



Getting to Know the AT&T M2X Data Service API

AT&T M2X Data Service



Table of Contents

Introduction	2
Prerequisites	3
Conventions Used	4
Setting Up for the M2X Exercises	5
Sign Up for an M2X Beta Account.....	5
Lab #1: Creating and Using a Data Source Blueprint	6
API Details.....	6
Exercise Variant: REST Client (Postman)	8
Exercise Variant: Command Line Tool (curl)	14
Summary	20
Lab #2: mbed and M2X	21
Setting up the mbed FRDM-KL46Z Device.....	21
Obtaining the Baseline Code	21
Modifying the Code to Send Accelerometer Data.....	22
Setting the IDE's Platform.....	24
Running the Code	26
Troubleshooting.....	27
Summary	28
Lab #3: Triggers.....	29
API Details.....	29
Setting Up a Trigger.....	31
Using the Trigger.....	32
Creating a Trigger through the API	33
Summary	37
Lab Exercise #4: M2X Server Sample Application.....	38
Running the Sample M2X Server Using Heroku.....	38
How the Server Code Works.....	39
Appendix A: Installing Postman for Chrome.....	41
Appendix B: Installing curl for Windows	43
Appendix C: Software Clients	45



© 2014 AT&T Intellectual Property. All rights reserved. AT&T, AT&T logo and all other AT&T marks contained herein are trademarks of AT&T Intellectual Property and/or AT&T affiliated companies. All other trademarks are the property of their owners. Actual results and your experience may vary from those described in this case study. Information and offers subject to change.



Introduction

AT&T M2X Data Service

AT&T M2X Data Service is a cloud-based, time series data storage solution for network connected Machine-to-Machine (M2M) devices, applications, and services. M2X makes it easier for developers to gather real-time data from various M2M sources and translate it into meaningful information, make operational decisions based on the data, and share the data with collaborative partners.

The intent of this booklet is to guide you through a series of simple lab exercises to familiarize you with using the AT&T M2X Data Service API and to greatly simplify its integration into your own applications.

These lab exercises are targeted toward developers and will be shown in curl and a web-based REST client ([Postman REST client for Chrome](#)), but you can use any programming language or REST client that you prefer. In addition, the exercises use the mbed FRDM-KL46Z device and mbed online IDE, but again, you can use any device that you prefer, as long as it can connect to a network and make HTTP requests.

The following pages provide additional information about the M2X Data Service.

- [About M2X](#)
- [Getting Started](#)
- [API Documentation](#)
- [Client Libraries](#)
- [Tutorials](#)
- [Hardware Platforms](#)
- [Forums](#)
- [FAQ](#)



Prerequisites

In order to complete the exercises in this workbook, you will need:

- A browser and internet connection.
- An mbed FRDM-KL46Z device with WiFi shield and USB cord.
- Optionally: A REST client, such as the Chrome app Postman. See [Appendix A](#) on how to download and install Postman.
- Optionally: A command line tool called curl (officially known as cURL). See [Appendix B](#) on how to download and install curl.



Conventions Used

The following formatting conventions are used in this workbook:

Type	Style	Example
URL	Monospace	<code>https://api.att.com/speech/v3/speechToText</code>
Parameter	<i>Italic</i>	<i>client_id</i>
Header	Bold	application/json
Command line	Monospace	<code>curl -X POST --data-binary...</code>
JSON or XML	Monospace	<pre>{ "Recognition": { "Status": "OK",...</pre>
User interface elements	Bold	Join Now
File names	Bold	homeBy6.wav



Setting Up for the M2X Exercises

Pre-requisites: browser

Length: 10 minutes

In order to call the AT&T M2X Data Service APIs, you will first need to sign up for a free M2X Beta account. Once you have done so, you can create new data source blueprints and begin to push data into the service.

See the [Prerequisites](#) section on what you need before starting the exercises.

Sign Up for an M2X Beta Account

Follow these steps to sign up for an M2X account.

Note: If you have already signed up, simply log in.

1. Launch your browser and go to <https://m2x.att.com/signup>.
2. Either sign in using your GitHub account, or create a new username.
3. If you sign up using GitHub, then authorize AT&T to have access to your public data.
4. Enter in your name, email, and password.
5. An activation email is sent to your email address. Click on the link inside the email to activate your account

Congratulations! You're signed up.



Lab #1: Creating and Using a Data Source Blueprint

Pre-requisites: M2X account, curl or REST client

Length: 20 minutes

In this exercise:

API Details

Exercise Variant: REST Client (Postman)

Exercise Variant: Command Line Tool (curl)

Summary

A Data Source is any IP enabled device, application, or service that will be sending data to the M2X service. A Data Source Blueprint defines the attributes associated with the data source. Once the Blueprint has been tested, it can be used with multiple devices of the same type, which is accomplished through creating a Batch and launching it. Once the Blueprint has been tested, it can be used with multiple devices of the same type, which is accomplished through creating a Batch and launching it.

Note: A Data Source Blueprint is not simply the design. It can be used with a device to collect data, exactly like it would in production. This allows you to fully test the Blueprint before using it with multiple devices.

In this exercise, you will learn how to set up a Data Source Blueprint and then send data to it. Finally, you will make a request to obtain all of the data that the Blueprint has collected.

To make HTTP requests, you can use either a graphical client such as the Postman add-on for Chrome, or you can use the curl command line tool.

API Details

Sending Data

To send data from a Data Source, make an HTTP POST request to this URL:

```
http://api-m2x.att.com/v1/feeds/{feed-ID}/streams/speed/values
```

where {feed-ID} is the ID of the feed you created when creating the Data Source Blueprint. The POST contains the following headers:

Header Name	Description
Content-Type	application/json
X-M2X-KEY	Your Data Source's API Key



The POST body contains JSON with the following key/value pairs:

Key	Required	Value
values	Required	An array with values and timestamps
value	Required	A string that contains the value
at	Optional	A string that contains the time-stamp in ISO 8601 format. If omitted, the current time of the server is used.

If successful, no data is returned. For more information, see the documentation at <https://m2x.att.com/developer/documentation/feed#Post-Data-Stream-Values>.

Retrieving Data

To retrieve data that was sent from a Data Source, make an HTTP GET request to the same URL:

```
http://api-m2x.att.com/v1/feeds/{feed-ID}/streams/speed/values
```

where {feed-ID} is the ID of the feed you created when creating the Data Source Blueprint. The GET contains the following header:

Header Name	Description
X-M2X-KEY	Your Data Source's API Key

You can use the following optional query parameters to filter the returned data:

Parameter	Required	Description
start	Optional	The starting date and time in ISO 8601 format for data to be returned. Default is time when stream was created.
end	Optional	The ending date and time in ISO 8601 format for data to be returned. Default is current time.
limit	Optional	The maximum number of values to return. Default is 100.

If successful, JSON data is returned with the following key/value pairs:

Key	Value
start	Specified start time. null, if not specified.
end	Specified end time.
limit	Maximum number of values returned
values	An array with values and timestamps
value	A string that contains the value
at	A string that contains the time-stamp in ISO 8601 format.

For more information, see the documentation at <https://m2x.att.com/developer/documentation/feed#List-Data-Stream-Values>.

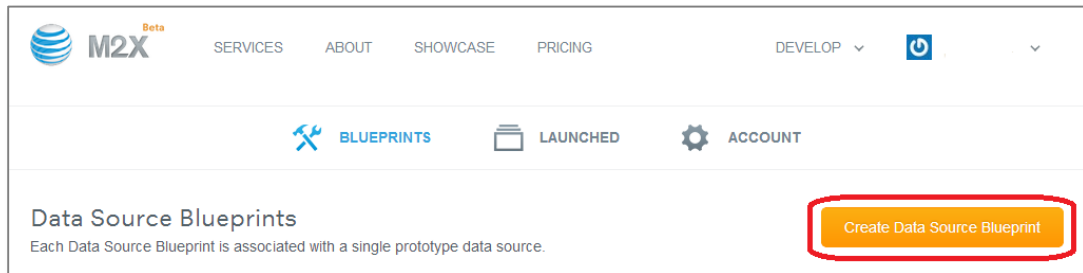


Exercise Variant: REST Client (Postman)

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 Blueprint, which is a way to test your data source before launch.

Follow these steps to set up a Blueprint:

1. Go to <https://m2x.att.com/blueprints>.
2. Click on **Create Data Source Blueprint**.



3. For **Data Source Name**, type "test". Click **Create**.


 The screenshot shows a modal window titled 'Create Data Source Blueprint'. It contains a description of what a Data Source Blueprint is. Below the description are three form fields: 'Data Source Name' with the value 'test', 'Data Source Description' (optional) with a placeholder 'Describe your Data Source...', and 'Visibility' with two radio button options: 'Private Data Source' (selected) and 'Public Data Source'. The 'Private Data Source' option has a sub-description: 'You use API keys to choose if and how you share data from a Data Source.' The 'Public Data Source' option has a sub-description: 'You agree to make this data source publicly available under the CC0 1.0 Universal license.' At the bottom right, the 'Create' button is highlighted with a red rectangle.

4. Your Blueprint will have a Feed ID and API Key associated with it. You will use these later.



FEED ID	84a7221ec4a9a3d9b19c30b1a95f2c52	Copy
API ENDPOINT	/feeds/84a7221ec4a9a3d9b19c30b0a95f2c52	
API KEY	8531911e8b2c361ad535fcb21a6a7064	Copy

5. Let's say that you were collecting information on a fleet of vehicles, and you wanted to keep track of the speed of each vehicle as a function of time. Let's add a data stream for feed. Scroll down and click on the **Add Stream** button.

STREAMS	API KEYS	TRIGGERS	LOCATION	API CHEATSHEET	HELP
<div>  </div>					
Name	Unit	Symbol	Min	Max	Current

6. Type in "speed" for the **Stream ID**, and then click on the **Aa** button.

Add Stream

Stream ID

speed

Stream IDs can only contain letters, numbers, underscores, and dashes — no spaces or special characters are allowed.

Units & Symbol

UNIT: e.g. Ohm, Watt

SYMBOL: e.g. W, Ø, Ω

Aa

7. Click on the **M**, then scroll down and select "miles per hour".



Stream ID: speed

Stream IDs can only contain letters, numbers, underscores, and dashes — no spaces or special characters are allowed.

Units & Symbol: UNIT: e.g. Ohm, Watt SYMBOL: e.g. W, ø, Ω

Log a Value (optional): e.g. Record Source

Select Unit & Symbol

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
												microgram	µg												
												mile	mi												
												miles per hour	mph												
												millimeter	mm												
												millibar	mbar												
												milligal	mGal												
												milligram	mg												

Learn more about Streams

8. Click on **Add Stream** to create the stream.

You now have a Data Source Blueprint and a stream. Next, you'll send some data to that Blueprint. Follow these steps to send a value of zero using the Postman add-on for Chrome:

1. Open the Postman add-on for Chrome.
2. In the **Enter request URL here** box type **http://api-m2x.att.com/v1/feeds/{feed-ID}/streams/speed/values**
3. Replace the {feed-ID} with the feed ID from the Blueprints page. You can use the **Copy** button to copy the ID to the clipboard.
4. Set the method dropdown to **POST**
5. If the POST body is not visible, select the **raw** button
6. Add the following into the POST body box:

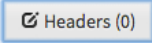
```
{
  "values": [
    { "at": "2014-08-21T19:17:00.624Z", "value": "0" }
  ]
}
```

Note: It is possible to send multiple values with one request, and each value can have a time-stamp. (If no time-stamp is specified, as in the example above, then the time-stamp is the

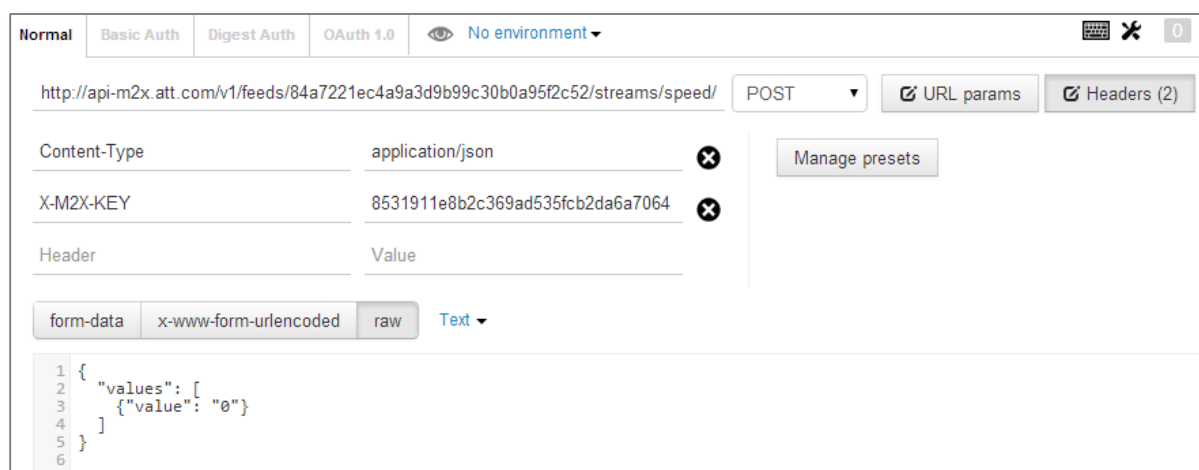


time the request is received.) For multiple values, use the "at" key, as shown in the following example:

```
{
  "values": [
    { "at": "2014-08-21T19:17:00.624Z", "value": "32" },
    { "at": "2014-08-21T19:18:00.522Z", "value": "30" }
  ]
}
```

7. Click the  button to display the headers section, if necessary.
8. Add a header with a name of **Content-Type** and value of **application/json**.
9. Add a header with a name of **X-M2X-KEY** and value of your API key from the Blueprints page. You can use the **Copy** button to copy the ID to the clipboard.

Your request should look like this:



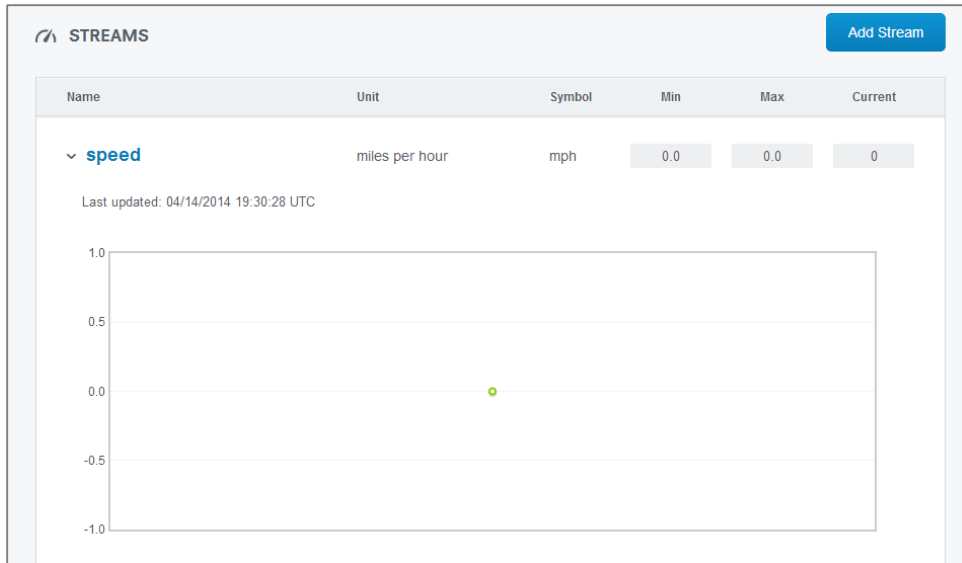
The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://api-m2x.att.com/v1/feeds/84a7221ec4a9a3d9b99c30b0a95f2c52/streams/speed/`
- Headers:**
 - Content-Type:** `application/json`
 - X-M2X-KEY:** `8531911e8b2c369ad535fcb2da6a7064`
- Body:**
 - form-data** (selected)
 - x-www-form-urlencoded**
 - raw** (selected)
 - Text** (selected)
 - Content:**

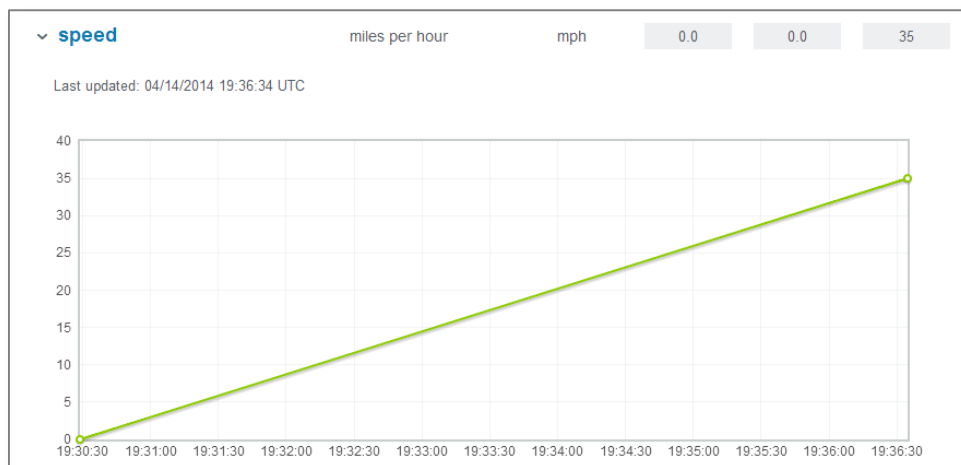
```
1 {
2   "values": [
3     {"value": "0"}
4   ]
5 }
6
```

10. Click **Send**. If successful, then an HTTP 204 response will be returned with no content.
11. Go back to your Blueprint page and scroll down until you see the **Streams**. There will be one stream for speed. Open it by clicking on it. Scroll down to see the graph of values. Note that the graph has one data point, which is at zero.



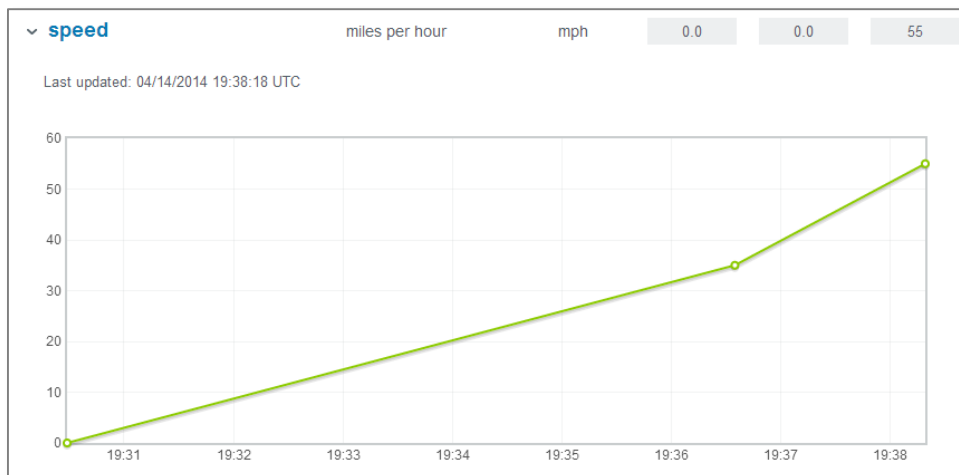


12. Return to Postman. In the POST body, change the value from "0" to "35". Copy and paste the modified curl command into the terminal. Your graph will immediately show a slope from 0 to 35, with a time scale on the x-axis.



13. Try it again, this time putting in a value of 55. You can see that each time it updates the graph.





10. Now, let's use an HTTP request to retrieve those values. Change the dropdown from **POST** to **GET** and click **Send**. This will return all values so far. Your output should look something like this, with the most recent values displayed first:

```
{
  "start": null,
  "end": "2014-08-21T15:19:28.632Z",
  "limit": 100,
  "values": [
    {
      "at": "2014-08-21T21:21:07.970Z",
      "value": "55"
    },
    {
      "at": "2014-08-21T21:20:45.269Z",
      "value": "35"
    },
    {
      "at": "2014-08-21T21:20:01.080Z",
      "value": "0"
    }
  ]
}
```



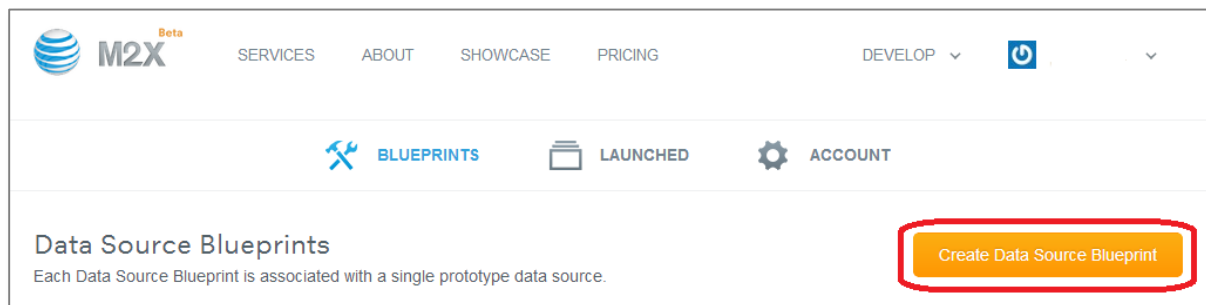
Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

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 Blueprint, which is a way to test your data source before launch.

Follow these steps to set up a Blueprint:

1. Go to <https://m2x.att.com/blueprints>.
2. Click on **Create Data Source Blueprint**.



3. For **Data Source Name**, type "test". Click **Create**.



Create Data Source Blueprint

A Data Source Blueprint acts as a template, where you can assign Data Source attributes like streams, triggers, location information, and metadata. A Data Source Blueprint has the same features as a Data Source, but a Data Source Blueprint is used to define and test Data Source attributes before launch.

Data Source Name

Data Source Description

optional

Visibility

☒ Private Data Source

You use API keys to choose if and how you share data from a Data Source.

☐ Public Data Source

You agree to make this data source publicly available under the CC0 1.0 Universal license.

COMING SOON

Learn more about Data Source Blueprints

Cancel

Create

4. Your Blueprint will have a Feed ID and API Key associated with it. You will use these later.

FEED ID	
84a7221ec4a9a3d9b19c30b1a95f2c52	Copy
API ENDPOINT	
/feeds/84a7221ec4a9a3d9b19c30b0a95f2c52	
API KEY	
8531911e8b2c361ad535fcb21a6a7064	Copy

5. Let's say that you were collecting information on a fleet of vehicles, and you wanted to keep track of the speed of each vehicle as a function of time. Let's add a data stream for feed. Scroll down and click on the **Add Stream** button.



STREAMS API KEYS TRIGGERS LOCATION API CHEATSHEET ? HELP

STREAMS

Add Stream

Name	Unit	Symbol	Min	Max	Current
------	------	--------	-----	-----	---------

6. Type in "speed" for the **Stream ID**, and then click on the **Aa** button.

Add Stream

Stream ID: speed

Stream IDs can only contain letters, numbers, underscores, and dashes — no spaces or special characters are allowed.

Units & Symbol: UNIT: e.g. Ohm, Watt SYMBOL: e.g. W, ø, Ω Aa

7. Click on **M**. Scroll down and select "miles per hour".

Add Stream

Stream ID: speed

Stream IDs can only contain letters, numbers, underscores, and dashes — no spaces or special characters are allowed.

Units & Symbol: UNIT: e.g. Ohm, Watt SYMBOL: e.g. W, ø, Ω Aa

Log a Value (optional): e.g. Record Source

Select Unit & Symbol

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
microgram																									µg
mile																									mi
miles per hour																									mph
millimeter																									mm
millibar																									mbar
milligal																									mGal
milligram																									mg

Learn more about Streams

8. Click on **Add Stream**.



You now have a Data Source Blueprint and a stream. Next, you'll send some data to that Blueprint. To do this, we will send a POST request with the following JSON in the POST body, which contains a value of zero:

```
{
  "values": [
    { "at": "2014-08-21T19:17:00.624Z", "value": "0"}
  ]
}
```

Note: It is possible to send multiple values with one request, and each value can have a time-stamp. (If no time-stamp is specified, as in the example above, then the time-stamp is the time the request is received.) For multiple values, use the "at" key, as shown in the following example:

```
{
  "values": [
    { "at": "2014-08-21T19:17:00.624Z", "value": "32" },
    { "at": "2014-08-21T19:18:00.522Z", "value": "30" }
  ]
}
```

Follow these steps to send a value of zero using curl:

1. Paste the following into a text editor. (You may have to put in the line breaks after pasting.)

```
curl --request POST --include \
--header "Content-Type: application/json" \
--header "X-M2X-KEY: {API-key}" \
--data '{"values": [{"at" : "2014-08-21T19:17:00.624Z","value": "0"}]}' \
http://api-m2x.att.com/v1/feeds/{feed-ID}/streams/speed/values
```

2. Replace {API-key} with your Blueprint's API key and {feed-ID} with your Feed ID, which you can find on the Data Source Blueprints webpage. You can use the **Copy** buttons to copy the values to the clipboard.

Note: Remove the curly brackets when replacing {API-key} and {feed-ID}. For example, your final version should look something like this:

```
curl --request POST --include \
--header "Content-Type: application/json" \
--header "X-M2X-KEY: 8a31911e8b2c361ad535fcb2da6a7064" \
--data '{"values": [{"at" : "2014-08-21T19:17:00.624Z","value": "0"}]}' \
http://api-m2x.att.com/v1/feeds/84a7221ec4a9a3d9b19c30/streams/speed/values?pretty=true
```

3. Open up a terminal (command prompt in Windows) that has curl.
4. Paste the edited curl command into the terminal and hit Enter. If successful, then the response will look something like this. (Note that no content is returned.)

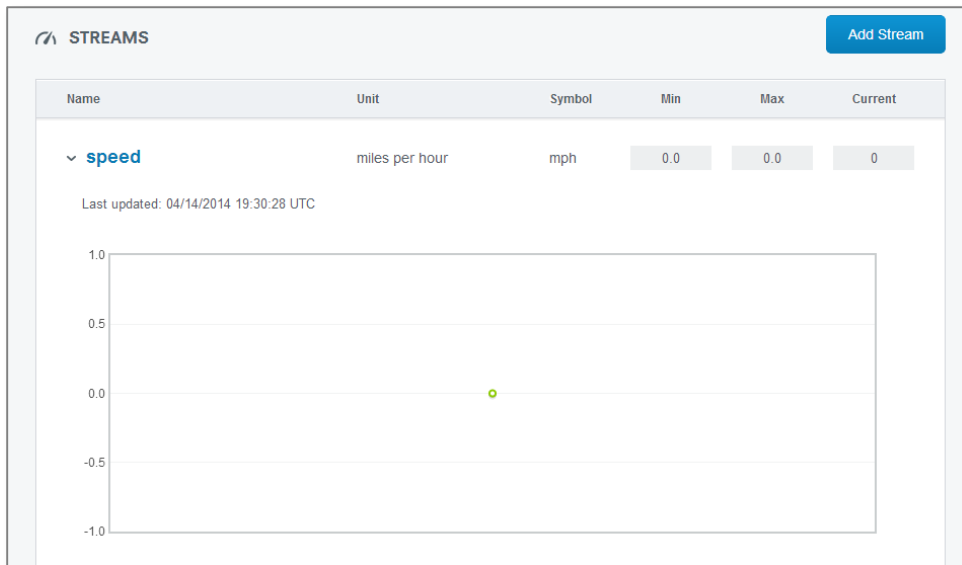


```

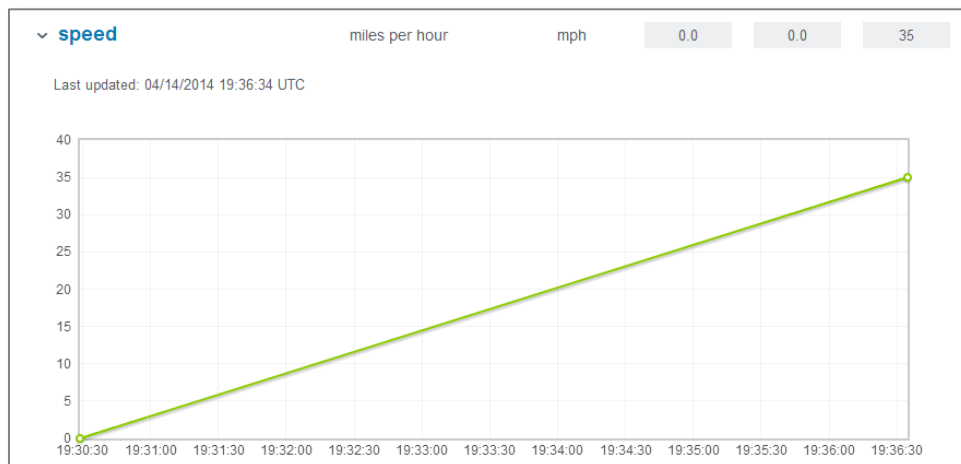
HTTP/1.1 204 No Content
Server: nginx
Date: Thu, 08 May 2014 18:05:51 GMT
Connection: keep-alive
Status: 204 No Content
X-M2X-VERSION: v1.0.0-beta
Vary: Accept

```

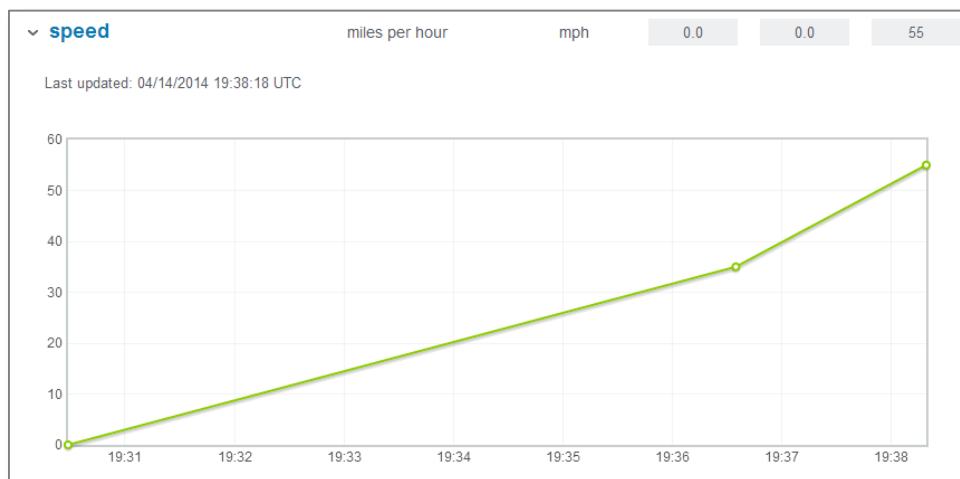
- Go back to your Blueprint page and scroll down until you see the **Streams**. There will be one stream for speed. Open it by clicking on it. Scroll down to see a graph. Note that the graph has one data point, which is at zero.



- In your text editor, change the value from zero to 35. Copy and paste the modified curl command into the terminal. Your graph will immediately show a slope from 0 to 35, with a time scale on the x-axis.



7. Try it again, this time putting in a value of 55. You can see that each time it updates the graph.



10. Now, let's use an HTTP request to retrieve those values. Change the POST request to GET and remove the --data. This will return all values so far. The curl command should look like this:

```
curl --request GET --header "Content-Type: application/json" \
--header "X-M2X-KEY: {API-key}" \
http://api-m2x.att.com/v1/feeds/{feed-ID}/streams/speed/values
```

11. Once formatted, your output should look something like this, with the most recent values displayed first:

```
{
  "start": null,
  "end": "2014-08-21T15:19:28.632Z",
  "limit": 100,
  "values": [
    {
      "at": "2014-08-21T21:21:07.970Z",
      "value": "55"
    },
    {
      "at": "2014-08-21T21:20:45.269Z",
      "value": "35"
    },
    {
      "at": "2014-08-21T21:20:01.080Z",
      "value": "0"
    }
  ]
}
```



Summary

You've learned how to create a DataSource Blueprint with a feed and a stream using the AT&T M2X Data Service website. You've also learned how to send data to that stream and retrieve it.



Lab #2: mbed and M2X

Pre-requisites: browser, mbed FRDM-KL46Z device with WiFi shield and USB cord
Length: 30 minutes

In this exercise:

- Obtaining the Baseline Code
- Modifying the Code to Send Accelerometer Data
- Running the Code
- Troubleshooting
- Summary

You can use the AT&T M2X Data Service with any device that can make HTTP requests. In this exercise, you will be using the mbed FRDM-KL46Z device that can connect to WiFi. We will provide you with baseline code that contains classes making it easy to communicate with both the device and the M2X service. You will add code that reads data from the device and makes the calls to the M2X service.

In this exercise, you will read accelerometer data from the mbed device and send it to an M2X stream if it calculates that the device is tilted above a certain angle.

Note: If you have trouble getting the device to work properly, see the Troubleshooting section.

Setting up the mbed FRDM-KL46Z Device

Follow these steps to set up the mbed FRDM-KL46Z device with a WiFi shield:

1. Screw the black antenna onto the antenna port.
2. Follow the wire that leads from the antenna port and make sure that it is connected to the board using the gold-plated connector
3. Attach the USB cord to the device using the mini-USB port that is nearest to the antenna.
4. Attach the other end of the USB cord to your computer's USB port. Note that the device is now powered by USB, so you do not have to use the power cord.

More information about the device can be found here: <http://mbed.org/platforms/FRDM-KL46Z/>.

Obtaining the Baseline Code

The mbed development platform is an online Integrated Development Environment (IDE) for writing software for ARM microcontrollers. It allows you to write code in C++, compile it, download the compiled code to your local computer, and then transfer that file onto the device. As an online IDE, it is easy to access your code from any computer, as well as share it easily with others.

Use the following steps to create an account on mbed.org. If you already have an account, then sign in.

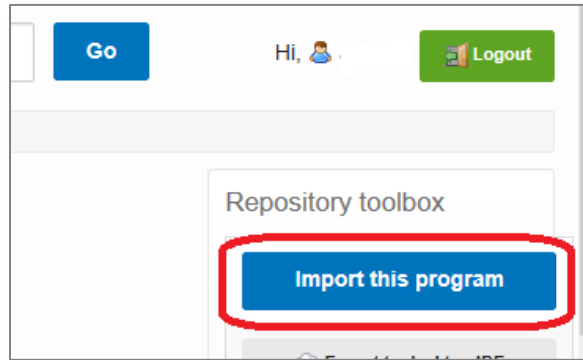
1. In a browser, navigate to <http://mbed.org>.
2. Click on the **Login or signup** button on the top right.



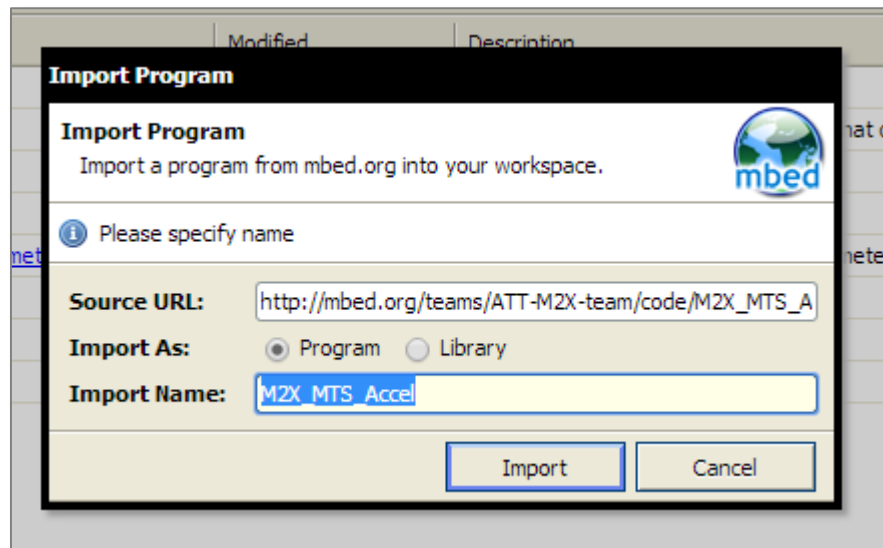
3. Click on the **Signup** button.
4. Click on the **No, I haven't create an account before** button.
5. Enter in your email address, username, and password, check the box to agree to the terms and conditions, and click the **Signup** button.

Next, you will need to import the sample code into your account. Follow these steps:

1. Open a new tab or window in your browser.
2. Navigate to http://mbed.org/teams/ATT-M2X-team/code/M2X_MTS_Accel/.
3. Click on the **Import this program** button. This will create a copy of the program in your own account.



4. Leave the name as is and click the **Import** button to complete the importing process.



5. You will now be taken to the online IDE with your project loaded. Note the project navigator on the left.

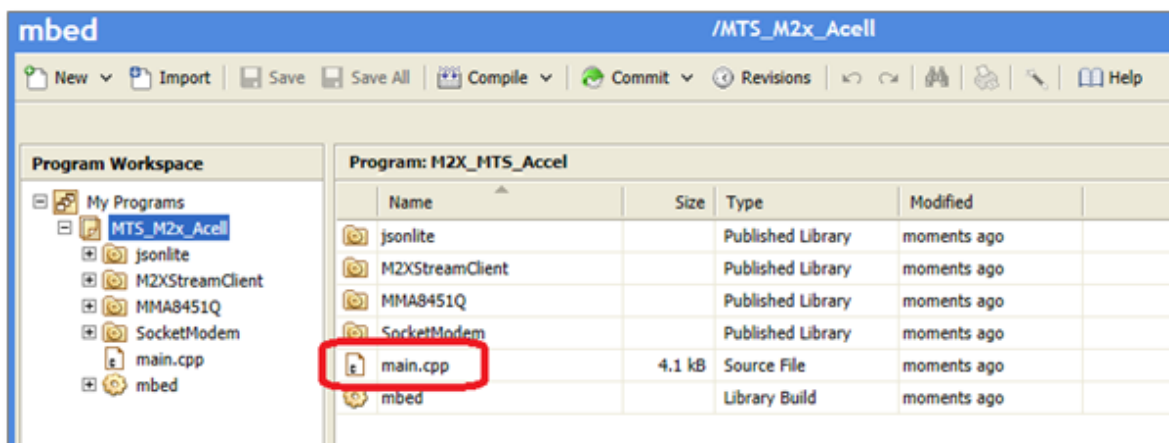
Modifying the Code to Send Accelerometer Data

Let's say your fleet of vehicles are carrying cargo that should be kept flat. You want to collect data when the cargo is tipped above 20 degrees. You'll need to add some code to read the accelerometer values, calculate the angle, and send data to your stream. **main.cpp** is the code that you will be working with. The other folders contain code that provide helper classes



to make it easier to access the M2X Data Service and the mbed device. Follow these steps to modify the code:

1. Double-click **main.cpp** to open it.



2. In lines **21**, **22**, and **23**, change **<key>** to your M2X key and **<feed>** to your feed ID, and **<stream>** to **"tilt"**, which is the name of the stream that will collect the data.
3. Depending on the type of hardware shield attached, follow one of these directions:
 - a. Wifi Shield: On line **27**, set the value for **CELL_SHIELD** to **0**. In lines **30** and **31**, change **<ssid>** to your wireless SSID, and **<password>** to your wireless password. In **32**, change the security type from WPA2, if necessary. Valid values are NONE, WEP64, WEP128, WPA, and WPA2.
 - b. 3G Shield: On line **27**, set the value for **CELL_SHIELD** to **1**.
4. At line **115**, add the following code. This will use the accelerometer values to calculate the pitch and roll angles of the device. If the maximum of these two is greater than 20 degrees, it will send the value to the tilt stream. Then, so that it doesn't overwhelm the stream with data, it waits 1 second before taking another accelerometer reading.



```

while (true) {
    // Get accelerometer data
    float x, y, z;
    x = acc.getAccX();
    y = acc.getAccY();
    z = acc.getAccZ();
    printf("Accel x: %1.2f, y: %1.2f, z: %1.2f\n\r", x, y, z);

    // Calculate pitch and roll. Find the maximum tilt angle.
    float pitch = atan(x / sqrt(y * y + z * z));
    float roll = atan(y / sqrt(x * x + z * z));
    float maxTilt =
        max(abs(roll), abs(pitch)) * 180.0 / 3.14159;
    printf("pitch: %5.1f roll: %5.1f maxTilt: %5.1f\n\r",
        pitch, roll, maxTilt);

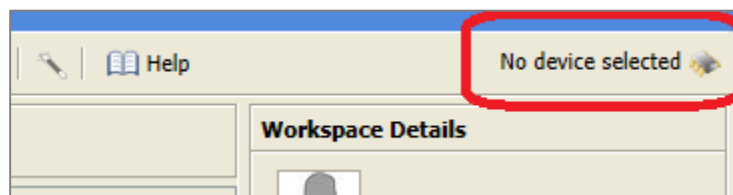
    // If the maximum title is over 20 degrees, then send
    // data to stream
    if (maxTilt > 20) {
        ret = m2xClient.post(feed, stream, maxTilt);
        printf("send() returned %d\n\r", ret);
        wait(1);
    }
}

```

Setting the IDE's Platform

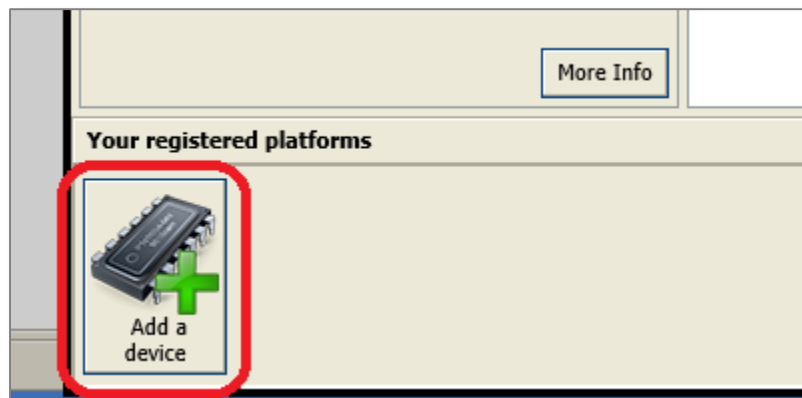
The IDE needs to be set to create compiled code for a given type of device (i.e., platform). Follow these steps to set the platform:

1. On the top right of the screen, click **No device selected**.

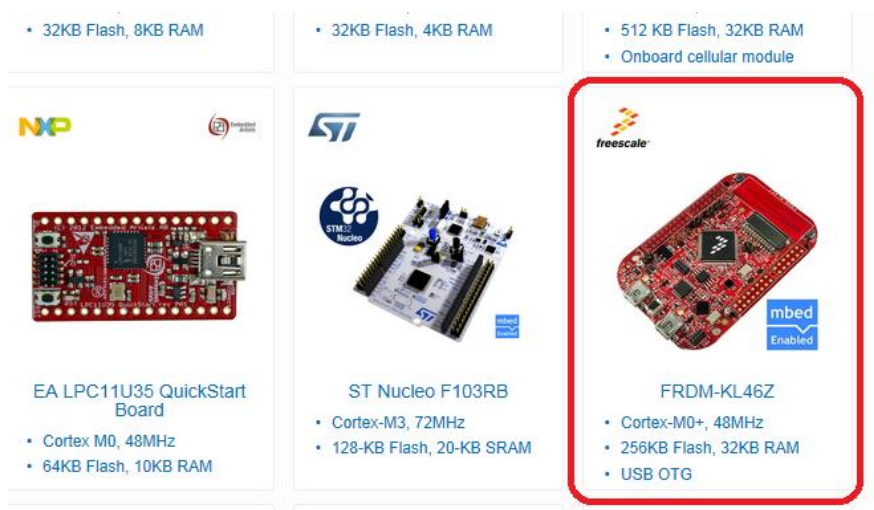


2. A new dialog will appear. In the bottom left, click **Add a Device**.





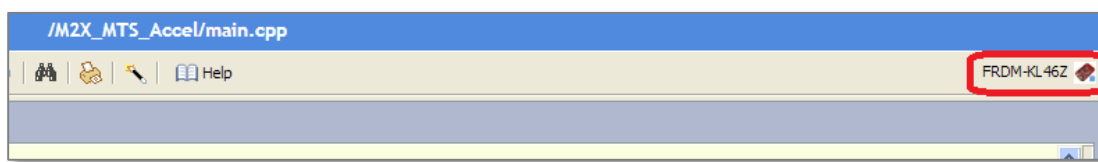
3. Scroll down until you find FRDM-KL46Z. Click on it.



4. Now, click on the button that says **Add to your mbed compiler**



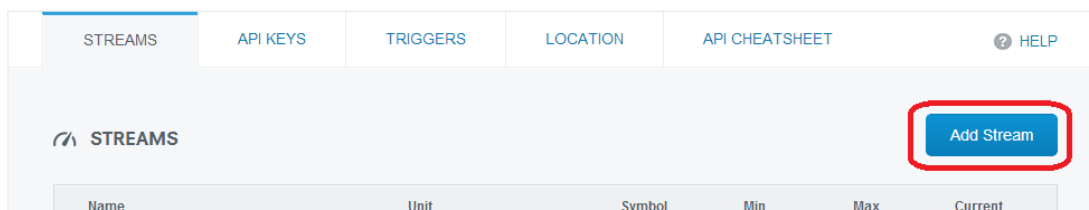
5. Return to your IDE workspace. You will see that in the upper right corner, it lists the FRDM-KL46Z as your platform.



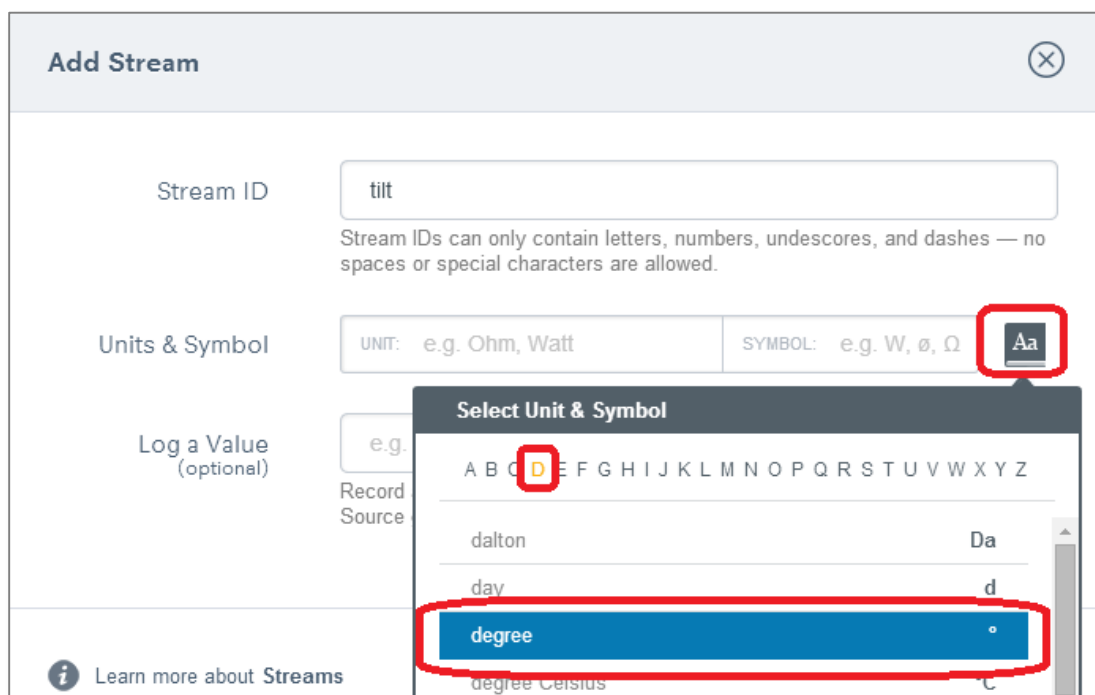
Running the Code

To run the code, first we need to set up a stream called "tilt" to collect the tilt angle data when it is greater than 20 degrees. Follow these steps:

1. Return to your Data Source Blueprint page and open the feed page for the feed you've been using. Click the **Add Stream** button.



2. Type in "tilt" for the Stream ID. Then click the **Aa** button on the **Units & Symbols** line. Click the letter **D**, and then choose **degree**.

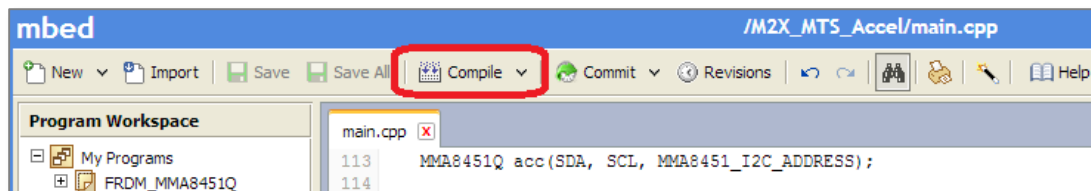


3. Click the **Add Stream** button.

Now return to the mbed online IDE and follow these steps to compile your program and load it onto the device.

1. Click the **Compile** button.



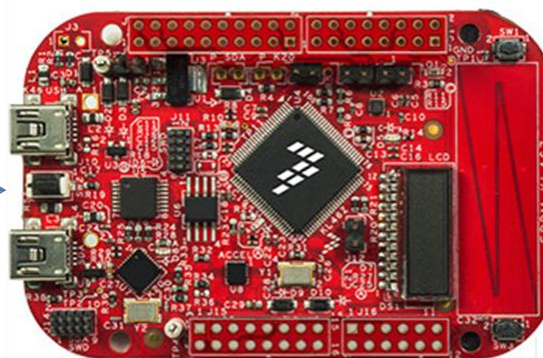


2. A file named **M2X_MTS_Accel_KL46Z.bin** will be downloaded. Make note of where it was saved.
3. On your computer, use Windows Explorer or Finder and navigate to the drive that's on the device.
4. Copy the **M2X_MTS_Accel_KL46Z.bin** from your computer to the device drive.

Note: Once the file has finished copying, the computer might disconnect the device and reconnect it. If you then open the device drive, you will not see the file that you copied there. This is normal. The file has been successfully uploaded.

5. To run the code, press the reset button, which is a small button between the two USB ports.

Reset button →



6. Put the device flat on the desk. Wait about 20 seconds for the device to start up and connect to the wireless network. In the meantime, open your "tilt" stream and scroll down to look for data to be collected.
7. You should not be seeing any data on the stream's graph because the angle is below 20 degrees. Pick up the device and hold it at an angle. After a few seconds, data will start showing up on the graph. Play around with larger and smaller angles and watch for the changes, which will take a few seconds to appear. If you drop below 20°, the data will not be sent to your stream.

Troubleshooting

You may be following the steps in this exercise and find that data is not being sent to your M2X stream. The mbed code contains printf lines that can help you troubleshoot and figure out if the problem is in the WiFi connection, the accelerometers, the M2X key, etc.



To be able to see the printf statements, you need to run a terminal application that uses a USB Virtual Serial Port, allowing the messages to be sent through the USB cable. Instructions on how to do that can be found in at <https://mbed.org/handbook/SerialPC>.

Note: Windows requires some extra steps, as detailed here: <https://mbed.org/handbook/Windows-serial-configuration>.

You can also search the mbed.org site for additional troubleshooting information.

Summary

You've learned how to import mbed code into the IDE editor and to modify it so that it reads accelerometer data and sends data to an M2X stream.



Lab #3: Triggers

Pre-requisites: access token without consent, curl or REST client

Length: 20 minutes

In this exercise:

API Details

Setting Up a Trigger

Using the Trigger

Creating a Trigger Through the API

Exercise Variant: REST Client (Postman)

Exercise Variant: Command Line Tool (curl)

Summary

The AT&T M2X Data Service allows you to create triggers so that when certain conditions are fulfilled, the M2X service will make HTTP requests to a URL that you have specified, with. This allows you to be notified when a value becomes too high or too low, or hits a certain value. Triggers can either be created through the web interface or through the API.

In this exercise, we will lead you through creating a trigger and then sending data to the stream that triggers it.

API Details

To create a new trigger through the API, make an HTTP POST request to this URL:

```
http://api-m2x.att.com/v1/feeds/{feed-ID}/triggers
```

where {feed-ID} is the ID of the feed you created when creating the Data Source Blueprint. The POST contains the following headers:

Header Name	Description
Content-Type	application/json
X-M2X-KEY	Your Data Source's API Key

The POST body contains JSON with the following key/value pairs:



Key	Required	Value
name	Required	The name of the trigger
stream	Required	The name of the stream
condition	Required	The condition, which can be <, <=, =, > or >=
value	Required	The value to use with the condition
callback_url	Required	The URL that will be called if data matches the condition
status	Required	Either "enabled" or "disabled"

If successful, JSON is returned with the following key/value pairs:

Key	Value
id	The ID of the trigger
name	The name of the trigger
stream	The name of the stream
condition	The condition, which can be <, <=, =, > or >=
value	The value to use with the condition
callback_url	The URL that will be called if data matches the condition
url	The URL of the trigger
status	Either "enabled" or "disabled"
created	Timestamp for when the trigger was created
updated	Timestamp for when the trigger was last updated

See the following documentation pages for more on triggers:

- List triggers: <https://m2x.att.com/developer/documentation/feed#List-Triggers>
- Create trigger: <https://m2x.att.com/developer/documentation/feed#Create-Trigger>
- View Trigger: <https://m2x.att.com/developer/documentation/feed#View-Trigger>
- Update Trigger: <https://m2x.att.com/developer/documentation/feed#Update-Trigger>
- Test Trigger: <https://m2x.att.com/developer/documentation/feed#Test-Trigger>
- Delete Trigger: <https://m2x.att.com/developer/documentation/feed#Delete-Trigger>



Setting Up a Trigger

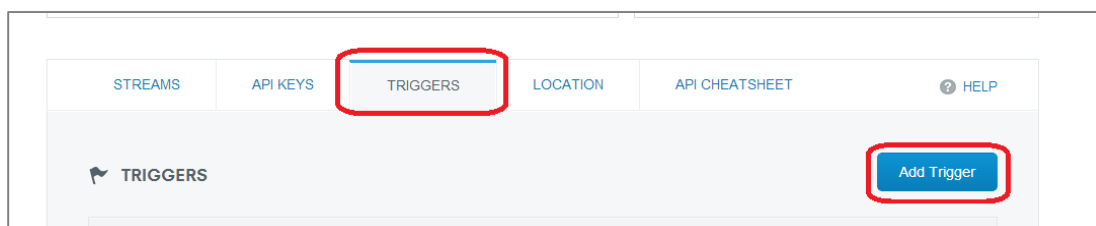
Let's go back to the example of a fleet of vehicles carrying cargo that needs to stay level. Let's say that, in addition to collecting data every time the tilt angle goes over 20°, you want to be notified every time the tilt angle goes over 40°, since this requires taking immediate action. We can set up a trigger to do this, which will make a POST request to a URL that we can monitor.

First, we need to set up a URL to receive the notification. We can do this by using a site called RequestBin, which allows you to set up URLs for testing purposes. Follow these steps:

1. In a new browser tab or window, navigate to <http://requestbin.in/>
2. Click on **Create a RequestBin**.
3. This will create a URL that we can use when creating a trigger. Save the page so that you can copy and paste the URL.

Use the following steps to create a trigger for the condition where the angle is greater than 40°:

1. In a different browser tab or window, open your Data Source Blueprint page and click the **Trigger** tab. Then click **Add Trigger**.



2. For the **Trigger Name**, type "serious tilt". For the **Stream**, choose **tilt**. For the **Condition**, choose **> is greater than**. For the threshold value, type "40". For the **Callback URL**, copy and paste the URL from the RequestBin site. Click **Create Trigger**.



Create Trigger

Trigger Name

serious tilt

Name used to identify this Trigger.

Stream

tilt

The stream this Trigger applies to.

Condition

> is greater than

Specify the condition required to fire this Trigger.

Threshold Value

40

Specify the value required to fire this Trigger. This must be a numeric value.

Callback URL

http://requestb.in/oej2njoe|

Callback URL of your choice that will receive the HTTP POST request when the condition has been met.

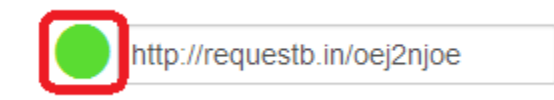
Learn more about Triggers

Cancel

Create Trigger

Using the Trigger

1. Your device should still be running. (If it's not, make sure it's plugged into the USB port, then press the reset button, and wait about 20 seconds for it to connect to the wireless network.) Pick up the device and hold it at about a 20° angle and watch the graph for the tilt stream to make sure the data is being picked up. This may take several seconds.
2. Now increase the tilt to over 40° and hold it for about 10 seconds. Check your RequestBin website and refresh it by clicking on the circle next to your URL, in the top right corner of the page.



3. You will see the data that was posted to the URL, which will take the following form. As you can see, it contains the feed ID, the stream name, the trigger name, the condition and threshold, as well as the value that crossed the threshold and its timestamp. If this were a real server that it was posting data to, you could parse the incoming data and take action.



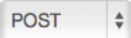

```
{
  "feed_id": "84a7221ec4a9a3d9b19c30b0a95f2c52",
  "stream": "tilt",
  "trigger_name": "serious tilt",
  "trigger_description": "tilt > 40",
  "condition": ">",
  "threshold": 40.0,
  "value": 55.1593,
  "at": "2014-04-30T23:21:35.908Z"
}
```

Next, you'll create a trigger by using the REST API.

Creating a Trigger through the API

Exercise Variant: REST Client (Postman Chrome App)

Sometimes a threshold is constant, and you will want to use the web interface to create a trigger, as you just did. However, sometimes a trigger will need to be dynamically generated or removed under certain circumstances. In this case, we can use the REST API to accomplish this. Follow these steps to add a new trigger for an even higher speed using the Postman Chrome App:

1. Open the Postman add-on for Chrome.
2. In the **Enter request URL here** box type **`http://api-m2x.att.com/v1/feeds/{feed-ID}/triggers`**
3. Replace the `{feed-ID}` with the feed ID from the Blueprints page. You can use the **Copy** button to copy the ID to the clipboard.
4. Set the method dropdown to 
5. If the POST body is not visible, select the  button
6. Add the following into the POST body box:

```
{
  "name": "dangerous tilt",
  "stream": "tilt",
  "condition": ">",
  "value": "60",
  "callback_url": "<url>",
  "status": "enabled"
}
```

7. Replace the `<url>` above with your ResponseBin URL.
8. Your page should look like this:



The screenshot shows the M2X API client interface. At the top, there are tabs for 'Normal', 'Basic Auth', 'Digest Auth', and 'OAuth 1.0'. The 'Normal' tab is selected. Below the tabs, the URL is 'http://api-m2x.att.com/v1/feeds/384a7221ec4a9a3d9b19c30b0a95f2c' and the method is 'POST'. There are buttons for 'URL params' and 'Headers (2)'. The 'Headers' section shows 'Content-Type' as 'application/json' and 'X-M2X-KEY' as '28531911e8b2c361ad535fcb2da6a7064'. Below the headers, there are tabs for 'form-data', 'x-www-form-urlencoded', and 'raw'. The 'raw' tab is selected, and the JSON body is:


```

1 {
2   "name": "dangerous tilt",
3   "stream": "tilt",
4   "condition": ">",
5   "value": "60",
6   "callback_url": "http://requestb.in/oej2njoe",
7   "status": "enabled"
8 }
9
10
11
  
```

 At the bottom, there are buttons for 'Send', 'Preview', 'Add to collection', and 'Reset'.

9. Click **Send**. You should see a response like this:

```

{
  "id": "243",
  "name": "dangerous tilt",
  "stream": "tilt",
  "condition": ">",
  "value": "60",
  "unit": "°",
  "callback_url": "http://requestb.in/oej2njoe",
  "url": "/feeds/384a7221ec4a9a3d9b19c30b0a95f2c5/triggers/243",
  "status": "enabled",
  "created": "2014-04-30T23:42:04.000Z",
  "updated": "2014-04-30T23:42:04.000Z"
}
  
```

10. Return to the M2X website with your feed information. Refresh the page, then click the **Triggers** tab. You should now see your new trigger listed.

The screenshot shows the M2X website interface for the 'tilt' feed. It lists two triggers: 'serious tilt' with a condition of '> 40 °' and 'dangerous tilt' with a condition of '> 60 °'. Each trigger has a toggle switch set to 'ON' and buttons for 'Delete', 'Edit', and 'Test Trigger'.



Note that the **id** returned is the ID for the trigger, which you can then use to [update the trigger](#) by using the PUT method and [remove it](#) by using the DELETE method. See [API Details](#) for more information.



Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Sometimes a threshold is constant, and you will want to use the web interface to create a trigger, as you just did. However, sometimes a trigger will need to be dynamically generated or removed under certain circumstances. In this case, we can use the REST API to accomplish this. Follow these steps to add a new trigger for an even higher speed using the curl:

1. Paste the following into a text editor. This will create a trigger for when the speed goes over 80. (You may have to put in the line breaks after pasting.) The `?pretty=true` parameter will return formatted JSON.

```
curl --request POST \
--header "Content-Type: application/json" \
--header "X-M2X-KEY: {API-key}" \
--data '{ "name": "dangerous tilt", "stream": "tilt", "condition":
">", "value": "60", "callback_url": "{url}", "status": "enabled" }' \
http://api-m2x.att.com/v1/feeds/{feed-ID}/triggers?pretty=true
```

2. Replace {API-key} with your API key. (There should not be curly brackets around it.)
3. Replace {feed-ID} with your feed ID. (There should not be curly brackets around it.)
4. Replace {url} with the URL from ResponseBin. (There should not be curly brackets around it.)
5. Open up a terminal (command prompt in Windows) that has curl.
6. Paste the edited curl command into the terminal and hit Enter. You should see a return value like this:

```
{
  "id": "243",
  "name": "dangerous tilt",
  "stream": "tilt",
  "condition": ">",
  "value": "60",
  "unit": "",
  "callback_url": "http://requestb.in/oej2njoe",
  "url": "/feeds/384a7221ec4a9a3d9b19c30b0a95f2c5/triggers/243",
  "status": "enabled",
  "created": "2014-04-30T23:42:04.000Z",
  "updated": "2014-04-30T23:42:04.000Z"
}
```



Note that the **id** is the ID for the trigger, which you can then use to [update the trigger](#) by using the PUT method and [remove it](#) by using the DELETE method. See [API Details](#) for more information.

7. Return to the M2X website with your feed information. Refresh the page, then click the **Triggers** tab. You should now see your new trigger listed.

tilt				
serious tilt	> 40 °	<input checked="" type="checkbox"/> ON	Delete	Edit Test Trigger
dangerous tilt	> 60 °	<input checked="" type="checkbox"/> ON	Delete	Edit Test Trigger

Summary

You've learned how to add and use a trigger, both using the M2X website and using the M2X Data Service API.



Lab Exercise #4: M2X Server Sample Application

Pre-requisites: git, Heroku (instructions provided)

Length: 30 minutes

In this section:

Running the Sample M2X Server Using Heroku How the App Works

This section shows you how to set up a server that uses the M2X APIs. The sample server code creates a new blueprint with three streams, and sends the server's load to the M2X service at regular intervals.

Running the Sample M2X Server Using Heroku

The sample app requires a web server, and in this exercise, we will use Heroku. Heroku is a Cloud Service Platform that provides free accounts for limited resources.

You will need git, which you can obtain at: <http://git-scm.com/>.

Follow these steps to install the sample code:

1. Sign up for an M2X account, as in the [Setting Up for the M2X Exercises](#) section.
2. Open a terminal (command prompt for Windows).
3. Use [git](#) to clone the git repository, using the command:

```
git https://github.com/attdevsupport/ruby_immn_sample.git
```

4. Follow the instructions at [Heroku](#) to create an account and log in.
5. Follow the steps at <https://devcenter.heroku.com/articles/quickstart> to install the Heroku Toolbelt on your local machine. Skip the step called "Deploy an Application."
6. cd to the directory where you want the code to be created and clone the sample repository with this command:

```
git clone https://github.com/attm2x/m2x-demo-heroku-ruby.git
```

7. cd into the directory that was created by the git clone:

```
cd m2x-demo-heroku-ruby
```

8. Create a new Heroku instance with this command:

```
heroku apps:create m2x-sample
```

9. Next you'll need to get your M2X API Master Key. Log into M2X, and click your name in the upper right-hand corner, then the "Account Settings" dropdown, then the "Master Keys" tab. (Or you can click here: <https://m2x.att.com/account#master-keys-tab>). Copy the Master Key into the clipboard.
10. Edit the file named **m2x_api_key.txt**. It should be empty. Paste in the Master Key and save.
11. Push the changes up to your Heroku app using these commands to add the file, commit it, and then push it.



```
git add m2x_api_key.txt
git commit -m "Updating to use my correct API key."
git push heroku master
```

12. Now your code should be uploaded. You are using a "Clock" process type, which means that the code will not be run automatically. Use this scaling command to set the number of clock processes to 1 so that the code will be running.

```
heroku ps:scale clock=1
```

13. The code will report the system load to the M2X platform every minute. Return to the M2X website and click on **Blueprints**. You should see a blueprint called **loadreport-heroku**.
14. Click on **loadreport-heroku**, and you will see three streams: **load_1m**, **load_5m**, and **load_15m**. These gather data that show the system load averages for the past 1, 5, and 15 minutes respectively.
15. Click on **load_1m**. The Stream graph should show a data point, which is the load. Wait for a couple of minutes, and you should see more than one data point.
16. When you are done, stop the process with the following command:

```
heroku ps:scale clock=1
```

How the Server Code Works

The M2X server demo code is built in Ruby and uses the Ruby M2X client library. The file **loadreport.rb** contains the process flow:

Step 1. Read the API key from the file **m2x_api_key.txt**.

```
APIKEY =
  File.read(File.join(File.dirname(__FILE__), 'm2x_api_key.txt')).strip
```

Step 2. Test to see if a blueprint named "loadreport-heroku" exists. If not, create it.

```
loadreport_blueprint_exists = false
blueprints = m2x.blueprints.list()

lr_blueprint = nil

blueprints.json["blueprints"].each { |bp|
  if bp['name'] == BPNAME
    loadreport_blueprint_exists = true
    lr_blueprint = bp
  end
}

if not loadreport_blueprint_exists
  puts "About to create the blueprint..."
  lr_blueprint = m2x.blueprints.create(name: BPNAME,
    visibility: "private", description: "Load Report")
end
```



Step 3. Call the Unix command **uptime**, and parse it to get the average load for 1 minute, 5 minutes, and 15 minutes.

```
UPTIME_RE = /(\d+\.\d+),? (\d+\.\d+),? (\d+\.\d+)$/
load_1m, load_5m, load_15m = `uptime`.match(UPTIME_RE).captures
```

Step 4. Send the data to **load_1m**, **load_5m**, and **load_15m** streams.

```
m2x.feeds.update_stream(lr_blueprint["id"], "load_1m", value: load_1m)
m2x.feeds.update_stream(lr_blueprint["id"], "load_5m", value: load_5m)
m2x.feeds.update_stream(lr_blueprint["id"], "load_15m", value: load_15m)
```



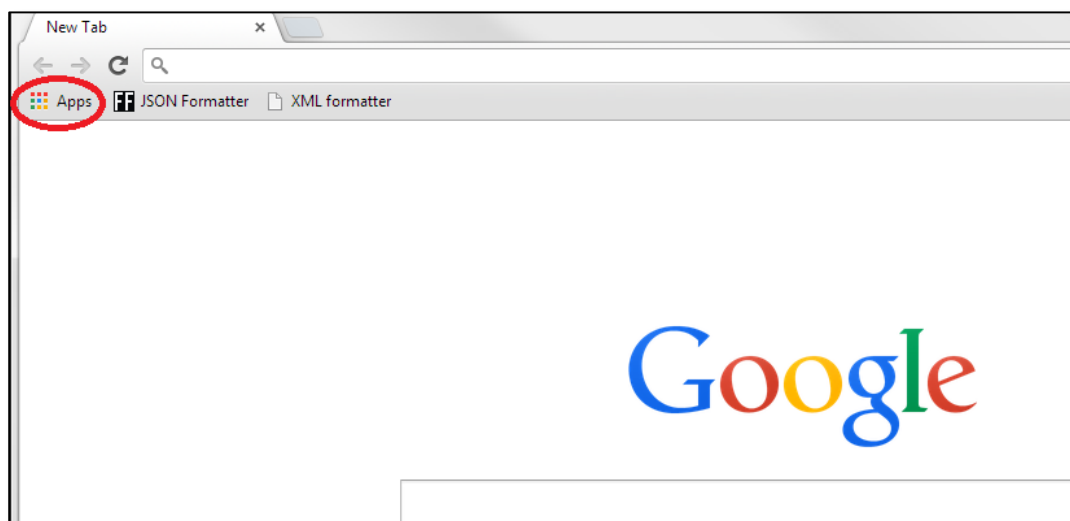
Appendix A: Installing Postman for Chrome

Pre-requisites: Google Chrome

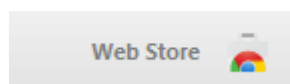
Postman is a Chrome app that lets you make REST calls from a graphical user interface.

Use the following steps to install Postman:

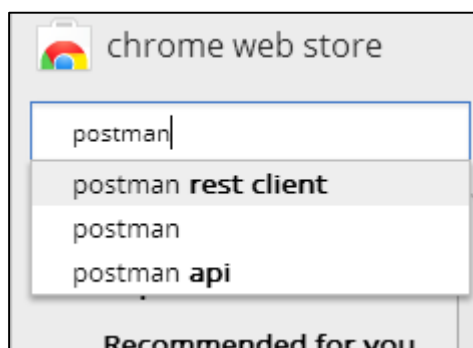
1. Open a new tab in Chrome.
2. In the bookmark bar, select Apps



3. In the very bottom right-hand corner, select **Web Store**.

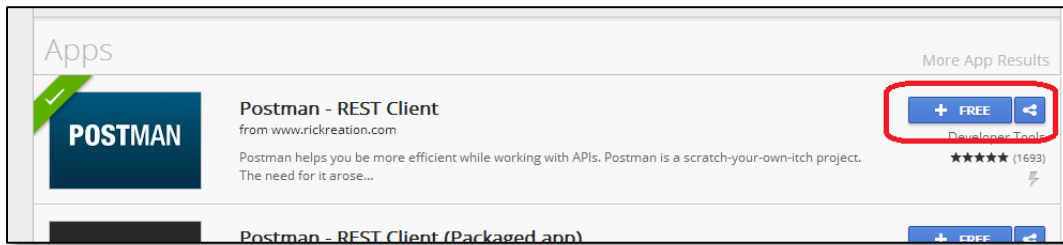


4. In the search box, type in **postman**, and then choose **postman rest client**.

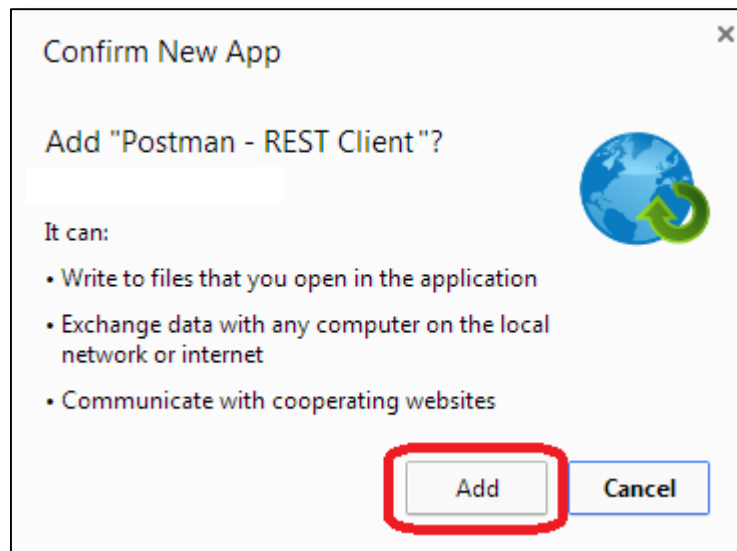


5. Next to Postman – REST client, click on + **Free**.





6. Click **Add** when asked to add the app.



7. Postman is now installed. Whenever you click on the **Apps** icon in the bookmarks bar, you can select Postman to run the app.



Appendix B: Installing curl for Windows

Pre-requisites: Windows 7 or later

curl (or cURL) is a powerful command-line tool for making HTTP calls. It is already installed on Mac computers.

Downloading curl

Use the following steps to install curl:

8. Open <http://curl.haxx.se/dlwiz?type=bin> in a browser.
9. Select your operating system in the dropdown box: either Windows /Win32 or Win 64. Click **Select!**
10. For Win 32, choose whether you will use curl in a Windows Command Prompt (**Generic**) or in a Cygwin terminal (**cygwin**). For Win 64, choose whether you will use curl in a Windows Command Prompt (**Generic**) or MinGW (**MinGW64**). Click **Select!**
11. If required, choose your Windows operating system. **Finish.**
12. Click **Download** for the version which has SSL enabled.
13. Choose a version with support for SSL.
14. Open the downloaded zip file. Extract the files to an easy-to-find place, such as C:\Program Files.

Testing curl

1. Open up the Windows Command Prompt terminal. (From the Start menu, click Run, then type **cmd**.)
2. Set the path to include the directory where you put **curl.exe**. For example, if you put it in **C:\Program Files\curl**, then you would type the following command:

```
set path=%path%;"c:\Program Files\curl"
```

3. Type **curl**.

You should see the following message:

```
curl: try 'curl -help' or 'curl -message' fo r more information
```

This means that curl is installed and the path is correct.

4. Type:

```
curl --insecure https://api.att.com
```

You should see JSON returned:

```
{
  "error": "invalid API request"
}
```



Troubleshooting

SSL certification error

If you see an SSL certification error, add a -k flag into your curl command.

https not supported error

If you get an error that says, "Protocol http not supported or disabled in libcurl", then you need a different version of curl. Make sure you have downloaded one that says SSL enabled. If you have downloaded the Win64 version, try the Win32 version instead, even if you are on a 64 bit machine.



Appendix C: Software Clients

Several software clients are available to make it easier to integrate the AT&T M2X Data Service API into your apps. Rather than requiring you to make low-level HTTP calls, these SDKs and tools provide classes and methods that reduce the amount of code to call the underlying API requests. The following libraries are available from the M2X website at [Client Libraries](#):

- .NET (Microsoft)
- Android (Google)
- Arduino
- BeagleBone
- C
- Cypress PSOC
- Electric Imp
- iOS (Apple)
- Java
- JavaScript
- LaunchPad Energia (Texas Instruments)
- mbed
- Node.js
- PHP
- Python
- Raspberry Pi
- Ruby

