xxxxx.

3. Enable SSL on the broker instance

This command enables SSL for the HTTP listener object:

```
mqsichangeproperties BROKER1 -b httplistener -o HTTPListener -n enableSSLConnector -v true
```

4. Modify broker properties to point to the keystore file

The keystore file is generated above in Step 1. Generate a keystore.

```
mqsichangeproperties BROKER1 -b httplistener -o HTTPSConnector -n keystoreFile
    -v /home/brkr/BROKER1.jks
```

5. Add broker keystore file to broker registry

```
mqsichangeproperties BROKER1 -o BrokerRegistry -n brokerKeystoreFile
    -v /home/brkr/BROKER1.jks
```

6. Add broker truststore file to broker registry

```
mqsichangeproperties BROKER1 -o BrokerRegistry -n brokerTruststoreFile
     -v /opt/IBM/mqsi/7.0/jre16/lib/security/cacerts
```

7. Set the registry password for keystore

```
mqsisetdbparms BROKER1 -n brokerTruststore::password -u temp -p changeit
```

8. Associate the broker with keystore password

```
mqsichangeproperties BROKER1 -b httplistener -o HTTPSConnector -n keystorePass
    -v <password>
```

9. Associate a port for broker to serve HTTPS requests

```
mqsichangeproperties BROKER1 -b httplistener -o HTTPSConnector -n port -v 7094
```

Now BROKER1 will run on port 7094 for HTTPS requests. Default port for HTTPS requests = 7083

10. Associate a port for broker to serve HTTP requests

```
mqsichangeproperties BROKER1 -b httplistener -o HTTPConnector -n port -v 7091
```

Now BROKER1 will run on Port 7091 for HTTP requests. The default port for HTTP requests is 7080.

11. Change the JVM attributes

You can change JVM heap sizes according to your requirements by modifying the object `ComIbmJVMManager`:

```
mqsichangeproperties BROKER1 -o ComIbmJVMManager -n jvmMaxHeapSize -v 1048576000
mqsichangeproperties BROKER1 -o ComIbmJVMManager -n jvmMinHeapSize -v 134217728
```

12. Verify the broker properties

```
mqsireportproperties BROKER1 -b httplistener -o HTTPConnector -n port
7091
BIP8071I: Successful command completion.

mqsireportproperties BROKER1 -b httplistener -o HTTPSConnector -n port
7094
BIP8071I: Successful command completion.
```

13. Restart the broker

```
mqsistop <Broker Name>
mqsistart <Broker Name>
mqsistop BROKER1
mqsistart BROKER1
```

Setting up ports exclusively for execution groups

To serve SOAP requests, a port needs to be configured at the execution group (EG) level. Each execution group has one listener, one HTTP port, and one HTTPS port. The default SOAP node port numbers are 7800 for HTTP and 7843 for HTTPS. In the example below, <EG Name> stands for execution group name.

1. Configure the SSL protocol

First tell the EG which SSL protocol type are using. SSLv3 is the default SSL protocol.

```
mqsichangeproperties BROKER1 -e <EG Name> -o HTTPSConnector -n sslProtocol -v SSLv3
```

2. Configure the port for SOAP over HTTP requests

Explicitly configure the port for SOAP over HTTP requests.

```
mqsichangeproperties BROKER1 –e <EG Name> -o HTTPSConnector
    -n explicitlySetPortNumber -v 7963
```

3. Associate the keystore file with the broker EG

The keystore file created earlier needs to be associated with the broker instance in order for it to know its repository file. To avoid confusion, do not have multiple keystore files on the server.

```
mqsichangeproperties BROKER1 -e <EG Name> -o HTTPSConnector
    -n keystoreFile -v /home/brkr/BROKER1.jks
```

4. Associate the keystore type.

You should configure the keystore type on the broker, because there are several other keystore types supported by broker. Information on these types is outside the scope of this article, which uses a Java Keystore (JKS).

```
mqsichangeproperties BROKER1 -e <EG Name> -o HTTPSConnector -n keystoreType -v JKS
```

5. Associate the keystore password

Associate the keystore password to the broker so that it can save it in its registry for authentication purpose, which is required when querying the new requests:

```
mqsichangeproperties BROKER1 -e <EG Name> -o HTTPSConnector -n keystorePass -v <password>
```

Setting up JVM attributes for execution groups

When an execution group is started in WebSphere Message Broker, it creates a JVM that is primarily used by the IBM primitive nodes that make use of Java functionality. The DataFlowEngine JVM can be configured either by passing parameters to it directly, or through the broker. When using the broker JVM by any of the means above, the DataFlowEngine memory may continue to grow and may cause resource problems. Use the following few commands to set up your min and max JVM heap size:

```
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n keystoreFile
    -v /home/brkr/BROKER1.jks
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n keystoreType
    -v JKS
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n keystorePass
    -v brokerKeystore::password
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n truststoreFile
    -v /home/brkr/BROKER1.jks
```

In this command, the keystore file type is associated with the ComIbmJVMManager object.

```
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n truststoreType -v JKS
```

When querying new requests, associate the keystore password with the broker's ComIbmJVMManager object so that it can be saved it in its registry for authentication purposes:

```
mqsichangeproperties BROKER1 -e <EG Name> -o ComIbmJVMManager -n truststorePass
    -v brokerTruststore::password
```

Problem scenario

In this scenario, the signed certificate supplied by CA is incorrect. This situation is a bit tricky to troubleshoot, but you can use the commands described above with close attention to their output.

We renewed the broker certificate and were able to display the certificate details on the same server with the correct start and expiration dates. To reconfirm it, we tried the URL `https://`

`<hostname of UNIX server:><port>` on the corresponding Message Broker server and it correctly displayed the renewed certificate with start and expiration dates. But applications could not connect to Message Broker and received certificate validation errors. Normally, the .der format certificate is imported as part of certificate renewal. We determined that the .der certificate did not have chained certificates because the CA team failed to include them. What made us think the chained certificates were missing? We displayed the complete list of certificates and compared the environments. What are those chained certificates? They are the certificates that identify the CA.

```
/var/mqsi/ssl/BROKER1>keytool -list -v -alias broker1 -keystore BROKER1.jks
    -storepass <password>
Alias name: broker1
Creation date: Dec 9, 2011
Entry type: keyEntry
Certificate chain length: 3
Certificate[1]:
Owner: CN=servername, OU=ZONE1, O=IBM, L=CN, C=IN
Issuer: CN=IBM_PKI_SubCA2, O=IBM, C=IN
Serial number: 3142a
Valid from: 11/7/11 8:43 AM until: 12/11/12 8:43 AM
Certificate fingerprints:
    MD5:  93:7F:6D:07:72:AA:47:0D:0A:BB:1C:9D:1B:3F:68:F8
    SHA1: D2:7E:1B:99:46:DB:88:24:4E:AE:35:B1:DF:D6:40:58:20:91:D1:18
Certificate[2]:
Owner: CN=IBM_PKI_SubCA2, O=IBM, C=IN
Issuer: CN=IBM_PKI_CA, O=IBM, C=IN
Serial number: 2
Valid from: 5/14/03 4:04 AM until: 5/14/13 4:04 AM
Certificate fingerprints:
    MD5:  BE:F7:0A:42:D7:C7:A8:40:B6:31:B1:93:E1:1B:6D:D6
    SHA1: 77:E1:05:21:74:3E:65:6A:11:DB:3D:BD:D2:34:99:7F:45:93:F8:5A
Certificate[3]:
Owner: CN=IBM_PKI_CA, O=IBM, C=IN
Issuer: CN=IBM_PKI_CA, O=IBM, C=IN
Serial number: 0
Valid from: 5/31/02 3:34 AM until: 5/31/32 3:34 AM
Certificate fingerprints:
    MD5:  8E:E6:5E:54:97:0E:DA:E9:12:65:7C:E1:C3:8A:5B:C6
    SHA1: B4:C2:C5:17:91:3D:2F:32:10:AB:2D:5A:99:5A:08:5C:10:4F:3E:2B
```

Conclusion

This article described the standard mechanism for implementing SSL communication in WebSphere Message Broker V6 and V7. It also described common problems in customer environments caused by incorrect certificates provided by the CA.

Acknowledgments

The author would like to thank the following individuals for their valuable input and feedback:

- Hermann Huebler -- Solutions Specialist and SME, Application Integration and Middleware, IBM India
- Muthukumar Manoharan -- Support Specialist, WebSphere MQ and WebSphere Message Broker Support, IBM India
- Vivek Grover -- Team Lead, WebSphere Message Broker and WebSphere Business Events Level-2 Support, IBM US

Related topics

- **WebSphere Message Broker resources**
  - WebSphere Message Broker V8 information center
    A single Web portal to all WebSphere Message Broker V8 documentation, with conceptual, task, and reference information on installing, configuring, and using your WebSphere Message Broker environment.
  - WebSphere Message Broker developer resources page
    Technical resources to help you use WebSphere Message Broker for connectivity, universal data transformation, and enterprise-level integration of disparate services, applications, and platforms to power your SOA.
  - WebSphere Message Broker product page
    Product descriptions, product news, training information, support information, and more.
  - Download free trial version of WebSphere Message Broker
    WebSphere Message Broker is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.
  - WebSphere Message Broker documentation library
    WebSphere Message Broker specifications and manuals.
  - WebSphere Message Broker forum
    Get answers to your technical questions and share your expertise with other WebSphere Message Broker users.
  - WebSphere Message Broker support page
    A searchable database of support problems and their solutions, plus downloads, fixes, and problem tracking.
  - WebSphere Message Broker V8 Development Training
    In this IBM Training course, you will learn about the components of the WebSphere Message Broker development and runtime environments. The course also explores message flow problem determination, and shows you how to construct message flows that use ESQL, Java, and PHP to transform messages.
  - Youtube tutorial: Integrating Microsoft .NET code in a WebSphere Message Broker V8 message flow
    This five-minute youtube tutorial shows you how simple it is to use WebSphere Message Broker V8 to build a message flow that includes Microsoft .NET code. Microsoft Visual Studio is used to build .NET code in C#, which is then integrated into a message flow using Message Broker and an HTTP RESTful interface.
- **WebSphere resources**
  - developerWorks WebSphere developer resources
    Technical information and resources for developers who use WebSphere products. developerWorks WebSphere provides product downloads, how-to information, support resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals.
  - developerWorks WebSphere application integration developer resources

How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application integration and business integration solutions.
- developerWorks WebSphere business process management developer resources
  WebSphere BPM how-to articles, downloads, tutorials, education, product info, and other resources to help you model, assemble, deploy, and manage business processes.
- Most popular WebSphere trial downloads
  No-charge trial downloads for key WebSphere products.
- WebSphere forums
  Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
- WebSphere on-demand demos
  Download and watch these self-running demos, and learn how WebSphere products and technologies can help your company respond to the rapidly changing and increasingly complex business environment.
- developerWorks WebSphere weekly newsletter
  The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.
- WebSphere-related books from IBM Press
  Convenient online ordering through Barnes & Noble.
- WebSphere-related events
  Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.

- **developerWorks resources**
  - Trial downloads for IBM software products
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - developerWorks blogs
    Join a conversation with developerWorks users and authors, and IBM editors and developers.
  - developerWorks cloud computing resources
    Access the IBM or Amazon EC2 cloud, test an IBM cloud computing product in a sandbox, see demos of cloud computing products and services, read cloud articles, and access other cloud resources.
  - developerWorks tech briefings
    Free technical sessions by IBM experts to accelerate your learning curve and help you succeed in your most challenging software projects. Sessions range from one-hour virtual briefings to half-day and full-day live sessions in cities worldwide.
  - developerWorks podcasts
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - developerWorks on Twitter
    Check out recent Twitter messages and URLs.
  - IBM Education Assistant

A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.