|  |
| --- |
| Connecting to M2X using LuvitRED |
|  |



**Franco Arboleda**

**27-Apr-16**

Table of Contents

1 Introduction 3

2 Configuring the M2X platform. 4

2.1 Creating a new device on a new account 4

2.2 Creating a new device on an old account 5

2.3 Adding a data stream 7

3 LuvitRED configuration. 8

3.1 Sending data to the M2X platform. 9

3.2 Receiving commands from the M2X platform. 15

# Introduction

This document explains how to send and receive data to and from AT&T’s M2X platform using LuvitRED.

This document was written using CloudGate’s firmware version 2.58.0 and LuvitRED version 2.7.5. Although older versions of firmware and LuvitRED might work in the same way, we strongly recommend to upgrade to the above mentioned versions or newer in order to ensure the same results.

For this explanation we are using a free account on AT&T’s M2X platform (<https://m2x.att.com/>).

# Configuring the M2X platform.

## Creating a new device on a new account

When creating a new account on the M2X platform, the user is prompted to create a new device. This section explains how to setup a new device in this case.

1. On the left hand side, click on Device:

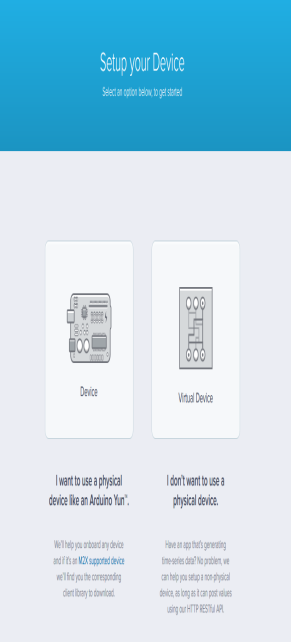


Figure 1: M2X: Device.

1. Name your device (e.g. CloudGate) and click on Save:

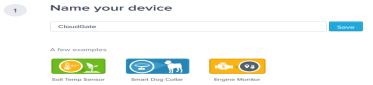


Figure 2: M2X: Name your device.

1. Do not set any data stream yet, instead, go to the bottom of the page and click on “Go to Device”:

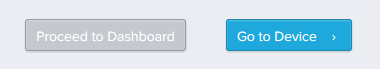


Figure 3: M2X: Go to Device.

1. Once on the newly created device page, take note of the DEVICE ID and PRIMARY API KEY of the device:

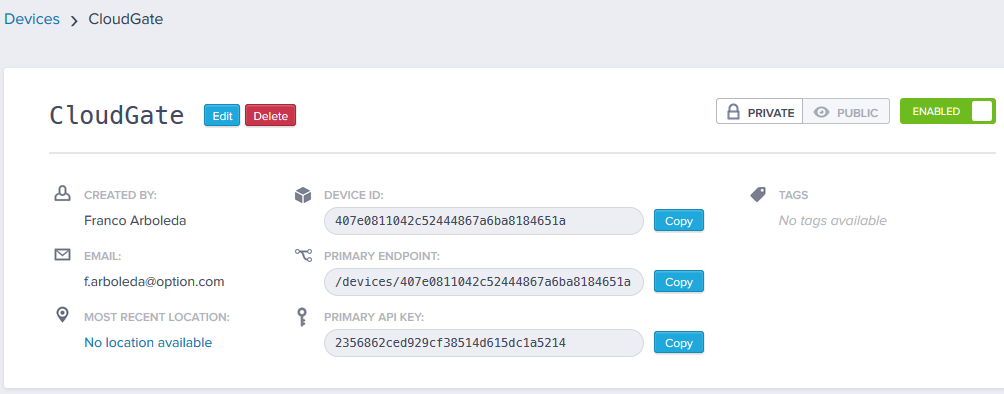


Figure 4: M2X: DEVICE ID and PRIMARY API KEY.

## Creating a new device on an old account

1. Login to your account and go to Devices. Then click on Create New and select Device:

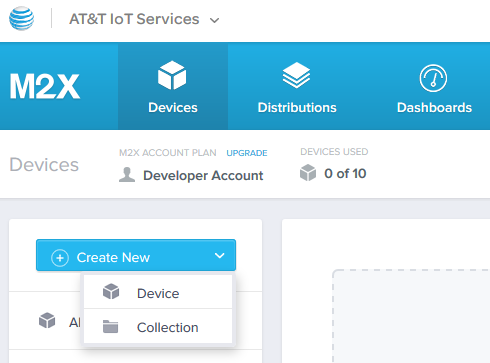


Figure 5: M2X: Create New - Device.

1. Provide a Device Name and make it a Private Device. Click on Create to finalize:

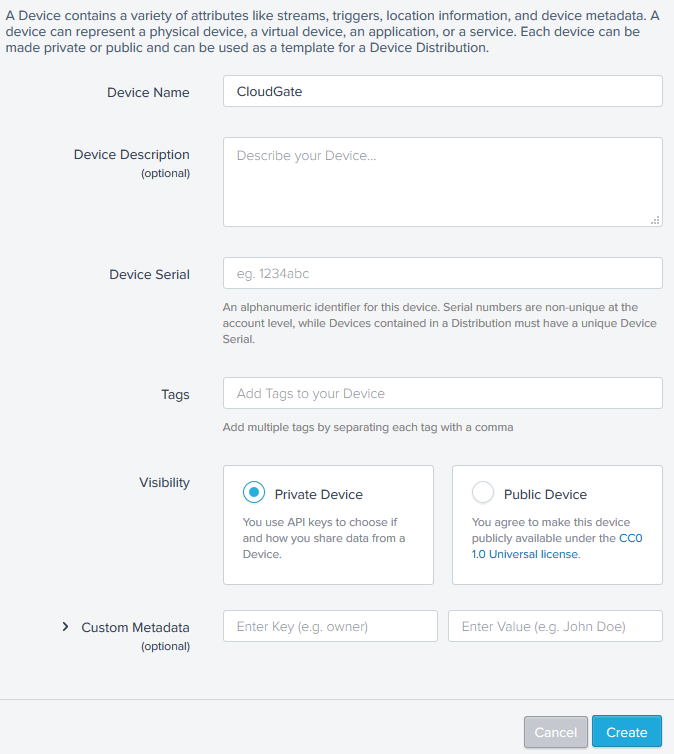


Figure 6: M2X: Device setup.

1. Take note of the DEVICE ID and PRIMARY API KEY as explained on section 2.1, step 4.

## Adding a data stream

1. Add a new Stream under the Overview section:

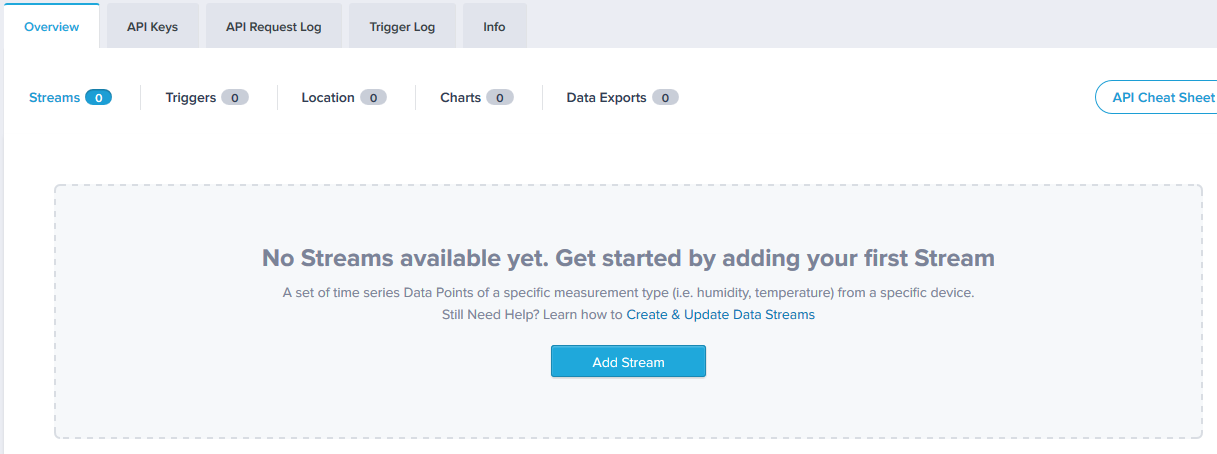


Figure 7: M2X: Add Stream.

1. Provide a Stream ID (rand\_num1) and Stream Type (Numeric) to the new stream and click on Save:

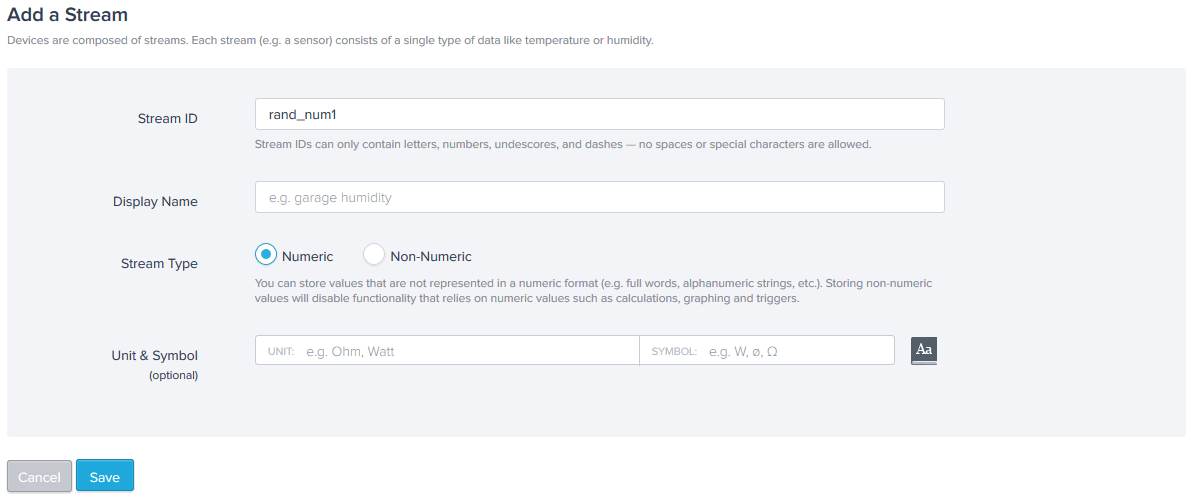


Figure 8: M2X: Add a Stream.

1. A new Stream called ***rand\_num1*** is created under the Stream section:

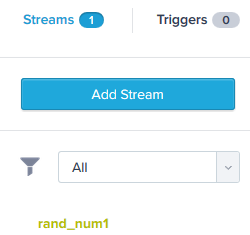


Figure 9: M2X: New Stream.

# LuvitRED configuration.

Go to the web interface of the CloudGate and then to the "Plugin" tab and sub-tab called "LuvitRED"



Figure 10: Plugin tab, LuvitRED.

**NOTE:** Do not focus on the SNMP (Simple Network Management Protocol) sub-tab. This tab is not going to be used on this document.

Under the “Advanced Editor” of LuvitRED, there are two nodes that are related to M2X (See Figure 11):



Figure 11: M2X nodes.

## Sending data to the M2X platform.

1. Drop a M2X out node:



Figure 12: M2X out node.

1. Configure the M2X out node in the following way:

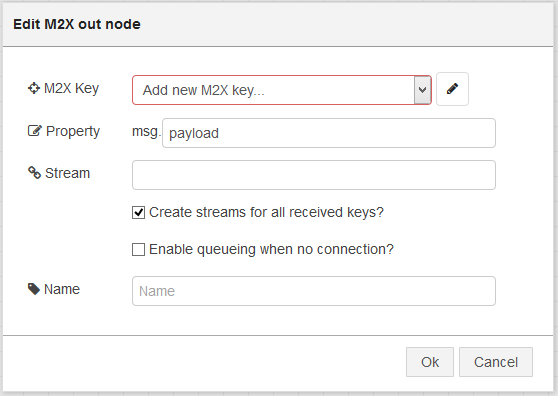


Figure 13: Default M2X out node.

* 1. Add a **new** M2X Key by clicking on the pencil icon and configure it for the DEVICE ID (Device ID) and PRIMARY API KEY (API Key) noted on the previous section. Change the “Name” of the configuration node and click on “Add”:

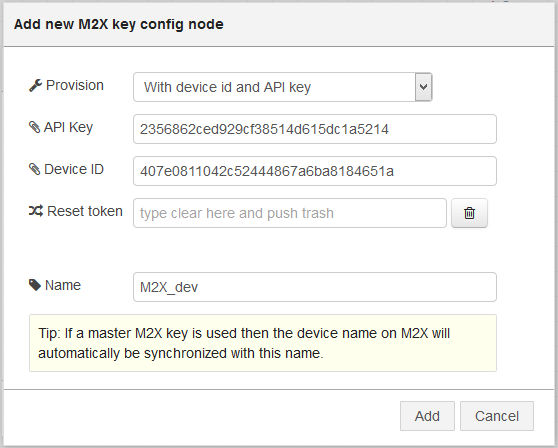


Figure 14: Adding Device's ID and API Key.

* 1. On the node configuration write the “Stream” name created on the M2X platform (***rand\_num1***), uncheck the “Create streams for all received keys?” and change the name of the node. Click on “OK” when finished:

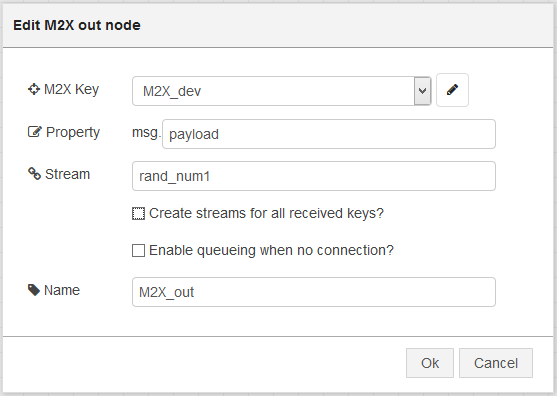


Figure 15: Final node configuration.

This node is now configured to send one data stream that will be called ***rand\_num1***. Now we need to actually send data.

1. Drop an inject node and configure it so that it outputs a random integer (0-100) every 10 seconds (give the node a name):

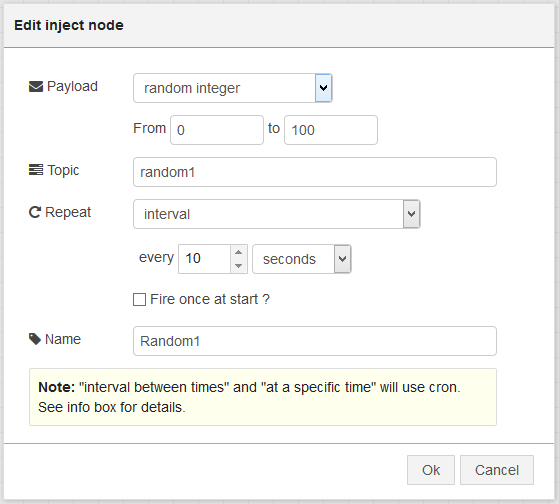


Figure 16: Inject node configuration.

1. Let’s connect the nodes together in the following way:



Figure 17: Final Configuration.

1. Click on “Deploy”

If we now go back to our device on the M2X platform, we should be able to see that the rand\_num1 stream is being updated with the values that our CloudGate is sending:

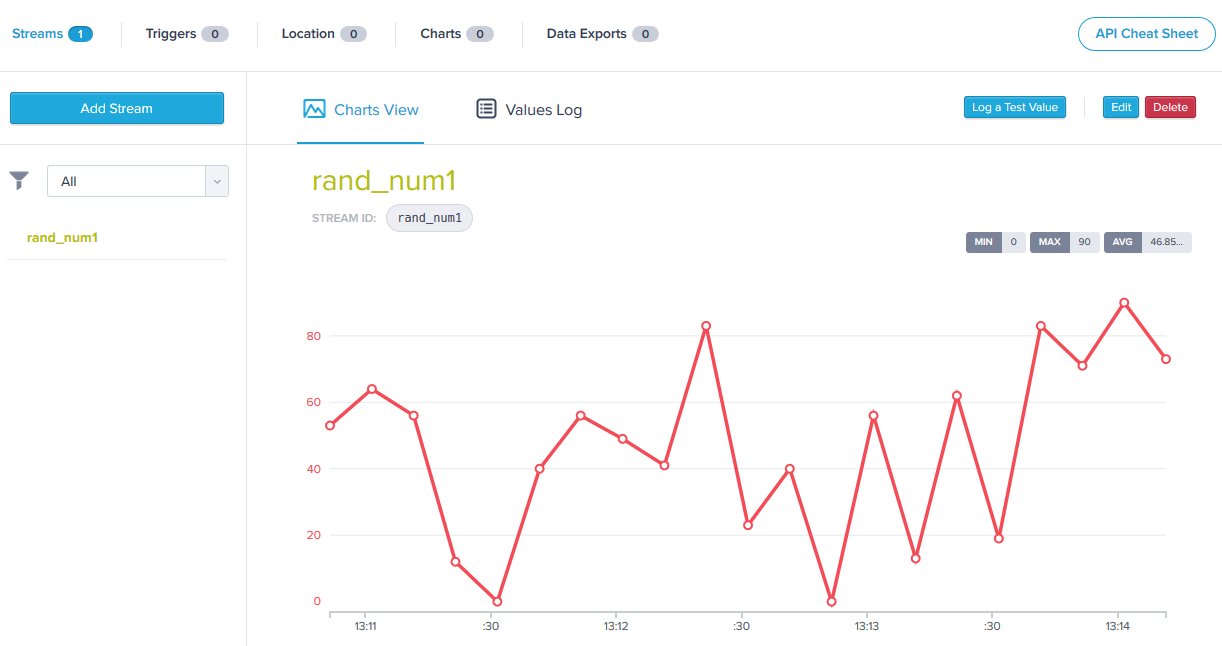


Figure 18: M2X: rand\_num1 stream item updated.

Multiple streams can be added by replicating the above steps we used to add ***rand\_num1***, but this is only a good solution for a reduced amount of streams. If we need to post data for a hundred streams, this solution is not the best. The M2X out node can send information for multiple streams as long as this information is provided to the node as a table in which case each key of the table need to be the name of the stream we want to update (See Figure 19):



Figure 19: Extract of the information tab of the M2X out node.

Let’s adapt the configuration we already have so that we can post data for multiple streams using one single M2X out node:

1. Open the M2X out node and delete “Stream” name (Remember: the stream name is now going to be provided as the key of the table). Check the “Create streams for all received keys?” option (This allows us to create new streams on the M2X platform without the need to manually adding them) and click on OK:

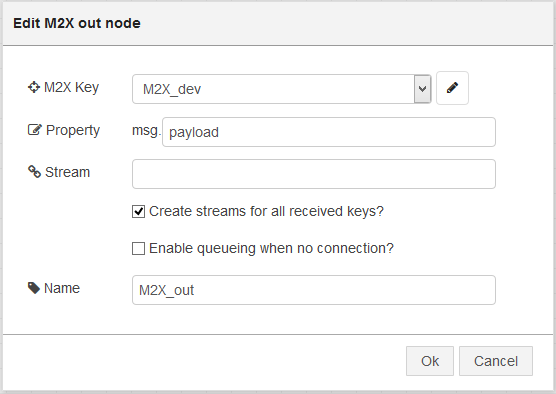


Figure 20: M2X out node with no Stream name.

1. Now, lets drag and drop a second inject node and configure it as follows:

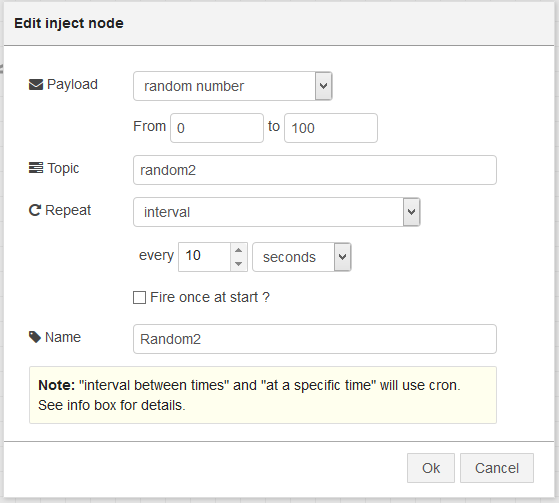


Figure 21: Random2 inject node (random number).

**NOTE:** We can now use a combine node to combine the messages coming from both nodes into one message and using their topics as keys.

1. Drag and drop a combine node and configure it the following way:

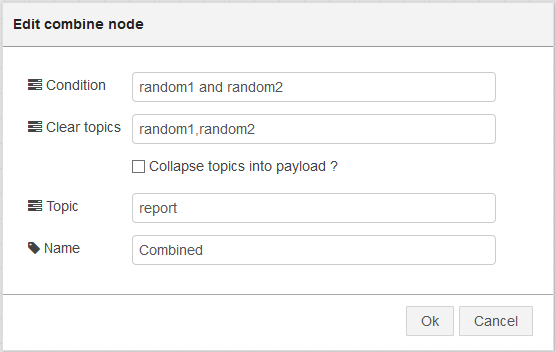


Figure 22: Combine node.

**NOTE:** This configuration does the following:

* The condition is to wait for both random1 and random2 topics to arrive.
* After they arrive, clear the topics, so that the node waits for two new values to come.
* It changes the final topic to “report”

1. Drop a debug node (This is just to show the format of the data after the combine node).
2. Let’s connect the nodes together in the following way and click on “Deploy”:

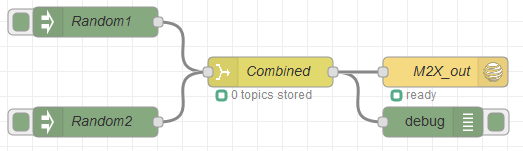


Figure 23: Configuration using combine and one M2X out node.

On the debug node, we should be able to see the table created by the combine node containing both random1 and random2 values:

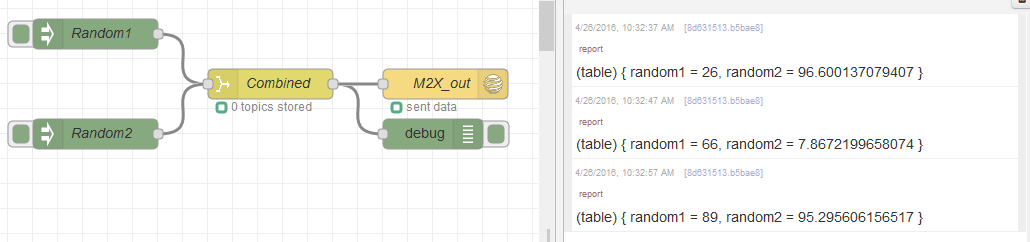
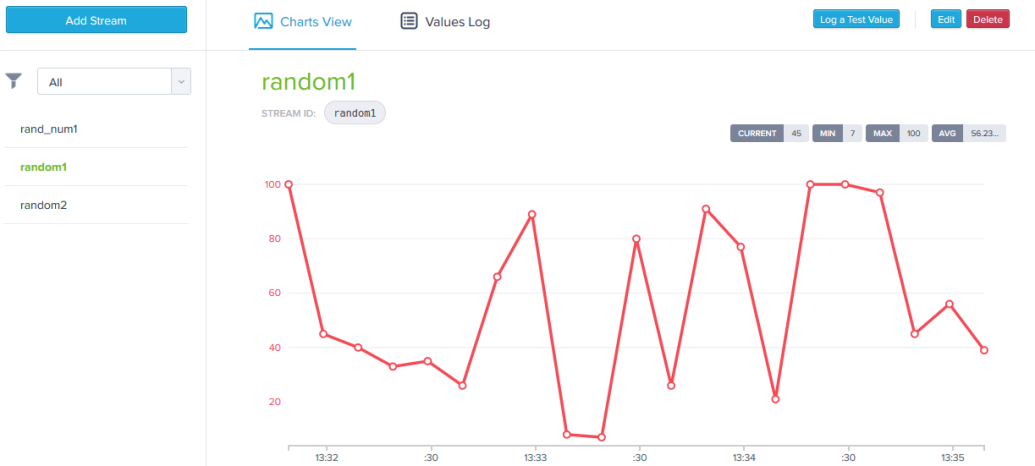


Figure 24: Table of data shown under debug tab.

On the M2X platform we should see the values being updated (Notice that the two new streams are added into the platform):



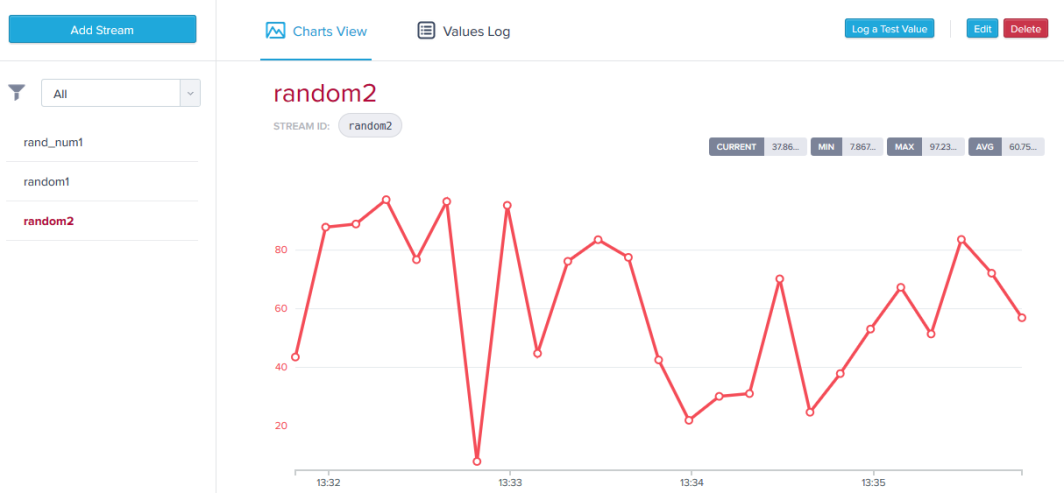


Figure 25: M2X: New Streams created and values updated.

**NOTE:** Since we are now creating the Streams directly from LuvitRED, we could delete the ***rand\_num1*** stream from the platform without affecting the configuration.

## Receiving commands from the M2X platform.

We have been successful in sending data from our device to the M2X platform, but how about receiving commands on our device? M2X provides a method of sending commands to the device. Each received commands needs to be marked accordingly to let the M2X platform know that it has been processed or rejected. For more information about the M2X commands, got to <https://m2x.att.com/developer/documentation/v2/commands>.

1. Drop an M2X in node:



Figure 26: M2X in node.

1. Configure the M2X in node in the following way:

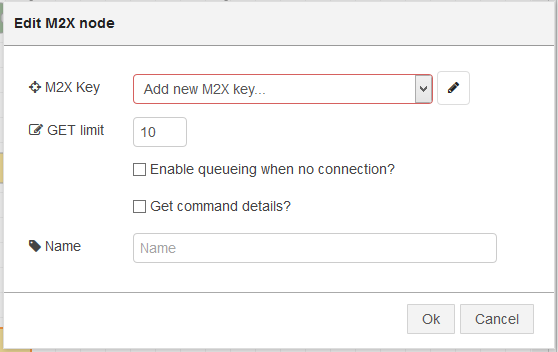


Figure 27: Default M2X in node.

* 1. Select the already configured M2X Device. If no device is already configured, please follow the steps explained on section 3.1. Check the “Get command details?” option and change the name of the node:

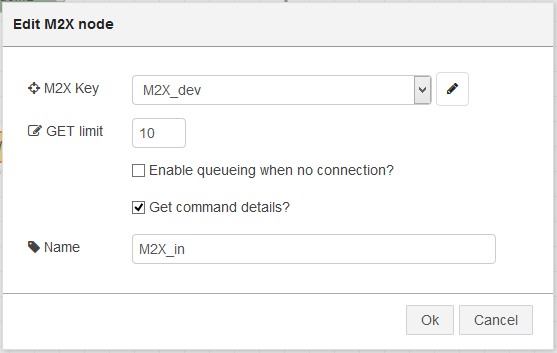


Figure 28: M2X in node configuration.

1. Drag and drop an inject node and configure it as follows:

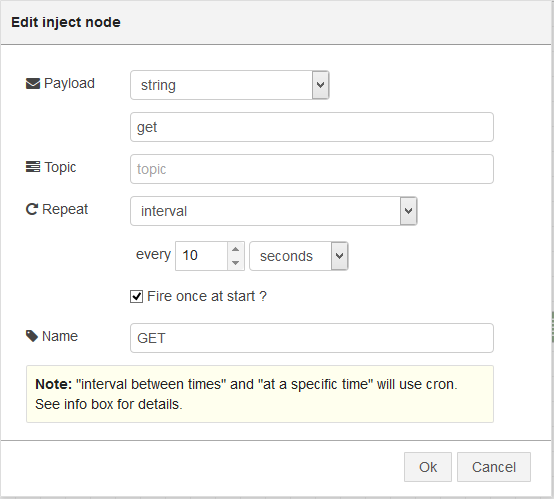


Figure 29: Inject node configured to send "get" command every 10 seconds.

Keep in mind the information on the M2X in node:

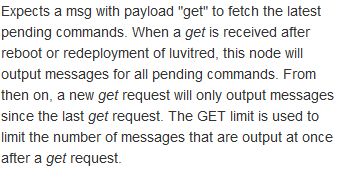


Figure 30: Extract of the information on the M2X in node.

1. Drag and drop a change node and configure it to set the status of the received command to processed:

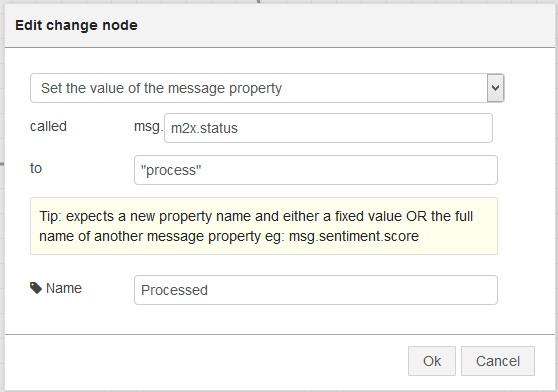


Figure 31: Change node configured to set the status to process.

Keep in mind the information on the M2X in node:

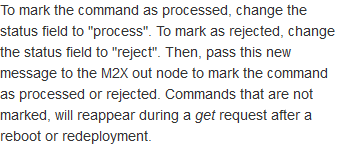


Figure 32: Extract of the information on the M2X in node.

1. Drag and drop two debug nodes, but we need to reconfigure the Output from “payload only” to “complete msg object” in order to see the m2x command information:

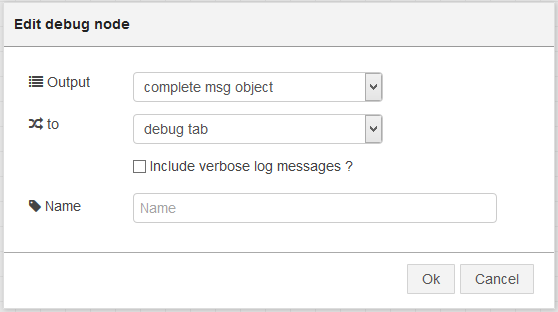


Figure 33: Debug node output configuration.

1. Let’s connect the nodes together in the following way and click on “Deploy”:

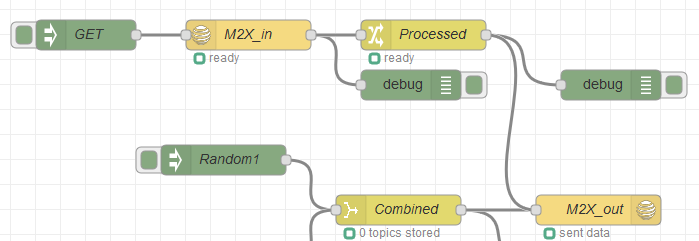


Figure 34: Final configuration.

On this configuration we are reusing the M2X\_out node from section 3.1 in order to send the “process” command back to the M2X platform.

Now, let’s send a command to the CloudGate. For this, we can use a command line tool called “cURL” to send commands to a Device using the M2X platform. We are going to test using cURL on from a Linux machine. The following command will be send:

***curl -i -X POST https://api-m2x.att.com/v2/commands/ -H X-M2X-KEY: d64c59e39dbac0cef0d45a958685aea0 -H Content-Type: application/json -d {"name":"update", "data":{"key1":"value1","key2":"value2"}, "targets":{"devices":["407e0811042c52444867a6ba8184651a"]}}***

* X-M2X-KEY: Is the account MASTER KEY (found under the account settings).
* data: is the JSON formatted data sent to the devices
* devices: Is a list of the DEVICE IDs that should receive the command.

**NOTE:** For more information about the structure of the command and the information contained on it, please refer to the M2X documentation.

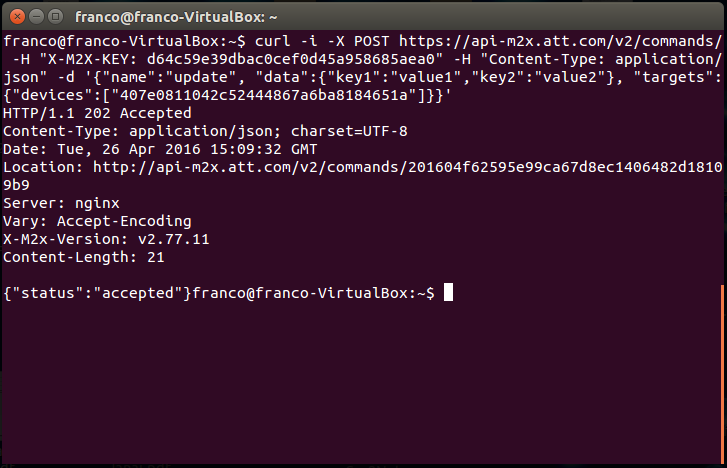


Figure 35: M2X command sent from a Linux machine.

The new data should be shown on the Debug tab in LuvitRED:

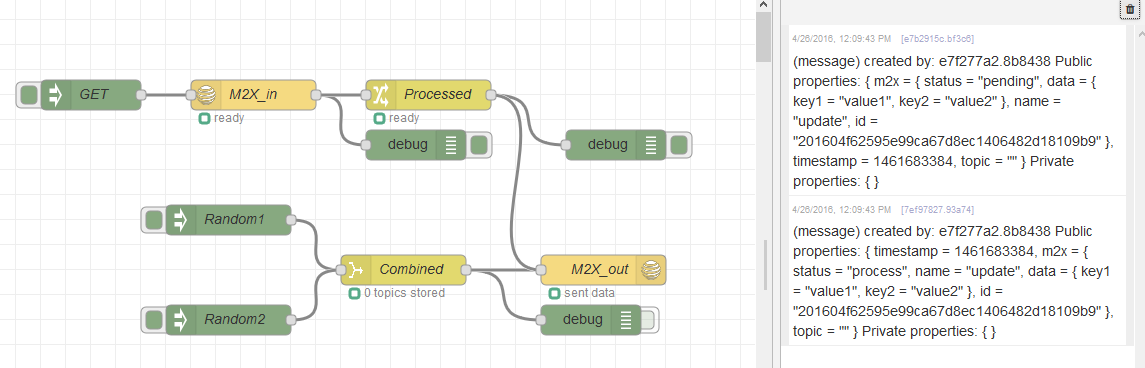


Figure 36: Command coming from M2X.

The actual data looks like this:

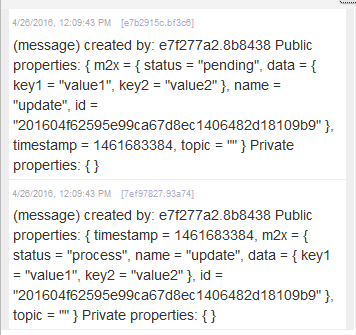


Figure 37: Commands coming in and going out.

The first line on the debug is the command coming from the Linux computer and it contains the M2X command status set to ***pending***, and the data set to ***Key1=”value1”*** and ***Key2=”value2”*** (these items are setup on the command we sent from the Linux computer).

The second line on the debug tab is the command response we are sending out on the M2X\_out node, and it contains the M2X status set to ***process***. In this way, we are telling the M2X platform that this specific command was processed by us. Of course, in this example, we are not using the command information to perform any action (we are just printing the command), but we could do something with the information first and then set the command to processed.