Public facing github: https://github.com/Tantiv4/giot

Quick Start Guide for Google IoT Core with Qualcomm 4020

Introduction

This guide is a step by step process to set up Google IOT with Qualcomm 4020.
Step 1: Cloud Side Setup
This step involves , creating a Google IOT project,Pub Sub topics and Registry.
Step 2: Steps to be run on Host Machine
This is for secure device creation. User needs to download a tool and follow the instructions to create a device in the cloud. **The device has to be created via this step to work with 4020**. Multiple devices can be created in the same registry using this tool. This sets up the JWT secret string using Tantiv4 key server making the process of setup easier and not having to recompile the code with the secret key inside.
Step 3:Device Side Steps
Three parts to this:

- Environment Setup: Set up the build environment in the host machine
- Build Instructions: Steps for downloading and compiling the SDK
- Flashing Instructions: Steps to flash the board from a windows PC

Step 4:Running GIOT
Steps to run giot from the CDB 4020 command line. Device to cloud communication, cloud to device communication with some examples.
Basic Overview of Google IOT for beginners.

Cloud Side Setup

Setting up the Google IOT infra from the Host computer.

Install the Google Cloud Command Line Tool

Follow all the commands in the link to install google command line tool

Installing the components

```
gcloud components install beta
```

Authenticate with Google Cloud

```
gcloud auth login
```

Create cloud project - choose your unique project name

```
gcloud projects create YOUR_PROJECT_NAME
```

Add permissions for IoT Core

```
gcloud projects add-iam-policy-binding YOUR_PROJECT_NAME
--member=serviceAccount:cloud-iot@system.gserviceaccount.com
--role=roles/pubsub.publisher
```

Set default values for gcloud

```
gcloud config set project YOUR_PROJECT_NAME
```

Create PubSub topic for device data

```
gcloud beta pubsub topics create <iot-topic>
```

Create PubSub subscription for device data

```
gcloud beta pubsub subscriptions create --topic <iot-topic> <iot-subscription>
```

Create device registry

```
gcloud beta iot registries create <iot-registry> --region us-central1
--event-pubsub-topic=<iot-topic>
```

For basic project creation billing needs to be enabled.

Steps to be run on Host machine

Download the below tool to your computer

- For Linux
- For Windows

- For Mac OS

After Downloading the appropriate Tool, please follow the steps:-

Pre requisites

projectid, registryid and regionid(us-central1 in this case) are the parameters which you have already created and used during the cloud setup. Please use the appropriate values for these three parameters to run in this tool.

deviceid is the new device which you are creating using this tool. You can choose any device name. This device will be created under the same projectid, registryid and regionid

Running the tool:

- In linux system, please run the following command:-

```
chmod 777 giot-linux
./giot-linux --projectid=<> --registryid=<> --regionid=<> --deviceid=<>
```

- In a Mac, please run the following command:-

```
chmod 777 giot-macos
./giot-macos --projectid=<> --registryid=<> --regionid=<> --deviceid=<>
```

- In a windows machine, please run the following command:-

```
giot-win.exe --projectid=<> --registryid=<> --regionid=<> --deviceid=<>
```

Checking if device is created:

Check if your registry and device is created in google console

Device Side Setup

Environment Setup

Installing Docker :

Step 1:

Download and install the docker image for your HOST computer from this link

Step 2:

Download the docker container. This download takes a while.

Step 3:

Unzip the container and navigate to the folder *"dockerContainerGiot"* in Command. That folder contains the Dockerfile and gcc tools in compressed form needed to build the binary for the qualcomm platform. These two files are needed for the next step.

Step 4:

Build the docker container by executing

```
docker build -t qc4020 .
```

Create a folder in the host machine to be shared with docker(<host-directory>).

Upon successful completion of the build command the docker container can be started by executing

```
docker run -v <host-directory>:/QCOMM4020/Code/ --interactive --tty qc4020
```

-v option is used to setup shared directory for code.

Host-directory -> absolute path to the directory created in the host machine in the previous step. On Windows the path needs to start with the drive letter colon slash.

Eg. *“docker run -v D:/Code:/QCOMM4020/Code/ --interactive --tty qc4020”* for windows.

“docker run -v /Users/xxx/Code/:/QCOMM4020/Code/ --interactive --tty qc4020” for OSX

This mounts the directory *D:/Code or /Users/xxx/Code/* to your working folder */QCOMM4020/Code/* inside the container.

Step 5:

The successful execution of the above command will give you a working shell inside of docker container irrespective of your Host environment.

If the above steps to setup the docker build environment are successful you will get the shared folder mounted at */QCOMM4020/Code/*

Build Instructions

Step 1:

Register and download the SDK from Qualcomm.

Download the Google IOT patch from here.

For docker: Move the SDK and Patch file to the shared folder (<host-directory>)

Unzip the file in the <host-directory>

Step 2:

Apply the Google IOT patch to Qualcomm standard SDK.

Move the giot_CDB_v1_4.patch to QCA4020xxxxxxx directory

example:

```
mv giot_CDB_v1_4.patch QCA4020.OR.1.1_PostCS1
```

Go to the QCA4020xxxxxxx directory and apply the patch

Example

```
patch -p1 < giot_CDB_v1_4.patch
```

This will apply the Google IOT Patch. To check if the step has gone correctly please check that on the docker shell /QCOMM4020/quartz/demo/QCLI_demo/src - there are two directories “wifi” and “giot”. If the patch is applied correctly then these directories should be there.

Step 3:

In the docker shell navigate to /QCOMM4020/Code/quartz/demo/QCLI_demo/build/gcc/

```
make prepare && make
```

Once this is complete, the last message on docker shell should be “Build Completed…”

This will compile the code and create output folder with flash binaries.

Flashing Instructions

Note: Flashing can currently be done only via Windows Machine. If the compilation is done on a Linux host then the only the output folder under gcc needs to be copied to the output folder under the same directory on the Windows. Please check that all the files in the output directory have recent timestamp, that is one way to check to see the build is successful. If an incorrect build is flashed the Board may not respond at all on the serial terminal.

Step 1:

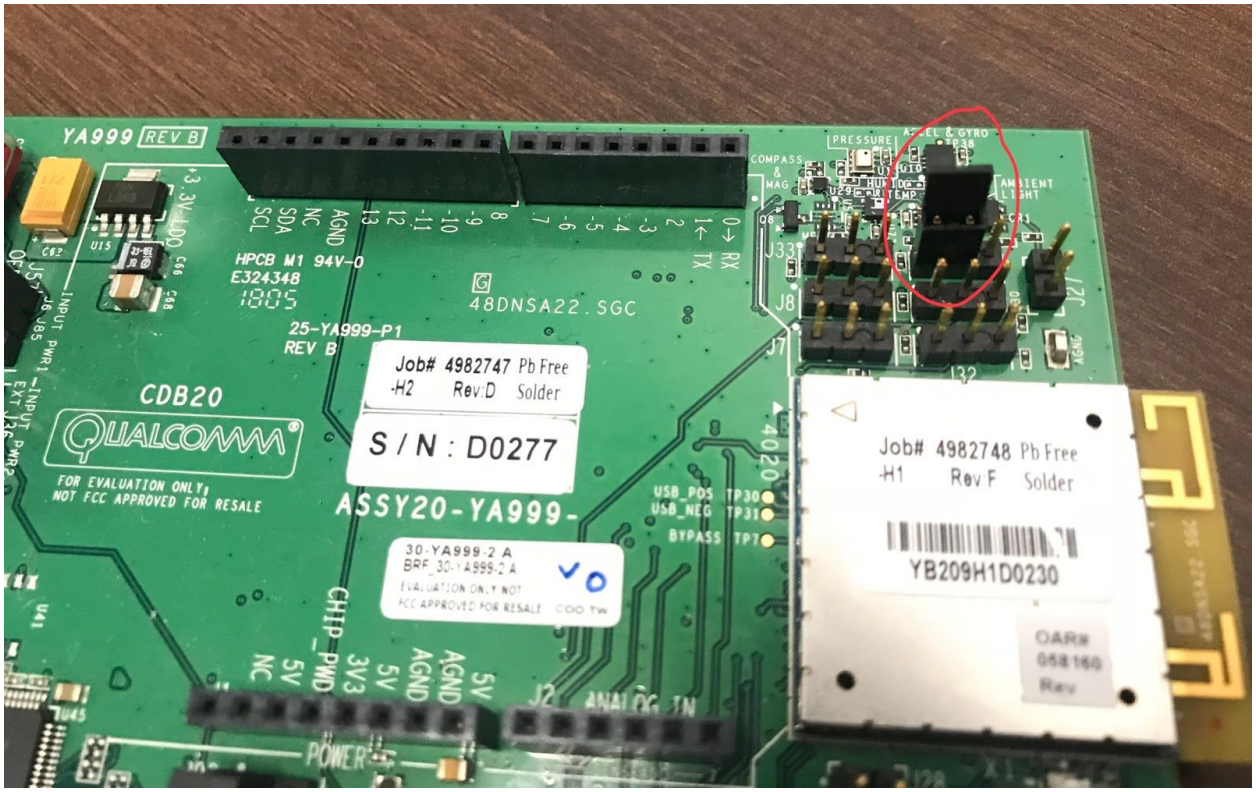
Download and install Python version 2.7 and set the PATH.

Connect both J6 and J85 of the QC4020 CDB board to HOST Machine using the supplied micro-usb cable.

Download and install the driver for Qualcomm UART Interface.

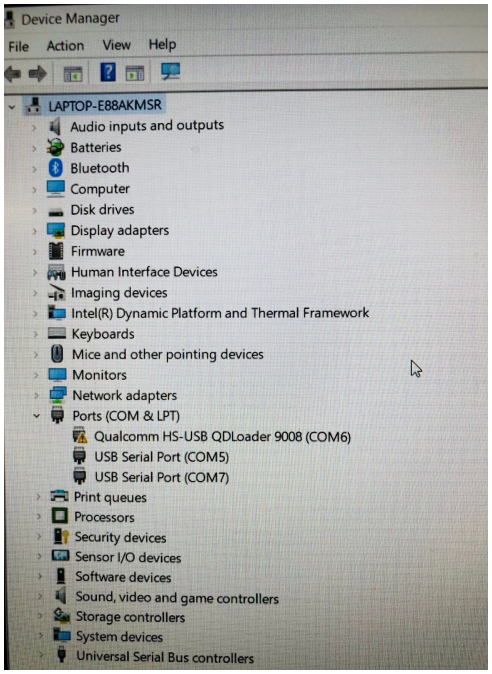
Step 2:

Connect Jumper J34 Pin 1-2 on CDB Board and Power On using switch S7.



Download QDL Driver.

Navigate to Device Manager and Expand the Ports Section.



Right-Click on “Qualcomm HS-USB QDLoader 9008” and select “Update Driver”

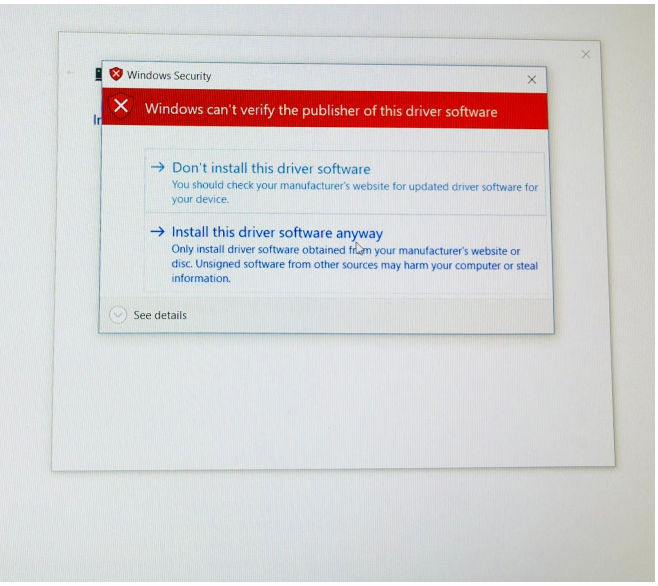
Select 2nd Option “Browse my computer for driver software”

Select “Let me pick up from the list of available drivers on my computer”

Click on “Have Disk” option and select qscer after navigating to Qualcomm_Drivers_QDLoader/Qualcomm Drivers/Qualcomm/Driver/ folder.

Click Next

On the next dialog select “Install this driver software anyway”



This will install the driver for Qualcomm QDLoader.

Note the COM Port number of QDL Loader.

Step 3:

Note: Use Windows Command Prompt only.

Run the following commands from the “gcc” directory on the host.

The gcc directory comes from the download of the Qualcomm SDK in the Step 1 of the Build Instructions earlier. Under unzipped set of files look for the following path: … \quartz\demo\QCLI_demo\build\gcc

The “output” directory under it is what is used for flashing and it should be copied from wherever the build took place. If the build was done on a Linux machine it needs to be copied from there. If there was no build just the output directory is passed around for flashing then it needs to copied at this place on the Windows machine which is being used for flashing. In this case, please download the Qualcomm SDK (step 1 in the build instructions) and unzip it in a folder. Then within that unzipped directory, change directory to path mentioned above.

```
set FS2IMG=KEEP
```

```
python ../../../../../build/tools/flash/qflash.py --comm 8
```

Once the prompt shows “Flash programming complete!” remove the jumper from j34 and power cycle the CDB board.

Running GIOT on CDB 4020

Connect the EVK to the Host computer via serial cable(J85)

J6 can be connected to a power outlet or to the host PC.

It will populate two COM(x) ports. Pick the one with higher value of x.

Open terminal 115200-8-N-1

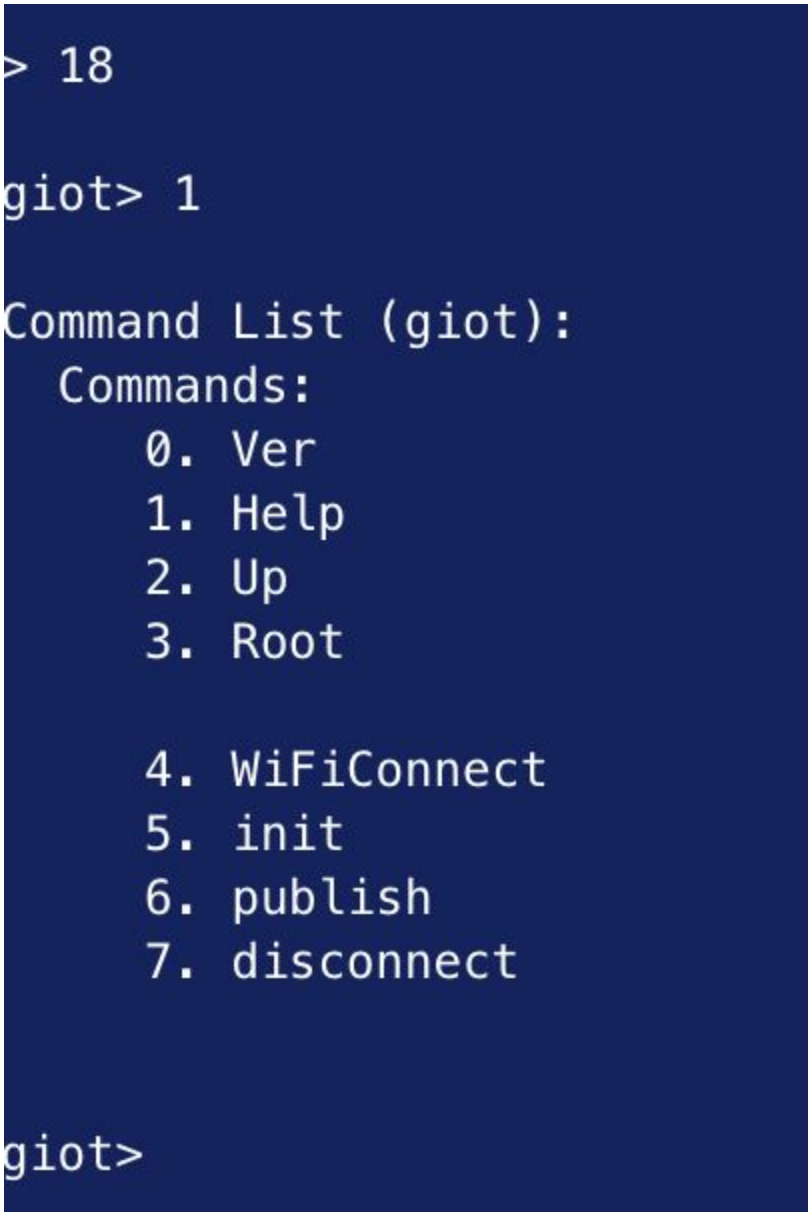
At the terminal prompt use command ‘1’ to see the available CLI options.

```
> 1

Command List:
  Commands:
     0. Ver
     1. Help
     2. Exit

  Subgroups:
     3. BLE
     4. HMI
     5. WLAN
     6. Net
     7. Coex
     8. FwUp
     9. LP
    10. Fs
    11. Ecosystem
    12. SecureFs
    13. Crypto
    14. ZigBee
    15. Thread
    16. Platform
    17. JSON
    18. giot
```

- Select option ‘giot’ to go to giot sub menu. Select 1 to see the options



Connecting to Network

- Use option 4 to connect to network
- Usage:

```
4 <SSID> <Passphrase>
```

- Passphrase can be left blank in case of a open network
- The device will connect to the network and get IP

GIOT initialization

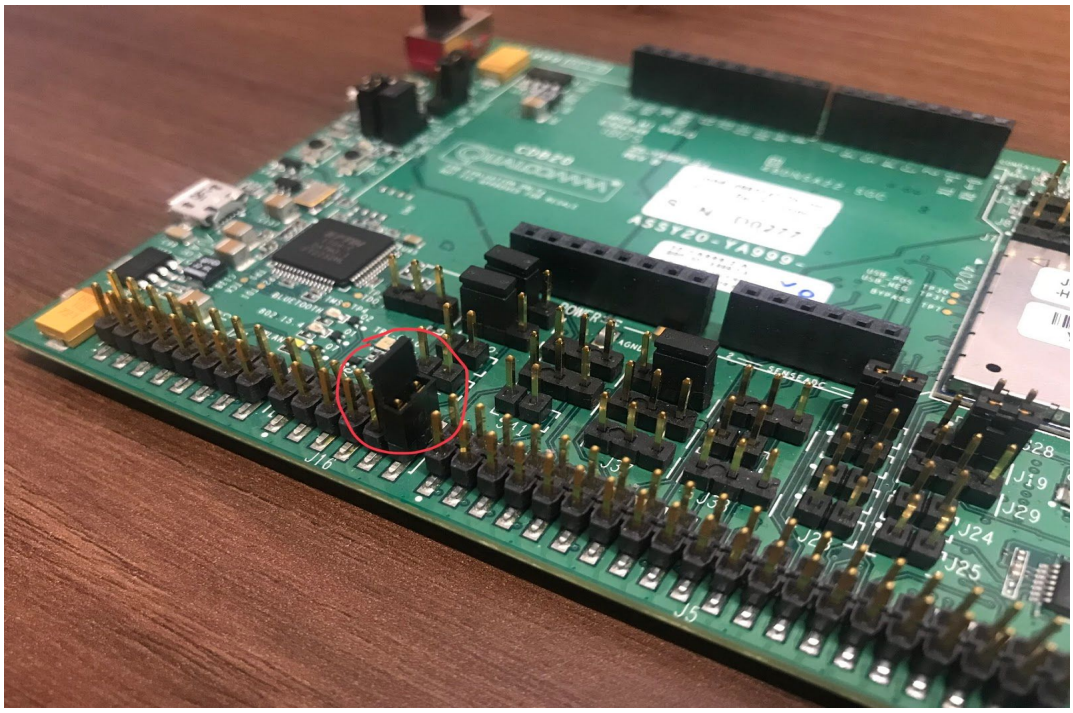
- Inside the GIOT menu
- init
    - init -p <project_id> -r <registry_id> -d <device_ID> - R <region ID>
    - Make sure the project id, registry id, device id and region id are the same as already used for device creation
- Upon successful Connection ,a message “Welcome to 4020” will be sent as a device state.
- In case of error(SSL error), disconnect from GIOT and connect again
- To disconnect use option 7 under GIOT commands
- To check the state go to Google IOT console.
    - Select the appropriate registry
    - Select the device from the devices list
    - Go to “Configuration and State History”

Sending Message from Cloud to Device

- To send a message to device use UPDATE CONFIG option in Google IOT Console. Any text message can be sent.
- The message can be seen in the serial console of the device

Configuring LED from IOT console

- LED connections for QC4020 CDB
    - Use the below jumper setting in J16



- To turn on the LED, send the message “Turn ON LED” from the UPDATE CONFIG in Google IOT console
- To turn off the LED, send the message “Turn OFF LED”

Sending Message from Device to Cloud

- To send message from Device to Cloud:
    - publish -m < message>-t <events/state>
    - Message:- Is the message that will be sent
    - -t is optional. Default is state. ‘-t’ will send the telemetry data
    - Example to publish state:

      publish -m “Hello world”

Hints and Troubleshooting

Device Side Setup

Step 2: Verify by:

C:\>docker --version

Docker version 18.03.1-ce, build 9ee9f40

Step 4: Before giving this command the docker service should be running on the host PC. On Windows sometimes the Docker does not start properly for various reason. It is best to restart it in such situations.

After flashing no activity on the serial terminal

Check the J34 jumper is removed. On Windows in device manager there should not be any QDL Loader port. That port is the flashing port and comes only when the the Jumer J34 is inserted.

It is possible that image being flashed is not built properly. At the end of the “Build Instructions” the “output” folder under “gcc” should have files built with recent time-stamp. Note that the date and time in the Docker container may not match the host PC time. The timestamp chgeck has ti be done on the shell prompt of the docker container because the code is built in the container. It is copied across for flashing from the shared folder on the host shared under Step 4 of the “Device Side setup.”