Setting Up WSO2 Enterprise Integrator Samples

This section describes the prerequisites and instructions on how to start WSO2 Enterprise Integrator with the sample configurations, how to start the Axis2 server and deploy the sample back-end services to the Axis2 server, as well as how to set up any listeners and transports that are required by the samples.

Once you have set up the samples, you can run them using the sample clients.

Prerequisites

1. Ensure that you have the following in your environment:
   - Oracle JDK 1.7.*/1.8.*.
   - JAVA_HOME environment variable is set to <JDK_HOME>.
   - Apache Ant 1.7.0 or above.
   - ActiveMQ or any other JMS provider to run the JMS samples.
2. Open a command prompt (or a shell in Linux) and go to the `bin` directory of the distribution. Then run the `ant` command to build the `build.xml` file.
3. Run WSO2 Enterprise Integrator in the DEBUG mode. To switch from the default INFO log messages to DEBUG log messages, edit the `<EI_HOME>/conf/log4j.properties` file and change the line `log4j.category.org.apache.synapse=INFO` to `log4j.category.org.apache.synapse=DEBUG`.

Understanding the samples

The samples use several sample clients, configurations and sample back-end services in order to explain different use cases. The diagram below depicts the interaction between the sample clients, WSO2 Enterprise Integrator and the services at a high level. The clients are able to send SOAP/REST or POX messages over transports such as HTTP/HTTPS or JMS with WS-Addressing, or WS-Security. They can send binary optimized content using MTOM, SwA, binary or plain text JMS messages. After mediation through WSO2 Enterprise Integrator, the requests are passed over to sample services.

Starting WSO2 Enterprise Integrator with a sample configuration

To start the WSO2 Enterprise Integrator with a selected sample configuration

1. Open a command prompt (or a shell in Linux) and go to the `<EI_HOME>\bin` directory.
2. Execute one of the following commands, where <n> denotes the number assigned to the sample.
    - On Windows: `wso2ei-samples.bat -sn <n>`
    - On Linux/Solaris: `./wso2ei-samples.sh -sn <n>`

For example, to start the WSO2 Enterprise Integrator with the Sample 0 configuration on Linux/Solaris, run the following command:

```
./wso2ei-samples.sh -sn 0
```

The `<EI_HOME>/repository/samples` directory contains all the configuration files of the samples as `synapse_sample_<n>.xml` files. Here again, <n> denotes the sample number of the sample you are running.

For example, the configuration file for Sample 0 is `synapse_sample_0.xml`.

Deploying sample back-end services

The sample back-end services come with a pre-configured Axis2 server. These sample services demonstrate in-only and in-out SOAP/REST or POX messaging over HTTP/HTTPS and JMS transports, using WS-Addressing and WS-Security. The samples handle binary content using MTOM and SwA.

Each back-end sample service can be found in a separate folder in the `<EI_HOME>/samples/axis2Server/src` directory. You need to compile, build and deploy each back-end service to the Axis2 server.

To build and deploy a back-end service

1. Open a command prompt (or a shell in Linux) and go to the required sample folder in the `<EI_HOME>/samples/axis2Server/src` directory.
2. Run `ant` from the selected sample directory.

    For example, to build and deploy **SimpleStockQuoteService**, run the `ant` command from the `<EI_HOME>/samples/axis2Server/src/SimpleStockQuoteService` directory, as follows:

```
user@host:/tmp/wso2ei-6.0.0/samples/axis2Server/src/SimpleStockQuoteService$ ant
Buildfile: build.xml
 ...
build-service:
   ....
      [jar] Building jar:
/tmp/wso2ei-6.0.0/samples/axis2Server/repository/services/SimpleStockQuoteService
.aar

BUILD SUCCESSFUL
Total time: 1 second
```

Sample back-end services explained

1. ***`SimpleStockQuoteService`***

    This service has four operations:
    - `getQuote` (in-out) - Generates a sample stock quote for a given symbol.
    - `getFullQuote` (in-out) - Generates a history of stock quotes for a symbol for a number of days.
    - `getMarketActivity` (in-out) - Returns stock quotes for a list of given symbols.

- `placeOrder` (in-only) - Accepts a one way message for an order.

2. ***`SecureStockQuoteService`***

   This service is a clone of the `SimpleStockQuoteService`, but has WS-Security enabled as well as an attached security policy for signing and encrypting messages.

3. ***`MTOMSwASampleService`***

   This service has three operations:
   - `uploadFileUsingMTOM` (in-out) - Accepts a binary image from the SOAP request as MTOM, and returns this image back again as the response.
   - `uploadFileUsingSwA` (in-out) - Accepts a binary image from the SOAP request as SwA, and returns this image back again as the response.
   - `oneWayUploadUsingMTOM` (in-only) - Saves the request message to the disk.

   This service also demonstrates the use of MTOM and SwA.

Starting the Axis2 server

For the samples, a standalone Apache Axis2 Web services engine is used as the back-end server, which is bundled with the WSO2 Enterprise Integrator distribution by default.

Once each back-end service is deployed to the Axis2 server, you need to start the Axis2 server before executing the sample client.

To start the Axis2 server

1. Open a command prompt (or a shell in Linux) and go to the `<EI_HOME>/samples/axis2Server` directory.
2. Execute one of the following commands
   - On Windows: `axis2server.bat`
   - On Linux/Solaris: `./axis2server.sh`

This starts the Axis2 server with the HTTP transport listener on port 9000 and HTTPS on port 9002 respectively.

If you want to start **multiple instances** of the Axis2 server on different ports, run the following commands. Give unique names to each server.

```
./axis2server.sh -http 9001 -https 9005 -name MyServer1
./axis2server.sh -http 9002 -https 9006 -name MyServer2
./axis2server.sh -http 9003 -https 9007 -name MyServer3
```

Running the commands as specified above will start three instances of the Axis2 server on HTTP ports 9001, 9002 and 9003 respectively.

Configuring WSO2 Enterprise Integrator to use the JMS transport

To run the JMS samples, you need to configure the JMS transport of WSO2 Enterprise Integrator with ActiveMQ 5.5.0 or higher. For instructions on configuring WSO2 Enterprise Integrator with ActiveMQ, see Configure with ActiveMQ.

Configuring WSO2 Enterprise Integrator to use the mail transport

To enable the mail transport sender for the samples

- Uncomment the mail transport sender configuration in the <EI_HOME>/conf/axis2/axis2.xml file and make sure it points to a valid SMTP configuration for an actual scenario.

```
<transportSender name="mailto"
class="org.apache.synapse.transport.mail.MailTransportSender">
    <parameter name="mail.smtp.host">smtp.gmail.com</parameter>
    <parameter name="mail.smtp.port">587</parameter>
    <parameter name="mail.smtp.starttls.enable">true</parameter>
    <parameter name="mail.smtp.auth">true</parameter>
    <parameter name="mail.smtp.user">synapse.demo.0</parameter>
    <parameter name="mail.smtp.password">mailpassword</parameter>
    <parameter name="mail.smtp.from">synapse.demo.0@gmail.com</parameter>
</transportSender>
```

To enable the mail transport receiver for the samples

- Uncomment the mail transport receiver configuration in the <EI_HOME>/conf/axis2/axis2.xml file.

```
<transportReceiver name="mailto"
class="org.apache.axis2.transport.mail.MailTransportListener">
</transportReceiver>
```

> You need to provide correct parameters for a valid mail account at the service level.

Configuring WSO2 Enterprise Integrator to use the NHTTP transport

WSO2 Enterprise Integrator uses the HTTP PassThrough Transport (PTT) as the default HTTP transport. Therefore, the default axis2.xml file in <EI_HOME>/conf/axis2 is PassThrough Transport enabled.

To run some of the samples, you need to configure WSO2 Enterprise Integrator to use the NHTTP transport as the default transport.

To enable the NHTTP transport

1. Rename the axis2.xml file in the <EI_HOME>/conf/axis2 directory to default_axis2.xml.
2. Rename the axis2_nhttp.xml file in the <EI_HOME>/conf/axis2 directory to axis2.xml.
3. Restart WSO2 Enterprise Integrator.

Configuring WSO2 Enterprise Integrator to use the FIX transport

To run the FIX samples, you need to have a local Quickfix/J installation. To download the latest version of Quickfix/J, go to http://www.quickfixj.org/downloads

To enable the FIX transport sender and receiver

- Uncomment the FIX transport sender and FIX transport receiver configurations in the <EI_HOME>/conf/axis2/axis2.xml file.

Configuring WSO2 Enterprise Integrator for FIX samples

In order to configure WSO2 Enterprise Integrator to run the FIX samples, you need to create the FIX configuration

files.

> You can find the config files in `<EI_HOME>/repository/samples/resources/fix` folder.

To create the FIX configuration files

1. Add the following entries in a file named `fix-synapse.cfg`

```
[default]
FileStorePath=repository/logs/fix/store
FileLogPath=repository/logs/fix/log
ConnectionType=acceptor
StartTime=00:00:00
EndTime=00:00:00
HeartBtInt=30
ValidOrderTypes=1,2,F
SenderCompID=SYNAPSE
TargetCompID=BANZAI
UseDataDictionary=Y
DefaultMarketPrice=12.30

 [session]
BeginString=FIX.4.0
SocketAcceptPort=9876
```

2. Add the following entries in a file named `synapse-sender.cfg`

```
[default]
FileStorePath=repository/logs/fix/data
FileLogPath=repository/logs/fix/log
SocketConnectHost=localhost
StartTime=00:00:00
EndTime=00:00:00
HeartBtInt=30
ReconnectInterval=5
SenderCompID=SYNAPSE
TargetCompID=EXEC
ConnectionType=initiator

 [session]
BeginString=FIX.4.0
SocketConnectPort=19876
```

The `FileStorePath` property in the `fix-synapse.cfg file and synapse-sender.cfg` file should point to two different directories in your local file system. Once the samples are executed, Synapse will create FIX message stores in these two directories.

Configuring sample FIX applications

If you use a binary distribution of Quickfix/J, the two samples and their configuration files are all packed to a single JAR file called `quickfixj-examples.jar`.

To configure sample FIX applications

1. Either modify the `quickfixj-examples.jar` file or pass the new configuration file as a command line

parameter.

To modify the `quickfixj-examples.jar` file:

- Extract the JAR file, modify the configuration files and pack them to a JAR file with the same name again.

To pass the new configuration file as a command line parameter:

- Copy the `config` files from the `<EI_HOME>/repository/samples/resources/fix` directory to the `<QFJ_HOME>/etc` directory. Then execute the sample apps from `<QFJ_HOME>/bin`, `./banzai.sh/bat ../etc/banzai.cfg executor.sh/bat ../etc/executor.cfg`.

2. Locate and edit the FIX configuration file of Executor to be as follows. This file is usually named `executor.cfg`

```
[default]
FileStorePath=examples/target/data/executor
ConnectionType=acceptor
StartTime=00:00:00
EndTime=00:00:00
HeartBtInt=30
ValidOrderTypes=1,2,F
SenderCompID=EXEC
TargetCompID=SYNAPSE
UseDataDictionary=Y
DefaultMarketPrice=12.30

[session]
BeginString=FIX.4.0
SocketAcceptPort=19876
```

3. Locate and edit the FIX configuration file of Banzai to be as follows. This file is usually named `banzai.cfg`

```
[default]
FileStorePath=examples/target/data/banzai
ConnectionType=initiator
SenderCompID=BANZAI
TargetCompID=SYNAPSE
SocketConnectHost=localhost
StartTime=00:00:00
EndTime=00:00:00
HeartBtInt=30
ReconnectInterval=5

[session]
BeginString=FIX.4.0
SocketConnectPort=9876
```

The `FileStorePath` property in the two files above should point to two different directories in your local file system. The launcher scripts for the sample application can be found in the bin directory of the Quickfix/J distribution.

For more information regarding the FIX sample applications, see http://www.quickfixj.org/quickfixj/usermanual/1.5.0/usage/examples.html

For more information on configuring Quickfix/J applications, see http://www.quickfixj.org/quickfixj/usermanual/1.5.0/usage/configuration.html

Configuring WSO2 Enterprise Integrator to use the VFS transport

To run some of the samples, you need to configure WSO2 Enterprise Integrator to enable the VFS listener and the VFS sender. For instructions on configuring WSO2 Enterprise Integrator to use the VFS transport, see Enable the VFS transport.

Configuring WSO2 Enterprise Integrator to use the AMQP transport

To configure WSO2 Enterprise Integrator to use the AMQP transport, you need to have QPid version 1.0-M2 or higher installed and started. You also need to copy the following client JAR files into the `<EI_HOME>/repository/lib` directory to support AMQP. These files can be found in the `lib` directory of the QPid installation.

- qpid-client-1.0-incubating-M2.jar
- qpid-common-1.0-incubating-M2.jar
- geronimo-jms_1.1_spec-1.0.jar
- slf4j-api-1.4.0.jar **
- slf4j-log4j12-1.4.0.jar **

> To configure FIX (Quickfix/J 1.3) with AMQP (QPid-1.0-M2), copy the `sl4j-*` libraries that come with QPid and ignore the `sl4j-*` libraries that come with Quickfix/J.

To enable the AMQP transport over the JMS transport

1. Uncomment the JMS transport listener configuration.

   To enable AMQP over JMS for WSO2 Enterprise Integrator:
   - `Update the <EI_HOME>/conf/axis2/axis2.xml` file by uncommenting the JMS transport listener configuration.

   To enable JMS support for the sample Axis2 server:
   - Update the `<EI_HOME>/samples/axis2Server/repository/conf/axis2.xml` file by uncommenting the JMS transport listener configuration.

   ```
   <\!--Uncomment this and configure as appropriate for JMS transport support, after
   setting up your JMS environment \-->
   <transportReceiver name="jms">
   </transportReceiver>

   <transportSender name="jms">
   </transportReceiver>
   ```

2. Locate and edit the AMQP connection settings file for the message consumer. This file is named `direct.properties` and can be found in the `<EI_HOME>/repository/samples/resources/fix` directory.

```
java.naming.factory.initial =
org.apache.qpid.jndi.PropertiesFileInitialContextFactory
# register some connection factories
# connectionfactory.[jndiname] = [ConnectionURL]
connectionfactory.qpidConnectionfactory =
amqp://guest:guest@clientid/test?brokerlist='tcp://localhost:5672'
# Register an AMQP destination in JNDI
# destination.[jniName] = [BindingURL]
destination.directQueue =
direct://amq.direct//QpidStockQuoteService?routingkey='QpidStockQuoteService'
destination.replyQueue = direct://amq.direct//replyQueue?routingkey='replyQueue'
```

3. Locate and edit the AMQP connection settings file for WSO2 Enterprise Integrator. This file is named `conn.properties` and can be found in the `<EI_HOME>/repository/samples/resources/fix` directory.

```
#initial context factory
\#java.naming.factory.initial
=org.apache.qpid.jndi.PropertiesFileInitialContextFactory
# register some connection factories
# connectionfactory.[jndiname] = [ConnectionURL]
connectionfactory.qpidConnectionfactory=amqp://guest:guest@clientid/test?brokerli
st='tcp://localhost:5672'
# Register an AMQP destination in JNDI
# destination.[jndiName] = [BindingURL]
destination.directQueue=direct://amq.direct//QpidStockQuoteService
```

Configuring WSO2 Enterprise Integrator to use the TCP transport

To enable the TCP transport for samples

- Open the `<EI_HOME>/conf/axis2/axis2.xml` file in a text editor and add the following transport receiver configuration and sender configuration.

```
<transportReceiver name="tcp"
class="org.apache.axis2.transport.tcp.TCPTransportListener">
    <parameter name="transport.tcp.port">6060</parameter>
</transportReceiver>

<transportSender name="tcp"
class="org.apache.axis2.transport.tcp.TCPTransportSender"/>
```

If you want to use the sample Axis2 client to send TCP messages, you have to uncomment the TCP transport sender configuration in the `<EI_HOME>/samples/axis2Client/client_repo/conf/axis2.xml` file.

Configuring WSO2 Enterprise Integrator to use the UDP transport

To enable the UDP transport for samples

- Open the `<EI_HOME>/conf/axis2/axis2.xml` file in a text editor and add the following transport configurations.

```
<transportReceiver name="udp" class="org.apache.axis2.transport.udp.UDPListener"/>
<transportSender name="udp" class="org.apache.axis2.transport.udp.UDPSender"/>
```

If you want to use the sample Axis2 client to send UDP messages, add the UDP transport sender configuration in the `<EI_HOME>/ samples/axis2Client/client_repo/conf/axis2.xml` file.

Configuring WSO2 Enterprise Integrator for script mediator support

The Script mediator is a Synapse extension. Therefore, pre-requisites for all BSF supported scripting languages may not be bundled by default with the WSO2 Enterprise Integrator distribution. Before you use some of the script mediators, you need to manually add the required JAR files to the `<EI_HOME>/repository/lib` directory, and optionally perform other installation tasks as required by the individual scripting language. The following section describes this in detail.

JavaScript support

The JavaScript/E4X support is enabled by default and comes ready-to-use with the WSO2 Enterprise Integrator distribution.

Ruby support

For Ruby support you need to install **WSO2 Carbon - JRuby Engine for Mediation**. This is available in the WSO2 P2 repository. Use the WSO2 Carbon Component Manager UI in the management console to connect to the WSO2 P2 repository and install **WSO2 Carbon - JRuby Engine for Mediation**.

Alternatively, you can download and install the JRuby engine manually. Download the `jruby-complete-1.3.0.wso2v1.jar` file from the WSO2 P2 repository and copy it into the `<EI_HOME>/dropins` directory.

Setting up the databases

To run the samples that involve database integration, you need to setup the databases, sample tables and reusable data sources as required by the samples. Most samples require a remote Derby database, whereas a few samples require a MySQL database.

Setting up with Apache Derby

For instructions on installing and configuring the remote Derby database, see Setting up the remote Derby database.

When the database is installed and configured, you can create the sample tables and insert sample data into the tables.

To create a new table in the database

- Execute the following statement.

```
CREATE table company(name varchar(10) primary key, id varchar(10), price double);
```

To insert sample data into the table

- Execute the following statement.

```
INSERT into company values ('IBM','c1',0.0);
INSERT into company values ('SUN','c2',0.0);
INSERT into company values ('MSFT','c3',0.0);
```

Setting up with MySQL

For instructions on installing and configuring the MySQL database, see Setting up the MySQL database.

When the database is installed and configured, you can create the sample tables, insert sample data into the tables and create the two stored procedures.

To create a new table in the database

- Execute the following statement.

```
CREATE table company(name varchar(10) primary key, id varchar(10), price double);
```

To insert sample data into the table

- Execute the following statements.

```
INSERT into company values ('IBM','c1',0.0);
INSERT into company values ('SUN','c2',0.0);
INSERT into company values ('MSFT','c3',0.0);
```

To create the stored procedures

- Execute the following statements.

```
CREATE PROCEDURE getCompany(compName VARCHAR(10)) SELECT name, id, price FROM
company WHERE name = compName;
CREATE PROCEDURE updateCompany(compPrice DOUBLE,compName VARCHAR(10)) UPDATE
company SET price = compPrice WHERE name = compName;
```

Setting up Synapse DataSources

Definition of a reusable database connection pool or datasources can be done using the `datasources.properties` file. It is possible to configure any number of datasources. Currently only two types of datasources are supported and those are based on the Apache DBCP datasources. The two types are `BasicDataSource` and `PerUserPoolDataSource` (based on Apache DBCP). The following configuration includes both definitions.

`datasources.properties configuration`

```
\###############################################################################
# DataSources Configuration
\###############################################################################
synapse.datasources=lookupds,reportds
synapse.datasources.icFactory=com.sun.jndi.rmi.registry.RegistryContextFactory
synapse.datasources.providerPort=2199
# If following property is present , then assumes that there is an external JNDI
provider and will not start a RMI registry
\#synapse.datasources.providerUrl=rmi://localhost:2199

synapse.datasources.lookupds.registry=Memory
synapse.datasources.lookupds.type=BasicDataSource
synapse.datasources.lookupds.driverClassName=org.apache.derby.jdbc.ClientDriver
synapse.datasources.lookupds.url=jdbc:derby://localhost:1527/lookupdb;create=false
# Optionally you can specifiy a specific password provider implementation which
overrides any globally configured provider
synapse.datasources.lookupds.secretProvider=org.apache.synapse.commons.security.secret
.handler.SharedSecretCallbackHandler
synapse.datasources.lookupds.username=esb
# Depending on the password provider used, you may have to use an encrypted password
here\!
synapse.datasources.lookupds.password=esb
synapse.datasources.lookupds.dsName=lookupdb
synapse.datasources.lookupds.maxActive=100
synapse.datasources.lookupds.maxIdle=20
synapse.datasources.lookupds.maxWait=10000

synapse.datasources.reportds.registry=JNDI
synapse.datasources.reportds.type=PerUserPoolDataSource
synapse.datasources.reportds.cpdsadapter.factory=org.apache.commons.dbcp.cpdsadapter.D
riverAdapterCPDS
synapse.datasources.reportds.cpdsadapter.className=org.apache.commons.dbcp.cpdsadapter
.DriverAdapterCPDS
synapse.datasources.reportds.cpdsadapter.name=cpds
synapse.datasources.reportds.dsName=reportdb
synapse.datasources.reportds.driverClassName=org.apache.derby.jdbc.ClientDriver
synapse.datasources.reportds.url=jdbc:derby://localhost:1527/reportdb;create=false
# Optionally you can specifiy a specific password provider implementation which
overrides any globally configured provider
synapse.datasources.reportds.secretProvider=org.apache.synapse.commons.security.secret
.handler.SharedSecretCallbackHandler
synapse.datasources.reportds.username=esb
# Depending on the password provider used, you may have to use an encrypted password
here\!
synapse.datasources.reportds.password=esb
synapse.datasources.reportds.maxActive=100
synapse.datasources.reportds.maxIdle=20
synapse.datasources.reportds.maxWait=10000
```

The configuration is similar to the `log4j` appender configuration.

It requires two databases, follow the steps in the section above to create the two databases `jdbc:derby://localhost:1527/lookupdb` and `jdbc:derby://localhost:1527/reportdb` using the user name and password as `esb`. Fill in the data for the two databases as described in the section above.

To secure data sources password, you should use the mechanism for `securing secret Information`. If that mechanism is used, the passwords that have been specified are considered as aliases and those are used for

picking actual passwords. To get password securely, you should set the password provider for each data source. The password provider should be an implementation of the following:

- `org.apache.synapse.commons.security.secret.SecretCallbackHandler`.

To run this sample

- Uncomment `secret-conf.properties`, `cipher-text.properties` and `datasources.properties`. These files can be found in the `<EI_HOME>/repository/conf` directory.

Using the sample clients

After setting up the samples, you can run them using sample clients. The sample clients can be executed from the `<EI_HOME>/samples/axis2Client` directory by specifying the relevant `ant` command.

You can execute `ant` from the `<EI_HOME>/samples/axis2Client` directory, in order to view the available sample clients and some of the sample options used to configure them.

This section describes the sample clients in detail.

Stock quote client

This is a simple SOAP client that can create and send stock quote requests, as well as receive and display the last sale price for a stock symbol.

The required stock symbol and operation can be specified as follows:

```
ant stockquote [-Dsymbol=IBM|MSFT|SUN|..]
    [-Dmode=quote | customquote | fullquote | placeorder | marketactivity]
    [-Daddurl=http://localhost:9000/soap/SimpleStockQuoteService]
    [-Dtrpurl=http://localhost:8280]
    [-Dprxurl=http://localhost:8280]
    [-Drest=true]
    [-Dwsrm=true]
    [-Dpolicy=../../repository/samples/resources/policy/policy_1.xml]
```

The stock quote client is able to operate in the **smart client mode**, **gateway/dumb client mode** as well as **proxy client mode**, and can send the payloads listed below as SOAP messages:

- **quote** - Sends a quote request for a single stock as follows. The response contains the last sale price for the stock which would be displayed.

```
<m:getQuote xmlns:m="http://services.samples/xsd">
    <m:request>
        <m:symbol>IBM</m:symbol>
    </m:request>
</m:getQuote>
```

- **customquote** - Sends a quote request in a custom format. WSO2 Enterprise Integrator would transform this custom request into the standard stock quote request format and send it to the service. Upon receipt of the response, it would be transformed again to a custom response format and would be returned to the client, which will then display the last sales price.

```
<m0:checkPriceRequest xmlns:m0="http://services.samples/xsd">
    <m0:Code>symbol</m0:Code>
</m0:checkPriceRequest>
```

- **fullquote** - Gets quote reports for the stock over a number of days (for example, for the last 100 days of the year).

```
<m:getFullQuote xmlns:m="http://services.samples/xsd">
    <m:request>
        <m:symbol>IBM</m:symbol>
    </m:request>
</m:getFullQuote>
```

- **placeorder** - Places an order for stocks using a one way request.

```
<m:placeOrder xmlns:m="http://services.samples/xsd">
    <m:order>
        <m:price>3.141593E0</m:price>
        <m:quantity>4</m:quantity>
        <m:symbol>IBM</m:symbol>
    </m:order>
</m:placeOrder>
```

- **marketactivity** - Gets a market activity report for the day (for example, quotes for multiple symbols).

```
<m:getMarketActivity xmlns:m="http://services.samples/xsd">
 <m:request>
  <m:symbol>IBM</m:symbol>
  ...
  <m:symbol>MSFT</m:symbol>
 </m:request>
</m:getMarketActivity>
```

Smart client mode

The `addurl` property sets the WS-Addressing EPR, and the `trpurl` sets a transport URL for a message. Therefore, by specifying both properties, the client can operate in the `smart client` mode, where the addressing EPR can specify the ultimate receiver, and the transport URL set to WSO2 Enterprise Integrator ensures that any mediation required takes place before the message is delivered to the ultimate receiver. For example:

```
ant stockquote -Daddurl=<addressingEPR> -Dtrpurl=<ei>
```

Gateway/dumb client mode

By specifying only a transport URL, the client operates in the `dumb client` mode, where it sends the message to WSO2 Enterprise Integrator and depends on WSO2 Enterprise Integrator rules for proper mediation and routing of the message to the ultimate destination.

```
ant stockquote -Dtrpurl=<ei>
```

Proxy client mode

The client uses the `prxurl` as an HTTP proxy to send the request. Therefore, by setting the `prxurl` to WSO2 Enterprise Integrator, the client ensures that the message is sent to WSO2 Enterprise Integrator for mediation. The client can optionally set a WS-Addressing EPR if required.

```
ant stockquote -Dprxurl=<ei> [-Daddurl=<addressingEPR>]
```

Generic JMS client

This is a client that can send plain text, plain binary content, or POX content by directly publishing a JMS message to a specified destination. The JMS destination name should be specified with the `jms_dest` property. The `jms_type` property can be text, binary or pox to specify the type of message payload.

The plain text payload for a text message can be specified through the `payload` property. For binary messages, the `payload` property will contain the path to the binary file. For POX messages, the `payload` property will hold a stock symbol name to be used within the POX request for stock order placement request.

```
ant jmsclient [-Djms_type=text]
[-Djms_dest=dynamicQueues/JMSTextProxy][-Djms_payload="24.34 100 IBM"]
ant jmsclient [-Djms_type=pox] [-Djms_dest=dynamicQueues/JMSPoxProxy]
[-Djms_payload=MSFT]
ant jmsclient [-Djms_type=binary] [-Djms_dest=dynamicQueues/JMSFileUploadProxy]
    [-Djms_payload=./../../repository/samples/resources/mtom/asf-logo.gif]
```

Note

Be sure that you have ActiveMQ or any other JMS provider installed to run the JMS samples.

Optimize client

This is a client that can send a binary image file as a MTOM or SwA optimized message, and receive the same file again through the response and save it as a temporary file. The `opt_mode` can be `mtom` or `swa` respectively for the optimization. Optionally, the path to a custom file can be specified through the `opt_file` property, and the destination address can be changed through the `opt_url` property if required.

```
ant optimizeclient [-Dopt_mode=mtom | swa]
    [-Dopt_url=http://localhost:8280/soap/MTOMSwASampleService]
    [-Dopt_file=./../../repository/samples/resources/mtom/asf-logo.gif]
```

FIX client

This is a client that can post a FIX message of type **Order-Single** embedded into a SOAP message.

```
ant fixclient -Dsymbol=IBM -Dqty=5 -Dmode=buy
-Daddurl=http://localhost:8280/soap/FIXProxy
```

JSON client

This is a client that can send get-quote requests using the JSON content interchange format over HTTP.

```
ant jsonclient
    [-Daddurl=http://localhost:8280/services/JSONProxy]
    [-Dsymbol=DELL]
```