AT&T IoT Starter Kit

LTE Shield, M2X and AT&T Flow

Getting Data from the IoT Starter Kit to the Cloud With AT&T IoT Platforms Flow Designer and M2X

AT&T cellular shield featuring WNC LTE Module with NXP developer board. Hardware configurations may vary.

This guide will show you how to get the Starter Kit talking to the cloud based time-series data storage (M2X) and AT&T Flow application service.

If you don't have the Starter Kit hardware you can purchase it at starterkit.att.com.

This guide includes a virtual device to use in testing. All the code is open-source and ready for you to use and expand.

Getting started

Create an AT&T IoT Services account

IoT Developer Services account is free and takes moments to setup up. It will give you access to IoT Starter Kit, M2X, AT&T Flow, and cellular connectivity API access, if you purchase an IoT Starter Kit hard- ware.

IoT Starter Kit

You have a choice between purchasing a SIM Card only, or a complete hardware kit with cellular shield and LTE module built-in. Both options come with a data plan and a connectivity API.

All you need to get up and running fast.
To purchase IoT Starter Kit, or sign-up for an AT&T IoT Services account, go to: starterkit.att.com

M2X

Navigate to https://m2x.att.com.
It comes to you free for up to 10 devices.
At its heart, M2X is a time-series data store – but it's also more, including a device manager, authorization manager, and dashboard widget builder. We'll come back to it soon.

AT&T Flow

Select "Flow" from the "AT&T IoT Services" dropdown in the top-left, or navigate to https://flow.att.com. Flow is GUI-based IoT development tool based on NodeJS with many ways to get data in and out of it, as well as implement business rules – all with version control, teams, and scalable application containers.

AT&T Flow gives you up to 100,000 data points per month free of charge.

If all this is new to you might check out https://flow.att.com/start. Login with the same credentials you used for your AT&T Developer account.

Setting up M2X

The AT&T M2X Data Service can receive, store, and react to data from a data source. In order to prepare the service for your data source, the first step is to set up a Device, which is a way to prototype your Device before launch.

Follow these steps to set up a Device:

1. Go to https://m2x.att.com/onboarding.

2. You will be asked to set up your first Device as either a physical device or a virtual device. If you choose physical, it will assist you in finding the correct client library. However, we will provide you with client library for this event. Therefore, click on Virtual Device.

3. The first time you log in to the M2X website, you can be led through the first-time process of creating a new Device and Stream. However, for this exercise, we will use the more standard procedure. In the top right corner, click on Skip Setup.

4. Click on Create Device.

5. For Device Name, type “test”. Leave everything else the same. Click Create.

6. Your Device will have a Device ID and API Key associated with it. You will use these later.

7. Let’s say that you were collecting information on ambient temperatures, and you wanted to keep track of the temperature for each asset as a function of time. Let’s add a data stream for device. Scroll down and click on the Add Stream button.

8. Type in “temp” for the Stream ID, “Temperature” for the Display Name, leave the Steam Type as Numeric, and then click on the **Aa**

button.

9. Click on the **D**, then scroll down and select “degrees Celsius”.

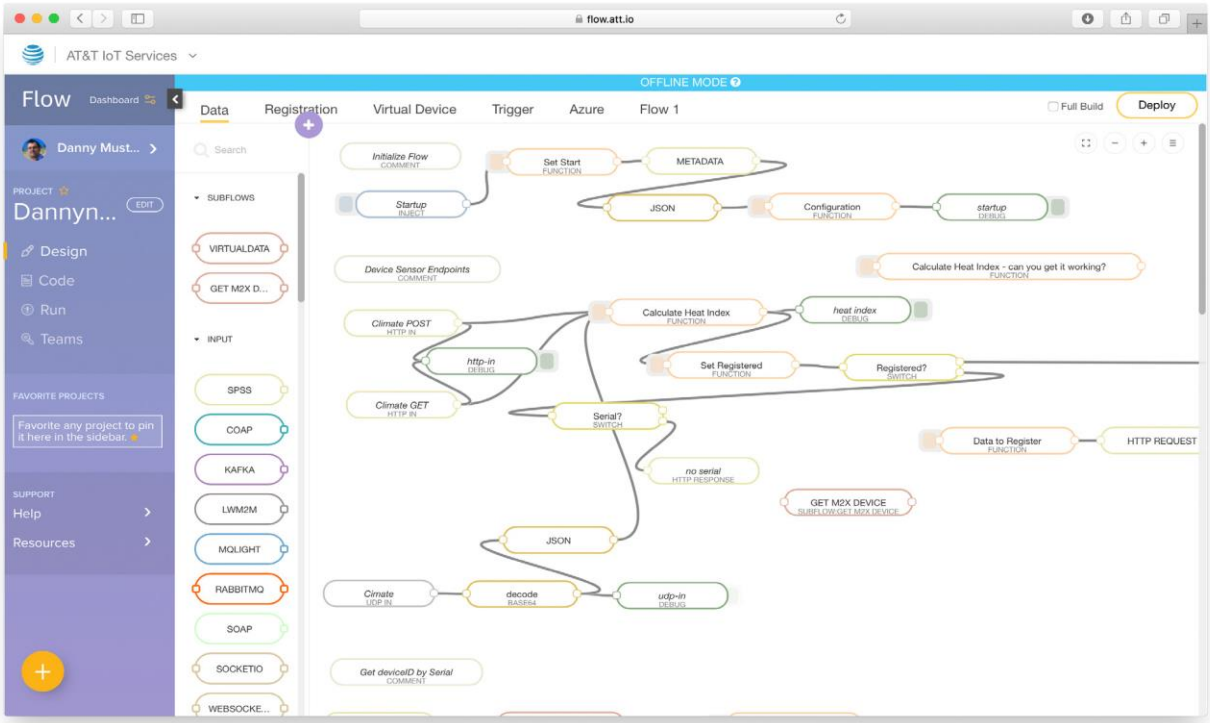
10. Click on Save to create the stream.

11. In the Overview tab you’ll see a link to Streams followed by Triggers. Click on Triggers and then on the “Add Trigger” button.

12. Give the trigger a name like “Hot Temp”, select Temperature from the Stream drop-down, “>” from the Condition drop-down, enter a threshold value of 40 (remember, this is in Celsius), and reset condition of 37. Leave the Notification Frequency set to Single and click on the Save button at the bottom.

Setting up AT&T Flow

You now have a Device, a stream, and a trigger. Next, you’ll send some data to that Device. Follow these steps to create a flow in Flow Designer to talk to M2X. Projects contain the flows that together make your application.



Let's start by creating your first project: This step is to make a simulated value of a specific stream.

1. In a new tab navigate to https://flow.att.com/ or select Flow from the AT&T IoT Services drop-down at the top of the page.

2. Click Login. As long as you are signed in with M2X, you will be signed into Flow Designer. (If not, use your M2X username and password.) If this is the first time you've logged into Flow Designer, a new Flow Designer account will be created for you.

3. Click the + button to create a new project.

4. Give your project a name, something like 'My first project'. You can also add a description if you like, 'Testing Flow Designer' for example.

5. Keep Visibility set to private (you can always change it later). Internal or Public visibilities allow others to see your project, clone it, etc. and is great for collaborative work. For now, we'll keep this first project Private.

6. Click "Create". The system will import the required template files and will set the environment for you. Once done, your new project is in 'Design' mode, ready for you to write your first flow.

7. Your project was created with a default first flow called Flow
   1. From the node palette on the left grab an Inject node from the Input section (you can find it easily by scrolling down or typing in "inject" into the Search box at the top of the palette) and drop it onto the canvas. By default this simply sends a Unix timestamp to the next node. Inputs for nodes connect on the left and outputs connect on the right so that the flow, itself, reads from left to right.

8. Grab a Function node from the Function section and drop it to the right of your Inject node. Click and drag on the output connector (white box on the right) to create a line connecting it to the input of the Function node. Now, the Inject node will send a msg object

to the Function node with msg.payload set to the timestamp.

9. Double click on the Function node to open it. Name the function node something like "M2X Lead-in" and add the following code:

```
msg.topic = "devices";
msg.action = "postMultiple";
msg.topic_id = "";
var timestamp = new Date(msg.payload).toISOString();
        // change timestamp to Date format
msg.payload = {
    "temp": [  // this should be same as stream name
        { "timestamp": timestamp, "value": 34 }]
};
return msg;
```

10. Set the msg.topic_id to your *device ID* from M2X (found at the top of the device's detail page).

11. Click on the OK button at the bottom to save your function.

12. Take a M2X node from the Storage section and drop it onto the canvas. Double-click on it to open it up. Next to the Feed click on the pencil icon to create a new M2X configuration. Name it "My M2X Account" and copy and paste your **Master API Key** from the M2X account page (https://m2x.att.com/account). Click Add to save it, then OK to save the M2X node. Link the Function node to the M2X node.

13. From the Output section, drop a Debug node to the right of the M2X node. Link the output of the M2X node to the Debug node.
    Your output should look like this:

14. Click on the Deploy button in the upper right hand corner.
    You'll see a progress report updated you on the status of the build.
    Once successfully completed, your flow has been built and deployed to the AT&T IoT Platform as a Service (PaaS), to a special default environment created for each project called 'sandbox'.
    Once finished, you may need to click on the 'Enter Online Mode'
     Button in the upper right corner to move to 'Online Mode'.
    Online mode means you are actually running your code on a designated 'Docker' (lightweight portable code execution container, equivalent to virtual machine, but much more efficient), as if you were running it on your own machine. Offline mode in con- trast, means as if you are looking at an architectural drawing of your house (the flow), but unable to actually walk up the stairs to 'test' or debug their pitch.

    Note: The Online button icon indicates the state that you will be in once you've clicked it, not the state that you are currently in. So you will see a "connected" icon when you are offline and a "disconnected" icon when you are Online.

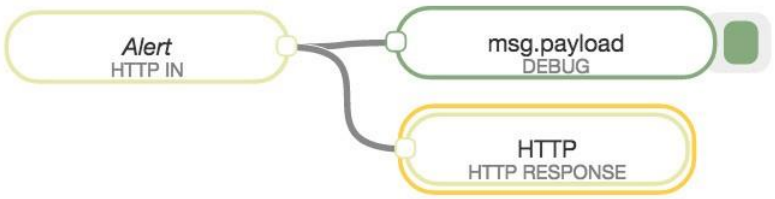
15. Once deployed, click on the tab to the left of the Inject node to initiate the flow. Click on the Debug panel at bottom to pull it up and you should see an accepted message. If you look at your Temperature stream in M2X you'll see a single data point of 34 °C.

16. A function node is written using NodeJS and could check the values to make sure they're in your given bounds, or you can rely on M2X to send triggers. M2X integrates with many services, including IFTTT. We'll use the M2X triggers here. Grab an HTTP node from the Input section and drop it below the Inject node. Double-click on it to open it and name it "Alert", change the Method to POST, then set the URL to "/alert". Click OK to save it.

17. Connect up a Debug node to the right of the HTTP In node. Deploy your project.

18. Grab an HTTP RESPONSE node from the Output section and drop it below the Debug node. Connect the output of the HTTP node to the input of the HTTP RESPONSE node. The HTTP RESPONSE is

required to close the trigger sender's connection. Your new nodes should look like this:



19. Deploy your project.

20. Pull up the Endpoints panel at the bottom and you'll see the URL for sending data to the Alert node labeled, "HTTPS (Alert)". Copy that URL. Then, open up M2X and the trigger you created by click on the pencil icon on the same line as the trigger. Paste that URL into the Callback URL text field and then click on the Save button.

21. Modify the Function node to set the temperature higher than 40, deploy the project, and then execute the Inject node by clicking on its tab. In the Debug panel in Flow you'll see a JSON object sent from M2X. You can modify the Function node to go back to a value less than 38, which will reset the trigger.

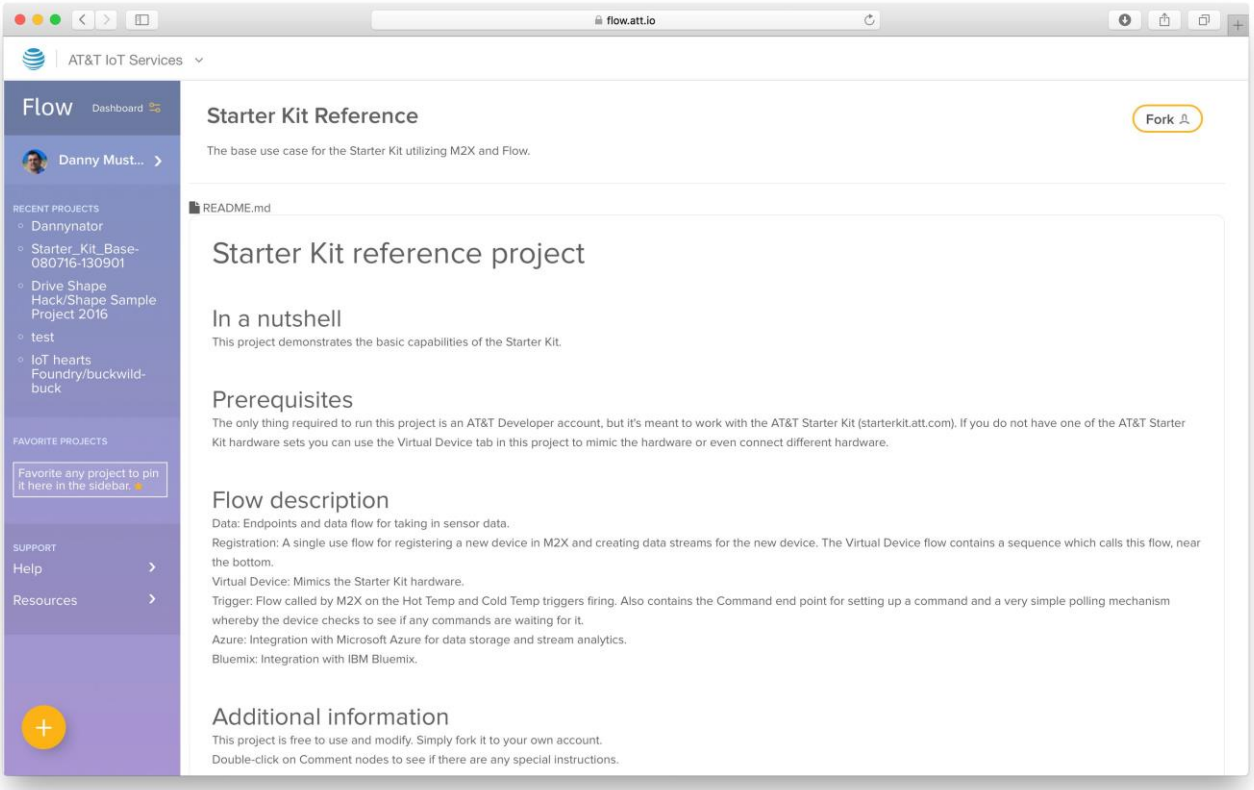
22. You can stop this node, preserving resources, by clicking "Run" in the left hand column. For example, you may need to do this because your Developer account supports a limited number of executing projects.

23. Click on "Sandbox"

24. Click on the "Stop" button on the upper right-hand corner. This will stop your flows, while retaining your code in Flow Designer's repository.

Starter Kit Example Project

Now, you've gotten a crash course in Flow and M2X. The Starter Kit Reference project implements several nodes and includes a virtual device sending data as if it were a real device. It even registers itself in M2X and creates the streams and triggers through the API. This way, if you were to build a project for several devices you wouldn't have to add them all to M2X manually.



1. Navigate to https://flow.att.io/starter-kit-core/starter-kit-base/ home to view the details of the Starter Kit Reference project. Click on "Fork" in the top-right corner of the page. In the confirmation dialog box you can assign a team.

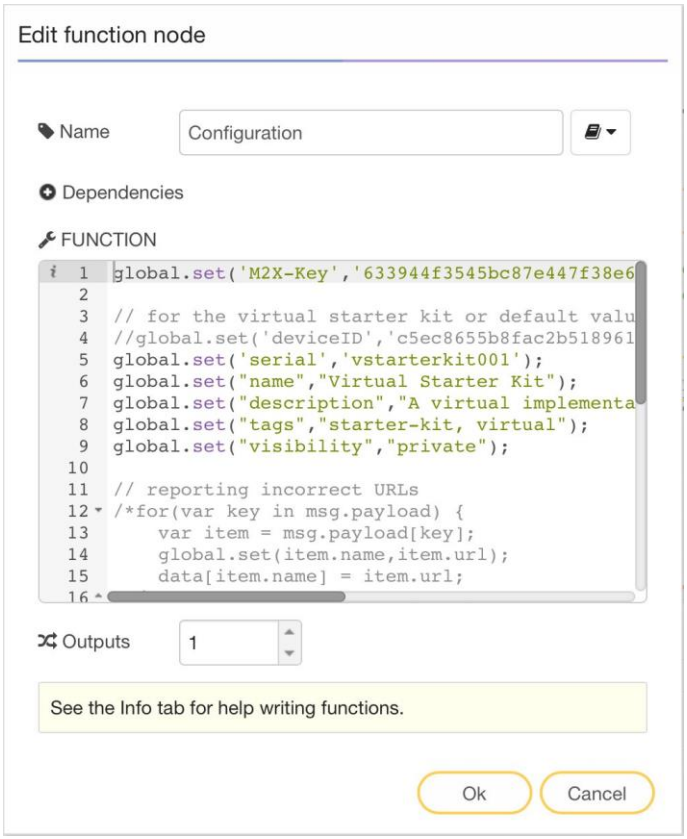
   IMPORTANT: Change the name to suit you.

   At this time a team can only be assigned when a project is created, though a team can have members added to it later on. Click "Con-

firm" to create your own copy of this project.

2. Take a moment to look around. The readme for this project describes each flow (the tabs along the top) and what they're for. We've laid out some boilerplate flows you may find useful in other projects as well. Take a look at the comment nodes, which typically contain useful instructions if you double-click on them.

3. To get started you need to deploy this project to initialize the endpoints by using the "Deploy" button in the top right-hand corner. We'll copy those end-points into a block of configuration code.



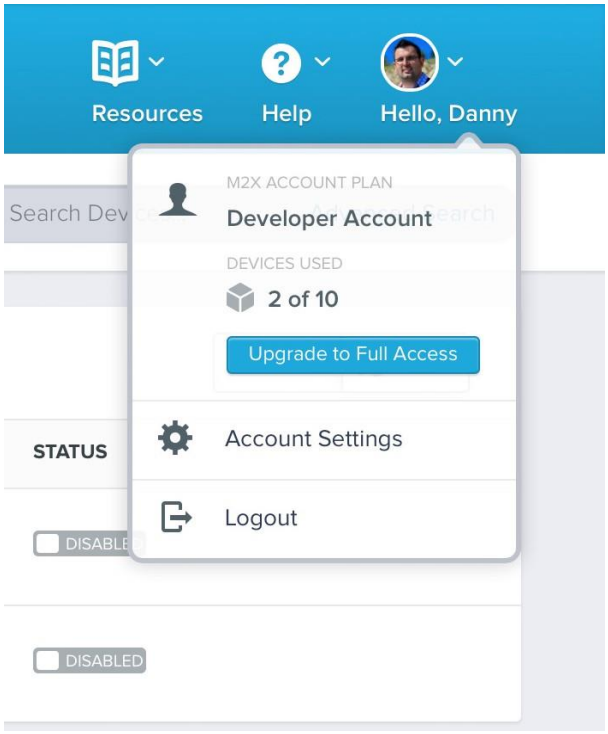
   This process takes a while the first time you execute it. It's packaging up the requirements to run Flow into a container and then deploying that to the cloud.

4. Once deploy is complete, open the Endpoints panel at the bot-tom of the canvas. You'll see the Base URL listed at the top. Copy this URL, then open the "Configuration" function node at the top of the Data flow by double-clicking on it. These nodes contain chunks of NodeJS code. Typically, they take the msg object (and the msg. payload property in particular), manipulate it, and pass the results to the next node. Change the line which begins, "*var base_url = …*" by pasting your value over the one that's there. Click "OK" to save your changes.

5. Go back to the Endpoints panel at the bottom of the canvas. Scroll down to note the port assigned to the UDP (Climate) end-

point. Go back to the “Configuration” function node and update the line of code ‘global.set(“ClimateUDP_Port”,….);’ and change to your port.

6. Go back to M2X (at this point, you’ll probably want to keep Flow and M2X open in separate browser tabs). In the top right-hand corner, click on the account drop-down menu and select “Account Settings”. This will show you a page with all your API keys. Copy the Master Key and head back over to Flow. Double-click the “Configuration” function node and alter the line starting, “global.set(‘M2X-Key’,…” to use the key you just copied by pasting over the sample key value. Save your changes.



NOTE: Without this crucial step, your project will fail to connect to your M2X account.

7. In the starterkit.att.com web site, on the “Api” page you’ll see your Control Center API key. You’ll need your Control Center username and API key for using the Control Center functionality in Flow. These values are set toward the bottom of the Configuration node. Enter them if you want to test this functionality, including SMS.

8. Deploy the project again. Once that’s done, go to the “Virtual Device” flow. You’ll see several inject nodes along the left side of the flow. The little tabs on the left side of the node executes it when clicked on. On the other side you’ll see several green debug nodes. The tabs on the right side of those nodes indicate if they are active or not. Active debug nodes send their data to the Debug panel at the bottom of the canvas. At the bottom of the node is an inject node labeled “Initialize”. Execute it.

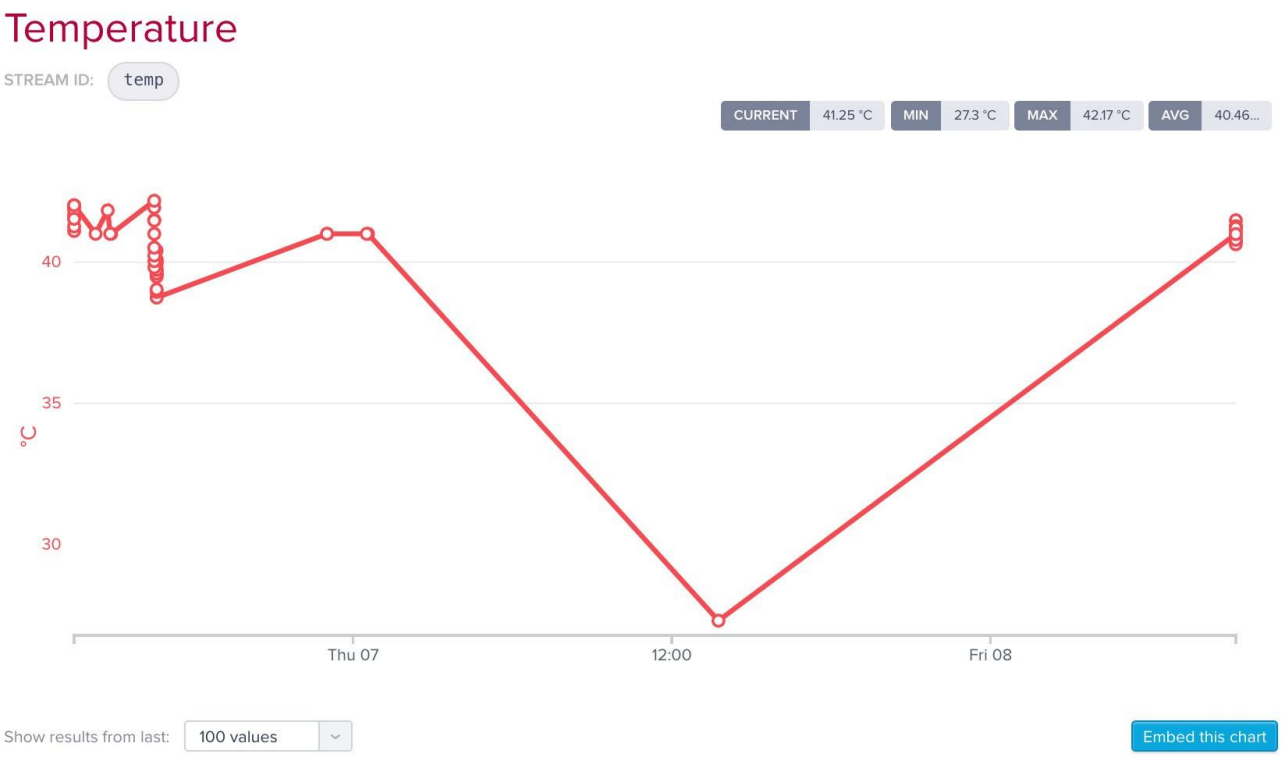
9. Flip back to M2X. Click on the Devices link at the top to pull up the list of devices in your account. If everything is configured correctly, you'll see a new device, "Virtual Starter Kit", which includes three streams of data.

10. Click on the + next to the device's name to list the streams. Click on the any stream or the device name to pull it its detail. In the Overview panel you'll see that the device has three streams and two triggers. Click on "Triggers" to pull those up. All of this was created with the nodes in the Registration flow. Make note of the Device ID and API Keys. If you call the Initialize inject node again it will create a new device and override the deviceID stored in the global context. You can use the Configuration function node to manually set this if you accidentally lose (or do another full deploy which will wipe out these values).

11. Return to Flow and to the Virtual Device flow. At the top you'll find a handful of nodes implementing a way to push data to the Data flow. Execute the one labeled "Sensor Reading". Go back to M2X and look at the device streams. In 5 seconds or less you'll see the values reflected on the chart there. Normally, Flow is stateless, however the "Data to Send" function node uses context variables to maintain state and create a "random walk" of simulated data readings.

12. Back in Flow you can execute the "manual values – hot" to trigger the Hot Temp trigger, and then "manual values – reset" to reset the trigger. You'll see this in action in the Debug panel below. The triggers are calling a HTTP endpoint on the Trigger flow. Edit the trigger in M2X to see how these values are set, where the URL is set, and how

the trigger timing is handled. You can see how it's done programmatically on the Registration flow in the "Set up Hot Temp Trigger" or "Set up Cold Temp Trigger" nodes. Along with the trigger coming in you'll also see the device periodically calling the command Poll flow, where it checks for waiting commands. This is a simple system for sending commands down to a remote device. You can also use an MQTT broker or RabbitMQ for a more robust implementation.



13. Click on the "Sensor Reading" inject node on the Virtual Device flow to create several random entries. In M2X you'll see these data points creating a chart.

14. While you're here, click on "Create Dashboard" link at the top of M2X. Click on the "Add Dashboard" button, name it, and click on the Add button.

    On the next screen, click on "Create a Widget", name it "Temperature vs Humidity", select "Line Chart" from the "Choose a Widget" drop-down, then "Virtual Starter Kit" in the Search Devices text box.

    Select the "Virtual Starter Kit" device in the results, which pops up a dialog showing the available streams.

Select Humidity and Temperature, then on the Update Data Source button. After that closes, click on the Add to Dashboard button at the bottom. Experiment with the different chart types by creating a radial graph of one of the streams.

15. Now that you're familiar with Flow, edit the code for your Starter Kit hardware to connect to Flow and start sending real data to the cloud. Set the base_hostname to the hostname for your Flow endpoints, (i.e. "run-east.att.io") and set the base_uri to the rest of the Base URL after the hostname (i.e. "/lkj32l2k3j432k/lk23j4l23k4j/kljlk243/in/flow"). You can also edit the unique ID for your device.

    Compile the source and then copy the source to the MBED removable drive to load it onto the MCU. If you have configured everything correctly, it should start to send data up through Flow and you'll see it through your Debug nodes.

    You will need to create a device in M2X manually (see the M2X Exercise above) or edit the default values in the Configuration node and use the Initialize inject node on the Virtual Device flow to do it for you.

Good luck and have fun hacking;
AT&T IoT Developer Services Team

- Welcome to Node-RED
- 2017-05-09 05:49:29.007 UTC

US-E
#1
DEBUG
==========
- 2017-05-09 05:49:29.008 UTC

US-E
#1
DEBUG
- 2017-05-09 05:49:29.009 UTC

US-E
#1
INFO
Loading palette nodes
- 2017-05-09 05:49:30.691 UTC

US-E
#1
ERROR
(node) sys is deprecated. Use util instead.
- 2017-05-09 05:49:32.537 UTC

US-E
#1
INFO
Starting flows
- 2017-05-09 05:49:32.546 UTC

US-E
#1
INFO
Started flows
- 2017-05-09 06:03:49.223 UTC

US-E
#1
DEBUG
[info] "2017-05-09T06:03:49.222Z" 'Using configured X-M2X-KEY [a09ef5d865e11f82fa507acbd1a2a434]'
- 2017-05-09 06:03:49.225 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:49.224Z" ['id', 'values', 'callback']
- 2017-05-09 06:03:49.226 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:49.226Z" 'PARAMETER [id] Value [0f67612bb9eda75c359d947c159db4d7]'
- 2017-05-09 06:03:49.227 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:49.227Z" 'PARAMETER [values] Value [[object Object]]'

- 2017-05-09 06:03:49.228 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:49.228Z" 'PARAMETER [callback] Value [function (error, response) {\n
_this.handle_msg_response(msg, error, response);\n }]'

- 2017-05-09 06:03:49.229 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:49.229Z" 'TOPIC [devices] ACTIONS [postMultiple]'

- 2017-05-09 06:03:49.432 UTC

US-E
#1
DEBUG
[info] "2017-05-09T06:03:49.431Z" 'Successful M2X Api call [202]'

- 2017-05-09 06:03:52.649 UTC

US-E
#1
DEBUG
[info] "2017-05-09T06:03:52.649Z" 'Using configured X-M2X-KEY [a09ef5d865e11f82fa507acbd1a2a434]'

- 2017-05-09 06:03:52.651 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:52.650Z" ['id', 'values', 'callback']

- 2017-05-09 06:03:52.651 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:52.651Z" 'PARAMETER [id] Value [0f67612bb9eda75c359d947c159db4d7]'

- 2017-05-09 06:03:52.652 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:52.652Z" 'PARAMETER [values] Value [[object Object]]'

- 2017-05-09 06:03:52.652 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:52.652Z" 'PARAMETER [callback] Value [function (error, response) {\n
_this.handle_msg_response(msg, error, response);\n }]'

- 2017-05-09 06:03:52.653 UTC

US-E
#1
DEBUG
[debug] "2017-05-09T06:03:52.653Z" 'TOPIC [devices] ACTIONS [postMultiple]'