IBM Software Group

SSL Configuration of the IBM Java EE Application Client and the WebSphere Application Server V7 Service Integration Bus

Rich Montjoy (rrmontjo@us.ibm.com)
WebSphere Platform Messaging Support
29 September 2010

WebSphere® Support Technical Exchange

Agenda

- Introduction to
  - Java Message Service
  - Service Integration Bus
  - launchClient tool
- SSL and JSSE
  - Introduction
  - Terminology
  - SSL Handshake
- Connecting the Application Client to SIB
  - Transport Chains
  - Provider Endpoints

Agenda Continued

- SSL Connection to SIB
  - Code example
  - Configuration via Admin console
  - launchClient tool example
- Tracing the message
- Useful Links
- Questions and Answers

What is JMS?

- Java™ Message Service is a standard (java based) enterprise messaging API
- Defined by a specification developed under Java Community Process JSR 914 as part of J2EE and implementation has been required since J2EE 1.3 (WAS v5)
- Supports both synchronous and asynchronous messaging
- Loosely coupled
- Supports Point-to-point and Publish-Subscribe messaging

JMS Components

- JMS provider: A messaging system that implements the JMS specification.
- JMS clients: Java applications that produce and consume messages off destinations.
- Messages: An object that contains the data being transferred by a JMS client.
- Administered objects: Pre-configured JMS objects that are created by an administrator for the use of JMS clients. For example connection factories and destinations.

What is the Service Integration Bus?

- Default JMS messaging provider with IBM® WebSphere Application Server v6.x, v7
- Consists of interconnected message engines that provide messaging functionality
- Supports JMS 1.1
- Pure Java implementation

Service Integration Bus Components

- Bus
  - Logical entity where applications connect to send or receive messages.
  - Consists of one or more interconnected message engines.
- Message engine
  - The runtime component for messaging.
  - Created when a bus member (either a server or cluster of servers) is added to a bus as a bus member.
- Destinations
  - Logical target (queue or topic space) for each message.

What is the IBM Java EE Application Client?

- Uses the runtime component of either the Application client installation or the WebSphere Application Server installation to access system resources such as security, transactions, jndi lookups and database access
- Can access EJBs, JDBC databases and JMS destinations
- Provides support for XML deployment descriptors
- Installed from the product CD

launchClient

- The launchClient tool is used to start Java EE application clients
  - Windows® C:\Program Files\IBM\WebSphere\AppClient\bin\launchClient.bat
  - Linux® /opt/IBM/WebSphere/AppClient/bin/launchClient.sh
  - Aix /usr/IBM/WebSphere/AppClient/bin/launchClient.sh

- Syntax

  launchClient [-profileName pName | -JVMOptions options | -help | -?] <userapp> [-CC<name>=<value>] [app args]

- Example usage

  launchClient c:\earfiles\myapp.ear -CCBootstrapHost=myWASServer

launchClient - parameters

The launchClient tool uses client container parameters for configuration as specified by the –CC name/value arguments. Some examples are:

-CCBootstrapHost - The name of the host server you want to connect to initially.

-CCBootstrapPort - The server port number. If none is specified the default value is used.

-CCD - Used for JVM system properties

-CCproviderURL - Provides bootstrap server information used to obtain an initial context. It can use either a CORBA object URL or an IIOP URL

-CCtrace - Use this option to obtain debug trace information.

-CCtracefile - Indicates the name of the file to which trace information is written

- C:\"Program Files"\IBM\WebSphere\AppClient\bin\launchClient -help

Agenda

- Introduction to
  - Java Message Service
  - Service Integration Bus
  - launchClient tool
- SSL and JSSE
  - Introduction
  - Terminology
  - SSL Handshake
- Connecting the Application Client to SIB
  - Transport Chains
  - Provider Endpoints

What is SSL?

- Secure Sockets Layer is a standard to ensure secure communications and allow reliable authentication.
- The SSL protocol is based on public key infrastructure (PKI), and uses public key encryption, shared key encryption and hashing algorithms in combination with certificates and certificate authorities for transport layer security.
- Sits between the Application layer and the Transport layer.
- Transport Layer Security (TLS V1.0) is based on SSL V3.0.

What is JSSE?

- Java Secure Socket Extension – Provides transport layer security for the Java 2 platform.
- Provides an API for a Java version of SSL and TLS.
- Relies on X.509 certificate-based key pairs

SSL Terminology

- Keystore – database that can contain both public and private keys used to encrypt and decrypt data.
- Trust store – database that contains the public keys for the remote side stored as signer certificates.
- Digital certificates – allow unique identification of an entity.
- Personal certificates – certificates used to authenticate yourself.
- CA certificates – certificates verified by a 3rd party certificate authority.
- Public/Private keys – used during the SSL handshake. The public key is used to encrypt data to be decrypted by the private key and vice versa.
- Session keys - negotiated during the SSL handshake, and subsequently used to encrypt/decrypt data flowing over the connection.
- Self-Signed certificate- certificate that has been signed with its own private key.
- Ciphers – cryptographic algorithms used for authenticating the client and server.

SSL Handshake (short version)

- The client sends a message to the server and requests a secure connection.
- The server sends a reply with its certificate (containing its public key) to the client and the client verifies the server certificate (server authentication).
- The server may optionally request the client to send its certificate to the server so the server may verify the client (client authentication).
- The client verifies the server's public key, encrypts data using this public key which in turn is used by each end of the connection to create identical secret keys. These secret keys are used to generate session keys which are used to encrypt and decrypt information during the SSL session.
- The client sends a message to the server that future messages will be encrypted with the session key. The server sends a message to the client with the same.

What does SSL buy me?

- Privacy – allows the client and server to establish an encrypted connection to exchange messages.
- Authentication – verifying another hosts identity.
- Integrity – Data is tamper-proof.
- Nonrepudiation – Prevents either sender or receiver from denying they sent data.

Connecting to SIB - Transport Chains

Inbound transport chain – enables messaging clients to communicate to a message engine.

- Default transport chains
  - InboundBasicMessaging (Non-SSL) – uses the port for SIB_ENDPOINT_ADDRESS
  - InboundSecureMessaging (SSL) – uses the port for SIB_ENDPOINT_SECURE_ADDRESS

Connecting to SIB - Provider Endpoints

- Property of the connection factory which allows you to specify a comma-separated list of suitable bootstrap servers.

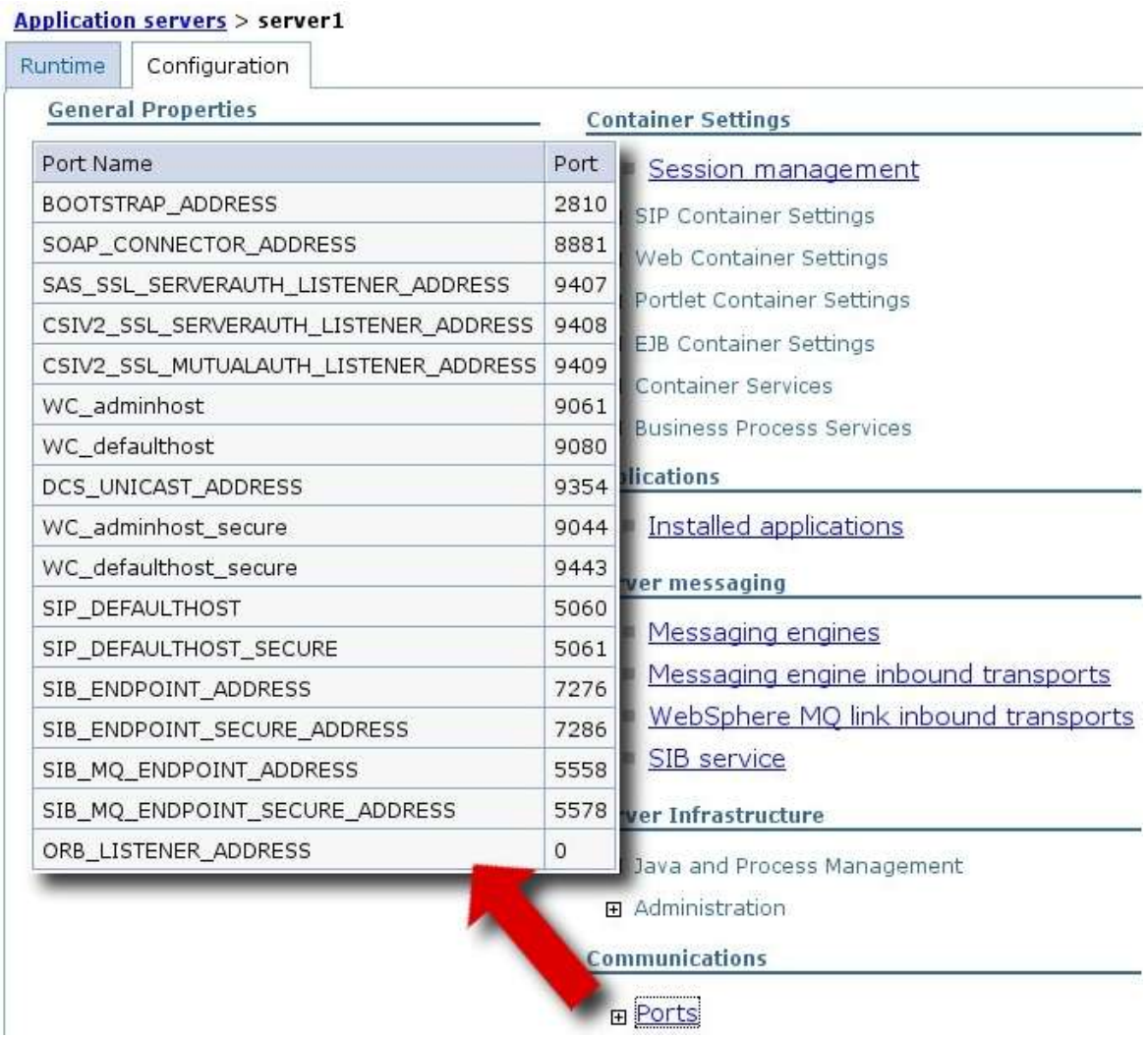
- Specified as a triplet of the form

  `hostname:port:transport chain`

  - Hostname - name of the host where the bootstrap server is running.
  - Port - port number for the SIB service for the bootstrap server which is determined from the inbound transport chain.
  - Transport chain - specifies the transport chain that will be used to send the bootstrap request to the bootstrap server.

  `BootstrapBasicMessaging` – used when sending bootstrap requests to the SIB_ENDPOINT_ADDRESS.

  `BootstrapSecureMessaging` – used when sending bootstrap requests to the SIB_ENDPOINT_SECURE_ADDRESS.



Port Name	Port
BOOTSTRAP_ADDRESS	2810
SOAP_CONNECTOR_ADDRESS	8881
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9407
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9408
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9409
WC_adminhost	9061
WC_defaulthost	9080
DCS_UNICAST_ADDRESS	9354
WC_adminhost_secure	9044
WC_defaulthost_secure	9443
SIP_DEFAULTHOST	5060
SIP_DEFAULTHOST_SECURE	5061
SIB_ENDPOINT_ADDRESS	7276
SIB_ENDPOINT_SECURE_ADDRESS	7286
SIB_MQ_ENDPOINT_ADDRESS	5558
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578
ORB_LISTENER_ADDRESS	0

Port Values

- The port value for **SIB_ENDPOINT_ADDRESS** is used for SIB clients not using SSL (unencrypted TCP/IP)

- The port value for **SIB_ENDPOINT_SECURE_ADDRESS** is used for SIB clients using SSL encryption

Agenda Continued

- SSL Connection to SIB
  - Code example
  - Configuration via Admin console
  - launchClient tool example
- Tracing the message
- Useful Links
- Questions and Answers

Client sample – imports and jndi

```
import java.io.*;
import javax.jms.*;
import java.util.*;
/**  Add SIB import below if you use the SIB ConnectionFactory
 *   import com.ibm.websphere.sib.api.jms.*;
 */
import javax.naming.*;
public class JMSClient
{
   public static void main(String[] args)
   {
     try {
          /**
           * Lookup queue connection factory object using jndi
           */
          System.out.println("* Looking up CF in jndi...");

          Context ctx = new InitialContext();
          ConnectionFactory cf = ConnectionFactory)ctx.lookup("jms/JMSClientCF");
          Destination jmsQ = (Destination)ctx.lookup("jms/JMSClientQ");
          /**
```

Client sample – connection and send

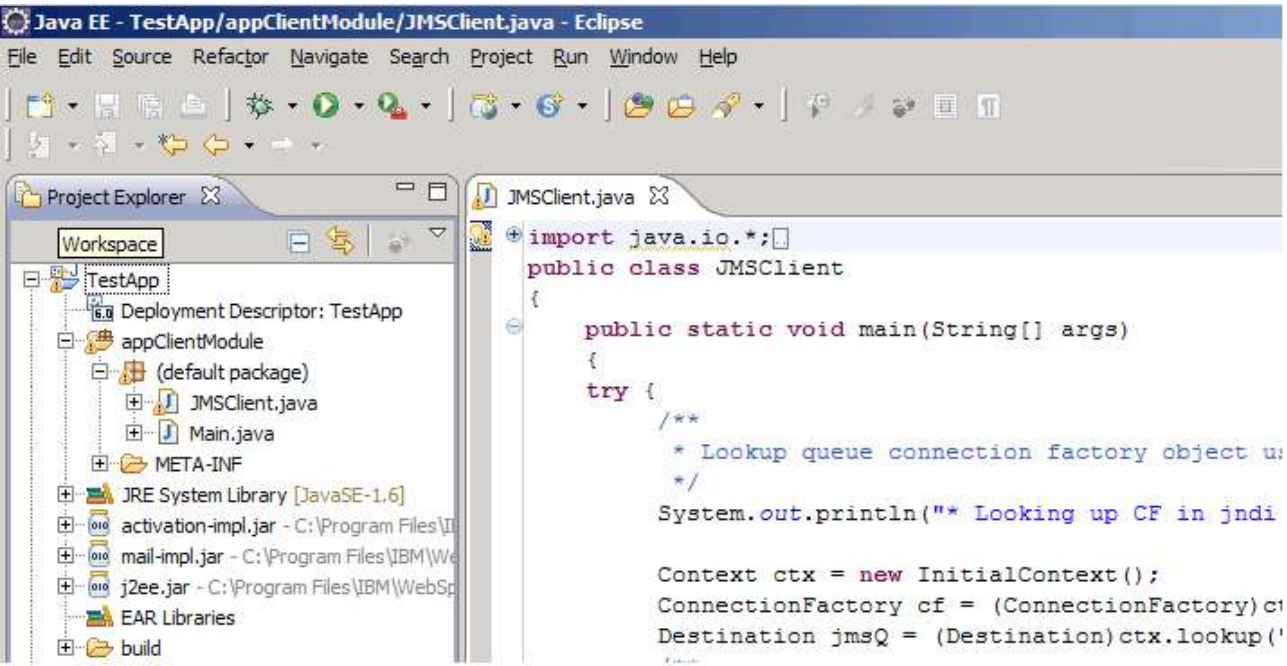
```
/**
 * Get a connection to the Bus
 */
   System.out.println("* Creating queue connection...");
   Connection conn = cf.createConnection();


     conn.start();
   System.out.println("* Queue connection created...");
   /**
    * Start a session
    */
   System.out.println("* Creating queue session...");
   Session session = conn.createSession(false,
   Session.AUTO_ACKNOWLEDGE);
   /**
    * send a simple text message.
    */
   System.out.println("* Creating queue sender to " + jmsQ.toString() +
                      "...");
   MessageProducer qsdr = session.createProducer(jmsQ);
   TextMessage msg = session.createTextMessage("This is a test message");
   qsdr.send(msg);
```

Client – close everything

```
            /**
            * Close everything
            */
            System.out.println("* Sample done");
            conn.close();
        } catch (JMSException j) {
            System.out.println("! Failed with JMSException " + j);
            System.out.println("! Linked Exception " +
                               j.getLinkedException());
            j.printStackTrace();
            System.exit(1);
        } catch (Exception e) {
            System.out.println("! Failed with Exception " + e);
            e.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }
}
```

Summary of steps used to build the TestApp1.ear



- Create a new Application Client project under

File > New > Application Client project

Fill in the project name and select "Add Project to EAR", then finish.

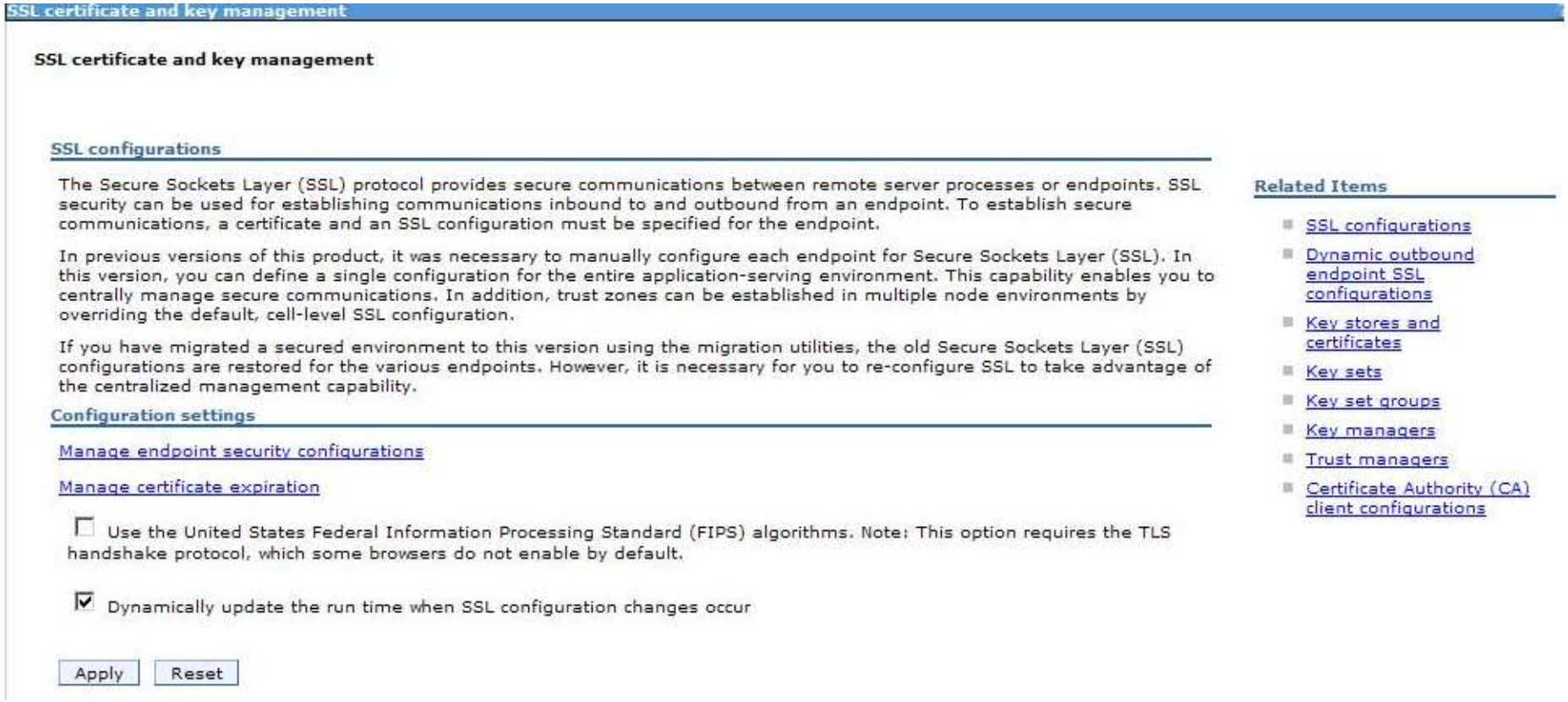
- Expand the new project and drag the source code to under AppClientModule> default package
- Right click on the top level project and choose
  > build path
  > configure build path
  > libraries tab
  > add external jars
  add the "j2ee.jar" from the WAS_INSTALL_ROOT\AppClient\lib directory
- Change the "Main-Class" in the Manifest.MF under META-INF to point to your class
- Export the project as an EAR

Connecting to SIB - Starting point on the SIB side

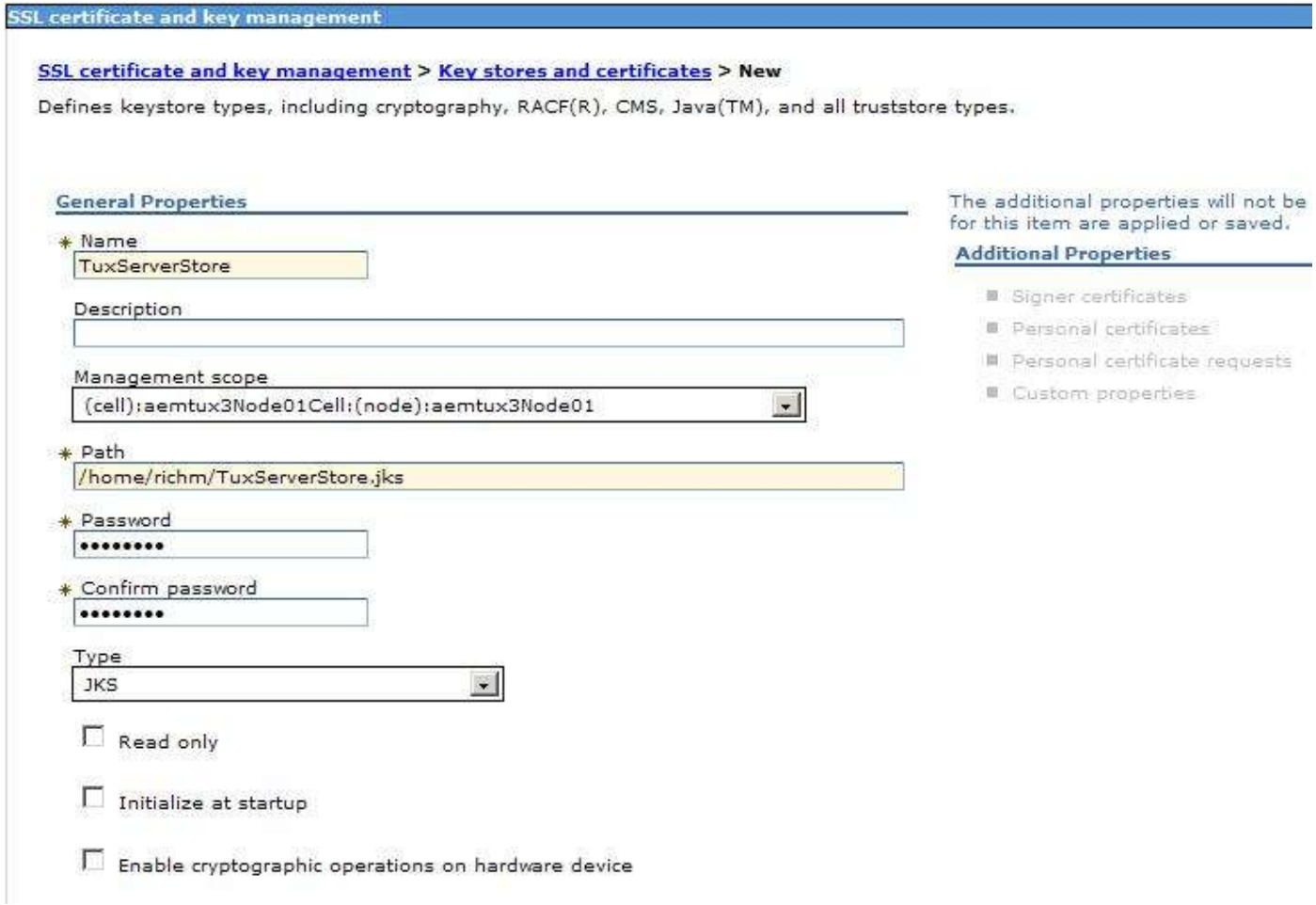
- Bus defined named “TuxBus”
- A Bus member named “server1”
- Messaging engine named “aemtux3Node01:server1
- Queue destination named “TESTQ”
- JMS Connection Factory named “jms/JMSClientCF”
- JMS Queue named “jms/JMSClientQ” pointing to the actual “TESTQ” queue destination.

Server SSL configuration

We will focus on
- Keystores and certificates
- SSL configurations
- Manage endpoint security configurations



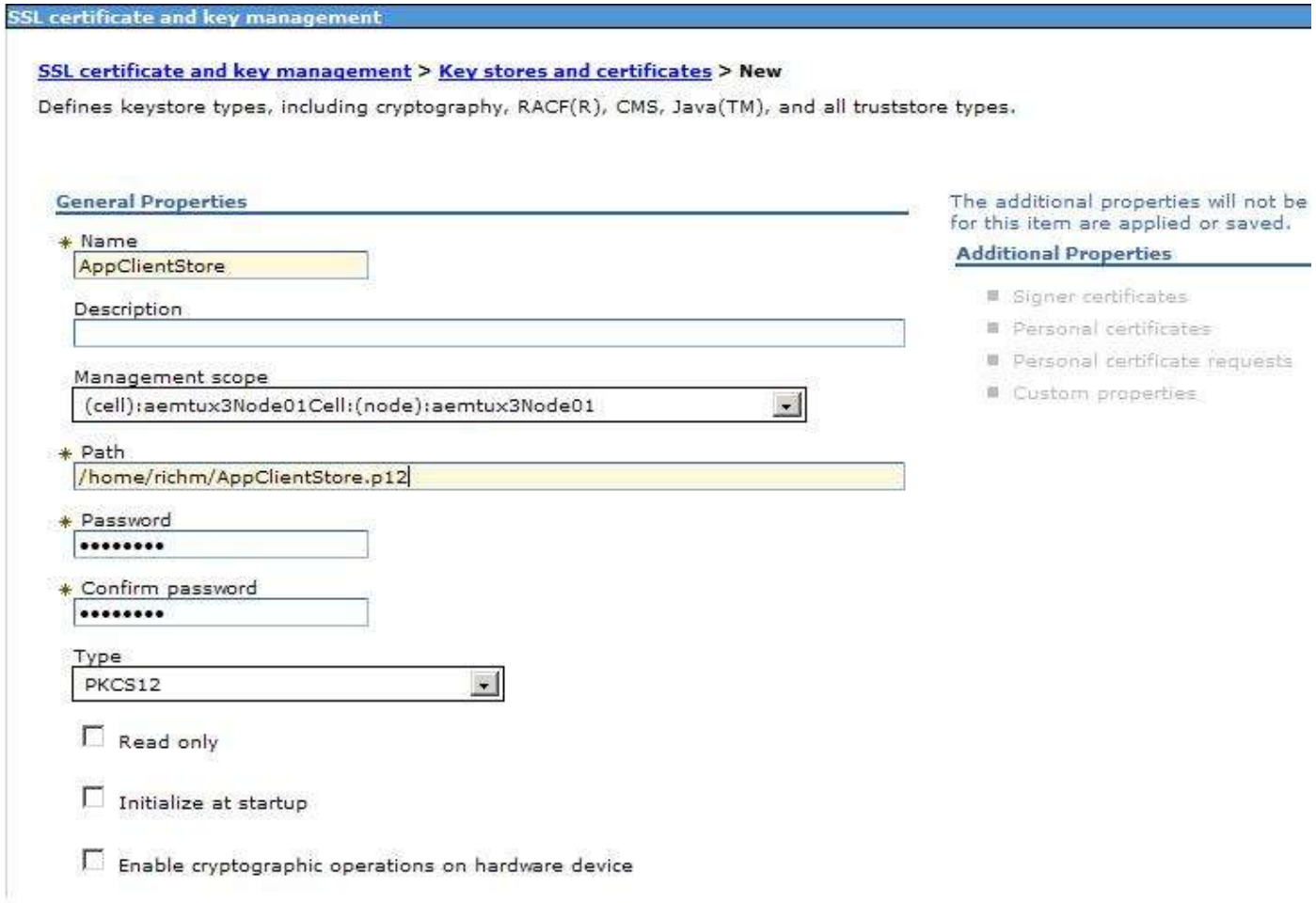
Create the server store



- Path is the fully qualified path to the keystore.
- JSSE defaults to JKS format type
- If the keystore does not already exist, it will be created.

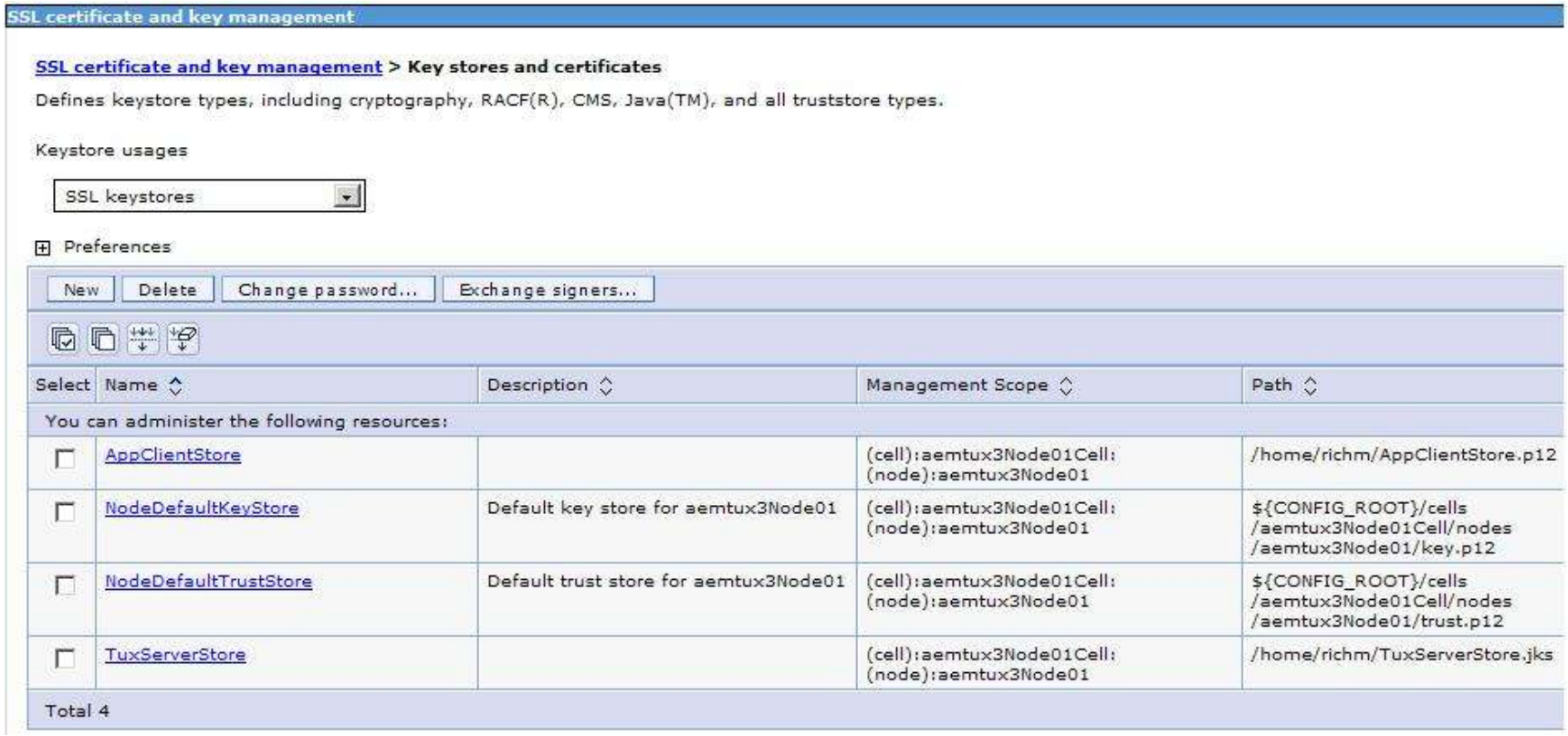
Create the client store

- The PKCS12 type is a standard keystore format



Configured stores

- Click on a keystore or truststore to create/view certificates.
- Select the client and server stores to exchange signers after you have created the personal certificates.



SSL certificate and key management

SSL certificate and key management > Key stores and certificates

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore types.

Keystore usages

SSL keystores

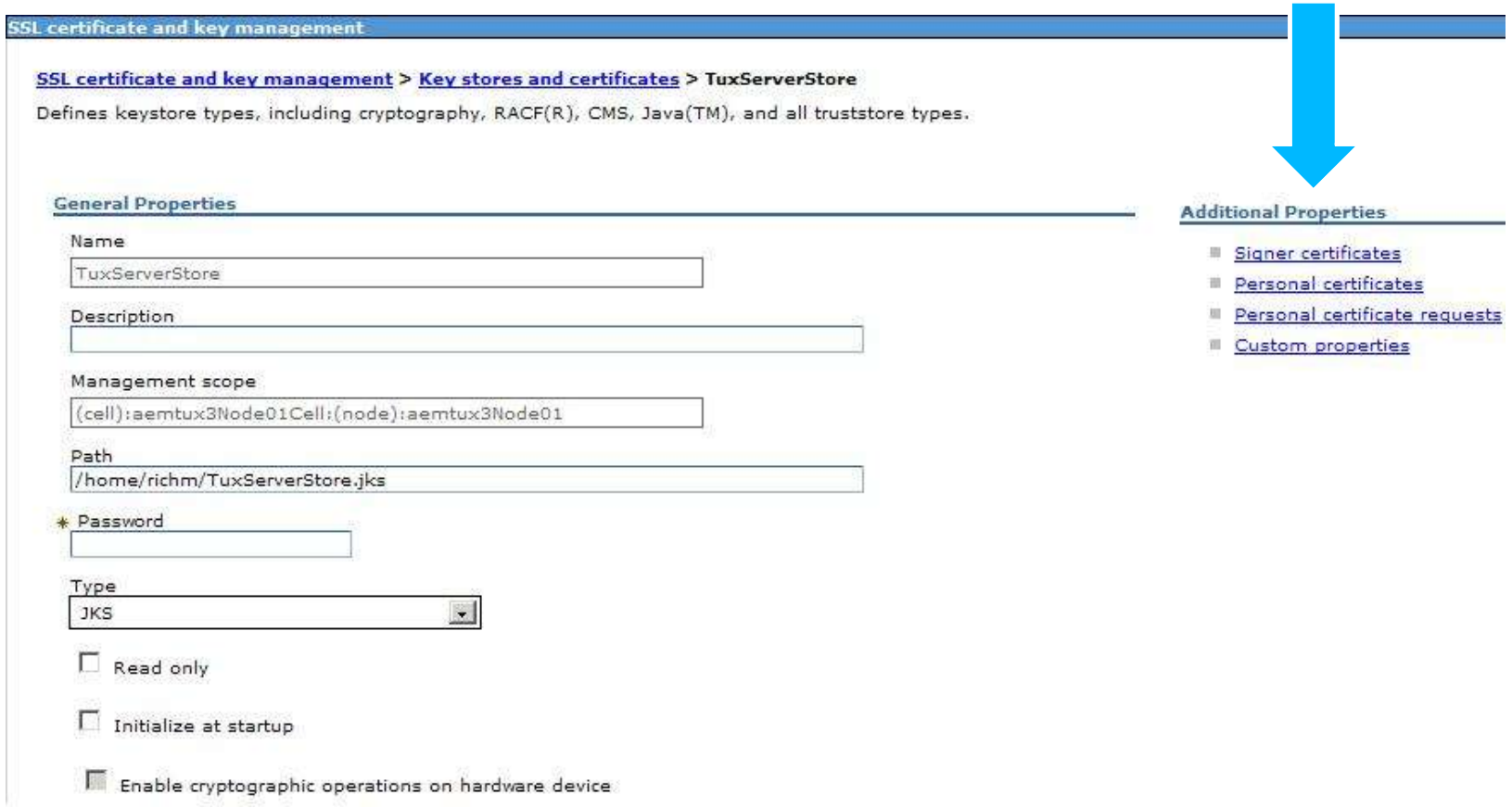
Preferences

New | Delete | Change password... | Exchange signers...

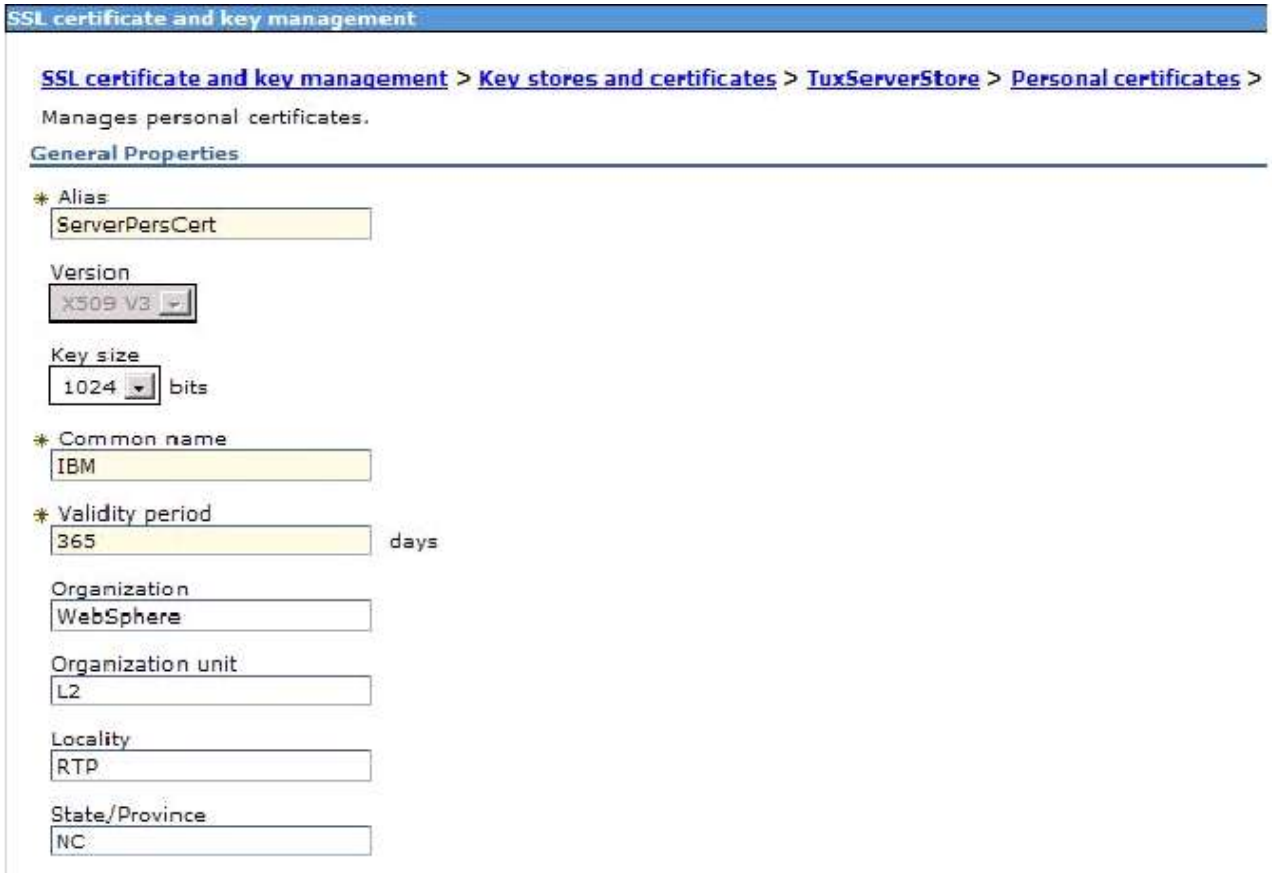
Select	Name	Description	Management Scope	Path
You can administer the following resources:				
☐	AppClientStore		(cell):aemtux3Node01Cell: (node):aemtux3Node01	/home/richm/AppClientStore.p12
☐	NodeDefaultKeyStore	Default key store for aemtux3Node01	(cell):aemtux3Node01Cell: (node):aemtux3Node01	${CONFIG_ROOT}/cells /aemtux3Node01Cell/nodes /aemtux3Node01/key.p12
☐	NodeDefaultTrustStore	Default trust store for aemtux3Node01	(cell):aemtux3Node01Cell: (node):aemtux3Node01	${CONFIG_ROOT}/cells /aemtux3Node01Cell/nodes /aemtux3Node01/trust.p12
☐	TuxServerStore		(cell):aemtux3Node01Cell: (node):aemtux3Node01	/home/richm/TuxServerStore.jks
Total 4				

Creating certificates

- Click ‘Personal certificates’ to create/view a self-signed certificate.
- Click ‘Signer certificates’ to extract/view signer certs.

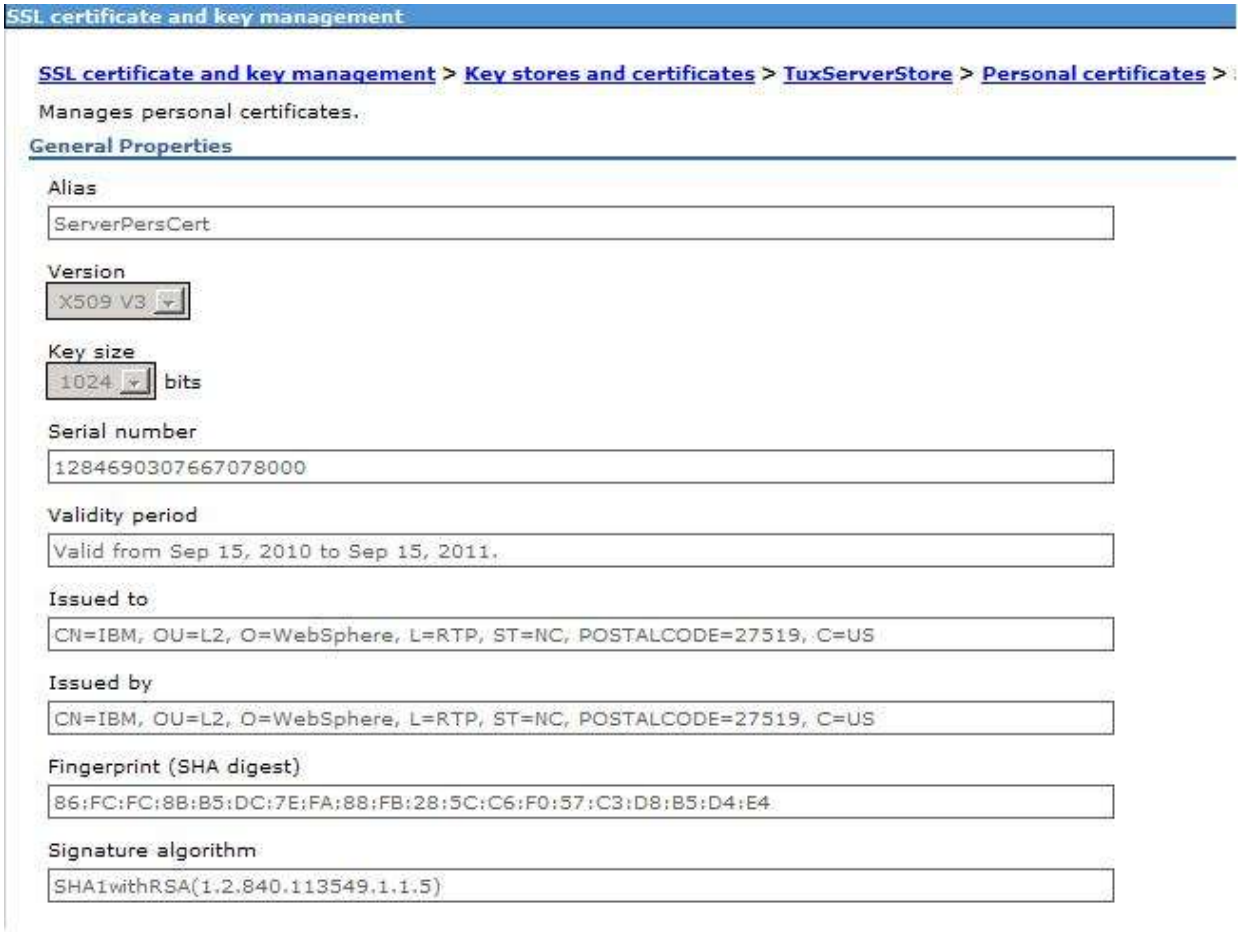


Create a self-signed certificate for the server

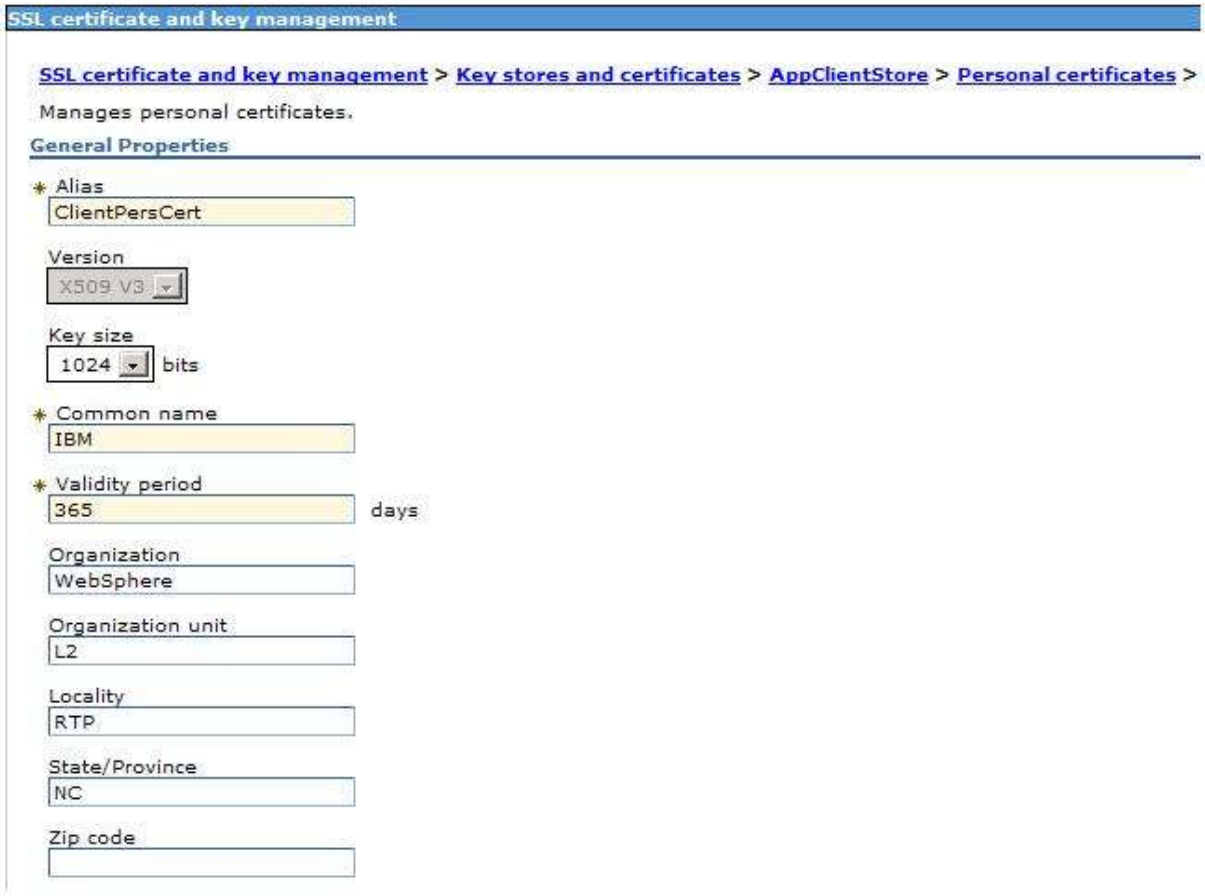


- The'Alias' is the name the certificate is known as by the keystore.
- The 'Common name' is typically the hostname where the certificate resides.

Display of server self-signed certificate

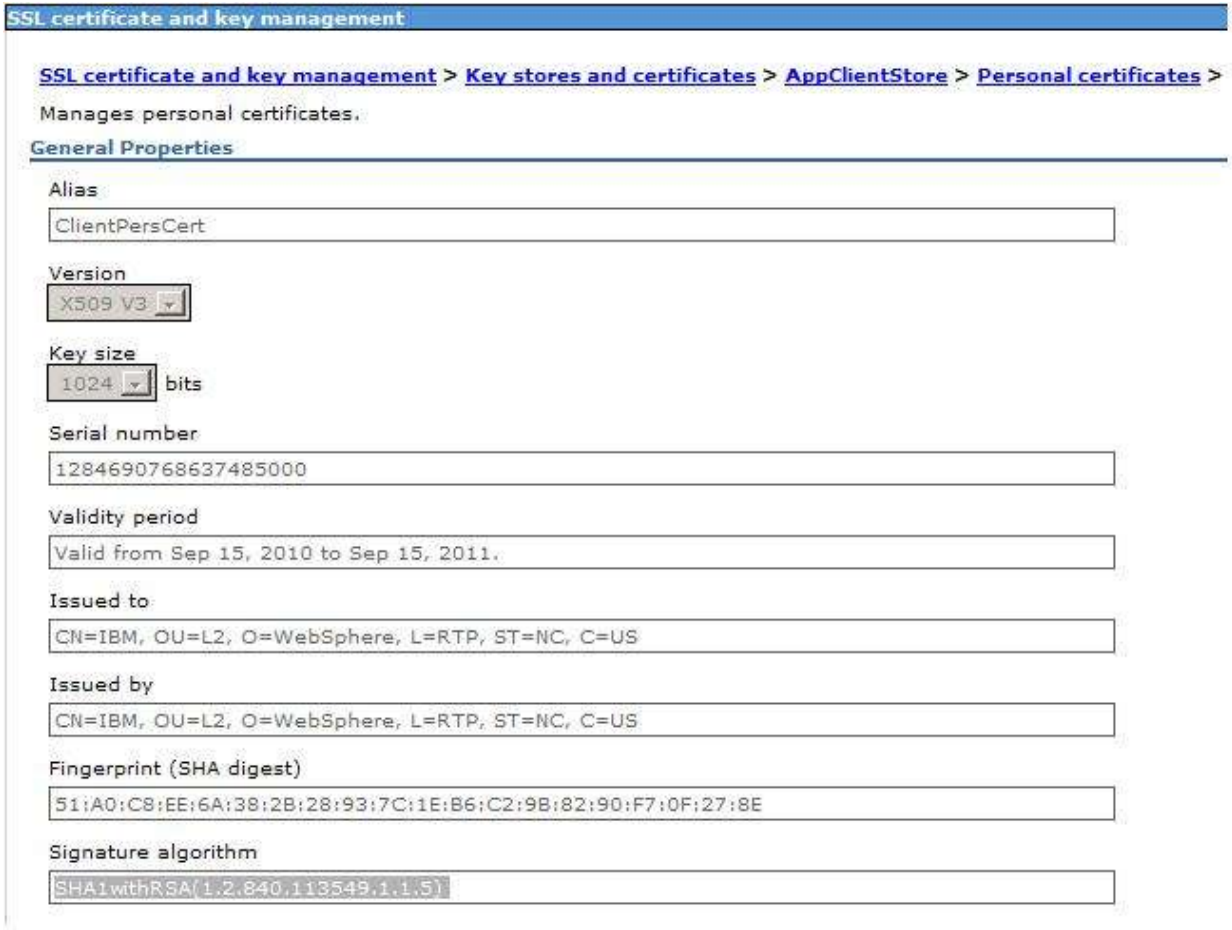


Create a self-signed certificate for the client



- The ‘Alias’ is the name the certificate is known as by the keystore.
- The ‘Common name’ is typically the hostname where the certificate resides but is not mandatory

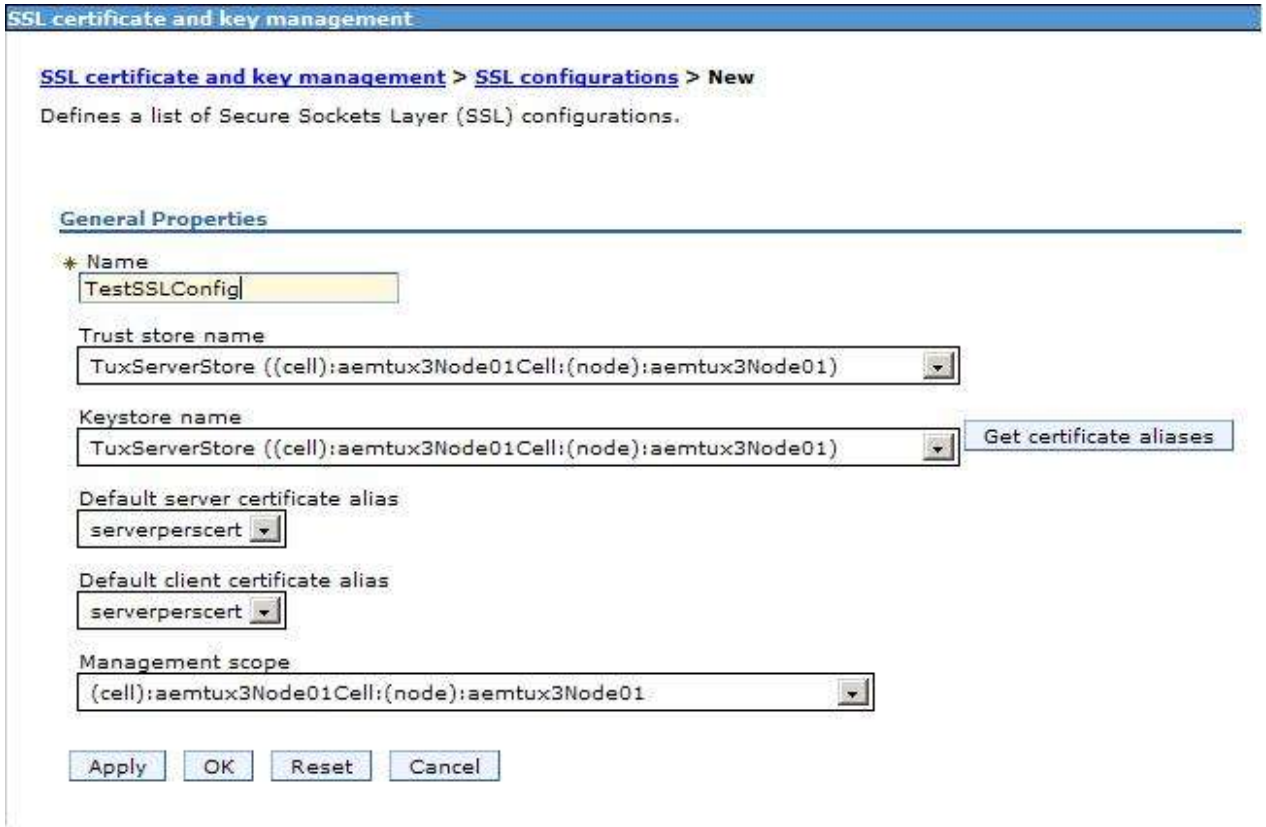
Display of client self-signed certificate



Exchange signers

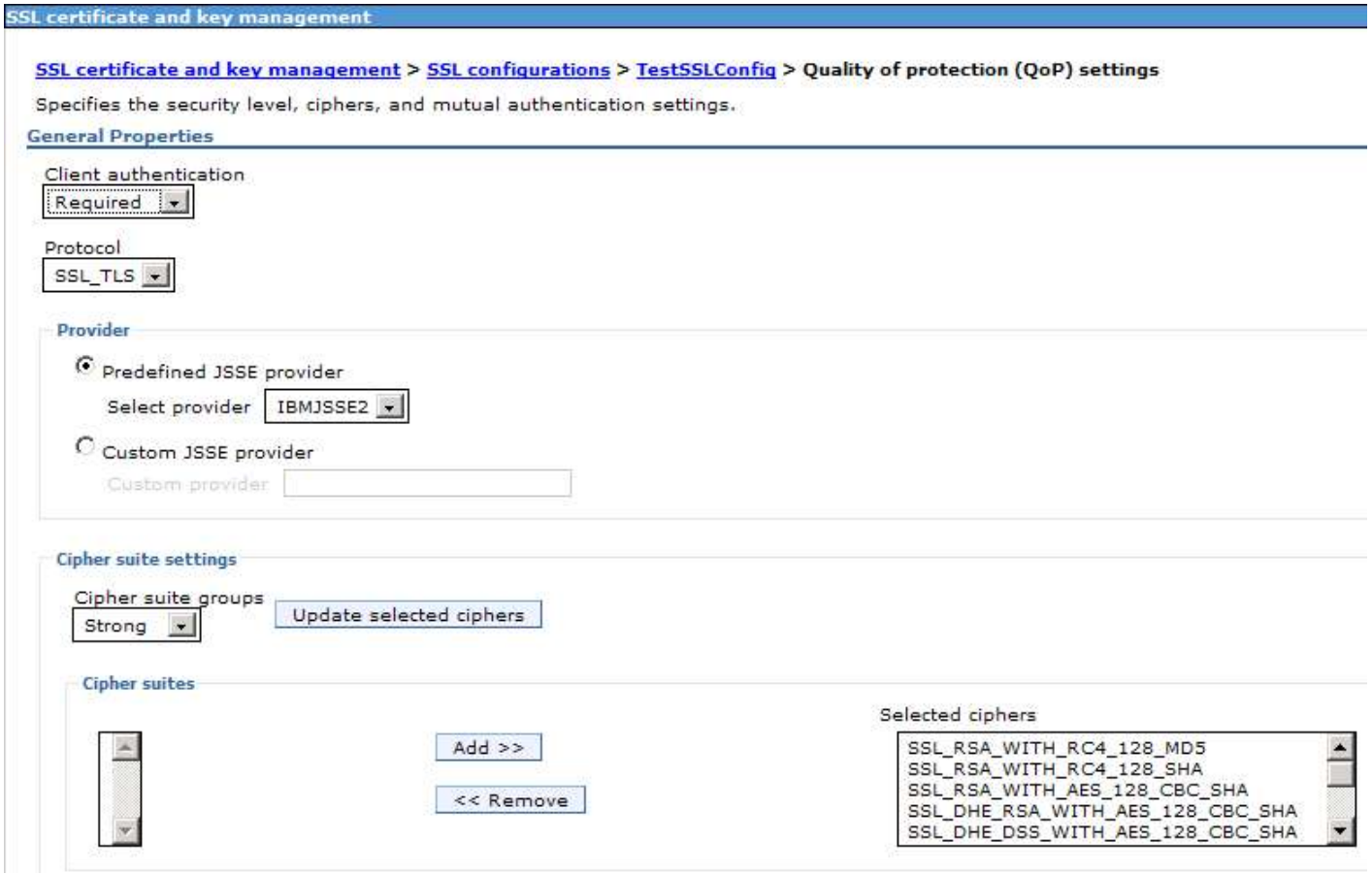
- Highlight each personal certificate and click ‘Add’ to include the signer certificates.

Create an SSL Configuration



- The SSL config points to the stores for the server.
- Once you point to the store, you then must click 'get certificate aliases' which will add the server certificate alias to the drop down.
- Click 'QoP' to enable client authentication.
- The trust store and keystore can be different store files.

Enable Client authentication



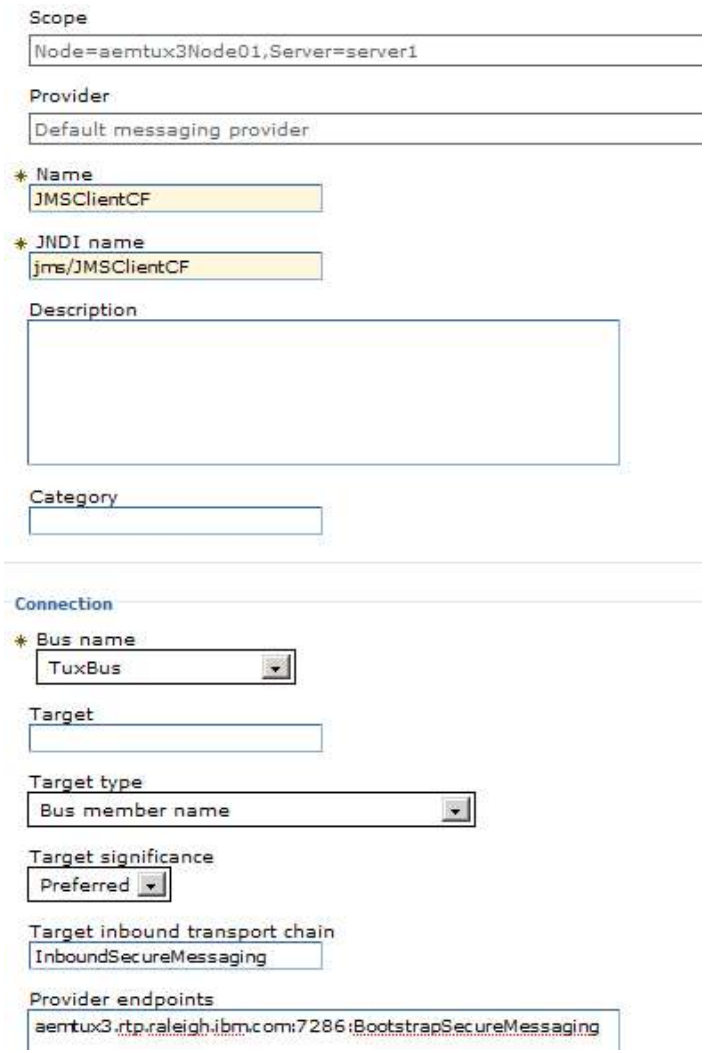
- 'Required' – Server will request cert from the client.
- 'Supported' – Server may request cert from the client.
- 'None' – Server does not request cert from the client.
- You can add or remove ciphers if you wish.

Manage endpoint security configurations

- Configure the inbound and outbound endpoints.
- Select the SSL config to use.
- ‘click’ the update certificate list.
- Choose the certificate alias in the Server store for the personal certificate.
- For the inbound and outbound connection you need to specify the SSL config to use, and at what scope.

Configured endpoints on the Server

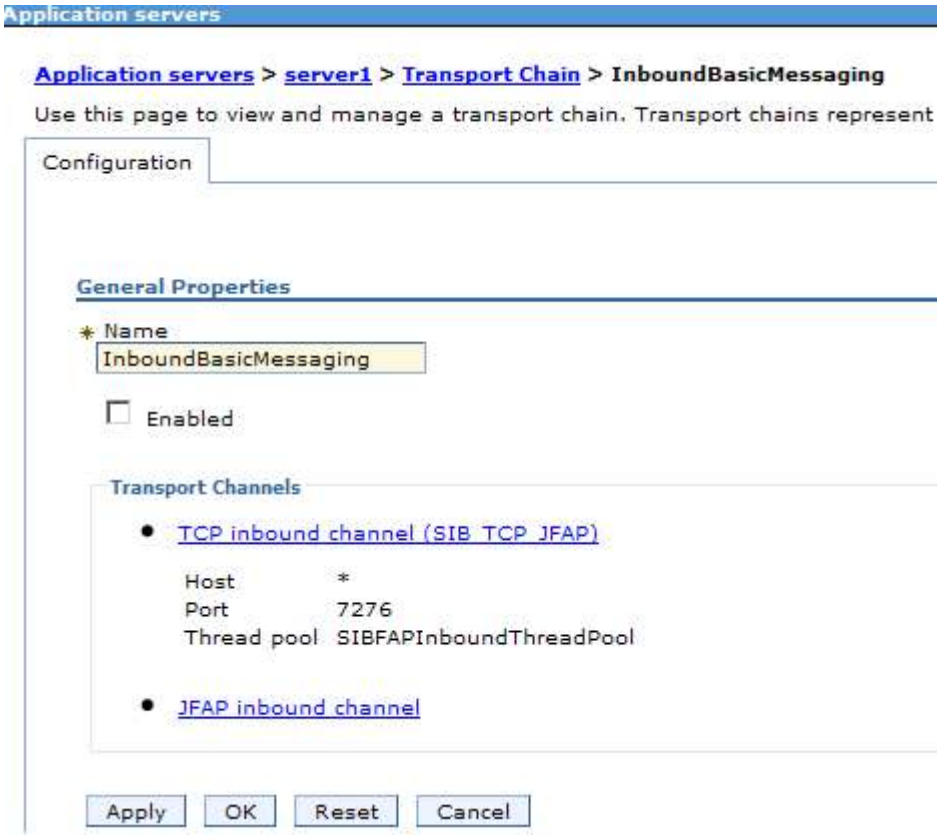
- Notice the TestSSLConfig is listed the for server1 scope now.



Provider Endpoints and Transport Chains

- The provider endpoints on the connection factory need to point to the port for the SIB_ENDPOINT_SECURE_ADDRESS and the inbound transport chain needs to point to the secure transport chain.

Disable the InboundBasicMessaging transport chain



- It is generally a good idea to disable the InboundBasicMessaging transport chain once you have SSL enabled.

Client side configuration - sib.client.ssl.properties

```
#############################################################################
#
#      (C) COPYRIGHT International Business Machines Corp. 2004, 2007
#        All Rights Reserved * Licensed Materials - Property of IBM
#
#                        SIB Client SSL Properties file
#
# This file contains properties that are used by Service Integration to
# determine client transport security settings.
#
#############################################################################

##
## Determines which SSL properties to use.
##
## Valid values are:
##    alias (default) ? use the alias referenced by the com.ibm.ssl.alias
##                      property.
##    file            - use the properties specified in this file and ignore
##                      any properties referenced by the com.ibm.ssl.alias
##                      property.
##
com.ibm.ws.sib.configurationSource=alias

##
## SSL configuration alias referenced in ssl.client.props
##
com.ibm.ssl.alias=ClientSSLConfig
```

- Edit the com,ibm.ssl.alias setting to point to the name of the SSL configuration you are using in the ssl.client.props.

Client side configuration - ssl.client.props

```
#-----------------------------------------------------------------------
# Another SSL configuration (this is a template, uncomment and modify)
# You can configure the dynamicSelectionInfo OR reference this alias
# from another protocol (e.g., soap.client.props or sas.client.props)
#-----------------------------------------------------------------------
com.ibm.ssl.alias=ClientSSLConfig
com.ibm.ssl.protocol=SSL_TLS
com.ibm.ssl.securityLevel=HIGH
com.ibm.ssl.trustManager=IbmX509
com.ibm.ssl.keyManager=IbmX509
com.ibm.ssl.contextProvider=IBMJSSE2
com.ibm.ssl.enableSignerExchangePrompt=true
com.ibm.ssl.keyStoreClientAlias=clientperscert
*com.ibm.ssl.customTrustManagers=
*com.ibm.ssl.customKeyManager=
*com.ibm.ssl.dynamicSelectionInfo=
*com.ibm.ssl.enabledCipherSuites=

# KeyStore information
com.ibm.ssl.keyStoreName=ClientStore
com.ibm.ssl.keyStore=C:\wste\aemtux3\AppClientStore.p12
com.ibm.ssl.keyStorePassword=password
com.ibm.ssl.keyStoreType=PKCS12
com.ibm.ssl.keyStoreProvider=IBMJCE
com.ibm.ssl.keyStoreFileBased=true

# TrustStore information
com.ibm.ssl.trustStoreName=ClientStore
com.ibm.ssl.trustStore=C:\wste\aemtux3\AppClientStore.p12
com.ibm.ssl.trustStorePassword=password
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
com.ibm.ssl.trustStoreReadOnly=false
```

- The com.ibm.ssl.alias specified in the sib.client.ssl.properties file points to the SSL configuration you want to use in the ssl.client.props file.
- The uncommmented entries on the left represent the modified SSL properties for the client.

Summary of steps for a secure connection

- Create keystores/truststores for the server and client.
- Create a self-signed personal certificate for each.
- Exchange the signer certificates.
- Create a new SSL config and point to the server keystore.
- Enable client authentication (for 2 way authentication).
- Point the inbound and outbound endpoints to the correct SSL server config.
- Set the proper provider endpoints on the connectionFactory to use the secure transport chain.
- Disable the InboundBasicMessaging transport chain.
- Move the client keystore/trust store to the client machine.
- Configure the sib.client.ssl.properties file to point to the proper SSL client configuration in the ssl.client.props file.
- Modify the ssl.client.props with the proper SSL properties for the client.
- Use launchClient to invoke the Application client EAR.

Connecting to SIB - launchClient

```
C:\"Program Files"\IBM\WebSphere\AppClient\bin\launchClient TestApp1.ear
   -CCproviderURL=corbaloc::aemtux3.rtp.raleigh.ibm.com:2809
   -CCtrace=SIB*=all -CCtracefile=C:\temp\trace.log
IBM WebSphere Application Server, Release 7.0
Java EE Application Client Tool
Copyright IBM Corp., 1997-2008
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the Java EE Application Client Environment.
WSCL0035I: Initialization of the Java EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class JMSClient
* Looking up CF in jndi...
* Creating queue connection...
* Queue connection created...
* Creating queue session...
* Creating queue sender to queue://TestQ?busName=TuxBus...
* Sample done
```

Tracking the message via client trace

C:\"Program Files"\IBM\WebSphere\AppClient\bin\launchClient TestApp1.ear
-CCproviderURL=corbaloc::aemtux3.rtp.raleigh.ibm.com:2809
-CCtrace=SIB*=all -CCtracefile=C:\temp\trace.log

Example entries in the client trace.log using SIB*=all

[9/18/10 23:09:21:734 EDT] 00000000 < UOW=3-46884688-26038281:rmontjoy
source=com.ibm.ws.sib.api.jms.impl.JmsSessionImpl method=createMessageID
(com.ibm.ws.sib.api.jms.impl.JmsSessionImpl) [:/2e052e05] org=IBM prod=WebSphere
component=Application Server thread=[P=749156:O=0:CT]
Exit parm0= 7d508aa6 87a0f2dc 5cc8691f 110a134f 00000000 00000001

[9/18/10 23:09:22:218 EDT] 00000000 3 UOW=3-46884688-26038281:rmontjoy
source=com.ibm.ws.sib.jfapchannel.impl.Connection org=IBM prod=WebSphere
component=Application Server thread=[P=749156:O=0:CT]
(com.ibm.ws.sib.jfapchannel.impl.Connection) [:/7e607e6] state = Connection state: OPEN,
threadsSendingData=1

Tracking the message via server trace

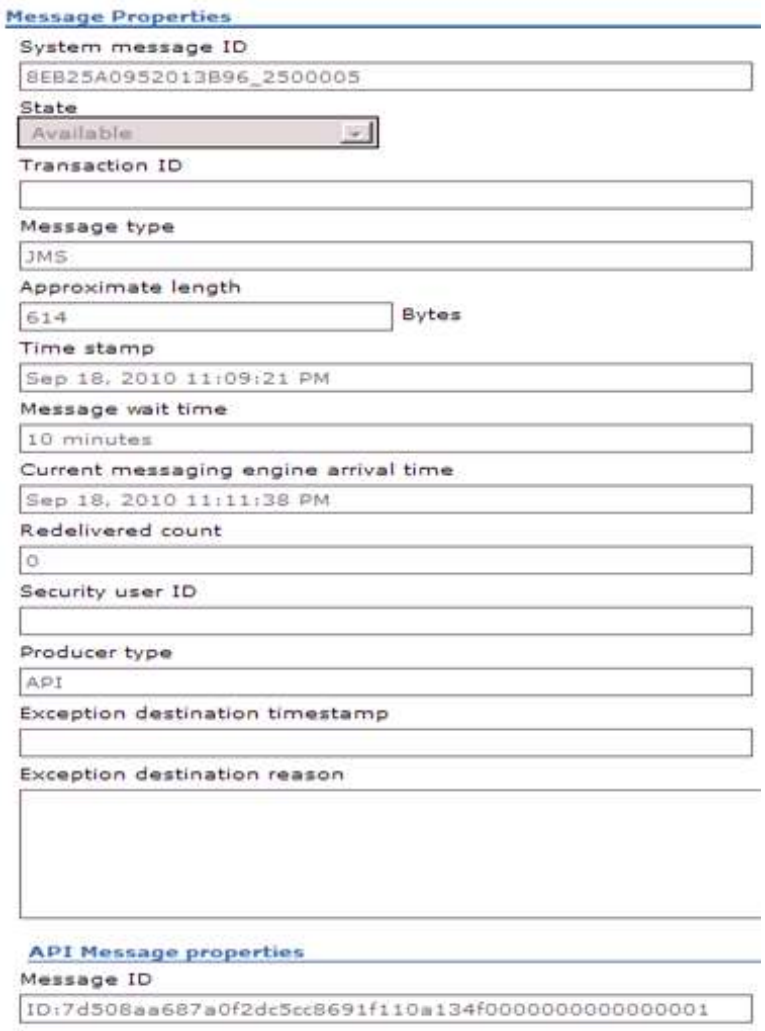
In the Administrative Console
Logging and Tracing > server1 > Diagnostic trace service >
=info: SIBMessageTrace=all

Example entries in the server trace.log using SIBMessageTrace*=all

[9/18/10 23:11:38:515 EDT] 00000020 UserTrace 3
(com.ibm.ws.sib.processor.utils.UserTrace) [:] CWSJU0004I: A message with ID
ID:7d508aa687a0f2dc5cc8691f110a134f0000000000000001, system message ID null
and correlation ID null is put to queue TestQ

[9/18/10 23:11:38:536 EDT] 00000020 UserTrace 3
(com.ibm.ws.sib.processor.utils.UserTrace) [:/41984198] CWSJU0003I: A message
with ID ID:7d508aa687a0f2dc5cc8691f110a134f0000000000000001, system
message ID 8EB25A0952013B96_2500005 and correlation ID null
is committed to destination TestQ, which is targeted for messaging engine
aemtux3Node01.server1-TuxBus.

Buses > TuxBus > Destinations > TestQ > Queue points > TestQ@aemtux3Node01.server1-TuxBus > Messages



Useful SIB and Application Client Links

- WebSphere Service Integration Bus MustGather Information
  http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg21266769
- Creating a Service Integration Bus
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tjj0002_.html
- launchClient tool
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rcli_javacmd.html

JSSE and SSL links

- IBM JSSE Reference Guide
  http://www.ibm.com/developerworks/java/jdk/security/50/secguides/jsse2Docs/JSSE2RefGuide.html
- WebSphere Application Server V7 Advanced Security Hardening
  http://www.ibm.com/developerworks/websphere/techjournal/1004_botzum/1004_botzum.html?ca=drs
- IBM WebSphere Application Server V7 Security Guide
  http://www.redbooks.ibm.com/abstracts/sg247660.html?Open
- JSSE and SSL Mustgather
  http://www-01.ibm.com/support/docview.wss?uid=swg21162961

Other Useful Links

- WebSphere Application Server V7: Deploying Applications
  http://www.redbooks.ibm.com/redpapers/pdfs/redp4583.pdf
- WebSphere Application Server V7: Messaging Administration Guide
  http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247770.html?Open
- WebSphere Application Server V7: System Management and Configuration
  http://www.redbooks.ibm.com/abstracts/sg247615.html?Open
- WebSphere Application Server Information Center
  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp
- WebSphere Application Server Main Support
  http://www.ibm.com/software/webservers/appserv/was/support

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

We Want to Hear From You!

Tell us about what you want to learn

Suggestions for future topics
Improvements and comments about our webcasts
We want to hear everything you have to say!

Please send your suggestions and comments to:
wsehelp@us.ibm.com

Questions and Answers