# Heating Monitor

## Block Diagram

Web client to monitor (and control) measurements

Internet

HTTP/REST

Sensors

WiFi/Internet

Measurement system, reads sensor data (temperature, humidity, etc) and send them to the cloud

HTTP

AZURE

HTTP

EVENT HUB

STREAM ANALYTICS

SQL DB

Web service (HTML, node.js)

## Setting up Azure Services

All resources will be set up on same data center, North-Europe region.

Naming of the resources: using project name Heating Monitor as base (HeatingMon).

### Azure Active Directory (AD)

### Service Bus / Event Hub

Service bus namespace : HeatingMon

When creating bus, the bus type (MESSAGING) and the messaging tier must be selected. In Service bus CONFIGURE menu, PRIMARY KEY is the key when sending data to system. The same key must be available for the sender devices. This means that key value must always stay same in this system. Otherwise there must be a method to send new key to measuring devices.

Under Service Bus there are the Event Hubs. Even hub name for the project is heatingmon. The only meaningful settings will be found in CONFIGURE menu. There is MESSAGE RETENTION which tells system how many days messages will be retained in the hub.

### IoT Hub

Name : HeatingMonIoT

There is not much to configure, but in “Shared access policies” menu, you can control access. You get the keys by clicking policy (device) name.

### SQL Database (and Server)

In QUICK CREATE menu you need to give name to the database (HeatingMonDB) and choose the server for the database. You may select “New SQL database server” or if you have any, use some previously created servers. If you select creating new, you will be asked admin login name and password for it. The automatically created server will get a funny name, for example fyn7a28mhq in our project. If you create server yourself, you may select also the name. However, then you need to take more care of server later yourself (Iaas service).

In server CONFIGURATION you need to set allowed IP addresses to manage the database. You should also allow WINDOWS AZURE SERVICES.

There is nothing in database settings that need to be set in menus. From database view, by selecting MANAGE, you can open Microsoft Azure Database management software that you can use managing database, add tables and columns. Another choice to manage database is Microsoft Visual Studio.

### Stream Analytics

Stream Analytics is needed to move data from hub to safe. In this case we will move data to SQL database.

When this resource was created, the name was given HeatingMon. As storage account was selected previously created SQL database. First thing to configure in stream handling is to select INPUTS tab and add new Data Stream job. In next page either Event Hub or IoT Hub should be selected depending which is in use. Following thing is to give some settings, especially the hub selection is important. Serialization format must be JSON and encoding UTF8.

Next task is the OUTPUTS menu where you need to add a job to output data to SQL database. Some settings again for output, where you need to give the database server name, login info and also the table where data will be put.

Last step is define the query which takes care of moving data from input to output. That can be done in QUERY tab. There we have a query box where in our project following SQL-like commands need to be given:

SELECT

cust\_id AS [cust\_id],

device\_id AS [device\_id],

meas\_time AS [meas\_time],

System.Timestamp AS [save\_time],

triggering\_event AS [triggering\_event],

sensor0 AS [sensor0],

sensor1 AS [sensor1],

sensor2 AS [sensor2],

sensor3 AS [sensor3],

sensor4 AS [sensor4],

sensor5 AS [sensor5],

sensor6 AS [sensor6],

sensor7 AS [sensor7]

INTO

[heatingMonOutput]

FROM

[heatingMonInput]

Here is also some query testing tools with which you can do some trials for your query by inputting json-messages from text file.

### Web Application (and Webjob)

The purpose of web application is to handle all database reading and writing through controlled service so that not anybody can access the database.

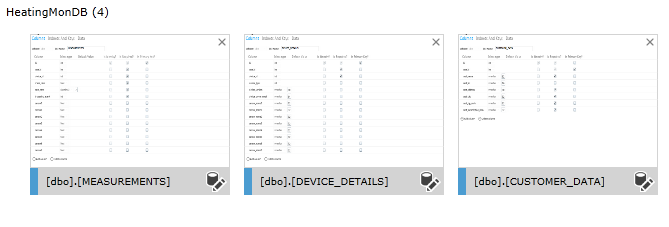
When creating a Web app, you need to give database name and login info. Setting up the service you may start from CONFIGURE. Most of the settings are fine by default but, if possible you should turn ALWAYS ON option to ON state. However, this is not always possible depending on subscription and service plan. Other thing here is to add our main HTML-file (HeatMonitor) to “default documents” list to make server give it as a web page when someone starts browsing our web address. The location on the list may affect to selection.

Still, the webjob to handle database queries must be added in WEBJOBS tab. The name for the job, the running mode and the webjob file(s) must be given.

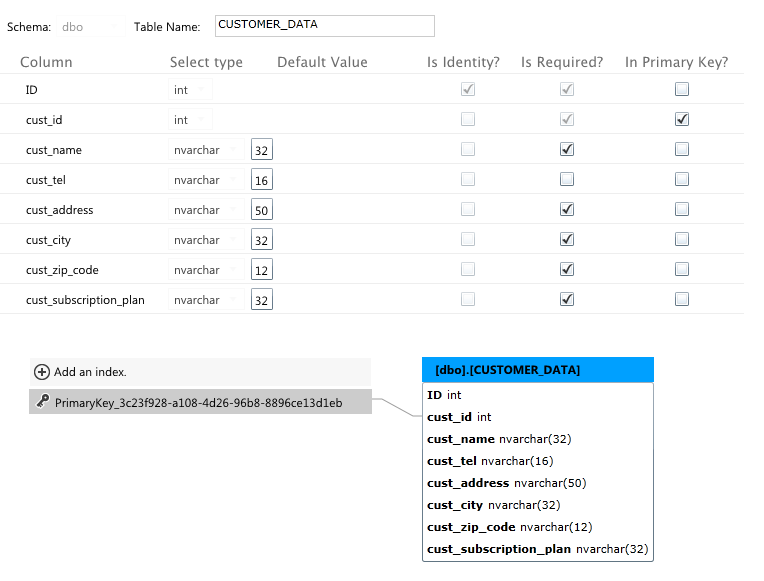
Also web.config file must be set up in web server to tell IIS (Internet Information Server) which process (webjob) REST-messages should be transferred.

## SQL Database (HeatingMonDB)

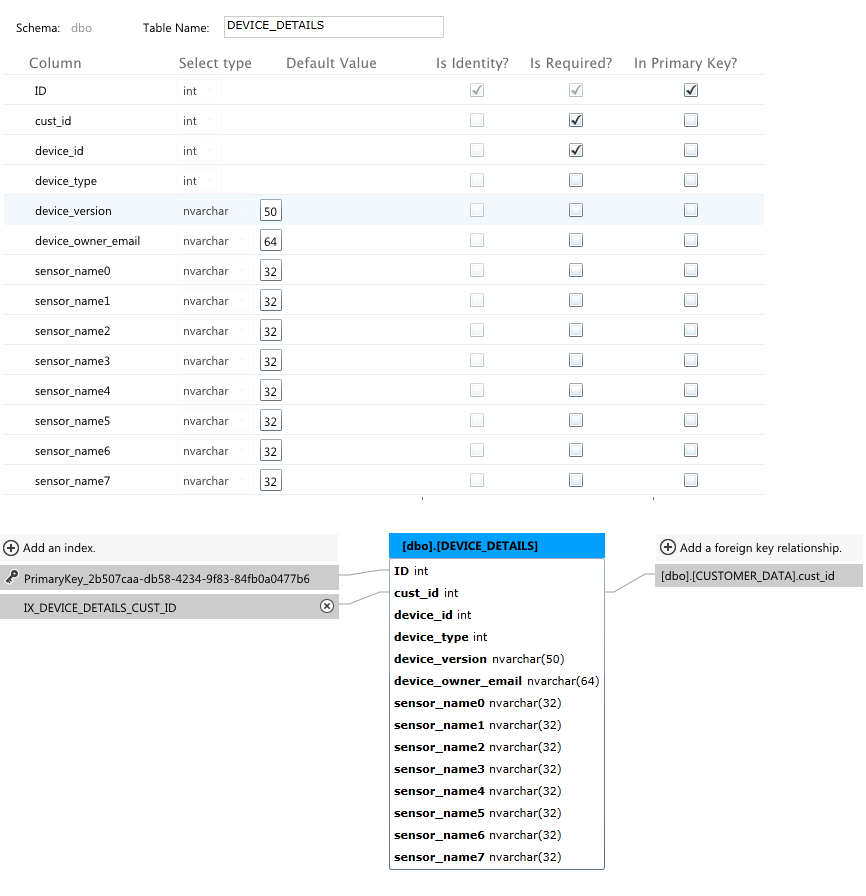
Currently there is three tables:



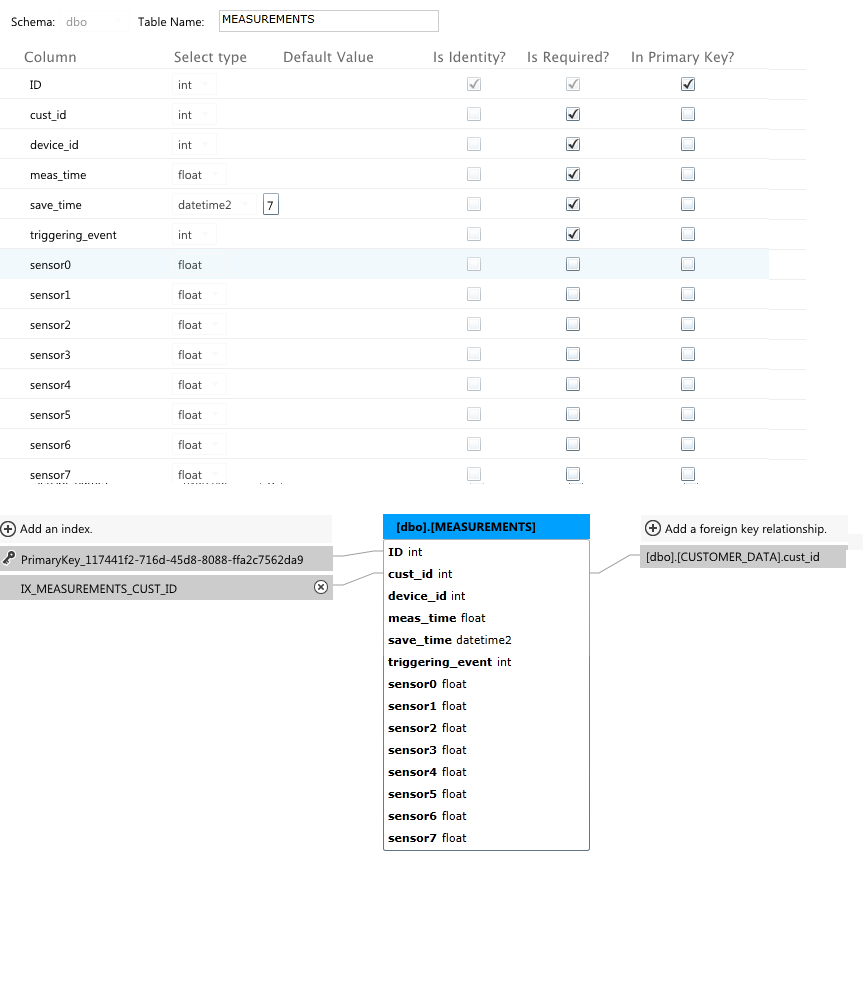
### CUSTOMER\_DATA



### DEVICE\_DETAILS

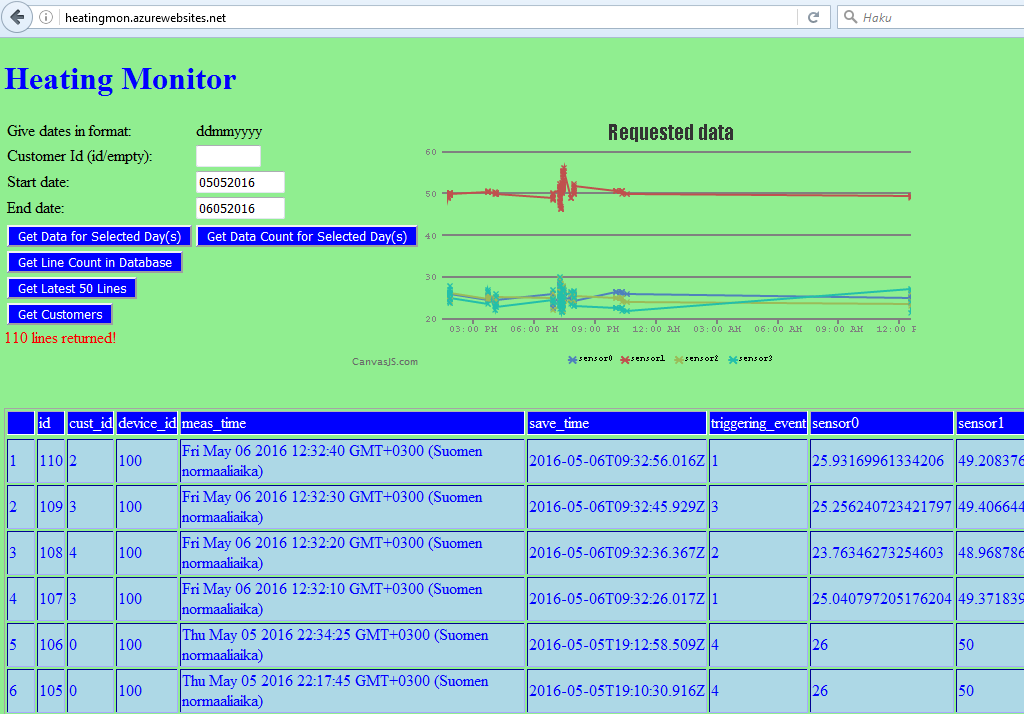


### MEASUREMENTS



## Related Software

Web page prototype (HeatMonitor.html) is set up in Azure to demonstrate the monitoring functionalities together with service process to handle REST-queries to database. This service process must be running on Azure server as WEBJOB.



Purely in demonstration purposes I have developed a command line program in node environment that creates records and sends them to Azure service bus.

Following is format the data must be sent:

-Request options----------------------------------

{ hostname: 'heatingmon.servicebus.windows.net',

port: 443,

path: '/heatingmon/messages',

method: 'POST',

headers:

{ Authorization: 'SharedAccessSignature sr=https%3A%2F%2Fheatingmon.servicebu

s.windows.net%2Fheatingmon%2Fmessages&sig=AXFqrfzxtspQdbI%2BTAiIhLQ6c73YRb1Ceggh2oqvN1s%3D&se=1463823130&skn=heatingmon',

'Content-Length': 191,

'Content-Type': 'application/atom+xml;type=entry;charset=utf-8' } }

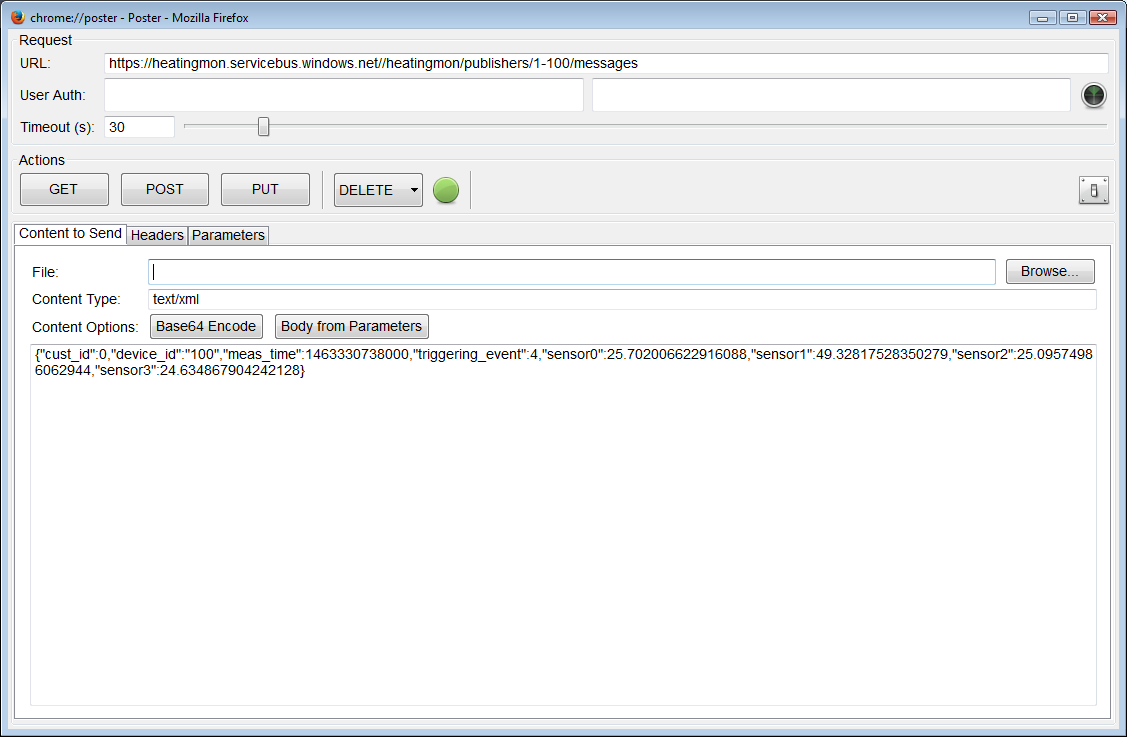
-Payload------------------------------------------

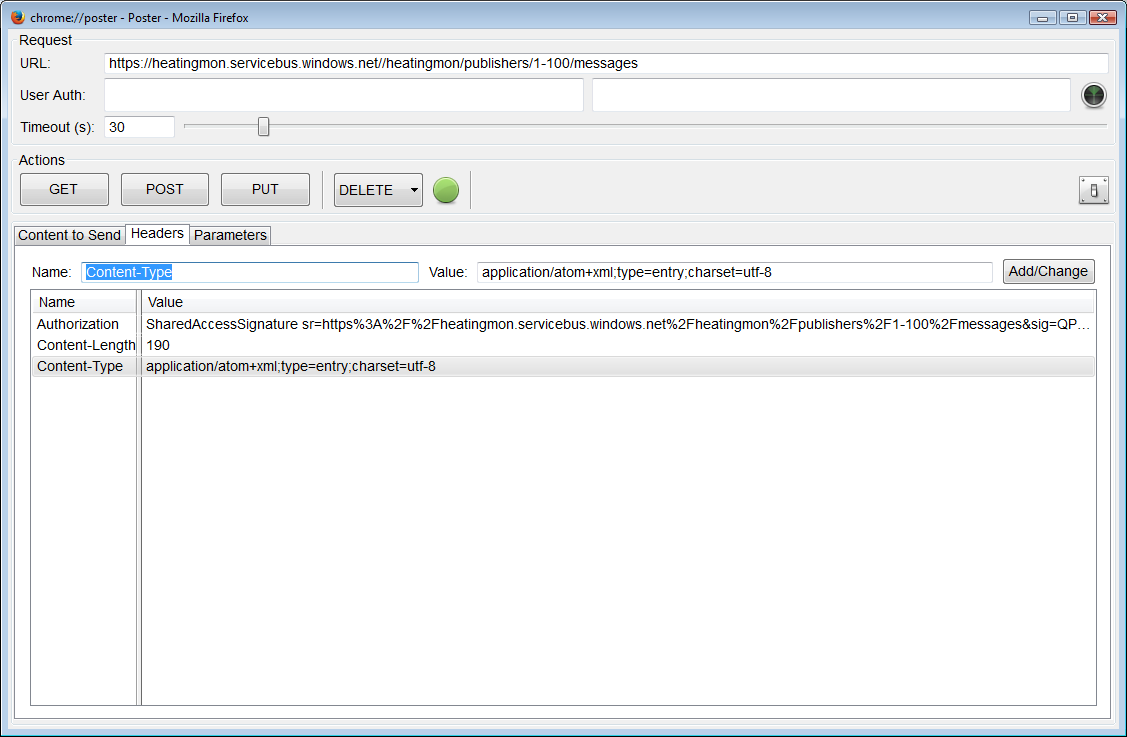
{"cust\_id":4,"device\_id":"100","meas\_time":1462527170000,"triggering\_event":2,"s

ensor0":26.0433927080594,"sensor1":48.47145873354748,"sensor2":26.52142556221224

4,"sensor3":24.464965699939057}

Messages can be sent to Azure also manually using Poster: (note that Authorization key must be correct and active)





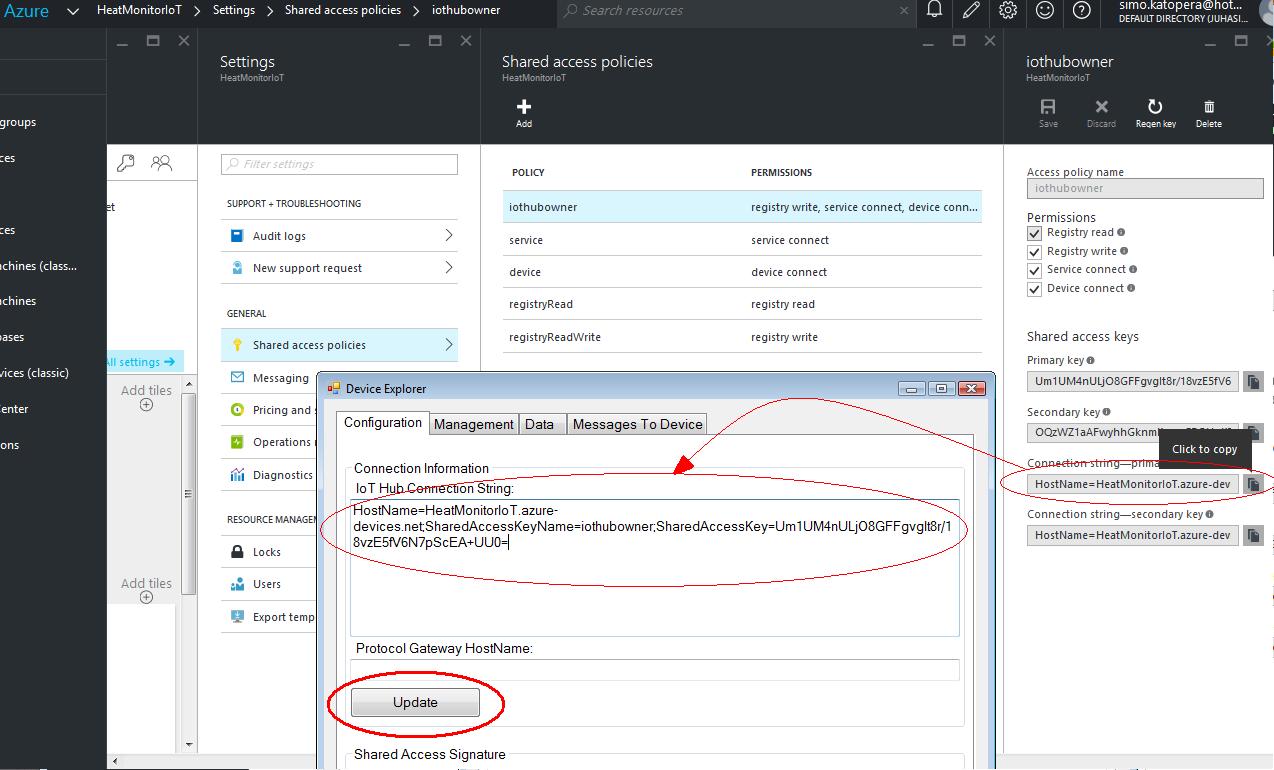
**Authorization format:** SharedAccessSignature sr=https%3A%2F%2Fheatingmon.servicebus.windows.net%2Fheatingmon%2Fpublishers%2F1-100%2Fmessages&sig=QPSRfwZQMSBGhQrz3cMhc%2F9ZZWDij4gI%2BfyjOOiM0XY%3D&se=1464627147&skn=heatingmon

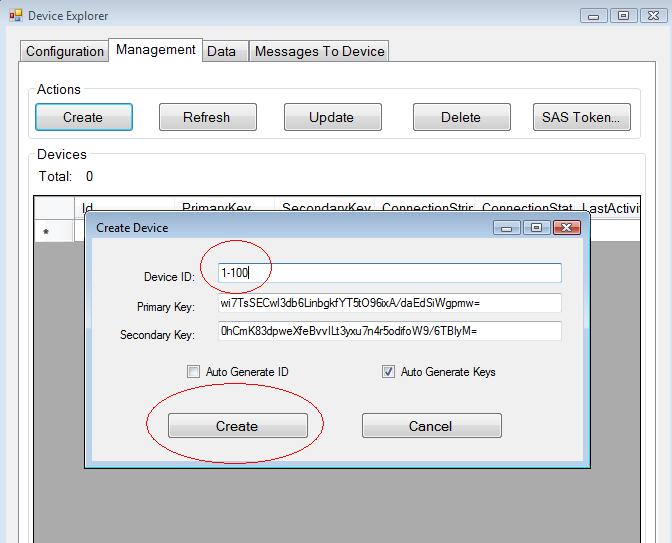
## IoT Hub

We can use iot hub instead of combinations of service bus and event hub . With Device Explorer (in GitHub) you can control devices connected to your iot hub. Each iot hub has got a Device Identity Registry which data need to be modified to allow/disallow devices connect to hub. The field in registry are deviceId, generationId, etag, auth, auth.symkey, status, statusReason, statusUpdateTime, connectionState, connectionStateUpdatedTime and lastActivityTime (see <https://azure.microsoft.com/en-us/documentation/articles/iot-hub-devguide/>).

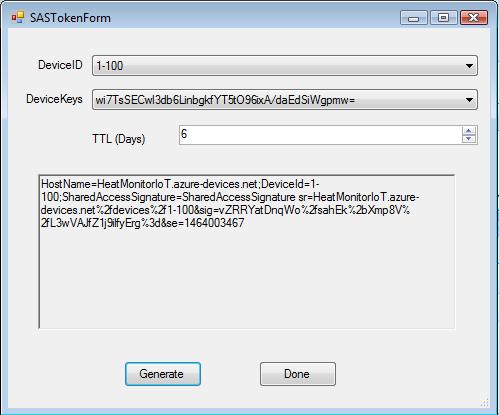
### Device Explorer

Here is how you get Device Explorer in use after loading and installing it. Go to IoT Hub “Shared Access Policies” in Azure portal. Select “iothubowner” policy and copy the Connection string for primary (or secondary) key by clicking “Click to copy” button on the right. Paste it on Device Explorer Configuration sheet’s “IoT Hub Connection String” field. Click Update button to take settings in use.



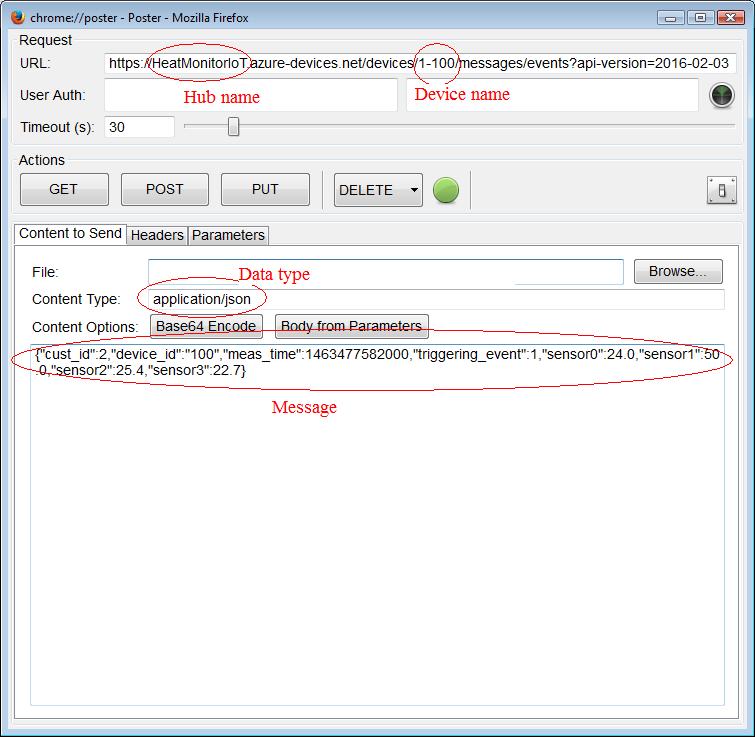
Go to Management sheet on Device Explorer. Click Create button. Give name for device and select Create. Every Device ID should be unique to later allow to control them separately.

Next you should generate “SAS Token” by clicking to button. Select the device you want to create token, give it duration and click “Generate”:

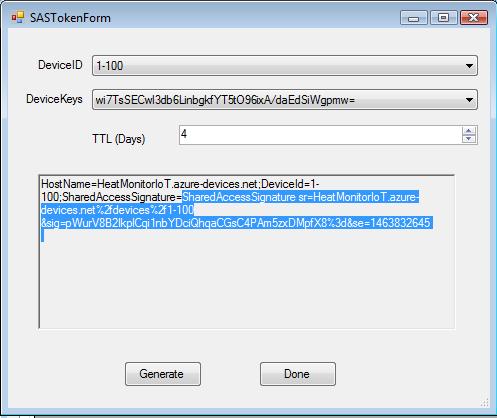


After these you should be able to monitor messages from device to hub (Data) and give commands to device from Device Explorer (Messages to Device).

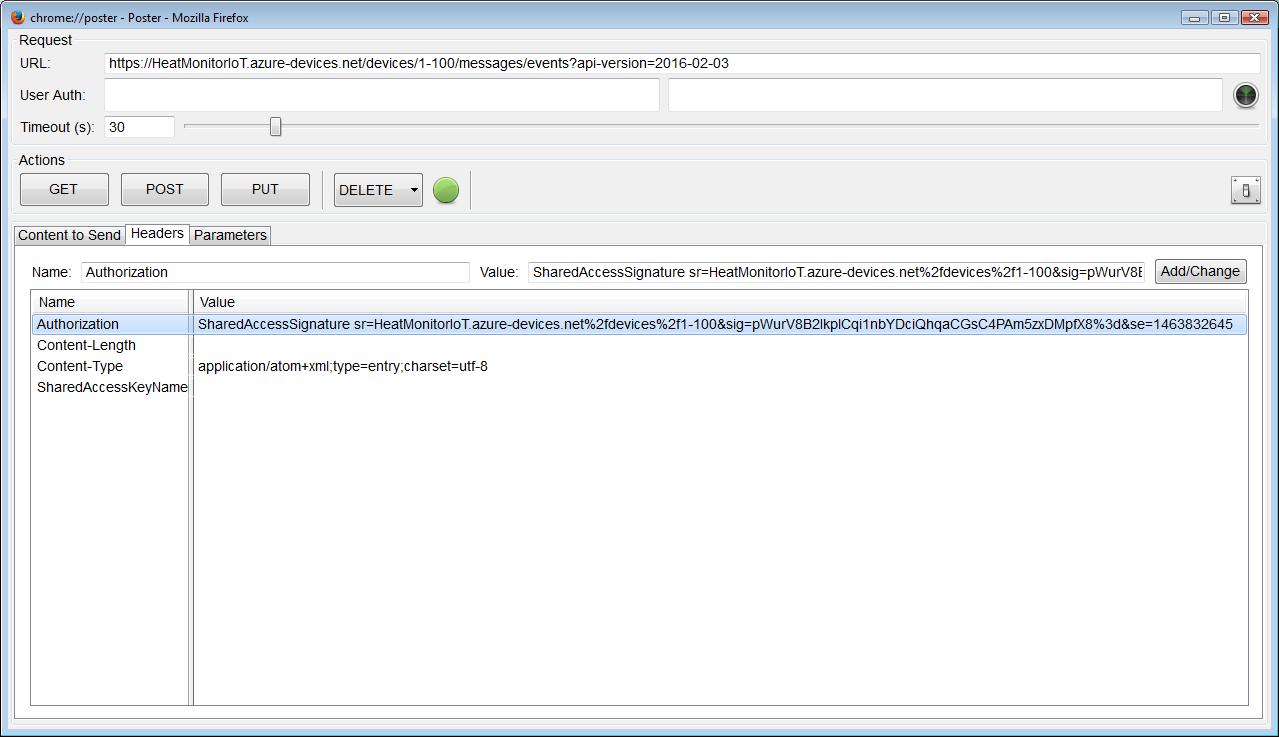
### Posting messages to Azure IoT hub

Sending test data to hub can be easily done using Poster plugin. To send messages, you need to set up Poster as follows. Give the path to IoT hub in URL field. The hub name and device name depend on your hub. In “Content to Send” page, the content type should be set to “application/json”. The data to be sent should be written in the last box in json format. (If the field “meas\_time” is not updated the values in database will show invalid time values.)

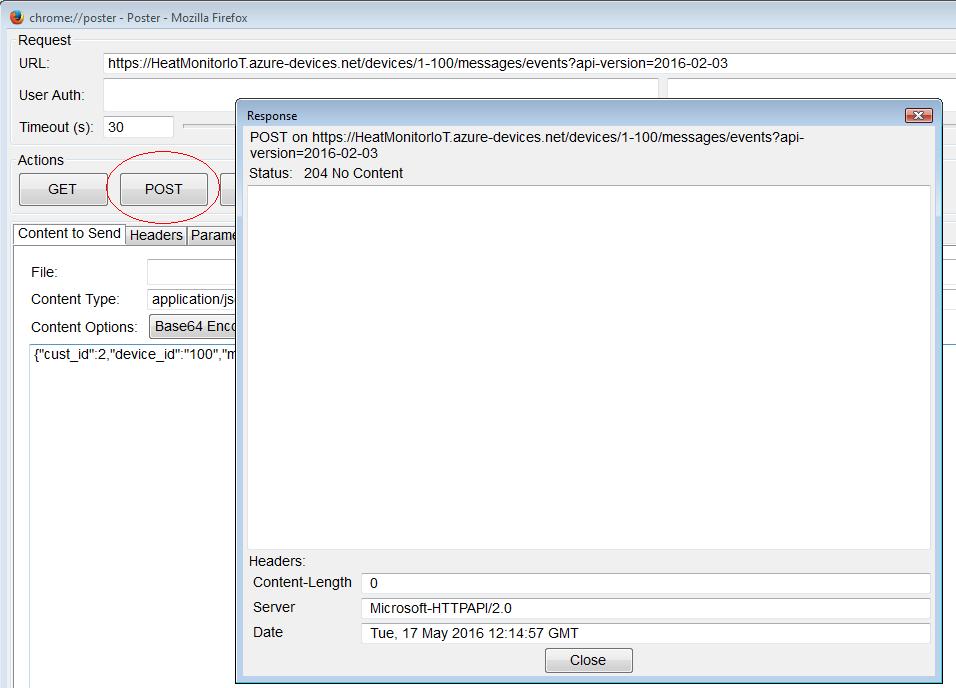
On Headers sheet you need to add “Content-Type” and set the value to “application/atom+xml;type=entry;charset=utf-8” and “Authorization”. The value for Authorization can be copied from Device Explorer’s “SAS token form:



This is what Headers page should look like (don’t care about empty Content-Length and SharedAccessKeyName fields).



Now you can push Post button and response should look like this:



Here, the “Status: 204 No Content” tells that IoT hub received data successfully but did not send any data back.