APS Vertx TCP Messaging Provider

User Guide

Version: 1.0.0

Author: Tommy Svensson

Table of Contents

1 APSVertxTCPMessagingProvider

This is an implementation of *APSMessageService* that uses Vert.x to provide the TCP communication. It is dependent on the **aps-vertx-provider** bundle which provides reusable *Vertx* instances.

1.1 Configuration

This bundle exposes a service interface: `APSVertxTCPMessagingOptions`. This should be implemented and published as an OSGi service. This provider will look for an implementation of that service and configure itself accoding to the supplied configuration.

Here is the API:

```
    package se.natusoft.osgi.aps.net.messaging.vertx.api

    /**
     * This is a service API for providers of options to provide an implementation of.
     */
    interface APSVertxTCPMessagingOptions {

        /**
         * @return A Map containing Vert.x Net server options. Contents: name, value
         */
        Map<String, Object> getServerOptions()

        /**
         * @return A Map containing Vert.x Net client options. Contents: name, value.
         */
        Map<String, Object> getClientOptions()

        /**
         * @return Mappings between topic and the URI for the topic.
         */
        Map<String, String> getTopicToURIMapping()
    }
```

Since this is build on top of Vert.x it takes Vertx options Map:s. See the Vertx documentation for more info.

The topic to URI mapping is a simple mapping per topic what it should listen to for received messages and where it should send published messages.

Here is an example from the test code:

```
    Map<String, String> getTopicToURIMapping() {
        return [
                testOne:
"tcp://localhost:13987?inst=4#in,tcp://localhost:13988/?inst=2#out,tcp://localhost:13989
/#out",
                testTwo:
"tcp://localhost:13988?inst=2#in,tcp://localhost:13987/#out,tcp://localhost:13989/#out",
                testThree:
"tcp://localhost:13989/#in,tcp://localhost:13988/#out,tcp://localhost:13987/#out"
        ]
    }
```

Note that this is Groovy syntax, but it shold be self explanatory what is topic and what is URIs :-).

- There are commas between each URI.
- The protocol is always 'tcp'.
- The fragment part (#) has either an "in" or and "out". This is required. There should always only be one #in, and one or more #out.
- The only query is *inst=n* This is optional.

- For an #in *inst* number of services listening on host and port of the URI will be created. This is how Vert.x distributes service jobs across processor cores. There should be at least as many as there are cores if you want to utilize the processor to its fullest.
- For an #out there will be a pool of clients that will hold at most *inst* number of clients. This is for multiple threads to be able to send messages to the same topic. Each client instance will have its own connection to the receiving service. If the pool gets full and no clients are free then a new client will be created, and then deleted again. A warning will be displayed on the log in this case.

As can be seen in the example above, each topic has one service (#in) and 2 clients (#out), one for each of the other 2 topics. So these 3 topics send their messages to the other 2, whatever is sent by one the other 2 will get.