Google Cloud

Last update: 15-October-2018

IoT Device Integration Cheatsheet

IoT Device Integration Command-line Summary

This Cheatsheet summarizes the commands used in the IoT Device Integration Guide.

The commands and responses shown were executed on Raspberry Pi 3 Model B+ with Raspian 9 to set up the device with all the necessary libraries to run the Python MQTT sample application, cloudiot_mqtt_example.py.

To use this document as a starting point, copy/paste the commands from the Appendix into a text editor and replace variables with your own names.

You must have an account on Google Cloud Platform.

1. To get a sense of your baseline system before starting, check the version of the OS. There are many ways to do that, but this is one of them:

```
cat /etc/os-release
```

Example response

```
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

2. You'll need OpenSSL for keys, so make sure it's on the system and that it's a recent version:

```
openssl version
```

Example response

```
OpenSSL 1.1.0f  25 May 2017
```

3. The Raspbian OS is a fresh new install, so as usual, update and upgrade all libraries. Note that **you're switching to root (#)** to do these and several subsequent steps:

```
sudo raspi-config
sudo -s
apt update && apt upgrade && apt dist-upgrade
```

Example response

```
Hit:1 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Hit:2 http://archive.raspberrypi.org/debian stretch InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
8 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  libfm-data libfm-extra4 libfm-gtk-data libfm-gtk4 libfm-modules libfm4
  python-unicornhathd python3-unicornhathd
8 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 592 kB of archives.
After this operation, 15.4 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.raspberrypi.org/debian stretch/ui armhf libfm-data all 1.2.5-1+rpi6
    [234 kB]
. . .
```

4. Update the firmware. Note you are still **root** and that the device will reboot:

```
rpi-update && reboot
```

5. Install all the libraries and other dependencies listed in “Set up required libraries on the device” so that the device can initiate TLS handshake over MQTT, handle JSON from Python, and sign JSON Web Tokens. These commands below handle the tasks. Note that you are still root (#):

```
apt install build-essential libssl-dev libffi-dev python-dev
```

Example response

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.3).
python-dev is already the newest version (2.7.13-2).
python-dev set to manually installed.
The following additional packages will be installed:
  libssl-doc
The following NEW packages will be installed:
  libffi-dev libssl-dev libssl-doc
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,987 kB of archives.
After this operation, 10.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
. . .
```

```
pip install pyjwt paho-mqtt cryptography
```

Example response

```
Requirement already satisfied: pyjwt in /usr/lib/python2.7/dist-packages
Collecting paho-mqtt
  Downloading
    https://files.pythonhosted.org/packages/25/63/db25e62979c2a716a74950c9ed658dce431b5cb01f
    de29eb6cba9489a904/paho-mqtt-1.4.0.tar.gz (88kB)
    100% |████████████████████████████████| 92kB 1.3MB/s
Requirement already satisfied: cryptography in /usr/lib/python2.7/dist-packages
Building wheels for collected packages: paho-mqtt
  Running setup.py bdist_wheel for paho-mqtt ... done
  Stored in directory:
    /root/.cache/pip/wheels/82/e5/de/d90d0f397648a1b58ffeea1b5742ac8c77f71fd43b550fa5a5
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.4.0
root@tech-lab-pi:/home/pi/dev#
```

6. Install Google API client libraries for Python (to run the example and test the setup):

```
pip install --upgrade google-api-python-client
```

Example response

```
Collecting google-api-python-client
  Downloading
    https://files.pythonhosted.org/packages/4e/92/e4746e646585c8c359781c19984fe8b6b8794a6cfe
    382cd481329d5252ac/google-api-python-client-1.7.4.tar.gz (141kB)
    100% |████████████████████████████████| 143kB 1.3MB/s
Collecting google-auth-httplib2>=0.0.3 (from google-api-python-client)
  Downloading
    https://files.pythonhosted.org/packages/33/49/c814d6d438b823441552198f096fcd0377fd6c8871
    4dbed34f1d3c8c4389/google_auth_httplib2-0.0.3-py2.py3-none-any.whl
Collecting google-auth>=1.4.1 (from google-api-python-client)
  Downloading
    https://files.pythonhosted.org/packages/58/cb/96dbb4e50e7a9d856e89cc9c8e36ab1055f9774f7d
    85f37e2156c1d79d9f/google_auth-1.5.1-py2.py3-none-any.whl (65kB)
    100% |████████████████████████████████| 71kB 1.9MB/s
Collecting httplib2<1dev,>=0.9.2 (from google-api-python-client)
  Downloading
. . .
```

7. Install the client libraries for Cloud IoT Core:

```
pip install --upgrade google-cloud-core
```

Example response

```
Collecting google-cloud-core
  Downloading
    https://files.pythonhosted.org/packages/0f/41/ae2418b4003a14cf21c1c46d61d1b044bf02cf0f8f
    91598af572b9216515/google_cloud_core-0.28.1-py2.py3-none-any.whl
Collecting google-api-core<2.0.0dev,>=0.1.1 (from google-cloud-core)
  Downloading
    https://files.pythonhosted.org/packages/a5/be/de30100034c391f4c56e2543f1507eb1b30b3030bd
    9a6764dd6cfe7a954e/google_api_core-1.4.1-py2.py3-none-any.whl (53kB)
    100% |████████████████████████████████| 61kB 1.8MB/s
Collecting protobuf>=3.4.0 (from google-api-core<2.0.0dev,>=0.1.1->google-cloud-core)
```

```
 Downloading
. . .
```

8. Install client libraries for Cloud Pub/Sub:

```
pip install --upgrade google-cloud-pubsub
```

Example response

```
Collecting google-cloud-pubsub
  Downloading
    https://files.pythonhosted.org/packages/e5/a0/3360ac59c93af4e65a4fb80e88c59139a953286668
    0efb1bc792d51909a2/google_cloud_pubsub-0.38.0-py2.py3-none-any.whl (100kB)
    100% |████████████████████████████████| 102kB 1.3MB/s
Requirement already up-to-date: enum34; python_version < "3.4" in
    /usr/lib/python2.7/dist-packages (from google-cloud-pubsub)
Requirement already up-to-date: google-api-core[grpc]<2.0.0dev,>=1.1.0 in
    /usr/local/lib/python2.7/dist-packages (from google-cloud-pubsub)
Collecting grpc-google-iam-v1<0.12dev,>=0.11.1 (from google-cloud-pubsub)
  Downloading
. . .
```

9. Switch back to non-root account:

```
exit
```

Example response

```
# exit
$
```

10. Go to the [Google Cloud SDK Documentation]() page and download the appropriate version of the SDK tools for your development platform or the device OS. The example installs the [Linux version of the Google Cloud SDK tools for Linux 32-bit]() on the Raspberry Pi device (version numbers and thus filenames change over time).

Note	Rather than installing the SDK on the device and running all commands from the device, you can use the [Google Cloud shell]() to perform the tasks.

When you have the file downloaded to your device, you can unpack it. For example:

```
tar -zxvf google-cloud-sdk-219.0.1-linux-x86.tar.gz
```

Example response

```
pi@tech-lab-pi:~/dev $
```

11. The gCloud command-line tool needs Python 2.7.9 (or later) on the system. Let's check (and it looks fine, so no need to upgrade):

`python --version`

Example response

```
Python 2.7.13
```

12. Add Google Cloud SDK tools to the path by running the install script:

`./google-cloud-sdk/install.sh`

Example response

```
Welcome to the Google Cloud SDK!

To help improve the quality of this product, we collect anonymized usage data
and anonymized stacktraces when crashes are encountered; additional information
is available at <https://cloud.google.com/sdk/usage-statistics>. You may choose
to opt out of this collection now (by choosing 'N' at the below prompt), or at
any time in the future by running the following command:

    gcloud config set disable_usage_reporting true

Do you want to help improve the Google Cloud SDK (Y/n)? n
The Google Cloud SDK installer will now prompt you to update an rc
file to bring the Google Cloud CLIs into your environment.

Enter a path to an rc file to update, or leave blank to use
[/home/pi/.bashrc]:
Backing up [/home/pi/.bashrc] to [/home/pi/.bashrc.backup].
[/home/pi/.bashrc] has been updated.

==> Start a new shell for the changes to take effect.

For more information on how to get started, please visit:
  https://cloud.google.com/sdk/docs/quickstarts
Open a new terminal so that the changes take effect.
```

13. Open new terminal session and configure gCloud so that it can login to your GCP account:

`./google-cloud-sdk/bin/gcloud init`

Example response

```
Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic (1/1 checks) passed.
You must log in to continue. Would you like to log in (Y/n)?  y
Your browser has been opened to visit:
https://accounts.google.com/o/oauth2/auth?redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&prom
```

```
pt=select_account&response_type=code&client_id=32555940559.apps.googleusercontent.com&scope=
https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2F
auth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%
2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.rea
uth&access_type=offline
 --disable-quic --enable-tcp-fast-open --disable-gpu-compositing
    --ppapi-flash-path=/usr/lib/chromium-browser/libpepflashplayer.so
    --ppapi-flash-args=enable_stagevideo_auto=0 --ppapi-flash-version=
Fontconfig warning: "/etc/fonts/fonts.conf", line 160: blank doesn't take any effect
    anymore. please remove it from your fonts.conf
[10265:10265:1006/183407.235443:ERROR:gpu_process_transport_factory.cc(1029)] Lost UI shared
    context.
Created new window in existing browser session.
You are logged in as: [kwiseth@neodada.org].

Pick cloud project to use:
 [1] neodada-project-002
 [2] neodada-testbed-001
 [3] Create a new project
Please enter numeric choice or text value (must exactly match list
```

14. If you don't create a new project during the initialization of gCloud (above), you can create one using the command:

```
gcloud projects create <project-id> --name=<project-name>
```

Example response

```
pi@tech-lab-pi:~/dev $ gcloud projects create neodada-integration --name=integration-test
```

15. Give the project permission to send messages from Cloud IoT Core to Cloud Pub/Sub:

```
gcloud projects add-iam-policy-binding <project-id>
--member=serviceAccount:cloud-iot@system.gserviceaccount.com--role='roles/p
ubsub.publisher'
```

Example response

```
bindings:
- members:
  - user:kwiseth@neodada.org
  role: roles/owner
- members:
  - serviceAccount:cloud-iot@system.gserviceaccount.com
  role: roles/pubsub.publisher
etag: BwV3rGXxFsY=
version: 1
pi@tech-lab-pi:~/dev $
```

16. Enable the necessary APIs:

```
gcloud services enable cloudiot.googleapis.com
```

Example response

```
Operation "operations/acf.e6050272-e2f4-4753-b689-50a39062e4e8" finished successfully.
```

```
gcloud services enable pubsub.googleapis.com
```

Example response

```
Operation "operations/acf.ac6448a0-737b-49b4-9bad-29168658655d" finished successfully.
```

17. Create a topic to add to the project:

```
gcloud pubsub topics create <topic-name>
```

Example response

```
Created topic [projects/neodada-integration/topics/integration-test-topic].
```

18. Create a subscription for the topic:

```
gcloud pubsub subscriptions create <subscription-name> --topic=<topic-name>
```

Example response

```
Created subscription [projects/neodada-integration/subscriptions/integration-test-sub].
```

19. Create a registry for devices:

```
gcloud iot registries create <registry-name> --region=us-central1
--project=<project-id> --event-notification-config=topic=<topic-name>
```

Example response

```
Created registry [int-test-registry].
```

20. Create the public/private key pair which will be associated with the device when you add it to the registry (in the next step):

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout my_private.pem -days 365
-out my_pub_cert.pem -subj "/CN=unused"
```

Example response

```
Generating a 2048 bit RSA private key
.................................+++
........+++
writing new private key to 'my_private.pem'
-----
```

21. Add the device to the registry:

gcloud iot devices create <device-name> --project=<project-id> --region=us-central1 --registry=<registry-name> --public-key path=my_pub_cert.pem,type=rs256

Example response

```
Created device [int-test-device].
pi@tech-lab-pi:~/dev $
```

22. The example MQTT [Python example] can now be run against your Google Cloud Platform instance. Download the example from GitHub, from the GoogleCloudPlatform Python doc samples repository. The path to this example is:

    python-docs-samples/iot/api-client/mqtt_example/cloudiot_mqtt_example.py

23. To run this example (and to run any other Google examples), you must have Google's trusted certificates file (roots.pem) available on the device to ensure that when the device tries to connect to GCP for any given session, the server that responds is indeed GCP and not an imposter service:

    wget https://pki.google.com/roots.pem

    Example response

```
--2018-10-05 21:13:40--  https://pki.google.com/roots.pem
Resolving pki.google.com (pki.google.com)... 2607:f8b0:4005:80a::200e,
   216.58.195.78
Connecting to pki.google.com (pki.google.com)|2607:f8b0:4005:80a::200e|:443...
   connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-pem-file]
Saving to: ‘roots.pem’

roots.pem               [ <=>                 ]  74.49K  --.-KB/s    in 0.03s

2018-10-05 21:13:40 (2.45 MB/s) - ‘roots.pem’ saved [76276]

pi@tech-lab-pi:~/dev $
```

24. Pass the specifics of your project ID, the name of the device, and other details to the Python script:

    python cloudiot_mqtt_example.py --project_id=<project-id> --registry_id=<registry-name> --cloud_region=us-central1 --device_id=<device-name> --private_key_file=my_private.pem --algorithm=RS256 --message_type=event

    Example response

```
Creating JWT using RS256 from private key file my_private.pem
Publishing message 1/100: 'int-test-registry/int-test-device-payload-1'
('on_connect', 'Connection Accepted.')
Publishing message 2/100: 'int-test-registry/int-test-device-payload-2'
Received message '' on topic '/devices/int-test-device/config' with Qos 1
Publishing message 3/100: 'int-test-registry/int-test-device-payload-3'
Received message '' on topic '/devices/int-test-device/config' with Qos 1
```

```
on_publish
on_publish
Publishing message 4/100: 'int-test-registry/int-test-device-payload-4'
on_publish
on_publish
Publishing message 5/100: 'int-test-registry/int-test-device-payload-5'
on_publish
Publishing message 6/100: 'int-test-registry/int-test-device-payload-6'
. . .
```

25. When the script completes, pull the subscriptions to see some of the details:

    `gcloud pubsub subscriptions pull <subscription-name>`

    Example response

```
┌─────────────────────────────────────────────┬─────────────────┬──────────────────
   ────────────────────┬───────────────────────────────────────────────────────────
   ────────────────────────────────────────────────────────────────────────────────
   ─────────────────────────┐
│                     DATA                    │    MESSAGE_ID   │
   ATTRIBUTES              │
   ACK_ID
    │
├─────────────────────────────────────────────┼─────────────────┼──────────────────
   ────────────────────┼───────────────────────────────────────────────────────────
   ────────────────────────────────────────────────────────────────────────────────
   ─────────────────────────┤
│ int-test-registry/int-test-device-payload-4 │ 224511094092737 │
   deviceId=int-test-device           │
   QV5AEkwmCERJUytDCypYEU4EISE-MD5FU0RQBhYsXUZIUTcZCGhRDk9eIz81IChFFgtTE1FcdgJZEGk
   zXHUHUQ0YdHtkdGhTEgRWQFl-VVsJPGh-Y3cFVQgYc31ocWpTEQYBQnv-88qz79pfZhg9XBJLLD5-LT
   1F │
│                                             │                 │
   deviceNumId=2813777014668441       │
    │
│                                             │                 │
   deviceRegistryId=int-test-registry │
    │
│                                             │                 │
   deviceRegistryLocation=us-central1 │
    │
│                                             │                 │
   projectId=neodada-integration      │
    │
│                                             │                 │ subFolder=
    │
    │
. . .
```

Appendix: Commands only

sudo raspi-config

```
sudo -s

apt update && apt upgrade && apt dist-upgrade

rpi-update && reboot

apt install build-essential libssl-dev libffi-dev python-dev

pip install pyjwt paho-mqtt cryptography

pip install --upgrade google-api-python-client

pip install --upgrade google-cloud-core

pip install --upgrade google-cloud-pubsub

exit

tar -zxvf google-cloud-sdk-219.0.1-linux-x86.tar.gz

python --version

./google-cloud-sdk/install.sh

./google-cloud-sdk/bin/gcloud init

gcloud projects create <project-id> --name=<project-name>

gcloud projects add-iam-policy-binding <project-id>
    --member=serviceAccount:cloud-iot@system.gserviceaccount.com--role='roles/pubsub.publisher'

gcloud services enable cloudiot.googleapis.com

gcloud services enable pubsub.googleapis.com

gcloud pubsub topics create <topic-name>

gcloud pubsub subscriptions create <subscription-name> --topic=<topic-name>

gcloud iot registries create <registry-name> --region=us-central1 --project=<project-id>
    --event-notification-config=topic=<topic-name>
```

```
openssl req -x509 -nodes -newkey rsa:2048 -keyout my_private.pem -days 365 -out my_pub_cert.pem -subj
    "/CN=unused"

gcloud iot devices create <device-name> --project=<project-id> --region=us-central1
    --registry=<registry-name> --public-key path=my_pub_cert.pem,type=rs256

python-docs-samples/iot/api-client/mqtt_example/cloudiot_mqtt_example.py

wget https://pki.google.com/roots.pem

python cloudiot_mqtt_example.py --project_id=<project-id> --registry_id=<registry-name>
    --cloud_region=us-central1 --device_id=<device-name> --private_key_file=my_private.pem
    --algorithm=RS256 --message_type=event

gcloud pubsub subscriptions pull <subscription-name>
```