JMS Provider Configuration

This topic provides a reference for configuring

SOAtest

for popular JMS providers.

Sections include:

Adding Required jar Files

To add the required jar files (listed in the tables below) to the

SOAtest

classpath, complete the following:

1. Choose **Parasoft> Preferences**.
2. Open the **Parasoft> System Properties** page.
3. Click the **Add JARS** button and choose and select the necessary JAR files to be added.

JNDI Authentication

Any time that a username and password are required to connect to a JMS system that is configured to require authentication for JNDI access, the following JNDI properties must be configured:

- java.naming.security.principal=<username>
- java.naming.security.credentials=<password>

Apache ActiveMQ

For	For Apache Active MQ
Minimum Required jars	Found under the ActiveMQ installation directory - activemq-all-[version].jar
Typical Provider URL Pattern	tcp://hostname:61616
Factory class	JNDI Initial Context factory class - org.apache.activemq.jndi.ActiveMQInitialContextFactory Default connection factory JNDI name - ConnectionFactory
Learn more	http://activemq.apache.org/jndi-support.html

Apache Qpid

For	For Apache Qpid
Minimum Required jars	Found in qpid-java-client-{ver}.tar.gz from http://qpid.apache.org/download.html. The jars vary depending on the version of Qpid. The jar for Qpid 0.8 includes the following: • backport-util-concurrent-2.2.jar • geronimo-jms_1.1_spec-1.0.jar • mina-core-1.0.1.jar • mina-filter-ssl-1.0.1.jar • qpid-all.jar • qpid-client-0.8.jar • qpid-common-0.8.jar • slf4j-api-1.6.1.jar
Connection URL JNDI property	connectionfactory.<jndiname> For example: connectionfactory.local = amqp://user:password@clientid/testpath?brokerlist='tcp://localhost:5672'
Factory class	JNDI Initial Context factory class - org.apache.qpid.jndi.PropertiesFileInitialContextFactory
Learn more	https://cwiki.apache.org/qpid/how-to-use-jndi.html

GlassFish MQ

For	For GlassFish
Minimum Required jars	GlassFish 3 - Found under [GlassFish install directory]/glassfish/lib: - gf-client.jar Note that the gf-client.jar references a few dozen other jars from the GlassFish 3 installation. You must reference gf-client.jar from the GlassFish installation directory so that the referenced jars can be found and loaded. GlassFish can be downloaded from http://glassfish.java.net/. GlassFish 2 - Found under [GlassFish install directory]/lib: - appserv-admin.jar - appserv-deployment-client.jar - appserv-ext.jar - appserv-rt.jar - javaee.jar - install/applications/jmsra/imqjmsra.jar
Typical Provider URL Pattern	iiop://yourhostname:3700
Factory class	JNDI Initial Context factory class - com.sun.enterprise.naming.SerialInitContextFactory Note that GlassFish has another JNDI Initial Context factory class called "com.sun.appserv.naming.S1ASCtxFactory". This class has poor performance and should not be used. Use the mentioned SerialInitContextFactory class instead which is available in both GlassFish 2 and GlassFish 3.
Notes	The GlassFish client and GlassFish server perform a reverse DNS lookup on each other. If the server does not recognize the client's host name, then you can add the client's host name and IP address to the hosts file on the server. If the client does not recognize the server's host name, then you can add the server's host name and IP address to the hosts file on the client. The hosts file is typically /etc/hosts on Unix or %SystemRoot%\system32\drivers\etc\hosts on Windows.
Learn more	http://glassfish.java.net/

IBM WebSphere Application Server (WAS)

For	For the WAS Default JMS provider. Parasoft recommends the use of IBM's JMS thin client that is provided by WAS 7.0 or later, and which can interoperate with WAS 6.0.2 and later.

Minimum Required jars	Found under [WAS installation dir]/runtimes - com.ibm.ws.ejb.thinclient_7.0.0.jar - com.ibm.ws.orb_7.0.0.jar - com.ibm.ws.sib.client.thin.jms_7.0.0.jar
Typical Provider URL Pattern	iiop://yourhostname:2809/
Factory class	JNDI Initial Context factory class - com.ibm.websphere.naming.WsnInitialContextFactory
Learn more	WebSphere Application Server Network Deployment, Version 7.0 documentation

What if you don't have WAS 7.0 or later?

If you aren't using (or don't have access to) a WAS 7 installation, download and install the IBM Client for JMS, which also works with WAS 6.0.2 and later:

1. Download the JMS Client installer jar from http://www-01.ibm.com/support/docview.wss?uid=swg24012804
2. Follow the instructions on the download page to install both the JMS and JNDI jars for the Sun JRE. The command to install the client jars is similar to:
   `"java -jar sibc_oeminst-o0902.06.jar jms_jndi_sun C:\ibmjms"`
   The client jars will be installed under the `lib` folder.
3. Add the JNDI property `com.ibm.CORBA.ORBInit=com.ibm.ws.sib.client.ORB`
   - For simplicity, this jndi property can be set globally in a jndi.properties file in the JRE's lib folder.
   - For the standalone build of this product, the JRE lib folder is under Parasoft/Test/{ver}/plugins/com.parasoft.xtest.jdk.eclipse.core.{platform}.{arch}_{ver}/jdk/jre/lib.
   - This JNDI property is not needed when using the jars from the WAS 7 runtimes folder.

IBM WebSphere MQ (MQ Series)

For	For the WebSphere MQ JMS provider
Minimum Required jars	**For MQ 6.0** *You can find these under [WebSphere MQ installation directory]/java/lib:* - com.ibm.mq.jar - com.ibm.mqjms.jar - connector.jar - dhbcore.jar - jta.jar *You need to download these:* - MS0B: WebSphere MQ Java classes for PCF — com.ibm.mq.pcf.jar - ME01: WebSphere MQ - Initial Context Factory — mqcontext.jar **For MQ 7.0** *You can find these under [WebSphere MQ installation directory]/java/lib:* - com.ibm.mq.jar - com.ibm.mqjms.jar - com.ibm.mq.commonservices.jar - com.ibm.mq.headers.jar - com.ibm.mq.jmqi.jar - connector.jar - dhbcore.jar - com.ibm.mq.pcf.jar *You need to download these:* - ME01: WebSphere MQ - Initial Context Factory — mqcontext.jar **MQ Client Download** The jars from *[WebSphere MQ installation directory]/java/lib* are also included in the MQ Client. The MQ 6 Client is no longer available from IBM. The MQ 7 Client is available at http://www-01.ibm.com/support/docview.wss?uid=swg24019253
Typical Provider URL Pattern	yourhostname:1414/SYSTEM.DEF.SVRCONN
Factory class	WebSphere MQ JNDI Initial Context factory class - com.ibm.mq.jms.context.WMQInitialContextFactory

SSL Configuration	WebSphere JMS clients can achieve SSL connections by setting the appropriate properties in the initial context. Both WebSphere MQ and WebSphere JMS clients will set the same properties for SSL connectivity.

JMS messaging without JNDI	Under the properties tab, define the following properties:
	Name: Value
	provider: WebSphere MQ
	queue.manager: [your queue manager name]
	channel: [your channel name]
	port: [port number, the default WebSphere MQ port is 1414]
	With this configuration, SOAtest/Virtualize will create the connection factory using the provided parameters as follows:
	`import com.ibm.mq.jms.*;`
	`...`
	`MQConnectionFactory cf = new MQConnectionFactory();`
	`cf.setHostName(hostname);`
	`cf.setPort(port);`
	`cf.setQueueManager(queuemanager);`
	`cf.setChannel(channel);`
	`cf.setTransportType(JMSC.MQJMS_TP_CLIENT_MQ_TCPIP);`
	`cf.setFailIfQuiesce(JMSC.MQJMS_FIQ_YES);`
	`cf.setUseConnectionPooling(true);`

Additional Notes	IBM's JNDI provider authenticates itself with the Websphere MQ server by sending the user's login name. This is typically the user name that was provided when logging into the workstation before starting SOAtest/Virtualize. If the MQ server does not recognize the user name, then your tool will fail with the error "javax.jms.JMSSecurityException: MQJMS2013: invalid security authentication supplied for MQQueueManager". This error can be resolved by adding a Windows user account on the Websphere MQ server machine. This account should 1) have the same user name as the account on the local machine where SOAtest is running and 2) be a member of the "mqm" IBM WebSphere MQ Administration Group. Alternatively, use a different user name by changing the value of the user.name Java System property. This system property can be configured by starting SOAtest using `soatest.exe` `-J-Duser.name=username` where `username` is the user name you want to use. In some configurations using an empty username via `soatest/virtualize.exe -J-Duser.name=` will work. Using `SYSTEM` for the user.name property may also work in some configurations. It is also possible to modify Java system properties during test suite execution by using an Extension Tool to call the java.lang.System.setProperty() method from Sun's Java API. For more details on this error, see http://www.mqseries.net/phpBB/view-topic.php?t=40640.
Learn more at	http://www-01.ibm.com/software/integration/wmq/library/ http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzaw.doc/jm10320_.htm

JBoss JMS

For	For JBoss JMS; the following jar list is based on JBoss 5.0

Minimum Required jars	Found under [JBoss install dir]/client - jboss-javaee.jar - jnp-client.jar - jboss-logging-spi.jar - jboss-messaging-client.jar - jboss-aop-client.jar - jboss-remoting.jar - jboss-common-core.jar - trove.jar - javassist.jar - jboss-mdr.jar - concurrent.jar - log4j.jar - jboss-serialization.jar
Typical Provider URL Pattern	yourhostname
Factory class	JNDI Initial Context factory class - org.jnp.interfaces.NamingContextFactory

Open Message Queue (OpenMQ)

Note that OpenMQ can run by itself or as part of Glassfish App Server, where it is called "Glassfish MQ". If you are using OpenMQ from Glassfish, see the GlassFish section.

For	For OpenMQ Server
Minimum Required jars	Found under [OpenMQ install dir]/mq/lib - fscontext.jar - imq.jar - jms.jar
Typical Provider URL Pattern	yourhostname
Factory class	JNDI Initial Context factory class - com.sun.jndi.fscontext.RefFSContextFactory
Learn more	http://mq.java.net

Oracle Advanced Queuing (AQ)

For	For Oracle Advanced Queuing (AQ)
Minimum required jars	From Oracle Database installation: - server/rdbms/jlib/aqapi.jar - server/jdbc/lib/ojdbc6.jar or From Oracle Weblogic Server installation: - server/lib/aqapi.jar - server/lib/ojdbc6.jar
Provider URL	Leave this field empty.
JNDI initial context factory class	oracle.jms.AQjmsInitialContextFactory
Connection factory JNDI name	One of the following: - QueueConnectionFactory - TopicConnectionFactory - ConnectionFactory - XAQueueConnectionFactory - XATopicConnectionFactory - XAConnectionFactory
Required JNDI properties	java.naming.security.principal=<username> java.naming.security.credentials=<password> db_url=jdbc:oracle:thin:@host:port:dbName

Queue/Topic names	Prefix a queue name with `Queues/` and a topic name with `Topics/`. For example:
	`Queue: Queues/<queue_name>`
	`Topic: Topics/<topic_name>`

Notes	The aqapi.jar from older version of Oracle Database, such as 10g, may be missing the Initial Context factory class. Use the jars from Oracle Database 11g or later. The latest Oracle Database Express Edition (XE) can be downloaded from oracle.com: Oracle Database Express Edition 11g Release 2
Learn more	Interoperating with Oracle AQ JMS

Oracle BEA WebLogic

For	For Oracle BEA WebLogic
Minimum Required jars (Thin Client) *	Found under [weblogic home]/wlserver_x/server/lib - wljmsclient.jar - wlclient.jar Note that additional jars may be needed. We recommend using a full client; this relieves you from having to find and add many jars.
Minimum Required jars (Full Client) *	Build the single wlfullclient5.jar as described in the instructions for "Creating a wlfullclient5.jar for JDK 1.5 client applications."
Typical Provider URL Pattern	**Thin Client:** iiop://yourhostname:7001 **Full Client:** t3://yourhostname:7001
Factory class	JNDI Initial Context factory class: - weblogic.jndi.WLInitialContextFactory
Learn more	http://docs.oracle.com/cd/E12840_01/wls/docs103/client/jarbuilder.html

Progress Sonic MQ/ESB

For	For Progress Sonic MQ/ESB
Minimum Required jars	Found under [sonic install dir]/MQ7.x/lib - mfcontext.jar - broker.jar - sonic_Client.jar Note that SonicMQ's broker.jar and TIBCO's tibcojms.jar cannot be used together.
Typical Provider URL Pattern	tcp://yourhostname:2506
Factory class	JNDI Initial Context factory class - com.sonicsw.jndi.mfcontext.MFContextFactory
Learn more	Refer to SonicMQ Application Programming Guide, Appendix A (Using the Sonic JNDI SPI) and which also refers to other relevant sections.

RabbitMQ

For	For RabbitMQ
Minimum Required jars	Download client library from Maven's Central Repository. rabbitmq-jms-<version>.jar

.bindings File example	#This file is used by the JNDI FSContext. #Wed Jan 31 15:36:15 PST 2018 ConnectionFactory/FactoryName=com.rabbitmq.jms.admin.RMQObjectFactory ConnectionFactory/ClassName=javax.jms.ConnectionFactory ConnectionFactory/RefAddr/0/Type=name ConnectionFactory/RefAddr/0/Encoding=String ConnectionFactory/RefAddr/0/Content=jms/ConnectionFactory ConnectionFactory/RefAddr/1/Type=type ConnectionFactory/RefAddr/1/Encoding=String ConnectionFactory/RefAddr/1/Content=javax.jms.ConnectionFactory ConnectionFactory/RefAddr/2/Type=factory ConnectionFactory/RefAddr/2/Encoding=String ConnectionFactory/RefAddr/2/Content=com.rabbitmq.jms.admin.RMQObjectFactory ConnectionFactory/RefAddr/3/Type=host ConnectionFactory/RefAddr/3/Encoding=String ConnectionFactory/RefAddr/3/Content=host.company.com ConnectionFactory/RefAddr/4/Type=port ConnectionFactory/RefAddr/4/Encoding=String ConnectionFactory/RefAddr/4/Content=5672
Typical Provider URL Pattern	The directory path containing the .bindings file file://C:/JNDI/rabbitMQ/
Initial Context	com.sun.jndi.fscontext.RefFSContextFactory
Connection Factory	JNDI Initial Context factory class - ConnectionFactory

Solace JMS

For	For Solace JMS
Minimum Required jars	commons-lang-<version>.jar sol-common-<version>.jar sol-jcsmp-<version>.jar sol-jms-<version>.jar
Typical Provider URL Pattern	smf://<host>:<port>
Factory class	JNDI Initial Context factory class - com.solacesystems.jndi.SolJNDIInitialContextFactory There is no default name; the specific JNDI name is needed to look up the connection factory. The additional JNDI property `Solace_JMS_VPN=myvpn` must be set in order to gain access to look up the connection factory. There is no default vpn name.

Sun Java System Message Queue (Sun MQ)

For	For Sun MQ Server
Minimum Required jars	Found under [Sun MQ install dir]Sun/MessageQueue/mq/lib - fscontext.jar - imq.jar - jms.jar
Typical Provider URL Pattern	yourhostname
Factory class	JNDI Initial Context factory class - com.sun.jndi.fscontext.RefFSContextFactory
Learn more	See the Sun Java System Message Queue Administration 3.x/4.x Guide, Section Quick-Start Tutorial. This document can be downloaded from http://docs.sun.com/app/docs/prod/message?l=en#hic. In the document, navigate to **Software> Application & Integration Services> Message Queue**.

TIBCO EMS

For	For TIBCO EMS
Minimum Required jars	Found under [TIBCO install dir]/ems/clients/java - tibjms.jar - tibcrypt.jar (required for SSL) Note that SonicMQ's broker.jar and TIBCO's tibcojms.jar cannot be used together.
Typical Provider URL Pattern	tcp://yourhostname:7222 yourhostname (if using default port numbers)
Factory class	JNDI Initial Context factory class - com.tibco.tibjms.naming.TibjmsInitialContextFactory
SSL Configuration	The following JNDI properties must be configured: - com.tibco.tibjms.naming.security_protocol=ssl - com.tibco.tibjms.naming.ssl_enable_verify_host=false - com.tibco.tibjms.naming.ssl_enable_verify_hostname=false For two-way SSL, the following additional properties must also be configured: - com.tibco.tibjms.naming.ssl_identity=<path to client keystore> - com.tibco.tibjms.naming.ssl_password=<keystore password> - com.tibco.tibjms.naming.ssl_trusted_certs=<path to trusted certificate> - com.tibco.tibjms.naming.ssl_vendor=j2se - com.tibco.tibjms.naming.ssl_auth_only=true/false
Notes	Any time that a username and password is needed to connect to Tibco EMS the following JNDI properties must be configured: - java.naming.security.principal=<username> - java.naming.security.credentials=<password>
Learn more	Refer to the TIBCO Enterprise Message Service User's Guide, section 9: Developing an EMS Client Application> Programmer Checklist

Sun JMS

SOAtest and Virtualize bundle a Sun JNDI implementation that stores JNDI bindings in directories and files on the local hard drive using com.sun.jndi.fscontext.RefFSContextFactory as the Initial Context and a user defined Provider URL (directory) of "`C:\JNDIRoot`" or something similar.

When using the Sun implementation, you must populate the "`C:\JNDIRoot`" directory with a .binding file so that `fscontext` can successfully look up the ConnectionFactory and Queue or Topic objects. Parasoft has put together an example that can be found in the SOAtest/Virtualize installation directory (/examples/jms/JndiFileProviderTest.java), which creates a .binding file.

Other JMS Providers

For other JMS providers, please refer to your vendor guides on how to configure a JMS client to communicate with your system.