**Example SNMP Edge Micro Server Implementation**

* **Source Code -**

Library used: SNMP4J - <http://www.snmp4j.org/>

**Purpose**: trapping SNMP events from an edge network, through a JAVA SDK EdgeMicroServer implementation then forwarding the trap information to the ThingWorx server.

**Background:** There are devices, like network elements (routers, switches) that raise SNMP traps in external networks. Usually there are third party systems that collect this information, and then you can query them, but if you want to catch directly the trap, you can use this starter-kit implementation.

**Architecture:**

The developed EMS contains a SNMPManager implementation that listens for SNMP traps on localhost – 127.0.0.1 on port 162. The SNMPManager class is responsible for gathering all the information from network devices that use SNMP. In order to generate SNMP traps for testing this implementation we used an opensource SNMP simulator.

If we discuss the EMS specific terminology, then we can say that we used:

1. A single VirtualThing We register the VirtualThing implementation class as a received for SNMP messages.
2. A single ConnectedThingClient. We use a Factory design pattern to initialize the client and return an instance of the newly created object.

The functionality of the SNMPManager can be customized by modifying or extending the SNMPManager class that is found in the /src directory. It provides a basic constructor

public SNMPManager(String UDPaddress) throws IOException, InterruptedException

that initialiazes the TransportMapping for the provided updAddress, sets the SNMP to version 3 and registers a trapHandler, that prints and sends a callback to a registered listener. The listener, SNMPManagerTrapListener, should be set using the setter method that is exposed on the SNMPManager class.

The functionality of the SNMPManager **can be extended to include interogations to the SNMP Agents or any of the 7 request types the protocol can handle.**

The design of the project is loosely coupled in order to be easily adaptable and extensible.

On the Thingworx Server side when installing you need to define a new appKey and a new Thing of type RemoteThing*.* When instantiating the SNMPManagerThing the 3rd parameter is the identifier used to bind the bind to the platform.

You will be able to receive the Trap information in 2 ways:

1. **Using the Trap property**, which holds the full to String conversion of the command received.
2. **Using the TrapTriggered event**, which has the capability to offer separate information (Message, OID, Timestamp etc.)

Everytime the SNMPManager receives a trap it will call a TrapHandler that is responsible of sending the information to the ThingWorx Platform.

Building:

This is a development example. It contains all the necessary files to build the project using most Java IDE’s.

Running the example:

1. Import the entities found the ***./entities*** folder into the ThingWorx instance.
2. Open your prefered **IDE** and create a **new project from existing sources** and add all the files and folders present.
3. Configure your project to use the ThingWorx EMS Java SDK by going to ***Project Structure >> Libraries >> Add Libraries >> [Add all the .jar files located in your SDK’s lib file]***.
4. Create a **new appKey** on the ThingWorx instance and edit the content of ***apikey*** variable in Main.java.
5. Modify the address of the ThingWorx server on the Main.java file.
6. Run the ***main*** function to start the example, with a default ***Run Configuration***.
7. The example accesses a ThingWorx example running on ***localhost:8002*** over HTTPS, and listens for SNMP traps coming from ***localhost***, this being the default configuration when running a SNMP Simulator.
8. Access the ThingWorx Instance and create a ***new Thing*** that extends the ***RemoteThing*** template and save it.
9. Access the ***Monitoring>>RemoteThing*** menu in the ThingWorx instance. Under the ***Unbound tab*** you will see a Thing with the identifier of SNMP1, or another identifier if you modified the constructor call.

Now you can choose either one, or both ways to access the trap message:

1. Property based: Go back to the Thing previously defined and ***under properties***, add a ***new property with remote bindings*** from the Manage Bindings section, and edit the property so that it is logged***.***
2. Event based: Under ***Events*** click on ***Browse Remote Events*** and add the available events. After you have added them, go into ***edit mode*** and select the ***TrapDataShape*** to be the datashape used.