Tutorial: Howto send/receive messages over a persistent ROS Service in the Happy Artist RMDMIA:

In this tutorial we teach Java programmers how to send/receive messages over a persistent **ROS** Service (multiple messages over a single persistent service connection).

System Requirements:

- ROS
- ROS Turtlesim (Configuration instructions)
- Java 1.6 or greater
- *RMDMIA vpr1_v7 slipstream or greater*

Prerequisites:

- RMDMIA Client Configuration for ROS Turtlesim Tutorial

Overview:

This tutorial explains how to implement a persistent service message handler to a ROS Service with the Happy Artist RMDMIA (These steps are identical to a topic subscriber). The ROS Turtlesim is used in this example to demonstrate a persistent service implementation. A service is set to persistent via a checkbox in the ROS Configuration Manager. The ROSNode sendServiceMessage technique used in the non-persistent service tutorial technically works for a persistent service, multiple performance issues exist to using ROSNode sendServiceMessage. Specifically, the sendServiceMessage runs synchronized on the ROSNode thread, and that can hold up the ROSNode execution, and additional logic is implemented to determine if the request is for a persistent connection, or a single request/response increasing latency sending/receiving messages to/from ROS.

The following code demonstrates how to implement a Java custom message handler on a persistent service:

```
<code>
package my.example;
import
org.happy.artist.rmdmia.rcsm.providers.ros.client.transport.AbstractSubscriberMessageHandl
er;
public class ConsoleWriterMessageHandler extends AbstractSubscriberMessageHandler
{
    // Define the ROS Message Decoder to read byte[] to String
    ROSMessageDecoder decode = new ROSMessageDecoder();

    // Implement the process methods for incoming data
    public void process(byte[] message, int dataLength)
   {
        System.out.println("Hex Message (single-block): " +
org.happy.artist.rmdmia.utilities.BytesToHex.bytesToHex(message));
        System.out.println("Hex to Text Message (single-block): "  +
decode.convertHexToString(org.happy.artist.rmdmia.utilities.BytesToHex.bytesToHex(messag
e).toCharArray()));
   }

    // Implement the process methods for incoming data
   public void process(byte[] message)
   {
        System.out.println("Hex Message (multi-block): " +
org.happy.artist.rmdmia.utilities.BytesToHex.bytesToHex(message));
        System.out.println("Hex to Text Message (multi-block): "  +
decode.convertHexToString(org.happy.artist.rmdmia.utilities.BytesToHex.bytesToHex(messag
```

e).toCharArray()));

}

}

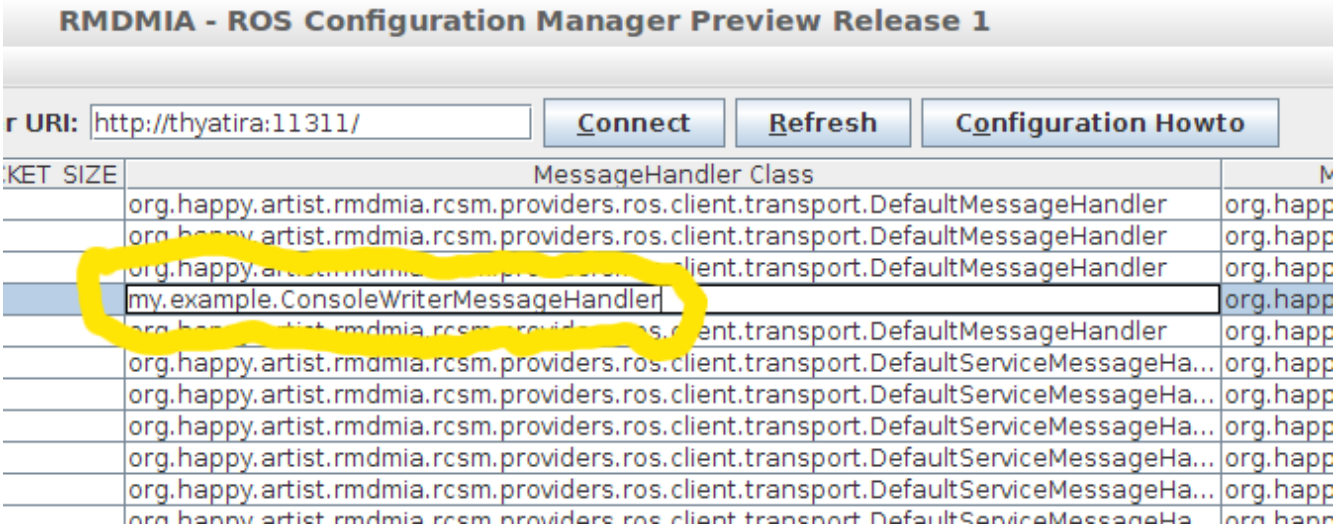
</code>

To test the above class:

1. *compile, and add the generated ConsoleWriterMessageHandler package structure with a class file to a jar file.*

2. *Copy the Jar to the RMDMIA plugins/rcsm directory.*

3. *Update the MessageHandler class for the associated service in the ROS Configuration Manager, and save/exit. Try the /rosout_agg topic for log messages.*



4. Launch the RMDMIA, and the new message handler will load.