

CC32xx MQTT Server

Overview

[Return to CC31xx & CC32xx Home Page](#)

MQTT(Message Queue Telemetry Transport) protocol is an extremely light weight machine to machine connectivity protocol. It is based on publish/subscribe messaging model and is designed to be used on the top of TCP/IP protocol. A small footprint implementation, low bandwidth requirement and ease of scalability makes it a popular choice for data transport for embedded systems in the realm of Internet-of-Things (IoT).

MQTT uses a client-server topology for data transactions. Multiple clients can communicate with a single server. More information about MQTT protocol can be obtained from the latest MQTT Protocol specification. Subsequent sections outline the support for tiny MQTT server that have been made available as part of the CC32xx SDK.

Application details

MQTT Network

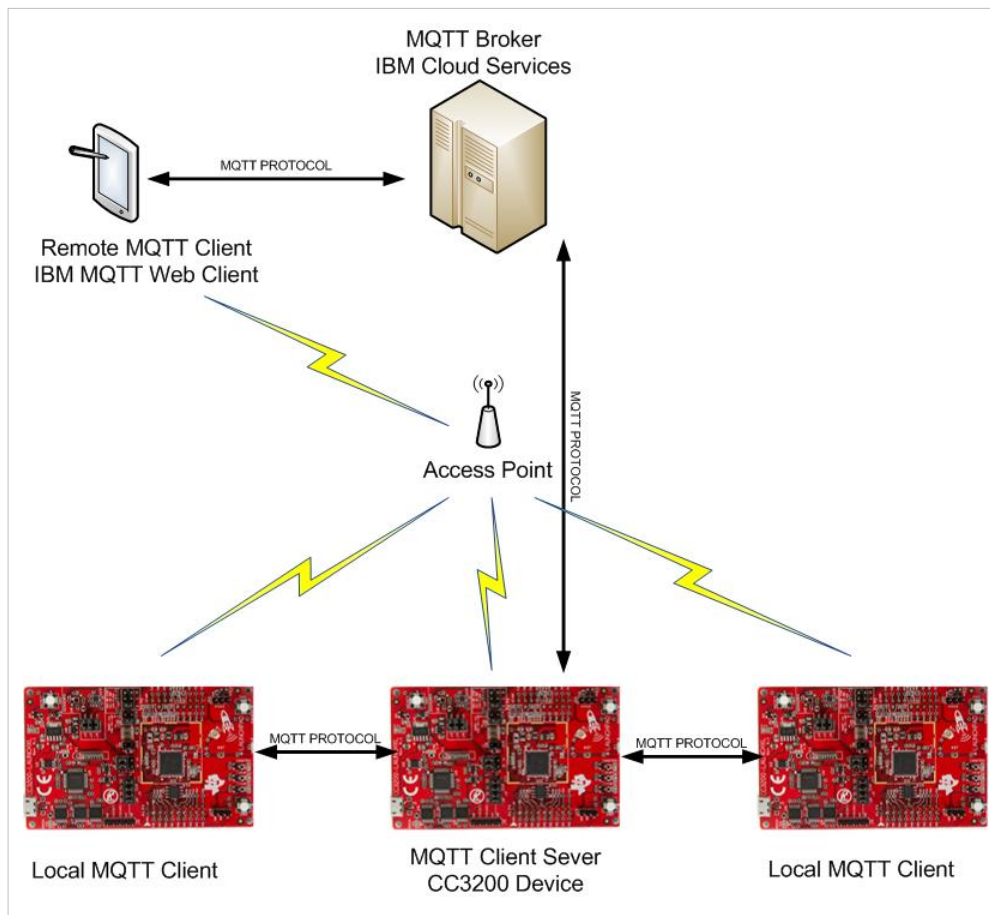
A simple MQTT network contains a server / broker (like a central hub) that can handle connections from multiple clients. Each of the connected client can publish data for any topic (token). Whereas, the server / broker is responsible for forwarding the data published for a topic by a client to all the other clients who have subscribed for that particular topic. This is a very simplistic description of a MQTT network to set the tone for the sample application provided in SDK.

MQTT Library

MQTT library abstracts out the underlying intricacies of MQTT network and provide you with an intuitive and easy to use APIs to implement the MQTT protocol on CC3200 device. The server library makes provisions to manage multiple clients and topics.

Application

This application implements a MQTT bridge that utilizes the services of the MQTT server and MQTT client running in parallel (on the same launchpad). The MQTT server manages the local MQTT network whereas, the MQTT client connects to a cloud server / broker. The bridge application enables data exchange between local-server and the cloud server. To receive data from the cloud server, the MQTT bridge, through the MQTT client, subscribes to certain topics. It also forwards the data / topics received from the local MQTT server to the cloud-server. The local MQTT server handles the connections from the multiple MQTT clients (also shown as Launchpads in the following figure, but can be running on another platform) in the local network.



Use Case

The motivation behind this application is to showcase a working setup where Two local MQTT clients can communicate with each other as well as talk to a remote client via an external broker. For the sake of simplicity following abbreviations are used:

1. LC1 - First local MQTT client
2. LC2 - Second local MQTT client
3. RC - Remote MQTT client
4. AS(Application Server) - MQTT server running on CC32xx device
5. AC(Application Client) - MQTT client running on CC32xx device
6. BR - External broker/ cloud server
7. AP - Access Point

The use-case consists of two Local MQTT client LC1 and LC2, which could be running on a PC or Smartphone or even can be CC32xx clients(as shown in the above figure), communicating with each other using the MQTT server(AS). These Local clients can also communicate with a Remote MQTT client(RC) through an external broker(BR). Both these cases are explained below. Please refer to the following diagrams for more clarification.

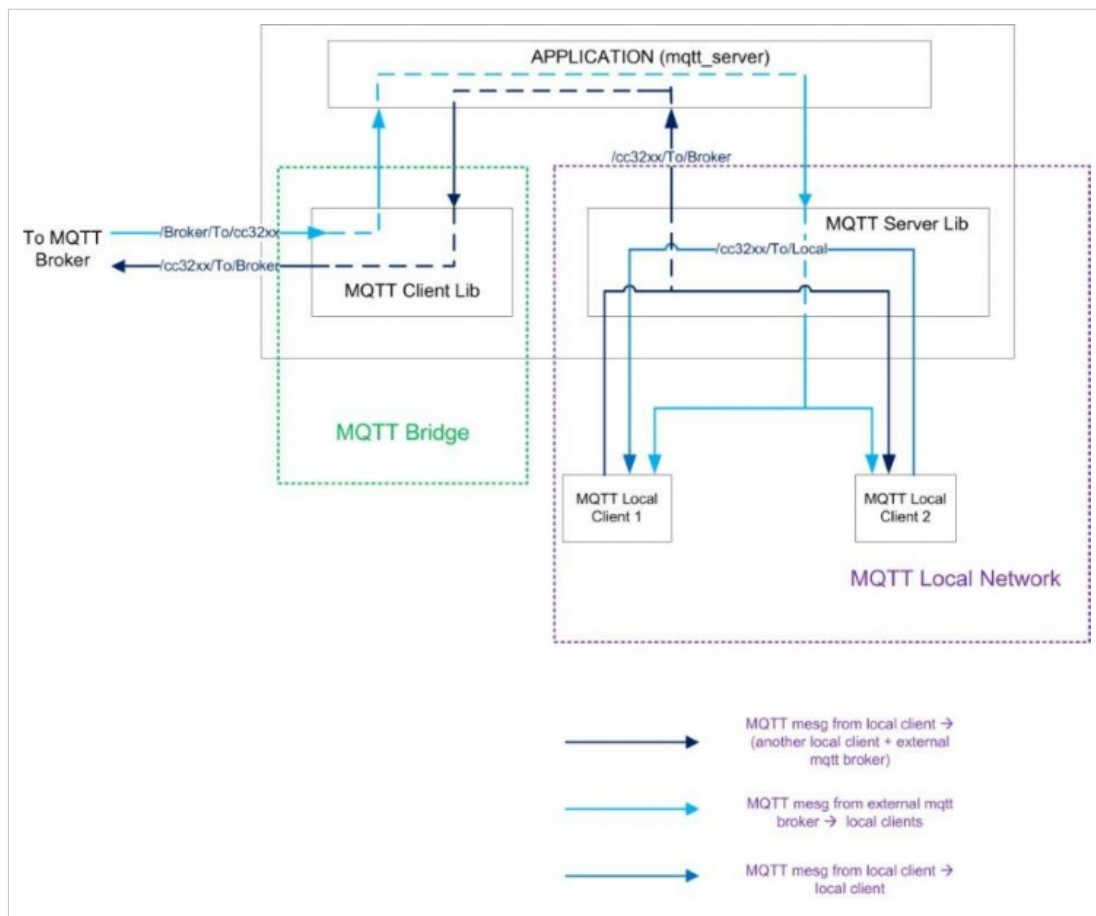
Local Communication

1. LC1 connects to AS
2. LC2 connects to AS
3. LC1 subscribes on a particular topic.
4. LC2 publish message on the same topic.
5. LC1 should get the message published by LC2. This will conclude the Local MQTT Network.

Remote Communication

1. RC and AC should be connected to the same external broker(BR)
2. AC has already subscribed to the topic SUB_TOPIC
3. RC will publish a message on SUB_TOPIC
4. Whatever message is received by AC will be passed to the AS.
5. If any of the local client has subscribed to the SUB_TOPIC, it will receive the message published by RC.
6. AC has also enrolled for the topic ENROLLED_TOPIC, which means if AS receives any message on ENROLLED_TOPIC from any local client, it will be passed to the AC. Which in turn will published to the BR.
7. If RC has subscribed on ENROLLED_TOPIC, it will receive the message published by the Local Client.

MQTT Application block diagram



Configurations

Following highlights most of the parameters that the user will want to configure

```
/*Operate Lib in MQTT 3.1 mode.*/
#define MQTT_3_1_1 false /*MQTT 3.1.1 */
#define MQTT_3_1 true /*MQTT 3.1*/

#define WILL_TOPIC "Client"
#define WILL_MSG "Client Stopped"
#define WILL_QOS QOS2
#define WILL_RETAIN false
```

```

/*Defining Broker IP address and port Number*/

#define SERVER_ADDRESS                "messagesight.demos.ibm.com"

#define PORT_NUMBER                    1883

#define LOOPBACK_PORT                 1882

/*Specifying Receive time out for the Receive task*/
#define RCV_TIMEOUT                    30

/*Background receive task priority*/
#define TASK_PRIORITY                  3

/* Keep Alive Timer value*/
#define KEEP_ALIVE_TIMER               25

/*Clean session flag*/
#define CLEAN_SESSION                  true

/*Retain Flag. Used in publish message. */
#define RETAIN                         1

/*Defining Publish Topic*/
#define PUB_TOPIC                      "/cc32xx/ButtonPressEvtSw2"

/*Defining Number of topics*/
#define SUB_TOPIC_COUNT                1

/*Defining Subscription Topic Values*/
#define SUB_TOPIC                      "/Broker/To/cc32xx"

/*Defining Enrolled Topic Values*/
#define ENROLLED_TOPIC                 "/cc32xx/To/Broker"

```

Apart from the MACROs shown above, the following structure can be configured as per requirement and use case.

```

/*Initialization structure to be used with sl_ExtMqtt_Init API*/
SlMqttClientCtxCfg_t Mqtt_ClientCtx ={
{
    SL_MQTT_NETCONN_URL,
    SERVER_ADDRESS,
    PORT_NUMBER,
    0,
    0,
    0,
    NULL
}

```

```

    },
    MQTT_3_1_1,
    true
};

SlMqttClientLibCfg_t Mqtt_ClientCfg ={
    0,
    TASK_PRIORITY,
    RCV_TIMEOUT,
    true,
    UART_PRINT
};

SlMqttServerCfg_t Mqtt_Server={
    {
        0,
        NULL,
        PORT_NUMBER,
        0,
        0,
        0,
        NULL
    },
    LOOPBACK_PORT,
    TASK_PRIORITY,
    KEEP_ALIVE_TIMER,
    true,
    UART_PRINT
};

SlMqttWill_t will_param={
    WILL_TOPIC,
    WILL_MSG,
    WILL_QOS,
    WILL_RETAIN
};

```

Source Files briefly explained

- **main.c** - The main file implementing the mqtt client server bridge.

Supporting Files

- **pinmux.c** - Generated by the PinMUX utility.

Common Files

- **button_if.c** - Interface file to handle button click events
 - **gpio_if.c** - Basic GPIO interface APIs. Used to control the LEDs.
 - **network_if.c** - Common functions to handle connection to AP and FreeRTOS hook functions.
 - **startup_ccs.c** - CCS related functions
-

- **startup_ewarm.c** - IAR related functions
- **timer_if.c** - Wrapper functions for timer module driver
- **uart_if.c** - To display status information over the UART

Usage

1. Setup a serial communication application (HyperTerminal/TeraTerm). For detail info visit CC31xx & CC32xx Terminal Setting
On the host PC, open a hyperterminal, with the following settings
 - **Port:** Enumerated COM port
 - **Baud rate:** 115200
 - **Data:** 8 bit
 - **Parity:** None
 - **Stop:** 1 bit
 - **Flow control:** None
 2. Run the reference application (Flashing the bin/IAR/CCS).
 3. Application requires the AP to have the internet connectivity.
 4. On the Hyperterminal, in case the connection to this default AP is unsuccessful, the user is prompted to enter the AP details.
 5. The GREEN LED denotes that the network processor has come up successfully, RED LED continuously blinks as long as a connection with AP is not established. Once established, the RED LED stays continuously ON for one second and then all the LEDs will be turned off.
 6. From another machine, open a web browser and type <http://m2m.demos.ibm.com/mqttclient/> in the url box.
You will reach the web client from IBM.
 7. Connect this web client to IBM broker by filling in the required fields(use below screenshot as reference for the field values).
 8. Connect one ore multiple local MQTT client to the MQTT server, it can be a PC or another CC3200 device. A local functional MQTT network can now be used to communicate between multiple local clients.
 9. At the same time, if a local client publishes on a topic enrolled by the on-board client, it will be published to the external broker and made available for remote clients.
 10. Any message from the broker to the on-board client will be passed on to the MQTT server, which will take responsibility of distributing the message to the local client(if subscribed to the same topic).
-

Web Client Screenshot

Connection

MQTT Client

Connected!

Connect

Server

messagesight.demos.ibm.com

Port

1883

Connect

Disconnect

Client ID

Client80248

Username

(optional)

Password

(optional)

Clean Session

OFF

ON

SSL

OFF

ON

Subscribe

Publish

Subscribe and Publish

MQTT Client

Connected!

Connect

Subscribe

Topic

/cc32xx/To/Broker

QOS

1

Subscribe

/cc32xx/To/Broker

/cc32xx/ButtonPressEvtSw2

Publish

Topic

/Broker/To/cc32xx

Message

message from the external client to the local server

QOS

1

Retained

OFF

ON

Publish

Log 4

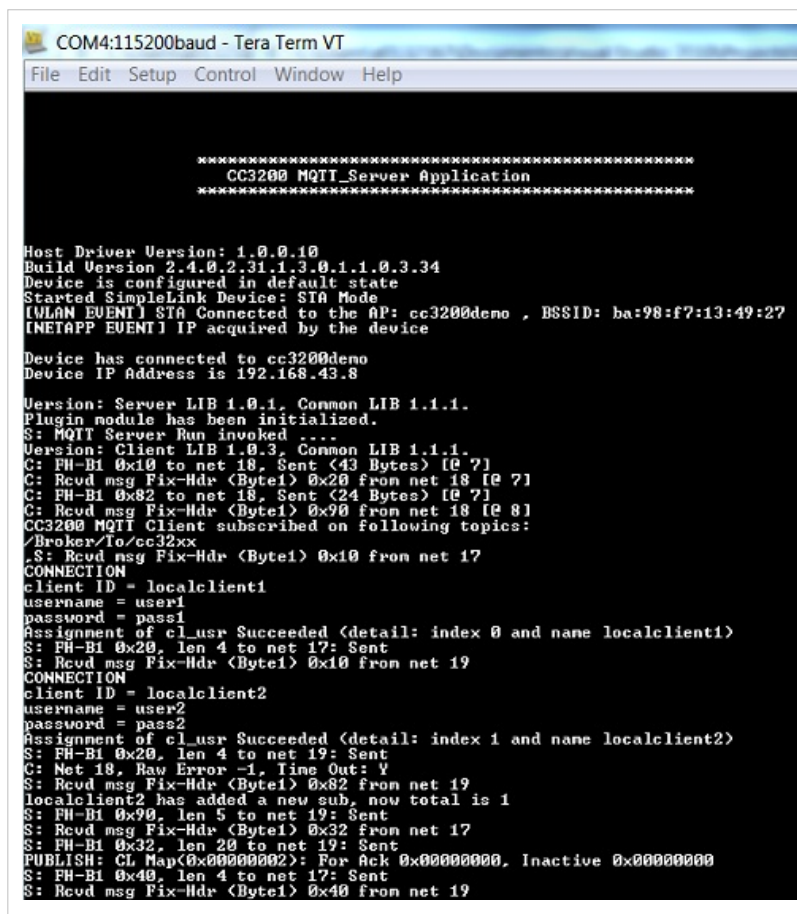
Clear

☐ Follow

(14:01:55.834) Connected to messagesight.demos.ibm.com:1883
(14:02:32.418) Subscribed to [/cc32xx/To/Broker][qos 1]
(14:03:25.841) Subscribed to [/cc32xx/ButtonPressEvtSw2][qos 1]
(14:03:46.185) << [/Broker/To/cc32xx][qos 1] message from the external client to the local server

Tera Term Screenshot

Local Communication



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

*****
CC3200 MQTT Server Application
*****

Host Driver Version: 1.0.0.10
Build Version 2.4.0.2.31.1.3.0.1.1.0.3.34
Device is configured in default state
Started SimpleLink Device: STA Mode
[VLAM EVENT] STA Connected to the AP: cc3200demo , BSSID: ba:98:f7:13:49:27
[NETAPP EVENT] IP acquired by the device

Device has connected to cc3200demo
Device IP Address is 192.168.43.8

Version: Server LIB 1.0.1, Common LIB 1.1.1.
Plugin module has been initialized.
S: MQTT Server Run invoked ----
Version: Client LIB 1.0.3, Common LIB 1.1.1.
C: FH-B1 0x10 to net 18, Sent (43 Bytes) [0 7]
C: Rcvd msg Fix-Hdr (Byte1) 0x20 from net 18 [0 7]
C: FH-B1 0x82 to net 18, Sent (24 Bytes) [0 7]
C: Rcvd msg Fix-Hdr (Byte1) 0x90 from net 18 [0 8]
CC3200 MQTT Client subscribed on following topics:
/Broker/To/cc32xx
.S: Rcvd msg Fix-Hdr (Byte1) 0x10 from net 17
CONNECTION
client ID = localclient1
username = user1
password = pass1
Assignment of cl_usr Succeeded (detail: index 0 and name localclient1)
S: FH-B1 0x20, len 4 to net 17: Sent
S: Rcvd msg Fix-Hdr (Byte1) 0x10 from net 19
CONNECTION
client ID = localclient2
username = user2
password = pass2
Assignment of cl_usr Succeeded (detail: index 1 and name localclient2)
S: FH-B1 0x20, len 4 to net 19: Sent
C: Net 18, Raw Error -1, Time Out: Y
S: Rcvd msg Fix-Hdr (Byte1) 0x82 from net 19
localclient2 has added a new sub, now total is 1
S: FH-B1 0x90, len 5 to net 19: Sent
S: Rcvd msg Fix-Hdr (Byte1) 0x32 from net 17
S: FH-B1 0x32, len 20 to net 19: Sent
PUBLISH: CL Map(0x00000002): For Ack 0x00000000, Inactive 0x00000000
S: FH-B1 0x40, len 4 to net 17: Sent
S: Rcvd msg Fix-Hdr (Byte1) 0x40 from net 19
```

Remote Communication

```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

=====
CC3200 MQTT_Server Application
=====

Host Driver Version: 1.0.0.10
Build Version 2.4.0.2.31.1.3.0.1.1.0.3.34
Device is configured in default state
Started SimpleLink Device: STA Mode
([MLAN EVENT]) STA Connected to the AP: cc3200demo , BSSID: ba:98:f7:13:49:27
([NETAPP EVENT]) IP acquired by the device

Device has connected to cc3200demo
Device IP Address is 192.168.43.8

Version: Server LIB 1.0.1, Common LIB 1.1.1.
Plugin module has been initialized.
S: MQTT Server Run invoked ....
Version: Client LIB 1.0.3, Common LIB 1.1.1.
C: FH-B1 0x10 to net 18, Sent (43 Bytes) [0 6]
C: Rcvd msg Fix-Hdr (Byte1) 0x20 from net 18 [0 8]
C: FH-B1 0x82 to net 18, Sent (24 Bytes) [0 8]
C: Rcvd msg Fix-Hdr (Byte1) 0x90 from net 18 [0 9]
CC3200 MQTT Client subscribed on following topics:
/Broker/To/cc32xx
C: Net 18, Raw Error -1, Time Out: Y
S: Rcvd msg Fix-Hdr (Byte1) 0x10 from net 17
CONNECTION
client ID = localclient1
username = user1
password = pass1
Assignment of cl_usr Succeeded (detail: index 0 and name localclient1)
S: FH-B1 0x20, len 4 to net 17: Sent
S: Rcvd msg Fix-Hdr (Byte1) 0x82 from net 17
localclient1 has added a new sub, now total is 1
S: FH-B1 0x90, len 5 to net 17: Sent
C: Rcvd msg Fix-Hdr (Byte1) 0x32 from net 18 [0 69]
C: FH-B1 0x40 to net 18, Sent (4 Bytes) [0 69]

Msg Recvd. by client
TOPIC: /Broker/To/cc32xx
PAYLOAD: message from the external client to the local server
QOS: 1
S: FH-B1 0x32, len 75 to net 17: Sent
PUBLISH: CL Map(0x00000001): For Ack 0x00000000, Inactive 0x00000000
S: Rcvd msg Fix-Hdr (Byte1) 0x40 from net 17
S: Rcvd msg Fix-Hdr (Byte1) 0x32 from net 17

Msg Recvd. by server
TOPIC: /cc32xx/To/Broker
PAYLOAD: message
QOS: 1
S: FH-B1 0x40, len 4 to net 17: Sent
C: FH-B1 0x32 to net 18, Sent (30 Bytes) [0 85]
C: Rcvd msg Fix-Hdr (Byte1) 0x40 from net 18 [0 85]
```

Suggestion

Note: Client name is unique for every client. It is recommended to change the client ID for CC3200 device to avoid connection issues.

Additional Info

Note: If anyone wishes to have prints(on serial comm port) about the packet level details of MQTT packets, just define `DEBUG_NET_DEV` as the predefined symbol while building the mqtt library.

Limitations/Known Issues

None.

Article Sources and Contributors

CC32xx MQTT Server *Source:* <http://processors.wiki.ti.com/index.php?oldid=195083> *Contributors:* A0132173, Jitgupta

Image Sources, Licenses and Contributors

File:Cc31xx cc32xx return home.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Cc31xx_cc32xx_return_home.png *License:* unknown *Contributors:* A0221015

File:Mqtt client server application model.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_client_server_application_model.jpg *License:* unknown *Contributors:* Jitgupta

File:Mqtt clietn server flow diagram.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_clietsn_server_flow_diagram.jpg *License:* unknown *Contributors:* Jitgupta

File:Mqtt client server connection.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_client_server_connection.jpg *License:* unknown *Contributors:* Jitgupta

File:Mqtt client server pub sub.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_client_server_pub_sub.jpg *License:* unknown *Contributors:* Jitgupta

File:Mqtt server local communication.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_server_local_communication.jpg *License:* unknown *Contributors:* Jitgupta

File:Mqtt server remote communication.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Mqtt_server_remote_communication.jpg *License:* unknown *Contributors:* Jitgupta