



**Hi3861V100 / Hi3861LV100 MQTT**

## **开发指南**

文档版本 03

发布日期 2020-07-24

版权所有 © 上海海思技术有限公司2020。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



**HISILICON**、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 上海海思技术有限公司

地址：            深圳市龙岗区坂田华为总部办公楼    邮编：518129

网址：            <https://www.hisilicon.com/cn/>

客户服务邮箱：  [support@hisilicon.com](mailto:support@hisilicon.com)



## 前言

### 概述

本文档主要介绍基于MQTT功能开发实现示例。

MQTT基于开源组件paho.mqtt.c-1.3.1实现，详细说明请参考官方说明：<https://www.eclipse.org/paho/files/mqtt/doc/MQTTClient/html/index.html>

### 产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3861	V100
Hi3861L	V100



### 读者对象

本文档主要适用于以下工程师：



- 技术支持工程师
- 软件开发工程师

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。



符号	说明
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

## 修改记录

文档版本	发布日期	修改说明
03	2020-07-24	<ul style="list-style-type: none"><li>在“<a href="#">1 API接口描述</a>”中删除paho.mqtt.c的版本号。</li><li>在“<a href="#">2.1 开发流程</a>”中删除paho.mqtt.c的版本号。</li><li>更新“<a href="#">3.1 支持加密通路</a>”的标题名称和内容。</li></ul>
02	2020-07-21	更新“ <a href="#">概述</a> ”中paho.mqtt.c的版本号为paho.mqtt.c-1.3.1。
01	2020-04-30	第一次正式版本发布。
00B01	2020-01-15	第一次临时版本发布。



## 目录

前言.....	i
1 API 接口描述.....	1
1.1 结构体说明.....	1
1.2 API 列表.....	1
1.3 配置说明.....	1
2 开发指南.....	2
2.1 开发流程.....	2
2.2 订阅示例代码.....	2
2.3 分发示例代码.....	4
3 注意事项.....	6
3.1 支持加密通路.....	6



# 1 API 接口描述

---

[1.1 结构体说明](#)

[1.2 API列表](#)

[1.3 配置说明](#)

## 1.1 结构体说明

paho.mqtt.c详细的结构体说明请参考官方说明文档：<https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/annotated.html>

## 1.2 API 列表

paho.mqtt.c详细的API说明请参考官方说明文档：[https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals\\_func.html](https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals_func.html)

## 1.3 配置说明

paho.mqtt.c详细配置说明请参考官方说明文档：[https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals\\_defs.html](https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/globals_defs.html)



# 2 开发指南

- 2.1 开发流程
- 2.2 订阅示例代码
- 2.3 分发示例代码

## 2.1 开发流程

使用paho.mqtt.c的应用程序通常使用类似的结构：

- 创建一个客户端对象
- 设置选项以连接到MQTT服务器
- 如果正在使用多线程（异步模式）操作，请设置回调函数（请参见官方说明“<https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/async.html>”）。
- 订阅客户需要接收的任何主题
- 重复直到完成：
  - 发布客户端需要的所有消息
  - 处理任何传入的消息
- 断开客户端
- 释放客户端正在使用的所有内存

具体实现可以参考官方说明中的示例：

- 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 订阅示例代码

```
#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 分发示例代码

```
#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 注意事项

---

## 3.1 支持加密通路

### 3.1 支持加密通路

- 如果需要实现MQTT加密传输，MQTT配置项中需要设置SSL参数。只做单端认证（客户端对服务端进行认证）时，需要提供认证服务端的根CA证书；做双端认证（客户端与服务端相互认证）时，除根CA证书外，还需要提供客户端证书与私钥。具体可参考/src/samples/lt\_mqtt\_005.c用例。