Thirty Meter Telescope (TMT)

Event Service Prototype – User Guide

Persistent Systems Ltd
Bhageerath, 402 Senapati Bapat Road
Pune 411 016, India

1. Purpose of this document

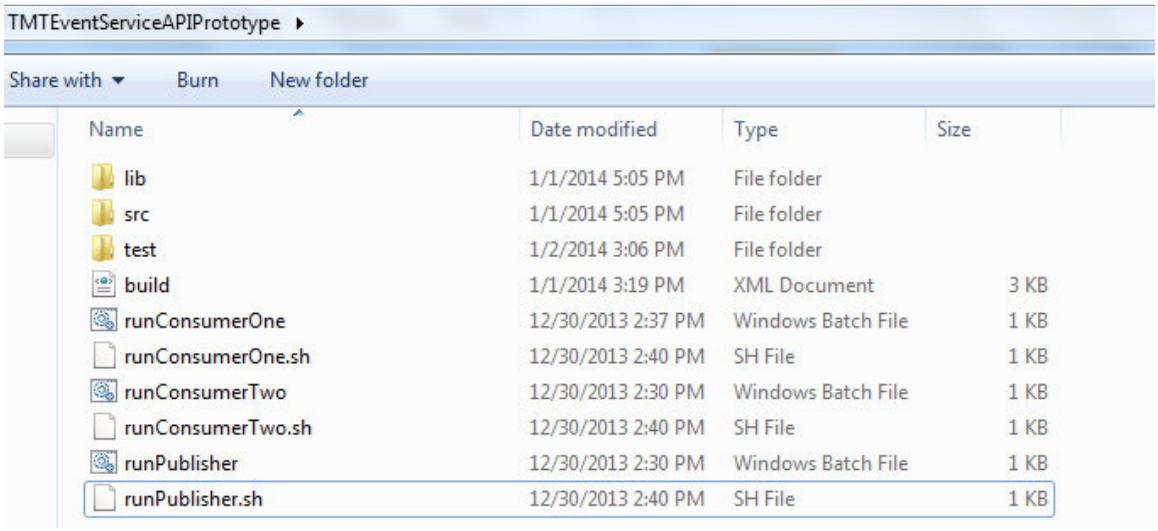
This document is a user-guide; it explains the code structure and procedure to build the code and to run the standalone Publisher and Consumer programs.

2. Get Code from GitHUB

Get code for prototype from GitHUB to any directory on the machine, hence forth we will refer to that directory as SOURCE_HOME.

3. Directory Structure

After getting the code from GitHUB on local machine, the code structure within SOURCE_HOME should look like following:



Following are the files and folder in this structure

1. **src:** this folder contains the source code for prototype, This folder contains 2 internal folders
   a. **config**: config folder contains properties files to configure hornetQ properties and JDK logger.

   b. **org**: this folder and sub-packages contains source code for the prototype.

2. **test:** This folder contains the source code for junit test cases.

3. **lib**: contains HornetQ jar files required to compile the code, additionally it contains junit jar file for running the junit tests.
   - hornetq-commons.jar
   - hornetq-core-client.jar
   - jnp-client.jar
   - netty.jar
   - junit-4.7.jar

4. **build.xml** : Ant build file

5. **runConsumerOne.bat/runConsumerOne.sh** : Script will run the demo Consumer Program which uses prototype code to subscribe.

   .bat file is for windows env. and .sh is for Linux env
   .sh files are tested on RedHat Linux env.

6. **runConsumerTwo.bat/runConsumerTwo.sh:** Script will run another demo Consumer Program which uses prototype code to subscribe.

7. **runPublisher.bat/runPublisher.sh:** Script is will run the demo Publisher Program which uses prototype code to publish a message to a topic.

4. Build

Ant is used to build the code.

Prerequisites:

SUN JDK (at least version 1.6) and Ant (at least version 1.9.2) must be installed on the machine and added to the PATH environment variable.
(Note: please see section 9. Appendix for more details.)

Running the ANT build

To run the ant build from command line, traverse to the SOURCE_HOME directory where build.xml file is located. For example if SOURCE_HOME on Windows OS is "c:\TMTEventServiceAPIPrototype" then from command line, traverse to that directory

On Windows it should look like c:\TMTEventServiceAPIPrototype>
On UNIX it might look like [SOURCE_HOME] #

And execute the ant command there
c:\TMTEventServiceAPIPrototype>ant compile
[SOURCE_HOME] # ant compile

build.xml contains multiple ant tasks for various purposes and should be run in following order :

a) Code Compilation:

To compile the code, execute command "ant compile" on the command prompt, this will generate the classes in SOURCE_HOME\build folder.

b) JUnit execution

To execute junit cases, execute command “ant junit” on the command prompt.

At the end of execution if the ant junit task runs successfully then it will create an html report for junit test execution summary which will contain the report of pass/fail cases. This report is available within SOURCE_HOME\junit directory.

Please open index.html file to view the report.

Please note that HornetQ should be available to execute the JUnit tests, IP address/Port of HornetQ server should be updated to the file SOURCE_HOME\src\config\configuration.properties

c) Packaging as ZIP file

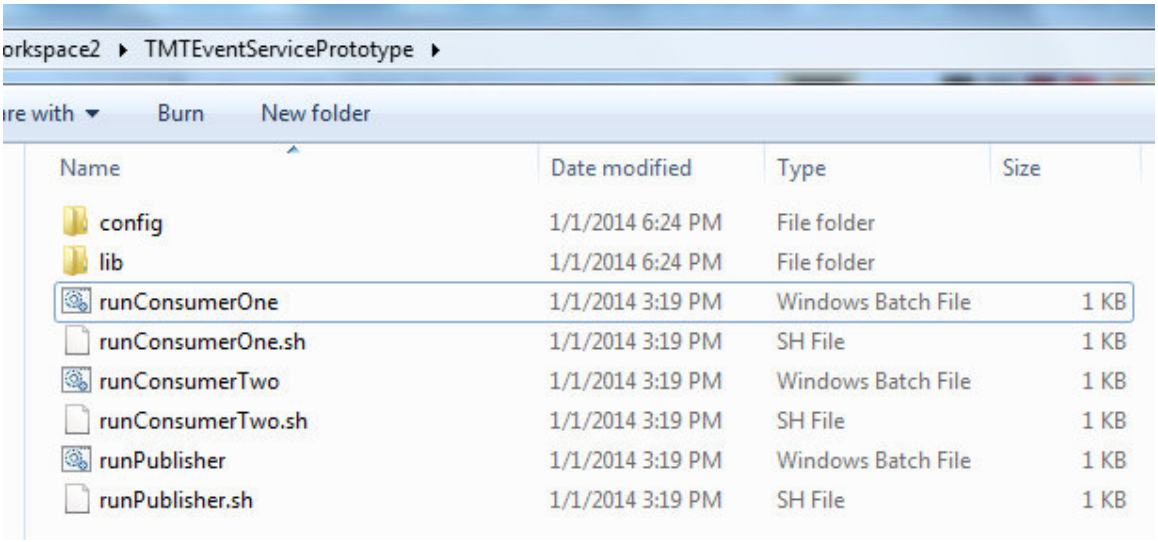
Executing the command “ant zip” on command prompt will package the entire prototype as a zip file (named TMTEventServicePrototype.zip), this zip file will contain prototype API as a jar file along with Two Consumers and One Publisher standalone programs.

This zip file (TMTEventServicePrototype.zip) will be created in location parallel to SOURCE_HOME.

5. Running the Publisher and Consumer Standalone Programs

Two Consumers and One Publisher standalone programs would be packaged along with the EventService API jar.

These programs uses prototype API to publish and subscribe, to run these programs, please unzip the TMTEventServicePrototype.zip created in previous step in any directory, hence forth we will call that directory as EXECUTABLE_HOME.
After unzip the directory structure should look like following:



Following are the files and folders available inside this directory:

1. **config**: config folder contains two properties files
   "configuration.properties" is to configure hornetQ properties in client programs.
   "logger.properties" is to configure JDK logger properties.

   Note: Before using Prototype API, following two parameters in configuration.properties file must be set

   ```
   brokerHost=<IP address of HornetQ server>
   brokerPort=5445 <Port of HornetQ server, which is by default 5445>
   ```

2. **lib**: lib folder contains prototype api jar file and hornetQ jar files to execute the Publisher/Consumer standalone programs.
   The prototype api will be available within **TMTEventServicePrototype.jar** file.

3. **runPublisher.bat/ runPublisher.sh:**
   .bat files are for windows environment and .sh files are for linux environment, sh files are tested on RedHat linux env.

   This file will execute the Publisher Standalone program which will use the prototype api to publish a message to a topic.
   To run Publisher program on windows machine, double click on runPublisher.bat file
   To run Publisher program on linux machine, traverse to EXECUTABLE_HOME directory and execute runPublisher.sh.

This program is easy to use and will run in an interactive mode, it will present multiple options to user to publish a message using command line.

4. runConsumerOne.bat/ runConsumerOne.sh:
   These files will execute ConsumerOne standalone program, which will use the prototype api to subscribe to message.

   To run ConsumerOne program on windows machine, double click on runConsumerOne.bat file

   To run ConsumerOne program on linux machine, traverse to EXECUTABLE_HOME directory and execute runConsumerOne.sh.

   This program will run in an interactive mode and will present multiple options to user for

   a. subscribing to a Topic
   b. unsubscribing from a Topic
   c. unsubscribeAll from a Topic
   d. stop the program

5. runConsumerTwo.bat/ runConsumerTwo.sh:
   These files will execute ConsumerTwo standalone program, it will execute as per given in step#4 above.

   Consumer Programs will receive the message posted by Publisher and these will be displayed on console for user to verify.

   Multiple ConsumerOne and ConsumerTwo programs can be run simultaneously in different command prompts/shell.

6. Logging

JDK logger is used for logging; logger can be configured using logger.properties file present in EXECUTABLE_HOME\config directory

JDK logger can be configured to output logs to console or to a specified log file, it can also be configured to emit logs written with INFO or SEVERE level or logging can also be turned OFF.

7. Scripts in BrokerScripts folder

BrokerScripts folder in github contains 3 files

1. **run.sh** : this script starts HornetQ
2. **stop.sh**: this script stops HornetQ
3. **setenv.sh**: this script sets PATH environment variable required for event service prototype
   This script can be run in 2 ways,
   Type either
   . setenv.sh
   OR
   source setenv.sh

These scripts are tested on Red Hat Linux and are available in folder "BrokerScripts"

4. **hornetq-configuration.xml :**
   hornetq-configuration.xml should be updated to provide HornetQ machine's IP address.
   This file is available at location:
   HORNETQ_HOME/config/stand-alone/non-clustered/

   Please update this line:
   <param key="host" value="${hornetq.remoting.netty.host:localhost}"/>
   With following:
   <param key="host" value="HornetQ machine's IP address"/>

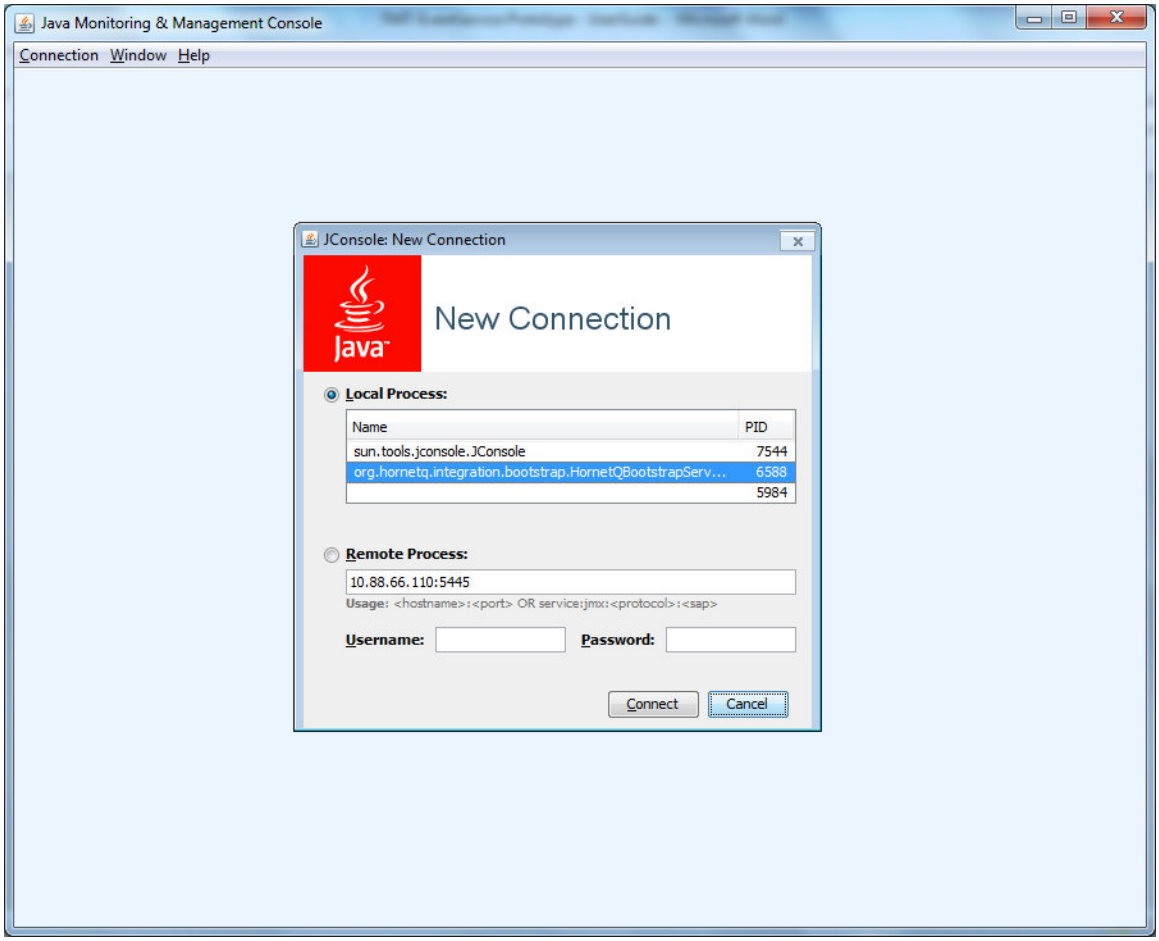
   Sample HornetQ configuration file is provided in "BrokerScripts" folder for reference only.

8. Using JConsole

JConsole is a GUI based tool, it comes with JDK1.6 and above, this tool usually is available from location JAVA_HOME\bin

1. ***Connecting JConsole to HornetQ where HornetQ is running on same machine:***

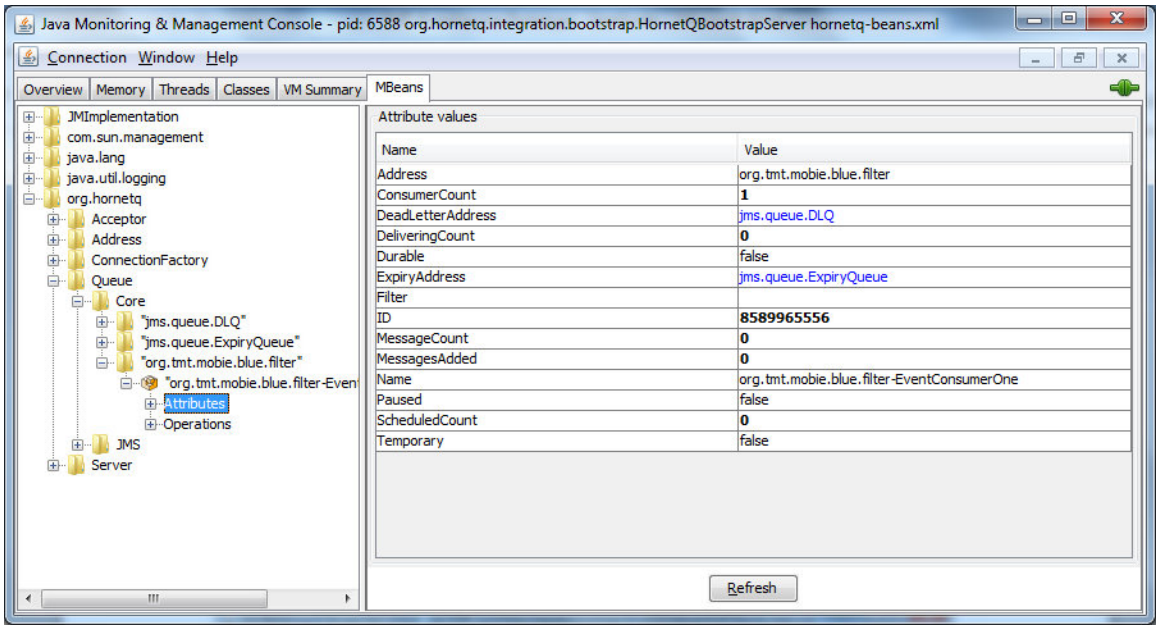
1a. Run JConsole, following GUI will open



1b. Select **Local Processes** and if HornetQ is already running then that process will already be listed in this section, select that HornetQ process and

1c. Click on connect button, this will connect JConsole to HornetQ server

1d. Next click on tab “MBeans”, from menu tree on left side of pane, expand “org.hornetq”, this section will show HornetQ related details like queues etc.



2. *Connecting JConsole to HornetQ where HornetQ is running on remote machine:*

2a. InOrder for JConsole to connect to HornetQ running on remote machine, run.bat/run.sh file used to start HornetQ (available at location HORNETQ_HOME\bin) should be modified to include additional parameter like Port on which HornetQ will be available to accept connections from JConsole.

For example on linux machine, open run.sh file and replace the last line

java $JVM_ARGS -classpath $CLASSPATH -Dcom.sun.management.jmxremote org.hornetq.integration.bootstrap.HornetQBootstrapServer $FILENAME

With following line

java $JVM_ARGS -classpath $CLASSPATH -Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=**3333**
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
org.hornetq.integration.bootstrap.HornetQBootstrapServer $FILENAME

Please note that port given above is 3333, you may choose any other port that is available

Please see the supplied file run.sh in “BrokerScripts” folder for reference, this script and above mentioned details are tested whereas JConsole was on windows machine and HornetQ server on another RedHat Linux OS machine.

2b. Next connect JConsole to remote HornetQ, Please see figure given in section **1a** above, select **Remote Process**, enter IP:Port combination (example 10.22.23:88:3333) and press connect button.
This will connect JConsole to remote HornetQ

Please follow step given in **1d** above

In case HornetQ throws following exception after making above changes
java.net.MalformedURLException: Local host name unknown:
java.net.UnknownHostException:

Then following additional changes needs to be made to resolve this exception

1. **Add Machine name in /etc/host file**

Contents of /etc/host file might look like following:
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

Please update above 2 lines to add your machine name as follows:
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4 **<machine name>**
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6 **<machine name>**

Machine name can be obtained by typing the command hostname

2. **Next update IP Address in hornetq-beans.xml**

This file is available in HORNETQ_HOME/config/stand-alone/non-clustered/ hornetq-beans.xml

- **Original file contents**

```
<bean name="StandaloneServer"
class="org.hornetq.jms.server.impl.StandaloneNamingServer">
      <constructor>
         <parameter>
            <inject bean="HornetQServer"/>
         </parameter>
      </constructor>
      <property name="port">${jnp.port:1099}</property>
      <property name="bindAddress">${jnp.host:localhost}</property>
      <property name="rmiPort">${jnp.rmiPort:1098}</property>
      <property name="rmiBindAddress">${jnp.host: localhost }</property>
   </bean>
```

- **After modification**

```
<bean name="StandaloneServer"
class="org.hornetq.jms.server.impl.StandaloneNamingServer">
      <constructor>
         <parameter>
            <inject bean="HornetQServer"/>
         </parameter>
      </constructor>
```

```
      <property name="port">${jnp.port:1099}</property>
      <property name="bindAddress">${jnp.host:10.88.66.110}</property>
      <property name="rmiPort">${jnp.rmiPort:1098}</property>
      <property name="rmiBindAddress">${jnp.host:10.88.66.110}</property>
   </bean>
```

After making the above changes in step 1 and 2, JConsole should be able to connect to HornetQ .

9. Appendix

1. **Software's referenced in this document and their versions**:
   HornetQ v2.3.0-Final and JDK v1.6 versions were used in phase-2; same versions of these software's are used for this prototype API.

   Apache Ant v1.9.2 is used for building the prototype API.
   SUN JDK1.6 should be used to run HornetQ and JUnit tests.

2. **Pre-requisites**
   The prototype API expects the software versions mentioned above in point#1 installed.

   JDK and ANT installation path upto bin folder should be added to PATH environment variable.

   For example on Windows machine if ANT and JDK are installed at following directories
   C:\Program Files (x86)\Java\jdk1.6.0_35
   C:\apache-ant-1.9.2

   Then following paths should be added to PATH environment variable.
   C:\Program Files (x86)\Java\jdk1.6.0_35\bin
   C:\apache-ant-1.9.2\bin

   Note: these paths are for reference purpose only; they might be different on different machines as per the installation path/directory chosen.

   Alternatively user can follow following approach
   Usually setting PATH environment variable is a one-time activity however if user wishes they can use the supplied file "setenv.sh" to set PATH variable, user needs to first open the setenv.sh file and update the following variables as per the installation directory on their machines.
   HORNETQ_HOME
   JAVA_HOME

ANT_HOME

After updating this script now can be used to set required PATH environment variable; However the PATH environment variable thus set is local to that shell/command prompt only, therefore user needs to execute this script in every shell window/prompt as a first activity before executing any functionality related to event service prototype, if user opens another new shell then setenv.sh needs to be executed again on that shell.

The scripts run.sh and setenv.sh are available in folder “BrokerScripts”