B: Raspberry Pi Java Client

Prerequisites:

- Installing the Java Runtime Environment
- Installing Ignition
- Installing the following MQTT Modules
    - MQTT Distributor
        - v3.1.0 or greater if using Ignition 7.9.X
    - MQTT Engine
        - v3.1.0 or greater if using Ignition 7.9.X
- Downloading the Sparkplug Sample Code onto a development system

Overview:

Sparkplug is an open source project developed by Cirrus Link Solutions which shows how devices or projects can be enabled to communicate with MQTT Engine and Ignition. This example will show how data can be published via MQTT from an emulated device running on a development machine. In addition, it will show how devices or projects can be controlled by writing to tags in Ignition. It will also show the caveats associated with establishing /ending an MQTT session and ensuring that the tag values in Ignition are valid.

Raspberry Pi Java Client:

This tutorial assumes:

- Ignition is running and in active trial mode or using a purchased license.
- MQTT Distributor is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- MQTT Engine is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- You have a Raspberry Pi 2 Model B running Raspbian Linux.
- You have a Pibrella I/O board.

With the Sparkplug sample code downloaded onto your development machine, change into the directory and build the code. In order for this to work you must have a Java Development Environment and Maven installed. Also, this example assumes the MQTT Server running is MQTT Distributor running with it's default configuration. If you are using a different MQTT Server, edit the following file to reflect your MQTT Server configuration:

- sparkplug_b/raspberry_pi_examples/java/src/main/java/com/cirruslink/example/SparkplugRaspberryPiExample.java

Since the IP address of the machine running the MQTT server (MQTT Distributor) is dependent on your network setup, the serverUrl must be changed in this file. Also, if not using MQTT Distributor you may need to modify the username and password. For simplicity this example does not use or support TLS over MQTT without modifications.

With the above steps completed, run the following commands to build the application:

cd sparkplug_b/raspberry_pi_examples/java/
mvn clean install

Now with the application built it must be transferred to the Raspberry Pi. This can be done at the command line with scp:

scp target/sparkplug_b_raspberry_pi_example-1.1.3-SNAPSHOT.jar pi@[RASP_PI_IP_ADDRESS]:/tmp/

Example if the Raspberry Pi IP address is 192.168.1.100:

scp target/sparkplug_b_raspberry_pi_example-1.1.3-SNAPSHOT.jar pi@192.168.1.100:/tmp/

Default Raspbian password is: raspberry

With the application now on the Raspberry Pi, it can be run as follows:

java -jar /tmp/sparkplug_b_raspberry_pi_example-1.1.3-SNAPSHOT.jar

At this point, the application will start, connect to the MQTT server, publish a Edge Node Birth Certificate, publish a Device Birth Certificate, and begin periodically reporting random data values to Ignition via MQTT Engine. This can be verified via Ignition Designer. Using a Web Browser, browse to the Ignition Gateway on your Ignition Gateway. If it is running on your development machine, that is: http://localhost:8088. You should see this:
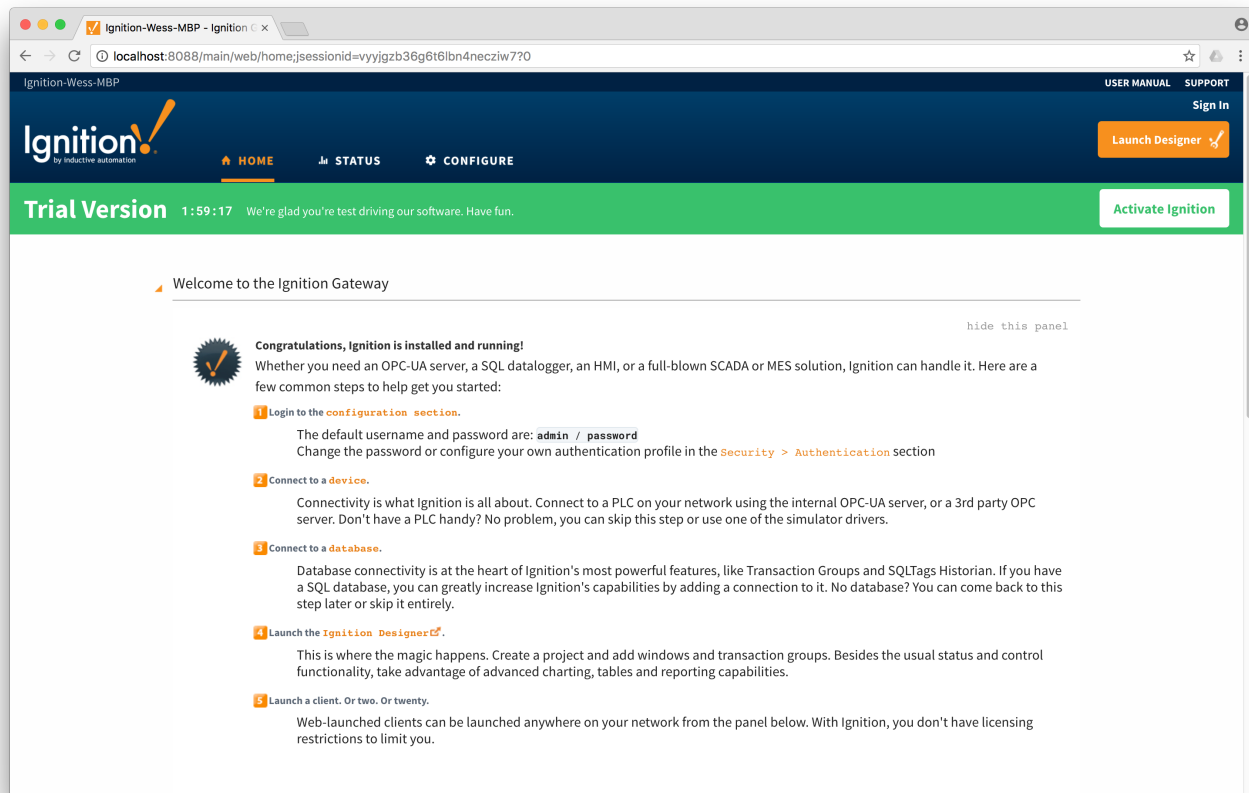
Near the upper right corner, click 'Launch Designer'. This will open the following window after downloading the .jnlp file and executing it. Note the default username/password is admin/password. Type those into the appropriate fields and click 'Login'.

This will bring you to a new Window where you can select an Ignition Project or create a new one. Create a new project by giving it a name and clicking 'Create New Project'.

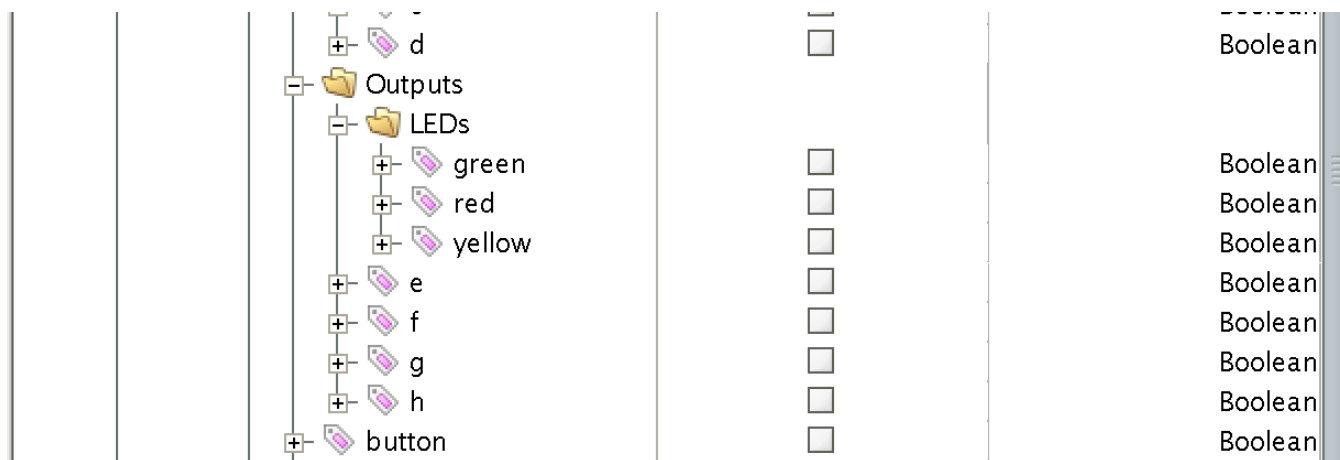Now you should be in designer. In the left hand side of the main window is a 'Tag Browser' window. In it, expand 'All Providers -> MQTT Engine -> Edge Nodes -> Sparkplug B Devices -> Java Raspberry Pi Example -> Pibrella'. You should see the following:
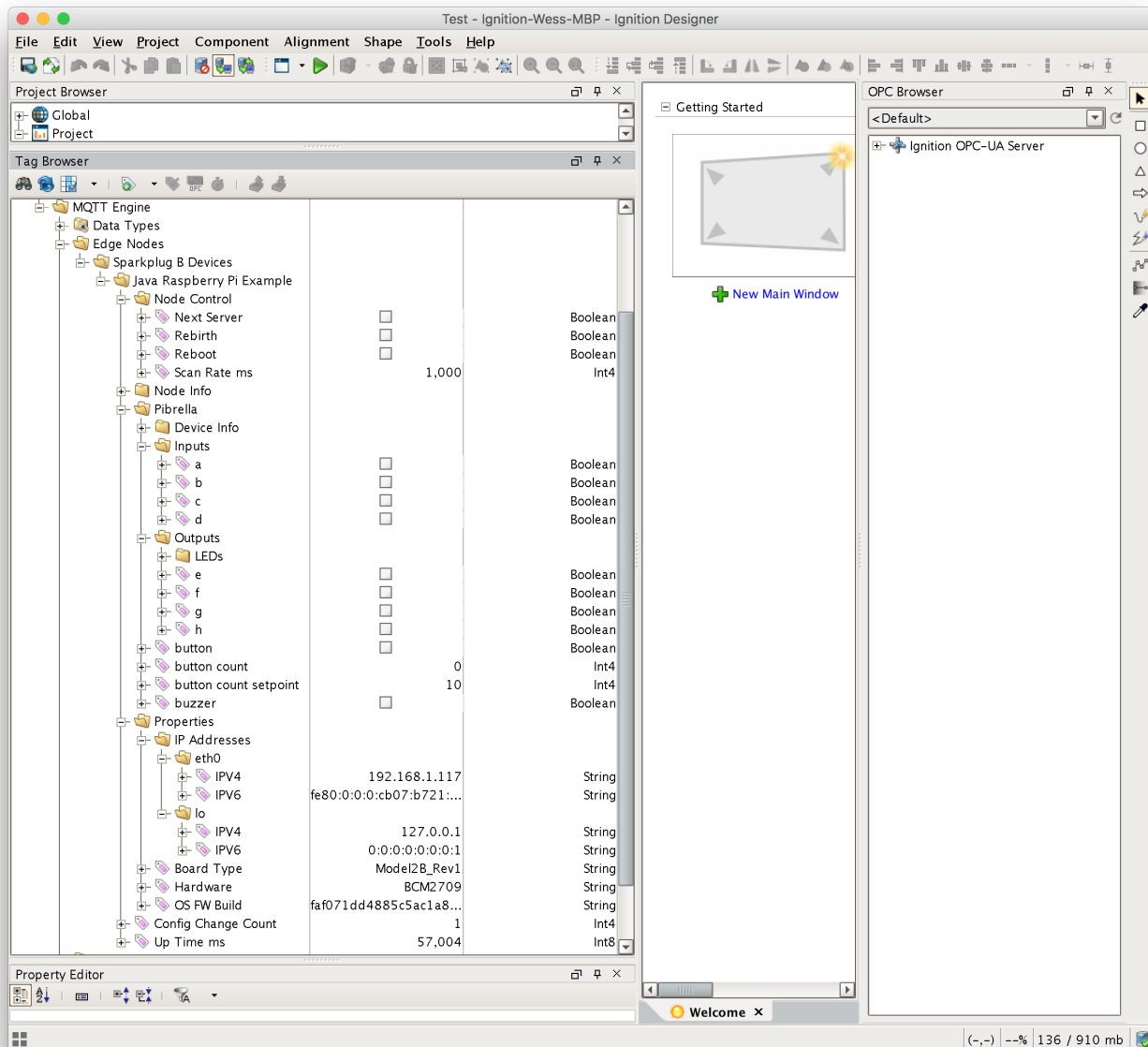
You will see that MQTT Engine saw a new device attach to the MQTT Server and publish a Birth Certificate. As a result, MQTT Engine created the Ignition Tags shown above. These are also dynamically updated as the values change. You can also write to the outputs after you Enable Device Writes from Ignition. This can be done by putting designer into read/write mode. Do so by clicking this button in the menu to enable read/write mode:

Then you can change any of the values on the outputs here:

```
       d                          □              Boolean
   Outputs
       LEDs
           green                  □              Boolean
           red                    □              Boolean
           yellow                 □              Boolean
       e                          □              Boolean
       f                          □              Boolean
       g                          □              Boolean
       h                          □              Boolean
   button                         □              Boolean
```

When you click on of the outputs, you should see the output on the Pibrella board change to reflect the state. Also, if any inputs change on the Pibrella you should see them update in the Tag Browser in the Ignition Designer. If you are not seeing the output and input values update to reflect the change you have made, make sure MQTT Engine is not configured to block outbound device tag writes as described here.