Administration Manual

Author:
Jacek Zagorski
Shasta NetWorks LLC
jaz@shastanetworks.com

September 15, 2003
JEngine 2.0, Doc Rev. 6

Requirements
Java 1.4 or greater
JAVA_HOME set
PATH includes path to java executable

INTRODUCTION

The new version of JEngine, 2.0, has two major differences from the previous versions, 1.x. One is the fact that it runs on top of Jboss 3.2.1 instead of Jboss 2.4. There were changes made to some of the source code to take into account new logging and java management extensions. See the source code for more details. The other major change is the addition of transformation (translation) capability. The transformation capability uses BeanShell to transform the fields and segments of an HL7 message.

There is a new directory structure for JEngine 2.0. From your root directory, you should see the following tree:

```
-jengine2
        -run
                -bin
                -client
                -docs
                -lib
                -server
                        -default
                                -conf
                                -data
                                -deploy
                                        -http-invoker.sar
                                        -jbossweb-jetty.sar
                                        -jms
                                        -jmx-console.war
                                        -jmx-invoker-adaptor-server.sar
                                        -jmx-rmi-adaptor.sar
                                        -management
                                -lib
```

-log
-tmp
-source
-test

The *run* directory contains all the files needed to run an instance of JEngine. The *source* directory contains all the source code and directories needed to run an instance of NetBeans (http://netbeans.org) to compile and build JEngine. The *test* directory contains the data files and scripts required to run basic tests. Listeners and senders are provided as examples – you can modify these to work for your particular setup.

CONFIGURATION

The configuration directory for the JEngine server resides in *c:\jengine2\run\server\default\conf\default*. Configuration for some of the services can also be found in *c:\jengine2\run\server\default\deploy*. While the core configuration for JEngine exists in the *config* directory, the *deploy* directory contains dynamic configuration that can be changed while JEngine is running. JEngine will automatically undeploy and then redeploy the services in the deploy directory (when a file is changed or a new file is created) – this includes the JMS configuration as well as the JEngine interface configuration. New queues and interfaces can be dynamically deployed by dropping in new configurations into the *deploy/jms* directory.

JMS for JEngine message queueing now uses the Hypersonic database for message persistence (PersistenceManager). The database provides more reliable and better performing persistence than file based persistence (JEngine 1.x). The size of the queue is set to 500MB. If the database limit is reached, secondary storage (file system) is used to store old data. The secondary storage (CacheStore) is configured to be *tmp\jbossmq*.

Interface Configuration : *jms/10-jengine-service.xml*

The previous JEngine configuration file was *jboss.jcml* in the core config directory. The new file for JEngine 2.0 is called *10-jengine-service.xml*.

All JEngine interface services are called Mbeans. First, in the file, the JMS Queues are defined. The queues are initialized first because the HL7 interfaces depend on them.

```
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_IN_HL7_MS4_MT_ADT">
   <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_OUT_HL7_MS4_MT_ADT">
   <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_IN_HL7_MT_MS4_CHG">
   <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_OUT_HL7_MT_MS4_CHG">
  <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_ERROR_OUT_HL7_MS4">
   <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
<mbean code="org.jboss.mq.server.jmx.Queue" name="JEngine:service=Queue,name=Q_ERROR_OUT_HL7_MT">
   <depends optional-attribute-name="DestinationManager">jboss.mq:service=DestinationManager</depends>
</mbean>
```

Then come the Inbound Interface definitions.

```
<mbean code="org.jengine.mbean.HL7ServerService" name="JEngine:service=Interface,name=TCP_IN_HL7_MS4">
   <!--
   <attribute name="Comment">Inbound ADT HL7 from MS4 (Cascade)</attribute>
   -->
   <attribute name="Port">9000</attribute>
   <attribute name="IFName">TCP_IN_HL7_MS4</attribute>
   <attribute name="Queues">Q_IN_HL7_MS4_MT_ADT</attribute>
</mbean>
<mbean code="org.jengine.mbean.HL7ServerService" name="JEngine:service=Interface,name=TCP_IN_HL7_MT">
   <!--
   <attribute name="Comment">Inbound Charges HL7 from MediTech</attribute>
   -->
   <attribute name="Port">9001</attribute>
   <attribute name="IFName">TCP_IN_HL7_MT</attribute>
   <attribute name="Queues">Q_IN_HL7_MT_MS4_CHG</attribute>
</mbean>
```

Then come the Outbound Interface definitions.

```
<mbean code="org.jengine.mbean.HL7ClientService" name="JEngine:service=Interface,name=TCP_OUT_HL7_MS4">
   <!--
   <attribute name="Comment">Outbound Charges HL7 to MS4 (Cascade)</attribute>
   -->
   <attribute name="Port">9001</attribute>
   <attribute name="IFName">TCP_OUT_HL7_MS4</attribute>
   <attribute name="IPAddress">172.16.18.11</attribute>
   <attribute name="Queue">Q_OUT_HL7_MT_MS4_CHG</attribute>
   <attribute name="QueueError">Q_ERROR_OUT_HL7_MS4</attribute>
   <attribute name="ResendFailedCount">10</attribute>
   <attribute name="ResendRetryInterval">5000</attribute>
   <attribute name="ConnectRetryInterval">20000</attribute>
</mbean>
<mbean code="org.jengine.mbean.HL7ClientService" name="JEngine:service=Interface,name=TCP_OUT_HL7_MT">
   <!--
   <attribute name="Comment">Outbound ADT HL7 to MediTech</attribute>
   -->
   <attribute name="Port">6000</attribute>
   <attribute name="IFName">TCP_OUT_HL7_MT</attribute>
   <attribute name="IPAddress">10.10.0.71</attribute>
   <attribute name="Queue">Q_OUT_HL7_MS4_MT_ADT</attribute>
   <attribute name="QueueError">Q_ERROR_OUT_HL7_MT</attribute>
   <attribute name="ResendFailedCount">10</attribute>
   <attribute name="ResendRetryInterval">5000</attribute>
   <attribute name="ConnectRetryInterval">20000</attribute>
</mbean>
```

Then come the Transformation definitions.

```
<mbean code="org.jengine.mbean.HL7XformService" name="JEngine:service=Interface,name=XFORM_ADT_MS4_MT">
   <attribute name="IFName">XFORM_ADT_MS4_MT</attribute>
   <attribute name="InQueue">Q_IN_HL7_MS4_MT_ADT</attribute>
   <attribute name="OutQueue">Q_OUT_HL7_MS4_MT_ADT</attribute>
   <attribute name="Transformation">org.jengine.xforms.bsh.Transformation</attribute>
   <attribute name="TransformationProperties">
            import org.jengine.tools.hl7.Segment;
            import org.jengine.tools.hl7.Field;
            log.info("transformation in progress");
            log.info("version is "+message.versionID);
            Segment pv1Segment;
            Segment evnSegment;
            Segment pidSegment;
            String eventTypeCode, patientClass;
            try {
                        pidSegment = message.getSegment("PID");
                        pv1Segment = message.getSegment("PV1");
                        evnSegment = message.getSegment("EVN");
            } catch (Exception e)
            { log.error("**** PID, PV1, EVN missing ****"); }

            eventTypeCodeField = evnSegment.getField(1);
            eventTypeCode = eventTypeCodeField.get(1);
            patientClassField = pv1Segment.getField(2);
```

```
		patientClass = patientClassField.get(1);

		String mappedPatientStatus = "";
		try {
			if (eventTypeCode.equals("A01")) mappedPatientStatus = "ADM";
			else if (eventTypeCode.equals("A02")) mappedPatientStatus =  "LTP";
			else if (eventTypeCode.equals("A03")) mappedPatientStatus = "DIS";
			else if (eventTypeCode.equals("A04")) mappedPatientStatus = "REG";
			else if (eventTypeCode.equals("A05")) mappedPatientStatus = "SCH";
			else if (eventTypeCode.equals("A06")) mappedPatientStatus = "ADM";
			else if (eventTypeCode.equals("A07")) mappedPatientStatus = "";
			else if (eventTypeCode.equals("A08")) mappedPatientStatus = "LTP";
			else if (eventTypeCode.equals("A09")) mappedPatientStatus = "";
			else if (eventTypeCode.equals("A11")) mappedPatientStatus = "LTP";
			else if (eventTypeCode.equals("A12")) mappedPatientStatus = "";
			else if (eventTypeCode.equals("A13")) mappedPatientStatus = "LTP";
			else if (eventTypeCode.equals("A17")) mappedPatientStatus = "LTP";
			else if (eventTypeCode.equals("A18")) mappedPatientStatus = "LTP";
			else mappedPatientStatus = "";
			Field f = new Field(mappedPatientStatus);
			pv1Segment.setField(41,f);
			log.info("EventTypeCode : [" + eventTypeCode + "]" + " >> PatientStatus : [" + mappedPatientStatus + "]");
		} catch (Exception e)
		{ log.error("Exception in mapping Patient Status"); e.printStackTrace(); }

		String mappedPatientType = "";
		try {
			if (patientClass.equals("B")) mappedPatientType = "CLI";
			else if (patientClass.equals("O")) mappedPatientType = "CLI";
			else if (patientClass.equals("P")) mappedPatientType = "CLI";
			else if (patientClass.equals("T")) mappedPatientType = "CLI";
			else if (patientClass.equals("B")) mappedPatientType = "CLI";
			else if (patientClass.equals("W")) mappedPatientType = "CLI";
			else if (patientClass.equals("E")) mappedPatientType = "ER";
			else if (patientClass.equals("I")) mappedPatientType = "IN";
			else if (patientClass.equals("N")) mappedPatientType = "IN";
			else if (patientClass.equals("D")) mappedPatientType = "INO";
			else if (patientClass.equals("R")) mappedPatientType = "RCR";
			else if (patientClass.equals("A")) mappedPatientType = "REF";
			else if (patientClass.equals("X")) mappedPatientType = "REF";
			else if (patientClass.equals("S")) mappedPatientType = "SDC";
			else mappedPatientType = "";
			Field f=new Field(mappedPatientType);
			pv1Segment.setField(18,f);
			log.info("PatientClass : [" + patientClass + "]" + " >> PatientType : [" + mappedPatientType + "]");
		} catch (Exception e)
		{ log.error("Exception in mapping Patient Type"); e.printStackTrace(); }

		Field location = pv1Segment.getField(3);
		location.set(1,4,"ACH");
		Field lf = new Field("ACH");
		pv1Segment.setField(39,lf);

		Field identifier = pidSegment.getField(3);

		String medRecNumber = identifier.get(1,1);

		String[] pcsMR = medRecNumber.split("-");
		String newMedRecNumber = "";
		for (int i=0; i != pcsMR.length; i++)  //a less-than sign does not work here (xml)!
			newMedRecNumber += pcsMR[i];

		Field newIdentifier = new Field(newMedRecNumber);
		pidSegment.setField(3,newIdentifier);
	</attribute>
</mbean>
<mbean code="org.jengine.mbean.HL7XformService" name="JEngine:service=Interface,name=XFORM_CHG_MT_MS4">
	<attribute name="IFName">XFORM_CHG_MT_MS4</attribute>
	<attribute name="InQueue">Q_IN_HL7_MT_MS4_CHG</attribute>
	<attribute name="OutQueue">Q_OUT_HL7_MT_MS4_CHG</attribute>
	<attribute name="Transformation">org.jengine.xforms.bsh.Transformation</attribute>
	<attribute name="TransformationProperties">
		import org.jengine.tools.hl7.Segment;
		import org.jengine.tools.hl7.Field;
		log.info("transformation in progress");
```

```
            log.info("version is "+message.versionID);
            Segment fhsSegment = message.getSegment("FHS");
            log.info("message.getSegment(FHS)");
            Segment ftsSegment = message.getSegment("FTS");
            log.info("message.getSegment(FTS)");
            message.removeSegment(ftsSegment);
            log.info("---FTS Segment Removed");
            message.removeSegment(fhsSegment);
            log.info("---FHS Segment Removed");

            //SET BHS SEGMENT FIELDS
            Segment bhsSegment = message.getSegment("BHS");
            Field fac = new Field("500");  //facilities
            Field app = new Field("MS4AR");//sending application
            bhsSegment.setField(4,fac);
            bhsSegment.setField(5,app);
            bhsSegment.setField(6,fac);
            Field batchType = new Field("C");
            bhsSegment.setField(9,batchType);
            Field batchControlId = new Field("");
            bhsSegment.setField(11,batchControlId);

            java.util.Vector mshSegs = new Vector();
            mshSegs = message.getSegments("MSH");
            log.info("Got mshSegs");
            for(java.util.Iterator it=mshSegs.iterator(); it.hasNext(); )
            {
               Segment seg=(Segment)it.next();
               Field fac = new Field("500");  //facilities
               Field app = new Field("MS4AR");//sending application
               seg.setField(4,fac);
               seg.setField(5,app);
               seg.setField(6,fac);
            }
   </attribute>
</mbean>
```

Next, we'll provide some more detail on the XML configuration and what you would need to do for a real interface. The MBean XML definition below is the Server bean that services (receives) HL7 messages from the external system, MS4. Note, that you should only change the Server MBean's Port number. The Queues define which Queue the incoming message will get sent to, to be picked up by an XForm MBean and then a outbound Client Mbean. Do not modify the Queue names unless you modify the corresponding interface that's next in the chain. For example, if you modify the Queue here that the Inbound (Server) interface **writes** from, you'll need to modify the Queue that the Transformation interface **reads** from.

```
<mbean code="org.jengine.mbean.HL7ServerService" name="JEngine:service=Interface,name=TCP_IN_HL7_MS4">
   <!--
   <attribute name="Comment">Inbound ADT HL7 from MS4 (Cascade)</attribute>
   -->
   <attribute name="Port">9000</attribute>
   <attribute name="IFName">TCP_IN_HL7_MS4</attribute>
   <attribute name="Queues">Q_IN_HL7_MS4_MT_ADT</attribute>
</mbean>
```

The MBean below is the Client MBean that communicates with external systems that receive messages from JEngine. Note that the Outbound Client defines both an IP Address and a Port number. The Queue for the Outbound interface defines which Queue the Client is picking messages up from following a Transformation.

```
<mbean code="org.jengine.mbean.HL7ClientService" name="JEngine:service=Interface,name=TCP_OUT_HL7_MS4">
   <!--
   <attribute name="Comment">Outbound Charges HL7 to MS4 (Cascade)</attribute>
   -->
   <attribute name="Port">9001</attribute>
   <attribute name="IFName">TCP_OUT_HL7_MS4</attribute>
   <attribute name="IPAddress">172.16.18.11</attribute>
   <attribute name="Queue">Q_OUT_HL7_MT_MS4_CHG</attribute>
```

```
   <attribute name="QueueError">Q_ERROR_OUT_HL7_MS4</attribute>
   <attribute name="ResendFailedCount">10</attribute>
   <attribute name="ResendRetryInterval">5000</attribute>
   <attribute name="ConnectRetryInterval">20000</attribute>
</mbean>
```

Logging Configuration : log4j.properties

The log file configuration is in log4j.properties. The JEngine server uses the standard Apache log4j mechanism to log its output giving fine control over levels of logging including DEBUG, INFO, WARN, ERROR and FATAL.
You can also set different logging levels for the console and the log file. The best place to find detailed documentation on log4j is http://jakarta.apache.org/log4j/docs/index.html
Most likely, you will only need to modify the log file name, whether or not it rolls over old files, how many files to keep and the maximum size each file should be. The parameters are listed below.

```
<appender name="FILE" class="org.jboss.logging.appender.DailyRollingFileAppender">
        <param name="File" value="${jboss.server.home.dir}/log/server.log"/>
        <param name="Append" value="true"/>
        <!-- Rollover at midnight each day -->
        <param name="DatePattern" value="'.'yyyy-MM-dd"/>
        <layout class="org.apache.log4j.PatternLayout">
                <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
        </layout>
</appender>
```

File defines the name and location of the log file. *Append* specifies that the server will append to an existing file on startup. *DatePattern* is the date pattern of the suffix of the file that's saved at midnight. *PatternLayout* defines what each line of the log file looks like : day, hour, subsystem, trace level, message.

ADMINISTRATION

Startup & Shutdown

JEngine can be started by double-clicking the *"JEngine Server" icon* on the desktop. You can also run the batch file in *c:\jengine2\run\bin\run.bat*.
To shutdown the JEngine server gracefully, hit *CTRL-C* in the JEngine terminal window.

Viewing Interface Status using a Browser

On the local JEngine machine, point your browser to
http://127.0.0.1:8080/web-console

or point your browser to the IP address of the JEngine server.

This will bring up a list of all MBeans in the JEngine system. Go to **System**, then **JMX Mbeans** and then the **JEngine** section to view JEngine Interface specific details. By following the links to the Q_*, you will be able to see how many messages exist in a particular Queue (see *QueueDepth*). By following the links to TCP_IN_* and TCP_OUT_*, the parameters of most importance are

- *ConnectionStatus* – a boolean, true or false, if the interface socket is connected to an external system
- *NumberMessages* – an integer, how many messages have been handled by the interface
- *NumberMessagesFailed* – an integer, how many messages have failed to be received or sent
- *TimeStampLastMsg* – the time of the last message going through the interface

DOWNLOADING FROM SOURCEFORGE AND TESTING INSTALLATION

The zip file containing the JEngine source and runtime is downloadable anytime from SourceForge (http://sourceforge.net/projects/jengine). The packaged configuration is configured for one inbound TCP/HL7 connection and 2 outbound TCP/HL7 connections.

Configure
The JEngine configuration can be found in /jengine2/run/server/default/deploy/jms. The file 10-jengine-service.xml contains the JEngine definitions.

Run
You may extract directly into /. Everything will unpack into directory /jengine2. The JEngine runtime is under /jengine2/run.
Go to /jengine2/run/bin and run either run.bat or run.sh depending on OS.
If Java is not in your path, modify the run file to customize for your machine's JDK location.

Test
The test scripts are located in /jengine/test. There, you can run listenXXX.bat(sh) and sendXXX.bat(sh) in order to set up receive and send sockets for testing. Again, if needed, modify the scripts to update IP address, port numbers, the path to Java, etc.