Delta Controls IoT

Google Cloud Platform – MQTT Bridge

Application Guide

Document version: 0.4

Contents

1 Google IoT

Documentation for the use of the python module that provides connectivity to Google IoT Core using the Google connection technique of a device connecting directly to GCP IoT Core.

1.1 Overview

This module is an application of IoT Connectivity using the MQTT library within the Version 4.13 (and later) firmware. The concept that it is designed to fulfil is described by Google in their how-to guides for Google Cloud (Configuring devices and getting state | Cloud IoT Core | Google Cloud)

This method presumes that a device establishes a secure connection to the Google MQTT Bridge using TLS and a JSON Web Token created with a Private/Public encryption key pair using openssl. This is implemented in V4.13 and later firmware, so that the device creates the key pair from a pre-loaded root certificate and only exposes the public key to allow connection to GCP IoT Core.

The connection to IoT Core requires a Google Cloud Platform Project, with a registry (in the appropriate cloud region) and then a device which the module will connect to. The only setup on the physical device is the connection to the IoT Core, which is defined in any CSV Object that must be named exactly (including case capitalisation) as:

1.1.1 UDMI Configuration

This CSV object must contain a JSON block that defines the key: value pairs as shown in the table and example below

Key	Description	Default	Optional / Required	Example
hostname	Name of IP host		Required	"mqtt.googleapis.com"
tcpPort	TCP port at host		Required	8883
project	GCP Project		Required	"bacnet-gateway"
location	Cloud region of GCP Project registry		Required	"europe-west1"
auth-type	Authentication method to use for MQTT connection	jwt	Optional	"jwt" / "none" / "simple"

	"jwt" = GCP "none" = MQTT3.1.1 with no authentication "simple" = MQTT3.1.1 with simple username / password authentication			
device	Device name of the Device within the GCP Project registry		Required	"FCU-101"
publish-topic	Pointset telemetry topic name for publish		Optional	"/devices/{{}}/events/pointset"
state-topic	State telemetry topic name for public		Optional	"/devices/{{}}/state"
debug	Diagnostic information level for debug info in the CSV.Description property 0 = Debug Off 1 = Minimal Info 20 = Normal Info 20 = Detailed Info 4 = Full Diagnostic Info	0	Optional	0-4
maxupdatems	Maximum update rate between MQTT requests (in milliseconds)	1000	Optional	1000 = 1 second
poll-interval	Polling interval between publish of proxy device telemetry data (in milliseconds)	900000	Optional	900000 = 900 seconds = 15 minutes
rpm	Use BACnet Read Property Multiple to collect proxy device telemetry data	1	Optional	1 = Use RPM 0 = Do not use RPM
cloud-write	Allow Cloud-to-Device commands (BACnet Write)	0	Optional	1 = Allow C2D 0 – Do not allow C2D
write-priority	BACnet priority level for Cloud write back	9	Optional	1-16
relinquish-always	When off (0) only Cloud-to-Device write backs to individual points that are subsequently cleared will be relinquished. When on all points are relinquished regardless of any prior write backs	0	Optional	0 = Relinquish all 1 = Relinquish changed
jwt_exp_mins	Interval between required refresh of JSON Web Tokens for connection (minutes)	20	Optional	20 = 20 minutes
cov	Use Change of Value	0	Optional	1 = Use COV (and poll) 0 = Poll only
cov-command	UDMI syntax for COV Increment	cov_increment	Optional	"cov_increment"

Example of minimally specified configuration file

```
{
  "hostName": "mqtt.googleapis.com",
  "tcpPort": 8883,
  "location": "us-central1",
  "project": "bacnet-gateway",
  "registry": "registrar-us",
  "device": "FCU-101",
}
```

Example of fully specified configuration file

```
{
  "hostName": "mqtt.googleapis.com",
  "tcpPort": 8883,
  "location": "us-central1",
  "project": "bacnet-gateway",
  "registry": "registrar-us",
  "auth-type": "jwt",
  "device": "FCU-101",
  "publish-topic": "/devices/{{}}/events/pointset",
  "state-topic": "/devices/{{}}/state",
  "debug": 4,
  "maxupdatems": 2000,
  "poll-interval": 60000,
  "rpm": 0,
  "cloud-write": 1,
  "write-priority": 7,
  "relinquish-always": 1,
  "jwt_exp_mins": 20,
  "cov": 0,
  "cov-command": "cov_increment"
}
```

1.2 Security

In order to connect to the Google IOT Core three pieces are needed:

Device Private Key: self-generated on the device. The Python module will generate this on the device if the file does not exist. This file is not exposed externally to maintain security.

Device Public Key: self-generated on the device. The Python module will generate this on the device when it discovers any FIL object named "**UDMI Public Key**" (exactly). This can be downloaded from the device from the FIL dialogue in enteliWEB and uploaded to the RSA key of the Gateway device in the GCP Project Registry. It is also copied into the FIL.Description property for copy/paste.

Google Root Certificate: A new Google Root certificate can be uploaded to any FIL object created on the device and exactly named "**UDMI Roots**". Loading a new root certificate will cause the Python module to recreate the private / public key pair.

2 Usage

Once the connection to GCP IoT Core has been established then IoT Core will publish to a topic named "**devices/[device]/config**", the Python module will have already subscribed to this and will be expecting to receive the payload defined in the UDMI Schema (UDMI Device Config JSON Block). This will define the points data for telemetry in the ['pointset']

The device will publish the status for the device to a topic named "**devices/[device]/status**" in accordance with the UDMI Schema Proxy Status.

Once attached and then at the relevant poll interval the Python module will collect the BACnet data into one JSON pointset payload defined in the UDMI Schema (UDMI Device Event Pointset Telemetry JSON Block) and publish this to the topic "**devices/[device]/events/pointset**"

2.1 UDMI Status Telemetry

The JSON payload that is published to the "**devices/[device]/status**" defaults to:

```
{
  "version": 1,
  "timestamp": "<TIME ISO-8601 UTC>",
  "system": {
    "make_model": "<BACNET DEV.Model_Name>",
    "firmware": {
      "version": "<BACNET DEV.Application_Software_Version>"
    },
    "serial_no": "N/A",
    "last_config": "",
    "operational": "<BACNET DEV.System_Status>"
  },
  "pointset": {
    "points": {}
  }
}
```

This can be modified using a template that resides in a CSV Object that must be named exactly (including case capitalisation) as:

2.1.1 UDMI State Telemetry

This CSV object must contain a JSON block that defines the key: value pairs that depict a frame for the telemetry. The device includes a pre-processor that will parse the template prior to publishing and replace any references to <BACnet > properties or <Time> values.