Application note: Google Cloud IoT Core API V1.0

Table of contents

Introduction

This application note describes how to use the functions in the `gci.inc` include file to communicate with the Google Cloud IoT Core.

Cloud IoT Core

A fully managed service to easily and securely connect, manage, and ingest data from globally dispersed devices. [1]

The `gci.inc` include file provides functions to monitor and manage the device, using the following features:

Device-to-cloud messages

Device-to-cloud messages can be sent using the `gciPublish` function, and can be used to send a payload to a number of endpoints on the Cloud IoT core for further processing.

Device configuration

The configuration is sent from the Cloud IoT core and can be used to configure the device.

The configuration can be received as a message from `gciReceiveMessage`.

Device state

The device state can be sent to the Cloud IoT core to report the current device configuration, using `gciSetState`.

Commands

Commands are a way of invoking code on the device from the Cloud IoT Core. This can e.g. be used to request the device to reset or to perform a specific task. When a command is invoked, the command request message is received and `gciGetMethodRequest` can be called to get the name of the command.

1 https://cloud.google.com/iot-core

Setup

It is necessary to install the client certificate containing the private key as well as any additional root certificates on the device before it is possible to connect to the Cloud IoT core service.

One way to get the needed root certificates is to install https://pki.goog/roots.pem which contains all the certificates needed for Google products and services[2].

The variables in the top of the VAR section must be set to the correct values matching the configuration of the Cloud IoT core:

project	The project ID
registry	The registry ID
region	The region, e.g. europe-west1
deviceId	The ID of the device
cert	The name of private key to use for communicating with the IoT Core. Must match the public key reported to IoT core.
cert_pass	The optional password for the private key.
tz_offset	The time offset in seconds between the device time and UTC, e.g 7200 seconds for UTC+2 and 0 for UTC.
jwt_lifetime	The lifetime of the authentication tokens in seconds. Must be less than 24 hours. When this time has elapsed, the token expires and a new token will be created. A smaller value will increase the network traffic but will decrease the window where a stolen token can be misused, increasing the security.

Application structure

This section describes how the VPL application for communicating with the Cloud IoT core could be structured. The `gci.inc` include file uses the MQTT functions for communicating with the Cloud IoT Core, so it can not be used at the same time.

2 https://pki.goog/faq.html "What roots should we trust for connecting to Google?"

The authentication uses JSON Web Tokens(JWT) which requires that the application is built with Large String Support.

The device must have been created on the Cloud IoT Core and the private key for the device must be installed on the device.

A connection to a network that can access the Cloud IoT Core must be established and the connection can then be created using `gciOpenJWT`. `gciOpenJWT` uses the internal function `_cloudGenJWT` to create a new token to use as the password and connects to the Cloud IoT Core.

When `gciOpenJWT` returns successfully, it is then necessary to wait for the connection to be established to the Cloud IoT Core, using `gciConnected` and `gciStatus` to monitor the connection.

The main loop for handling the communication must call `gciWaitEvent` to check for new messages and to keep track of the connection.
If a messages has been received, the `gciReceiveMessage` function block must be called to read the message and based on the type of message, different functions can be called to held handle the message.
When `gciWaitEvent` detects that the connection is re-established after it has been lost, it automatically subscribes to the topics again, in case the subscriptions have been lost on the server.
`gciWaitEvent` also detects when the token expires and generates a new token using `_cloudGenJWT.`

Functions

The following functions are used to communicate with the Cloud IoT Core and can be found in the `gci.inc` include file.

gciOpenJWT()

Open the connection to the Cloud IoT Core, using JSON Web tokens for authentication.

Input:

• project_id	- The project identifier.
• region_id	- The region identifier. This is the region where the server is located.
• registry_id	- The id of the registry.
• device_id	- The device identifier.
• server	- The server to connect to. Should normally be left at "mqtt.googleapis.com".
• port	- The port to connect to. Should normally be left at 8883.
• iface	- The network interface to use.
• cert	- The name of the certificate containing the private key.
• cert_pass	- The password for the private key, if needed.
• lifetime	- The number of seconds the token is valid for before it expires. Max 24 hours.
• tz_offset	- Difference between the device time and UTC, used to convert clockNow() to UTC.

Returns:

• 0	- Success.
• -1	- No available connections.
• -2	- Invalid parameter
• -3	- Connection is already in use.
• -4	- Failed to create token.

gciClose()

Close the connection to the Cloud IoT Core

Input:

- None.

Returns:

- None.

gciSubscribe()

Subscribe to the events on the connected Cloud IoT Core.

Input:

- None.

Returns:

0	- Success.
1	- Invalid connection.
2	- Not connected to server.
3	- Invalid parameter

gciConnected()

Tells if the connection to the Cloud IoT Core is established.

Input:

- None.

Returns:

TRUE	- Device is connected to Cloud IoT Core.
FALSE	- Device is not connected to Cloud IoT Core.

gciStatus()

Returns the status of the connection.

Input:

- None.

Returns:

- 0 - Connected to the Cloud IoT Core.
- 1 - Invalid connection.
- 2 - Not connected to network.
- 3 - Cloud IoT Core server not found.
- 4 - No reply from Cloud IoT Core
- 5 - Connection rejected, unacceptable protocol version.
- 6 - Connection rejected, client ID rejected.
- 7 - Connection rejected, server unavailable.
- 8 - Connection rejected, bad user-name or password.
- 9 - Connection rejected, not authorized.
- 20 - Secure connection failed.
- 21 - Secure connection require client certificate.
- 22 - Certificate verifications failed.
- 23 - Client certificate is incomplete.

gciWaitEvent()

Waits for a new message, blocking the thread.

Input:

- timeout - The number if seconds to wait before timing out. Use 0 to return immediately, use -1 to wait forever.

Returns:

- 1 - A messages is received. Use `gciReceiveMessage` to read it.
- 0 - Timeout.
- -1 - No connection.

gciReceiveMessage()

Function block that receives messages from the Cloud IoT Core.

Input:

- data - Address of a buffer to store the data of the message in.
- maxsize - The size of the buffer. Any message data after this limit is lost.

Output:

- ready - True if a message has been received.
- size - The size of the read data.
- type - The type of the message:

-1	- Unknown message.
2	- Command request. Use `gciGetMethodRequest` to read the name of the requested command. The data contains the parameters for the command.
6	- Configuration received. The data contains the configuration data.

gciPublish()

Sends a message to the connected Cloud IoT Core.

Input:

• subfolder	- The optional subfolder to send the message to.
• data	- Address of a buffer with the data to send.
• size	- Size of the data to send.

Returns:

0	- Success.
1	- Invalid connection.
2	- Not connected to server.
3	- Invalid parameter
4	- A publish is already in progress
5	- Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.

gciGetMethodRequest()

Retrieves the name of the requested command.

Output:

• method	- The name of the called command.

Returns:

• 0	- Success.
• -1	- No unread method request available.

gciSetState()

Sends the device state to the connected Cloud IoT Core.

Input:

- data - Address of a buffer with the state to send.
- size - Size of the state to send.

Returns:

0	- Success.
1	- Invalid connection.
2	- Not connected to server.
3	- Invalid parameter
4	- A publish is already in progress
5	- Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.