Thirty Meter Telescope (TMT)

Event Service Prototype – Implementation Notes

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

1. Purpose of this document

The purpose of this document is to explain in brief how each requirement is implemented.

2. Requirements

Please refer to section4 of document OSW TN010-EventServiceAPINotes_REL01.pdf

2.1. ES-1000 Use the HornetQ Core API

HornetQ Core API is used to develop the prototype.

2.2. ES-1002 Use the HornetQ configuration to set default parameters for Event Service

Prototype API reads HornetQ configuration properties from a file named configuration.properties and uses these properties to connect to broker; these properties are taken from phase-2

Following are the properties and their values

```
brokerHost=<IP Address of Broker Machine>
brokerPort=5445
tcpBuffer=700000000
tcpNoDelay=false
preAck=true
useNio=false
producerRate=1000
producerWindowSize=31457280
consumerWindowSize=-1
confirmationWindowSize=1310720
retryInterval=1000
retryIntervalMultiplier=1
maxRetryInterval=60000
reconnectAttempts=1000
useGlobalPool = false
threadPoolMaxSize = -1
scheduledThreadPoolMaxSize = 24
durableQueue=false
```

2.3. ES-1004 Do not use features that reduce performance.

Features that reduce performance like security, durable messages, transaction etc are not used.

2.4. ES-1006 Allow auto-reconnect of client to broker if broker goes down

This is achieved by following properties in configuration file-configuration.properties, prototype API reads these properties and uses them to connect to HornetQ.

```
confirmationWindowSize=1310720
retryInterval=1000
retryIntervalMultiplier=1
maxRetryInterval=60000
reconnectAttempts=1000
```

2.5. ES-1012 Provide a simple Bash script that starts the broker.

A script named "run.sh" is provided to start HornetQ, it will assume that HORNETQ_HOME variable is already set in PATH Environment Variable, JDK is installed on machine and JDK installation path is set to PATH variable.

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 variables by executing this file on a shell/command prompt, however the PATH variable thus set will be local to that shell/command prompt only, if user opens a new shell/command prompt then they will need to execute "setenv.sh" again in that shell, and this should be done before executing any functionality of event service prototype.

The scripts run.sh and setenv.sh are available in folder "BrokerScripts" & were tested on RedHat linux, please see user guide for details on this script.

2.6. ES-1013 Provide a single jar file for the broker if possible or necessary.

HornetQ standalone mode is used for the prototype, HornetQ binary is available as zip or tar.gz file, it needs to be unzipped on machine to install and run, packaging the HornetQ product as jar file is not required.

2.7. ES-1014 No sessions in the Event Service API

EventService Interface is provided, which is independent of any broker specific product API.

2.8. ES-1018 API should support callbacks (or Akka Java futures) when updates are received.

Callbacks are supported, API allows a consumer to subscribe to a topic with a callback class, and API will register a Topic Listener and spawns a new thread of Listener waiting for messages, which will notify the callback whenever new messages are received from HornetQ.

2.9. ES-1020 Consider generating a GUI that demonstrates both ends of service usage.

Standalone demo java programs are provided to showcase API usage. Programs would be interactive to support publish, subscriber and unsubscribe operations, one publisher program and two consumer programs are provided along with the code, these can be executed by running the provided bat/sh files.

Publisher Program can publish the event to a topic in HornetQ, two consumer programs can perform subscribe, unsubscribe and unSubscribeAll operations.

These programs are interactive and present multiple options to users to publish, subscribe, unsubscribe to a topic, these programs accepts user inputs from command line.

Detailed steps to run the demo publisher and subscriber programs are given in User-Guide document.

2.10. ES-1022 Use TMT fully qualified names as described in the SOW within the API for subscribing, posting and unsubscribing.

Fully qualified names like org.tmt.mobie.filter for Topic and Queues are supported by API for subscribing, posting and unsubscribing.

2.11. ES-1024: Messages as key-value, hierarchical map.

The API would internally use HashMap for event message communication. Publishers would pass Event objects to the API which would do the transformation to HashMap. Registered Callbacks would receive the Event objects after transformation from HashMap.

2.12. ES-1027: Event structure should include time stamp that indicates when the event was created and when the event was published. Event structure should include the source as a fully qualified name.

Event Message structure has two parts, message header and payload, message header includes the event creation timestamp, event publish timestamp and source.

2.13. ES-1028 Make callbacks asynchronous is possible.

Callbacks are asynchronous, API allows a consumer to subscribe to a topic with a callback class, and API will register a Topic Listener and spawns a new thread of Listener waiting for messages, which will notify the callback whenever new messages are received from HornetQ, , subscribers are not blocked to check new message arrival.

2.14. ES-1030 Provide a unit test suite that demonstrates API is working properly.

Test suite written in JUnit is provided to demonstrate the API, a list of Junit test cases are given in design document.

JUnit cases were tested with both the JUnit and HornetQ were running on Sun JDK 1.6

2.15. ES-1010 Consider the use of the management API to support diagnostics if easy.

JConsole can be used for this purpose, JConsole tool is a GUI based tool which comes with JDK1.6 and above, it can connect to HornetQ server running on local or remote machine and displays various useful information about HornetQ server to which it is connected like address, queues, consumer count, message count etc, steps to use JConsole are given in user guide.