MuleSource®

the open source choice for SOA infrastructure™

U1 TECHNOLOGIES

AmbrosiaMQ-MuleSource ESB Integration

Executive Summary

This document outlines the interoperation of MuleSource ESB and U1 Technologies' AmbrosiaMQ.

AmbrosiaMQ Installation

These installation instructions assume the target OS is Linux/Solaris (Unix) using the bash shell or Windows 2000/XP or later, and with Sun JDK 1.5.x or later. For other operating systems please contact U1 Technologies for the appropriate documentation.

Downloading and Running the Installer

Download and run the appropriate setup version for your platform from http://www.u1.com/downloads.html

Unix

Change the AmbrosiaMQ setup program to be executable and then run it. (assume `~/AmbrosiaMQ` is the install directory)

```
$ chmod +x ./setup.bin
$ ./setup.bin
$ cd ~/AmbrosiaMQ
```

Windows

Run the AmbrosiaMQ setup program (assume `C:\AmbrosiaMQ` is the install directory).

```
C:\> setup.exe
C:\> cd C:\AmbrosiaMQ
```

Setting the AmbrosiaMQ Environment

Configure the scripts for the current environment. Adjust these settings depending upon the location of your JVM.

Unix

```
$ export JAVA_HOME=/usr/bin/java
```

Change to the directory where AmbrosiaMQ was installed and run the `install.sh` script which configures the AmbrosiaMQ scripts with the proper environment paths for the current system.

```
$ cd ~/AmbrosiaMQ
[AmbrosiaMQ]$ ./install.sh
```

Set the `CLASSPATH`, `AMBROSIAMQ`, and other environment variables in your current shell.

```
[AmbrosiaMQ]$ source ./linux/setcp
```

or if using Solaris,

```
[AmbrosiaMQ]$ source ./solaris/setcp
```

Windows

```
C:\AmbrosiaMQ> install.bat
C:\AmbrosiaMQ> windows\setcp.bat
```

The `AMBROSIAMQ` environment variable will be set to the install directory that was previously entered.

Using the 3-Broker Test Configuration

Start the sample 3 broker configuration named "`basic-ib-sec`" which includes interbroker (`-ib`) and security (`-sec`) features.

Unix

```
[AmbrosiaMQ]$ cd config/basic-ib-sec
```

Make the scripts executable

```
[basic-ib-sec]$ chmod +x *.sh
```

Use the default Derby database for broker and initialize the databases

```
[basic-ib-sec]$ ./init.sh
```

Start the brokers and verify they have started

```
[basic-ib-sec]$ ./run.sh
```

The `./run.sh` command will execute a `tail -f broker1/log/broker.log` to show the startup messages from that broker. Verify the broker has started properly by reviewing the output..

Windows

```
C:\AmbrosiaMQ> cd config\basic-ib-sec
C:\AmbrosiaMQ\config\basic-ib-sec> init.bat
```

Note: If the `init.bat` file is not available, please run the following equivalent commands:

```
cd broker1
java com.u1.broker.InitBrokerDatabase create
java com.u1.tools.LoadPolicies policies.txt
cd ..\broker2
java com.u1.broker.InitBrokerDatabase create
java com.u1.security.InitSecurityDatabase create
cd ..\broker3
java com.u1.broker.InitBrokerDatabase create
cd ..
```

```
C:\AmbrosiaMQ\config\basic-ib-sec> run.bat
```

Note: If the `run.bat` file is not available, please run the following equivalent commands:

```
cd broker1
start java com.u1.broker.Broker
cd ..\broker2
start java com.u1.broker.Broker
cd ..\broker3
start java com.u1.broker.Broker
```

The messages near the end of the output should be similar to the ones below :

```
[2008-02-20 17:28:28] tcp Acceptor on port 8001 now accepting...
[2008-02-20 17:28:28] ssl Acceptor on port localhost:8002 now accepting...
[2008-02-20 17:28:58] Connected to broker3@localhost:8021, localhost:8025
[2008-02-20 17:28:58] Size of subscriptions transferred to broker3: 1825
[2008-02-20 17:28:59] Connected to broker2@localhost:8011
[2008-02-20 17:28:59] Size of subscriptions transferred to broker2: 1825
```

MuleSource Installation

Downloading and Installation

Download the Mule 1.4.3 stand alone server full archive from mulesource.org and extract Mule 1.4.3 into your selected directory.

Unix

```
[ ]$ cd ~
[ ]$ tar xzvf mule-1.4.3.tar.gz
[ ]$ cd mule-1.4.3
```

Windows

```
C:\> cd \
C:\> unzip mule-1.4.3.zip
C:\> cd mule-1.4.3
```

Set the environment variables and PATH for Mule and copy the AmbrosiaMQ client and JMS jar files to Mule's user library directory.

Unix

```
[mule-1.4.3]$ export MULE_HOME=`pwd`
[mule-1.4.3]$ cp $AMBROSIAMQ/lib/jms.jar $MULE_HOME/lib/user/
[mule-1.4.3]$ cp $AMBROSIAMQ/lib/AmbrosiaMQ-client-*.jar $MULE_HOME/lib/user/
```

Windows

```
C:\mule-1.4.3> set MULE_HOME=%CD%
C:\mule-1.4.3> copy %AMBROSIAMQ%\lib\AmbrosiaMQ-client-*.jar %MULE_HOME%\lib\user
C:\mule-1.4.3> copy %AMBROSIAMQ%\lib\jms.jar %MULE_HOME%\lib\user
```

Note: If you plan to perform reconnection testing, refer to the section titled *Reconnection Notes* for additional information.

MuleSource / AmbrosiaMQ Interoperation

Downloading and Installing the Echo Example

Download the "*MuleSource Echo Example*" application from http://www.u1.com/downloads.html.

Unzip the downloaded *MuleSource Echo Example* file into a directory that is a sibling to mule-1.4.3.

Unix/Windows

```
(mule-1.4.3) cd ..
unzip mule-echo.zip
cd amb-echo
```

Example Overview

This example uses two separate windows, one for running the AmbMule.java test client, and the other for running the Mule Echo client. Type into either window and press <enter> will display text sent over a Durable or Reliable Topic, or a via a Queue, to the other window.

Creating Queues

Before running tests with **Queues**, first create the Queues used by the example by executing the following script:

Unix

```
[amb-echo] $ ./createQueues.sh
```

Windows

```
C:\amb-echo> createQueues.bat
```

This script will set the `CLASSPATH` and execute the following commands:

```
java CreateQ localhost:8001 Administrator Administrator muleInQueue
java CreateQ localhost:8001 Administrator Administrator muleOutQueue
```

Running the AmbMule Sample Program and the Mule Echo Server

The `runAmbMule` and the `runMuleEcho` scripts will try to set the `MULE_HOME` environment variable if it is not yet defined. If Mule is not located in the `mule-1.4.3` directory with the same parent directory as `amb-echo`, then please set `MULE_HOME` manually, and return to the `amb-echo` directory to start the Mule server.

The Mule configuration xml file `echo-config-{durable,reliable,queue}.xml` and the `jndi.properties` files for this example are located in the `conf` directory under `amb-echo`. The `jndi.properties` file in `amb-echo` is used by the `AmbMule.java` program.

Run each of the following sets of commands, a pair at a time, one command from each of two windows. Type messages into the Mule Server console window and the messages will show up in the AmbMule demo application window, and vice versa..

Unix

```
(window1) [amb-echo] $ ./runAmbMule.sh --durable
(window2) [amb-echo] $ ./runMuleEcho.sh --durable
```

Windows

```
(window1) C:\amb-echo> runAmbMule.bat --durable
(window2) C:\amb-echo> runMuleEcho.bat --durable
```

Note: one of the parameters [-d|--durable|-r|--reliable|-q|--queue] is required. Both the `runAmbMule` and the `runMuleEcho` scripts should be given the same parameter when running a test.

Key Mule Configuration Parameters

Here are the contents of the `amb-echo/conf/echo-config-durable.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mule-configuration
```

```
    PUBLIC "-//MuleSource //DTD mule-configuration XML V1.0//EN"
    "http://mule.mulesource.org/dtds/mule-configuration.dtd">

<mule-configuration id="Mule_Echo_Sample" version="1.0">

    <description>
        This is a simple component example that demonstrates how to
        expose a component over multiple transports.
    </description>

    <mule-environment-properties>
        <connection-strategy
                className="org.mule.providers.SimpleRetryConnectionStrategy">
            <properties>
                <property name="doThreading" value="false" />
                <property name="retryCount" value="-1" />
                <property name="frequency" value="5000" />
            </properties>
        </connection-strategy>
    </mule-environment-properties>

    <!--
        The system stream connector is used to send and receive information via the
        System.in and System.out. Note this connector is only really useful for testing
        purposes.
        promptMessage - is what is written to the console
        messageDelayTime - the time in milliseconds before the user is prompted again.
        These properties are set as bean properties on the connector.
    -->
    <connector name="SystemStreamConnector"
            className="org.mule.providers.stream.SystemStreamConnector">
        <properties>
            <property name="promptMessage" value="Please enter something: " />
            <property name="messageDelayTime" value="1000" />
        </properties>
    </connector>

    <connector name="AmbrosiaMQJmsConnector"
            className="org.mule.providers.jms.JmsConnector">
        <properties>
            <property name="specification" value="1.1" />
            <property name="connectionFactoryJndiName" value="TopicConnectionFactory" />
            <property name="jndiInitialFactory" value="com.u1.naming.InitialFactory" />
            <property name="recoverJmsConnections" value="true"/>
```

```
        <property name="username" value="Administrator" />
        <property name="password" value="Administrator" />

        <property name="durable" value="true" />

        <map name="connectionFactoryProperties">
            <property name="DURABLE_SUBSCRIBER_LB_POOL" value="pool-2" />
            <property name="com.u1.naming.TopicConnectionFactory.host"
                value="localhost:8021, localhost:8011" />
            <property name="com.u1.naming.TopicConnectionFactory.appId"
                value="Mule-Server-Durable" />
            <!--
            <property name="RECONNECT_ENABLE" value="true" />
            <property name="RECONNECT_RETRY_INTERVAL_SEC" value="2" />
            <property name="RECONNECT_BLOCKING_ENABLE" value="true" />
            -->
        </map>
    </properties>
</connector>

<transformers>
    <transformer name="HttpRequestToSoapRequest"
        className="org.mule.providers.soap.transformers.HttpRequestToSoapRequest" />
</transformers>

<!--
    The Mule model initialises and manages your UMO components
-->
<model name="echoSample">
    <!--
     A Mule descriptor defines all the necessary information about how your
     components will interact with the framework, other components in the
     system and external sources.
     Please refer to the Configuration Guide for a full description of all the
     parameters.
    -->
    <mule-descriptor name="EchoUMO"
        implementation="org.mule.components.simple.EchoComponent">

        <!-- any number of endpoints can be added to an inbound router -->
        <inbound-router>
            <endpoint address="stream://System.in" />
            <endpoint address="vm://echo" />
            <endpoint address="jms://topic:muleInTopic">
                <properties>
```

```
                <!--
                 Set durableName. Mule's auto-generated name contains
                 "." (dot) characters which are used as subject separator
                 characters in AmbrosiaMQ
                -->
                    <property name="durableName"
                        value="my_subscription" />
                </properties>
            </endpoint>

            <!-- Example URL:
             http://localhost:65081/services/EchoUMO?method=echo&param=Echo_Test
            -->
            <endpoint address="axis:http://localhost:65081/services"
                      transformers="HttpRequestToSoapRequest" />
            <endpoint address="axis:http://localhost:65082/services"/>
        </inbound-router>

        <!--
         An outbound router can have one or more router configurations
         that can be invoked depending on business rules, message contents, headers
         or any other criteria.
         The OutboundPassthroughRouter is a router that automatically passes on
         every message it receives
        -->
        <outbound-router matchAll="true">
            <!-- added matchAll="true" so all routers listed below will
                receive the message -->
            <router className=
                   "org.mule.routing.outbound.OutboundPassThroughRouter">
                <endpoint address="stream://System.out" />
            </router>
            <router className=
                   "org.mule.routing.outbound.OutboundPassThroughRouter">
                <endpoint address="jms://topic:muleOutTopic" />
            </router>
        </outbound-router>
    </mule-descriptor>
</model>

</mule-configuration>
```

Reconnection Notes

If you plan to perform reconnection tests with Mule, please refer to the following links for additional information as well as to

determine if a new release is available that resolves the issues discussed in the links.

- JMSConnector fails to reconnect when used with SimpleRetryConnectionStrategy
  http://mule.mulesource.org/jira/browse/MULE-1720

  We have found that the using code contained in an attachment included in the above link, named *JmsConnector.patch* does improve reconnection capability.

- MuleWorkManager is stopped after JMS reconnection
  http://mule.mulesource.org/jira/browse/MULE-2616

  We have added the call to `disposeWorkManagers();` to the end of the `JmsConnector.onNotification()` method as described in the JIRA note and it also improves reconnection.

If you choose to implement the unofficial patches described above, the `JmsConnector.class` files can be placed in a jar with their full path and then copied to the `mule-1.4.3/lib/user/` directory so Mule will run with the modified classes.

Additional Information

Property settings for JMS - http://mule.mulesource.org/display/MULEUSER/Jms+Provider

Contact Information

For questions or assistance please contact:

Margaretta Colangelo
VP of Business Development
U1 Technologies
Direct: 415.721.7155
Main: 415.480.0318
Fax: 415.704.3275
Email: mc@u1.com