Hi3861 V100 / Hi3861L V100 MQTT

Development Guide

Issue	**01**
Date	**2020-04-30**

HiSilicon (Shanghai) Technologies Co., Ltd.

Address:	New R&D Center, 49 Wuhe Road, Bantian, Longgang District, Shenzhen 518129 P. R. China
Website:	https://www.hisilicon.com/en/
Email:	support@hisilicon.com

About This Document

Purpose

This document describes the Message Queuing Telemetry Transport (MQTT) development using examples.

MQTT is implemented based on the open-source component paho.mqtt.c-1.3.0. For details, see **https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/index.html**.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3861	V100
Hi3861L	V100

Intended Audience

The document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The following table describes the symbols that may be found in this document.

Symbol	Description
⚠ DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.

Symbol	Description
⚠ WARNING	Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
📖 NOTE	Supplements the important information in the main text. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Change History

Issue	Date	Change Description
01	2020-04-30	This issue is the first official release.
00B01	2020-01-15	This issue is the first draft release.

Contents

1 API Description

1.1 Data Structures

For details about the paho.mqtt.c-1.3.0 data structures, see **https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/annotated.html**.

1.2 APIs

For details about the paho.mqtt.c-1.3.0 APIs, see **https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals_func.html**.

1.3 Configuration

For details about the paho.mqtt.c-1.3.0 configuration, see **https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals_defs.html**.

2 Development Guidance

2.1 Development Procedure

Applications with the paho.mqtt.c-1.3.0 client library typically use a similar structure:

- Create a client object.
- Set the options to connect to an MQTT server.
- Set up callback functions if multi-threaded (asynchronous mode) operation is being used. For details, see **https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/async.html**.
- Subscribe to any topics the client needs to receive.
- Repeat until finished:
  - Publish any messages the client needs to.
  - Handle any incoming messages.
- Disconnect the client.
- Free any memory being used by the client.

See some simple examples:

- Synchronous publication example (**https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/pubsync.html**)
- Asynchronous publication example (**https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/pubasync.html**)
- Asynchronous subscription example (**https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/subasync.html**)

2.2 Sample Code for Subscription

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "MQTTClient.h"

#define ADDRESS     "tcp://192.168.43.101:1883"
#define CLIENTID    "ExampleClientSub"
#define TOPIC       "abc"
#define PAYLOAD     "Hello World!"
#define QOS         1
#define TIMEOUT     10000L
extern void hi_watchdog_enable(void);
extern void hi_watchdog_disable(void);


volatile MQTTClient_deliveryToken deliveredtoken;

void delivered(void *context, MQTTClient_deliveryToken dt)
{
    (void)context;
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    int i;
    char* payloadptr;
    (void)context;
    (void)topicLen;
    printf("Message arrived\n");
    printf("     topic: %s\n", topicName);
    printf("   message: ");

    payloadptr = message->payload;
    for(i=0; i<message->payloadlen; i++)
    {
        putchar(*payloadptr++);
    }
    putchar('\n');
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}

void connlost(void *context, char *cause)
{
    (void)context;
    printf("\nConnection lost\n");
    printf("     cause: %s\n", cause);
}

int mqtt_002(int argc, char* argv[])
{
    (void)argc;
    (void)argv;
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    int rc;
    int ch;

    MQTTClient_create(&client, ADDRESS, CLIENTID,
        MQTTCLIENT_PERSISTENCE_NONE, NULL);
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to connect, return code %d\n", rc);
        return rc;
    }
    printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
           "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);
    MQTTClient_subscribe(client, TOPIC, QOS);
    hi_watchdog_disable();
    do
    {
        ch = getchar();
    } while(ch!='Q' && ch != 'q');
    hi_watchdog_enable();
    MQTTClient_unsubscribe(client, TOPIC);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return rc;
}
```

2.3 Sample Code for Publication

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "MQTTClient.h"

#define ADDRESS     "tcp://192.168.43.101:1883"

#define CLIENTID    "ExampleClientPub"
#define TOPIC       "abc"
#define PAYLOAD     "Hello World!"
#define QOS         1
#define TIMEOUT     10000L

int mqtt_001(int argc, char **argv)
{
    printf("start mqtt sync publication test.\r\n");
    (void)argc;
    (void)argv;
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_deliveryToken token;
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID,
        MQTTCLIENT_PERSISTENCE_NONE, NULL);
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to connect, return code %d\n", rc);
        exit(EXIT_FAILURE);
    }
    pubmsg.payload = PAYLOAD;
    pubmsg.payloadlen = (int)strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for up to %d seconds for publication of %s\n"
            "on topic %s for client with ClientID: %s\n",
            (int)(TIMEOUT/1000), PAYLOAD, TOPIC, CLIENTID);
    rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
    printf("Message with delivery token %d delivered\n", token);
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return rc;
}
```

3 Precautions

3.1 Channel Encryption Unsupported

- In the paho.mqtt.c-1.3.0 source code, the encryption suite OPENSSL is used for encrypted communication. However, Hi3861 currently does not support OPENSSL. Therefore, only unencrypted MQTT communication is provided.