**Gateway API Introduction**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Change Description** | **Author** |
| 1.0 | 2018/5/8 | Initial draft | Hogen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Catalogue**

[1. Purpose 4](#_Toc523732766)

[2. Product Architecture 4](#_Toc523732767)

[3. Function Design 5](#_Toc523732768)

[3.1 Main function 5](#_Toc523732769)

[3.1.1 About MQTT Protocol 5](#_Toc523732770)

[3.1.2 Monitor Status about KES 6](#_Toc523732771)

[3.1.3 Update ESL Display Picture 6](#_Toc523732772)

[3.2 Function Call flow 7](#_Toc523732773)

[3.2.1 Alive message between Gateway and MQTT server 7](#_Toc523732774)

[3.2.2 ESL Status Report 7](#_Toc523732775)

[3.2.1 Update ESL picture success 8](#_Toc523732776)

[3.2.1 Update ESL picture failed 10](#_Toc523732777)

[3.2.2 Update password of ESL 11](#_Toc523732778)

[3.3 ESL ID definition rule 11](#_Toc523732779)

[3.4 Service Configuration 12](#_Toc523732780)

[3.4.1 Scanning paramaters 12](#_Toc523732781)

[4. Message format 12](#_Toc523732782)

[4.1 Alive message 12](#_Toc523732783)

[4.2 Advertisement packet report 13](#_Toc523732784)

[4.3 Update Picture 15](#_Toc523732785)

[4.3.1 Updating pictures in uncompressed mode 15](#_Toc523732786)

[4.3.2 Updating pictures in compressed mode 18](#_Toc523732787)

[4.3.3 Get parameters command 20](#_Toc523732788)

[4.4 Modify password 20](#_Toc523732789)

[4.5 Ack message 21](#_Toc523732790)

[4.6 Error Code for Ack message 21](#_Toc523732791)

[5. Gateway MQTT client demo 22](#_Toc523732792)

[6. MQTT Test Guidance 23](#_Toc523732793)

[6.1 Configuring Gateway MQTT Parameters 23](#_Toc523732794)

[6.2 Using third part MQTT client to verify Gateway 23](#_Toc523732795)

[6.2.1 Setup mqtt.fx 23](#_Toc523732796)

[6.2.2 Running mqtt.fx 24](#_Toc523732797)

[6.3 Mqtt message test 26](#_Toc523732798)

[6.3.1 Monitoring ESL advertisment Data 26](#_Toc523732799)

[6.3.2 Updating ESL paramaters 29](#_Toc523732800)

[6.4 Installing the MQTT Server 32](#_Toc523732801)

[6.4.1 Download software 32](#_Toc523732802)

[6.4.2 Install MQTT service 33](#_Toc523732803)

[6.4.3 Configuring User Name and IP Address 34](#_Toc523732804)

[6.4.4 Running MQTT service 35](#_Toc523732805)

[6.4.5 Verify MQTT service running 36](#_Toc523732806)

# Purpose

This article describes the API interface of the Gateway. It used for third-party software engineer to integrat the ESL and Gateway to their own system.

# Product Architecture

ESL

Gateway

ESL

Bluetooth4.0/5.0

MQTT broker

IP network

WiFI/Eth

Client manager

JSON

JSON

IP network

ESL

Gateway

ESL

Bluetooth4.0/5.0

WiFI/Eth

Interface:

* Communication between the ESL and the Gateway using Bluetooth 4.0/5.0. To ensure that the ESL and the Gateway are not illegally accessed, a two-way MD5 authentication protocol is required before the communication between the ESL and the Gateway.
* WiFI, Ethernet, and Bluetooth modules are available on the base station. The Gateway using MQTT protocol to communicate with MQTT broker/HTTPs server, and the protocol packets are JSON format.
* A multi-to-many redundant connection between the ESL and the Gateway is supported. When one Gateway is down, another Gateway can automatically assume the management function of ESL when the ESL under the Gateway.

**Role:**

1. ESL: After the ESL is powered on, it will periodically broadcast status packet to Gateway. The packet include the battery level, picture id, fault, temperture, version, etc.,

2. Gateway: After the Gateway is powered on, it will try to connect MQTT broker and start the scanning. Once the ESL broadcast packet is scanned, the Gateway will send the data to the MQTT server.

*When Gateway receiving the command request from the MQTT server, the Gateway will setup connection between ESL and Gateway, then Gateway will sends a corresponding control command or data to the ESL device.*

3, MQTT server: Using standard MQTT server architecture, mainly plays a role of data subscribe and publish.

4, Client manager: for user to monitor ESL/Gateway status or config the ESL.

# Function Design

## Main function

### About MQTT Protocol

The Gateway using MQTT protocol to communicate with MQTT broker. All communication messages are in JSON format, which makes it easy to use third-party tools for debugging and integration. Therefore, developers are advised to understand the MQTT protocol and the JSON protocol before development.

After the gateway is powered on, it will start the subscription and publish of three MQTT topics, as follows:

Publish Topic:

* + Status topic: The default topic name is ESL/publish/mac, which is used by the gateway to publish the ESL broadcast data, which is used for cloud server to monitor the ESL status;
  + pubaction topic: The default topic name is ESL/ pubaction /mac, which is used by the gateway to send command result to the MQTT client.

Subscribe topic:

* Subaction topic: The default topic name is ESL/subaction/mac, which is used to receive execution commands from MQTT client.

   The mac address of the gateway, and uses a 12-digit hexadecimal identifier, such as the A1A200F3A300, to uniquely identify a gateway.

### Monitor Status about KES

After the ESL is powered on, the gateway periodically monitors the status of the ESL. The periodicity of the following data is reported to the MQTT server, and the MQTT server reports the message to the client management module to monitor the status of the ESL.

* + ID of the ESL
  + The strength of the signal of the ESL (distance to the base station);
  + The picture ID currently displayed by the ESL;
  + The fault status of the ESL operation;
  + ESL battery level;
  + the temperature of the ESL environment;

Description:

The battery power is provided in the form of voltage, the unit is mV, for example, 3.3V corresponds to 3300mV, and the actual power can be converted according to the voltage;

The ambient temperature monitored by the tag sensor (with an error range of ±3 °C) provides a rough temperature monitoring;

### Update ESL Display Picture

After the Gateway is powered on, it will connect to the MQTT server and will listen to the message sent by the MQTT server. When the client management module needs to update the picture of the specified ESL, the information can be sent to the gateway through the MQTT server. The gateway then establishes a connection with the ESL, and sends the picture information to the ESL, and returns the execution result to the MQTT server.

## Function Call flow

### Alive message between Gateway and MQTT server

MQTT Server

Gateway

MQTT Client

Publish(pubaction topic)

(alive message)

Power on

Subscribe(pubactiontopic)

Publish(alive message)

Optional

Publish(pubaction topic)

(alive message)

After the Gateway is powered on, it periodically sends an Alive message to the MQTT server. The alive message includes the gateway ID (MAC address), the pubaction topic name, and the subaction topic name.

The sending period of this message is about 45~60 seconds and sent periodically.

Client management can subscribe to this message for monitor Gateway that is alive.

### ESL Status Report

Subscribe(status topic)

Optional

Beacon1

MQTT Server

Gateway

MQTT Client

Publish(status topic)

(status packet)

Status packet

Status packet

Publish(status topic)

(status packet)

Publish(status topic)

(status packet)

Beacon1

Beacon1

By subscribing to the Publish topic, the client management can get ESL status(status, version, battery, etc.,). Also the releastionship between the ESL and the Gateway can be established.

In addition, the status information of the ESL is reported by the multiple Gateway, and the signal strength information of the ESL is reported. Therefore, the client management software can establish a “ESL-Gateway” one-to-many relationship table. When the client management software sends the data to the ESL, it can select the Gateway with the strongest signal (the closest to the label) to send data. In addition, the advantage of this is that when a Gateway is down, the client management software can send data to the label through the Other Gateway.

The relationship between the ESL and the Gateway is self-organizing redundant, and no manual configuration is required. This greatly simplifies the configuration process. For example, when a gateway is down, the ESL automatically selects another gateway reporting status data within the coverage signal range.

### Update ESL picture success

KBeacon

MQTT Server

Gateway

MQTT Client

1.Publish: subaction

msg: dData{picture}

2.Publish:Pubaction

msg: picture, success1

5.MD5 request(random1)

13.Publish:Pubaction

msg: dAck, success: 0

4.BLE connection setup

11.{ picture }

12.update ack

3. Scanning device

6.MD5 response(auth1\_value, random2)

8.MD5 response(auth2\_value)

7. Device auth success

9. Gateway Auth success

10. MD5 success

Step 1.The Mqtt client send dData message which include picture information to Mqtt Server.

{

"msg":"dData",

"mac":"375692f98100",

"seq":7719,

"auth1":"00000000",

"dType": "hex",

"data":"XXXXX”}

xxxxx is picture information.

Step 2. After Gateway received the message, it will send success result with cause=1 to Mqtt Server/Client. This message means the Gateway has received the message but not execute.

{

"mac": "405192f98100",

"cause": 1,

"msg": "dAck",

"seq": 7719,

"rslt": "succ"

}

Step 3~4. The Gateway setup connection with ESL;

Step 5~10. The Gateway using MD5 to verify the device.

Step11. After MD5 success, the Gateway send the parameters to ESL device

Step12. After configuration success, the Gateway send the success result with cause=0 to MQTT server/client.

{

"msg": "dAck",

"mac": "405192F98100",

"rslt": "succ",

"seq": 7719,

"cause": 0

}

If command execute fail, Gateway will send fail result with cause. The error cause was describe in setcion*《4.8 Error Code for downdata Command》*

{

"msg": "dAck",

"mac": "405192F98100",

"rslt": "fail",

"seq": 7719,

"cause": 129

}

### Update ESL picture failed

KBeacon

MQTT Server

Gateway

MQTT Client

1.Publish: subaction

msg: dData{picture}

2.Publish:Pubaction

msg: cfg, success1

5. authentication failed

8.Publish:Pubaction

msg: dAck, fail)

4.BLE connection setup

3. Scanning device

Step 1.The Mqtt client send dData{picture} message to Mqtt Server.

Step 2. After Gateway received the message, it will send ack(cause=1) message to Mqtt Server/Client.

{

"mac": "405192f98100",

"cause": 1,

"msg": "dAck",

"seq": 7719,

"rslt": "succ"

}

Step 3~5. The Gateway setup connection with ESL but authentication failed.

Step6. Gateway send fail result with error code to MQTT server/client. The error cause was describe in setcion *<<4.6 Error Code for Ack Command>>*

{

"msg": "dAck",

"mac": "405192F98100",

"rslt": "fail",

"seq": 7719,

"cause": 257

}

### Update password of ESL

KBeacon

MQTT Server

Gateway

MQTT Client

1.Publish: subaction

msg: dData{newpassword}

2.Publish:Pubaction

msg: dAck, success1

5.MD5 authentication

4.BLE connection setup

6.{cfg new password}

7.{updae ack}

8.Publish:Pubaction

msg: dAck, success0

3. Scanning device

## ESL ID definition rule

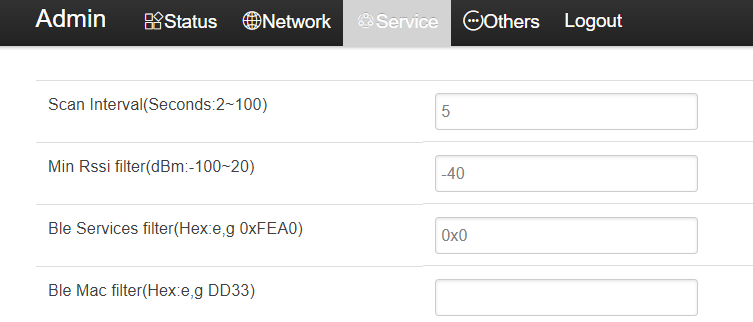
According to the Bluetooth protocol, the device ID address of the label is obtained byte-by-byte according to the MAC address defined by IEEE. The first two MAC addresses represent the manufacturer ID. At present, the first two labels are fixed to DD33. The QR code printed on the label is the MAC address.

As shown below, the QR code of the following tags is DD330A000986. Its corresponding tag ID is 8609000A33DD. The tag ID in all JSON messages is 8609000A33DD.



## Service Configuration

### Scanning paramaters



Once the Gateway power on, it will periodicately scanning ESL advertisment packet. You can configure following scanning paramaters on Gateway.

**1. Scan Interval:** This paramaters used for filter same Beacon report advertisment packet multi-time. the broadcast message of the same ESL will only be reported to MQTT server once in one **Scan interval**.

**2. Min Rssi filter:** If this paramaters was set, the Gateway will only report the advertisement packet which signal > **Min Rssi** value.

**3. Ble Services filter:** If this paramaters was set, the Gateway will only report the advertisement packet which include BLE services ID.

**4. Ble Mac filter:** If this paramaters was set, the Gateway will only report the Beacon advertisement packet which mac address include the filter value.

For example, if Ble Mac filter value set to 33DD, then following ESL advertisement packet will report to HTTPs cloud.

* ESL1: mac = 0x33DD01000002
* ESL1: mac = 0xA133DD010002
* ESL1: mac = 0xA10005033DD2

# Message format

## Alive message

Direction: Gateway=>MQTT Server

MQTT topic：publish

Message Name: alive

Example:

{

"msg": "alive",

"subaction": "ESL/subaction/D03304001182",

"pubaction": "ESL/pubaction/D03304001182",

"gmac": "D03304001182"

}

Gmac: the Gateway mac address.

## Advertisement packet report

Direction: Gateway=>MQTT Server

MQTT topic：publish

Message Name: advData

Note: The ESL ID in the status report message is not the same as the QR code MAC address on the tag shell, but the reverse of the QR code MAC address. For the correspondence between the two, see "3.3 ESL ID Address Definition Rule".

Example:

{

"msg": "advData",

"gmac": "66E800E29647",

"obj": [

{

"dmac": "795792F98100",

"rssi": "-71",

"data1": "0201060302A0FE10FF4B4D100000000B371E0001020304CA"

},

{

"dmac": "265892F98100",

"rssi": "-40",

"data1": "0201060302A0FE10FF4B4D100000000C83210001020304CA"

}

],

"seq": 70

}

1）gmac：mac address of Gateway；

2）Every device status include following infomration:

dmac：ID of ESL；

rssi：received RSSI signal about device;

"data1": "0201060302A0FE10FF4B4D100000000C83210001020304CA"

* 0201060302：reserved
* A0FE：services id, the services id about all ESL are A0FE；
* 10：next message length
* FF4B4D：Reserved
* 10：device type, 0x10 means ESL device；
* 00：The firmware version number of the label, the label supported by the different version number will have differences.
* 00：Capability field 1, high 4bit indicates working status, 0x1 identifies electronic label display fault, low 4bit reserved;
* 00：Capability field 2, high 4bit reserved; low 4bit indicates label type, Label type:
  + 0x0: 2.9-inch two-color screen;
  + 0x01: 2.9-inch three-color screen;
  + 0x2: 4.2-inch two-color screen;
  + 0x3: 4.2-inch three-color screen ;
  + 0x4: 2.1-inch two color screen;
  + 0x5: 2.1-inch three color screen
* 0C83：Battery voltage level, using big-end coding, (0C83 voltage is 3203mV)
* 2100：Enverioment temperature, signed number, big endian code, (2100 means 33 ° C)
* 01020304： The currently displayed picture ID is specified by the client management software when updating the picture. It is recommended to assign a unique picture every time the picture is updated, so that the picture currently displayed by each tag can be monitored.
* CA： The signal power of the tag at 1 meter, using the number of symbols, (0xCA = -54dBm) is used to calculate the distance between the tag and the base station;

Seq： For each sequence number of the reported message, increment by 1 each time

Firmware version

|  |  |
| --- | --- |
| version | capibility |
| 0x00 | - Support 2.9 inch black and white label  - Support for updating passwords  - Support to modify the broadcast period |
| 0x01 | - Added support for image compression algorithm, refreshing will be more faster |

dmac: ESL mac address

rssi: the rssi about ESL

data1: advertisement packet.

Seq: the seq will increase every message.

## Update Picture

### Updating pictures in uncompressed mode

Direction: MQTT=>Gateway

MQTT topic：subaction

Message name:dData

Example:

{

"msg":"dData",

"mac":"375692f98100",

"seq":4,

"auth1":"00000000",

"dType": "hex",

"data":"01010203041280FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFFFFFFFFFFF80FFFF80FFFFFFFFFFFFFFE7FFFFFFFF80FFFF80FFFFFFFFFFFFFF079FFFFFFF80FFFF80FFFFFFFFFFFFFF001FFFFFFF80FFFF80FFFFFFFFFFFFFF001FFFFE07FFF03FFFFFFFFFFFFFFFFFC01FFFFE07FFF03FFFFFFFFFFFFFFFFFE7FFFFFE07FFF03FFFFFFFFFFFFFFFFFFFFFFFFE07FFF03FFFFFFFFFFFFFFFFFE63FFFFE07FFF03FFFFFFFFFFFFFFFFFE43FFFFE07FFF03FFFFFFFFFFFFFFFFFC41FFFFE07FFF03F80FFFFFFFFFFFFFFC89FFFFE07FFF03F80FFFFFFFFFFFFFFC89FFFFE07FFF03F80FFFFFFFFFFFFFFC09FFFFE07FFF03F80FFFFFFFFFFFFFFE11FFFFE07FFF03F80FFFFFFFFFFFFFFF3BFFFFE07FFF03F80FFFFFFFFFFFFFFFFFFFFFE07FFF03F80FFFFFFFFFFFFFFF9FFFFFE07FFF03F80FFFFFFFFFFFFFFE17FFFFE07FFF03F80FFFFFFFFFFFFFFE13FFFFE03FFF03F80FFFFFFFFFFFFFFC11FFFFE0000003F80FFFFFFFFFFFFFFCD9FFFFE0000003F80FFFFFFFFFFFFFFC19FFFFE0000003F80FFFFFFFFFFFFFFE01FFFFE0000003F80FFFFFFFFFFFFFFF03FFFFE0000003F80FFFFFFFFFFFFFFF87FFFFE0000003F80FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80FFFFFFFFFFFFFFF1FFFFFFFFFFFFFF80FFFFFFFFFFFFFFE07FFFFFFFFFFFFF80FFFFFFFFFFFFFFC03FFFFFFFFFFFFF80FFFFFFFFFFFFFFC41FFFFFFFFFFFFF80FFFFFFFFFFFFFFEF9FFFFFFFFFFFFF80FFFFFFFFFFFFFF079FFFFFFFFFFFFF80FFFFFFFFFFFFFF003FFFFFFFFFFFFF80FFFFFFFFFFFFFF003FFFFFFF80000000FFFFFFFFFFFFFFF81FFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFF80000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE17FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE13FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC11FFFFFFFFFFFFFFFFFFFFFFFFFFFFFCD9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC19FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF03FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF87FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE3FFFFFE1FFFFFFFFFFFFFFFFFFFFFFFC01FFFFE01FFFFFFFFFFFFFFFFFFFFFFC01FFFFE003FFFFFFFFFFFFFFFFFFFFFC61FFFFE0003FFFFFFFFFFFFFFFFFFFFE7FFFFFE00007FFFFFFFFFFFFFFFFFFF03FFFFFF000007FFFFFFFFFFFFFFFFFF001FFFFFFF00000001FFFFFFFFFFFFFF801FFFFFFFFC000003FFFFFFFFFFFFFFFE1FFFFFFFFE000003FFFFFFFFFFFFFFFFFFFFFFFFC0000003FFFFFFFFFFFFFFE7FFFFFFF000000003FFFFFFFFFFFFFF079FFFFE000007FFFFFFFFFFFFFFFFFF001FFFFE0000FFFFFFFFFFFFFFFFFFFF001FFFFE0007FFFFFFFFFFFFFFFFFFFFC03FFFFE007FFFFFFFFFFFFFFFFFFFFFF7FFFFFE07FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE3FFFFFFFFFFFFFFFFFFFFFFFFFFFF0003FFFFFFFFFFFFFFFFFFFFFFFFFF000003FFFFFFFFFFFFFFFFFFFFFFFF00000003FFFFFFFFFFFFFFFFFFFFFF0000000003FFFFFFFFFFFFFFE07FFFFE0000000003FFFFFFFFFFFFFFE01FFFFE000000083FFFFFFFFFFFFFFFE01FFFFE0001FFF87FFFFFFFFFFFFFFFFF3FFFFE07FFFFF87FFFFFFFFFFFFFFFEF9FFFFE007FFFF87FFFFFFFFFFFFFFFE01FFFFE000000F87FFFFFFFFFFFFFFFE01FFFFE0000000007FFFFFFFFFFFFFFF81FFFFE0000000003FFFFFFFFFFFFFFFFFFFFFFF800000003FFFFFFFFFFFFFFFFFFFFFFFFFC000003FFFFFFFFFFFFFFF07FFFFFFFFFFC0003FFFFFFFFFFFFFFE03FFFFFFFFFFFFC01FFFFFFFFFFFFFFC01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFCF1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFC79FFFFFFFFFFFFFFFFFFFFFFFFFFFFFE01FFFFFFFFFFFFFC1FFFFFFFFFFFFFFE03FFFFFFFFFFFFFC3FFFFFFFFFFFFFFF03FFFFFFFFFFFFFC3FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC3FFFFFFFFFFFFFFEFFFFFFFFFFFFFFFC3FFFFFFFFFFFFFFE3FFFFFFFFFFFFFFC3FFFFFFFFFFFFFFF0FFFFFFFFFFFFFFC3FFFFFFFFFFFFFFFC3FFFFE0000000003FFFFFFFFFFFFFFF00FFFFE0000000003FFFFFFFFFFFFFFE007FFFE0000000003FFFFFFFFFFFFFFE0E3FFFE0000000003FFFFFFFFFFFFFFEFF3FFFE0000000003FFFFFFFFFFFFFFFFF7FFFE0000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC000FFFFFFFFFFFFFFFFFFFFC7FFFFFF80007FFFFFFFFFFFFFFFFFFFE7FFFFFF00003FFFFFFFFFFFFFFFFFFFE3FFFFFF00001FFFFFFFFFFFFFFFFFFFE07FFFFF00001FFFFFFFFFFFFFFFFFFFE01FFFFE01601FFFFFFFFFFFFFFFFFFFF01FFFFE0FFC1FFFFFFFFFFFFFFFFFFFFFDFFFFE0FFE1FFFFFFFFFFFFFFFFFFFFFFFFFFE0FFE1FFFFFFFFFFFFFFFFFFFF1FFFFFE0000010003FFFFFFFFFFFFFFE13FFFFE0000000003FFFFFFFFFFFFFFC13FFFFE0000000003FFFFFFFFFFFFFFCD9FFFFE0000000003FFFFFFFFFFFFFFC19FFFFE0000000003FFFFFFFFFFFFFFE01FFFFE0000000001FFFFFFFFFFFFFFE01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF03FFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE6FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE6FFFFFFFF81FFF807FFFFFFFFFFFFFFE07FFFFFF800FFC000FFFFFFFFFFFFFFF003FFFFF000FF80007FFFFFFFFFFFFFF001FFFFF000FE00003FFFFFFFFFFFFFFE01FFFFE000FC00003FFFFFFFFFFFFFE6FFFFFFE001F800001FFFFFFFFFFFFFE6FFFFFFE0FFF007FC1FFFFFFFFFFFFFE0FFFFFFE1FFE00FFC1FFFFFFFFFFFFFE03FFFFFE1FF801FFE1FFFFFFFFFFFFFF001FFFFE0FF007FFC1FFFFFFFFFFFFFF801FFFFE00000FFF81FFFFFFFFFFFFFFE61FFFFE00001FC001FFFFFFFFFFFFFFFFFFFFFF00003FC003FFFFFFFFFFFFFFFFFFFFFF8000FFC003FFFFFFFFFFFFFF80FFFFFF8001FFC007FFFFFFFFFFFFFF007FFFFFE007FFC01FFFFFFFFFFFFFFF003FFFFFFFFFFFFFFFFFFFFFFFFFFFFE3E1FFFFFFFFFFFFFFFFFFFFFFFFFFFFE3F1FFFFFFFFFFFFFFFFFFFFFFFFFFFFE3F1FFFFFFFFFFFFFFFFFFFFFFFFFFFFF021FFFFE0000000001FFFFFFFFFFFFFF003FFFFE0000000003FFFFFFFFFFFFFF803FFFFE0000000003FFFFFFFFFFFFFFE0FFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFF0000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC00000001FFFFFFFFFFFFFFFFFFFFFFF8000000007FFFFFFFFFFFFFFFFFFFFFF0000000007FFFFFFFFFFFFFFFFFFFFFF0000000003FFFFFFFFFFFFFFFFFFFFFF0000000003FFFFFFFFFFFFFFFFFFFFFF07FFFFFF03FFFFFFFFFFFFFFFFFFFFFE0FFFFFFF83FFFFFFFFFFFFFFFFFFFFFE0FFFFFFFC3FFFFFFFFFFFFFFFFFFFFFE0FFFFFFFC3FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFE0000000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF00003FFFFFFFE000001FFFFFFFF007FF00007FFFFFFFF800001FFFFFFFF007FF0000FFFFFFFFFC00001FFFFFFFF007FF0001FFFFFFFFFC00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001F80000007E00001FFFFFFFF007FF0001F00000007E00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF000007FFFFFF8000001FFFFFFFF007FF00007FFFFFFFF000001FFFFFFFF007FF0000FFFFFFFFF800001FFFFFFFF007FF0001FFFFFFFFFC00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0003F00000007F00001FFFFFFFF007FF0003F00000003F00001FFFFFFFF007FF0003F80000007F00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0000FFFFFFFFFC00001FFFFFFFF007FF00007FFFFFFFF800001FFFFFFFF007FF00003FFFFFFFF000001FFFFFFFF007FF000007FFFFFF0000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF00000FFFFFFF8000001FFFFFFFF007FF00003FFFFFFFF000001FFFFFFFF007FF0000FFFFFFFFF800001FFFFFFFF007FF0000FFFFFFFFFC00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0003F00000007F00001FFFFFFFF007FF0003F00000003F00001FFFFFFFF007FF0003FA000000FF00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0000FFFFFFFFFC00001FFFFFFFF007FF0000FFFFFFFFF800001FFFFFFFF007FF00003FFFFFFFF000001FFFFFFFF007FF000007FFFFFF0000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF00000FE07FFFC000001FFFFFFFF007FF00007FE07FFFF000001FFFFFFFF007FF0000FFE07FFFFC00001FFFFFFFF007FF0001FFE07FFFFC00001FFFFFFFF007FF0001FFE07FFFFE00001FFFFFFFF007FF0003FFE07FFFFE00001FFFFFFFF007FF0003FFE07FFFFE00001FFFFFFFF007FF0003F0007E007F00001FFFFFFFF007FF0003F0007E003F00001FFFFFFFF007FF0003F00000007F00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0003FFFFFFFFFE00001FFFFFFFF007FF0001FFFFFFFFFE00001FFFFFFFF007FF0000FFFFFFFFFC00001FFFFFFFF007FF0000FFFFFFFFF800001FFFFFFFF007FF00003FFFFFFFF000001FFFFFFFF007FF00000FFFFFFF8000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFF007FF0000000000000000001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"}

mac： ID of ESL；

seq：Message sequence number, the client management software sends a new message every time, the sequence number needs to be added 1

auth1：ESL device password, the factory default password is eight ascii zero, “00000000”

dType: Data format type, “hex” indicates that the data content of the picture is hexadecimal

data：Image content, specific fields are defined as follows:

Example: "data":"01010203041280FFFF………..

* 01：picture data type, 0x01 means hex uncompress data；
* 01020304： Picture ID, After the image is updated, the status report message of the ESL will carry the picture ID. You can use the picture ID to establish a releationship with the price. For example, picture ID 0x278 corresponding price of 14.7$.
* 1280：The length of the picture is hexadecimal with small endian code, ie 4736, corresponding to the 2.9-inch label;
* The content of the following content and the content of the image; the encoding method of the image refers to the document "ESL image format. docx"

### Updating pictures in compressed mode

Direction: MQTT=>Gateway

MQTT topic：subaction

Message name:dData

Example:

{

"msg": "dData",

"mac": "4631000A33DD",

"seq": 19871,

"auth1": "00000000",

"dType": "ascii",

"data": ""

}

*Note: ESL with a version number greater than 1 support compression to update the image.*

mac： ID of ESL；

seq：Message sequence number, the client management software sends a new message every time, the sequence number needs to be added 1

auth1：ESL device password, the factory default password is eight ascii zero, “00000000”

dType: Data format type, “ascii” indicates that the data content is encode by ascii compressed mode.

data：Image content, specific fields are defined as follows:

* 0x02: Picture update type, 0x02 means to update the picture with a compressed ASCII algorithm;
* 0x00000091：Picture ID
* 0x1280：Picture length
* 0x0A：Dictory code number
* FFFE3F1FC7000F0787C0：Code about dictory
* The next content is image content; the encoding method of the image refers to the document "electronic tag image encoding format. docx"

The principle of the image compression algorithm is as follows:

1. A character with a high frequency appears in the statistical picture, which is replaced by 1 character. For example, 0xFF appears frequently. Then, the 'G' is used instead, and the words 'G'~'Z' characters are used in order, up to 20 characters.

2. Use lowercase a~z to indicate the number of characters (1~). For example, 0xFFFFFF, the compression code is cG.

Please referance the compressed demo：https://github.com/kkmhogen/ESLBin2Json.git

### Get parameters command

{

"msg": "dData",

"mac": "405192f98100",

"seq": 7716,

"auth1": "0000000000000000",

"dType": "json",

"data": {

"msg": "getPara"

}

}

mac: ESL address

seq: message sequence, the mqtt client need increate the sequence in every message;

auth1: the password about the ESL device, the default password is 16 ASCII 0, “0000000000000000”

dType: download data format, it is fixed to “json” for ESL.

data: message content that send to ESL device.

## Modify password

Direction: MQTT=>Gateway

MQTT topic：subaction

Message name:dData

{

"msg": "dData",

"mac": "625692f98100",

"seq": 12,

"data": {

"tag": "mdf",

"pwd":"12345678",

},

"dType": "json",

"auth1": "00000000"

}

## Ack message

Direction: Gateway=>MQTT Server

MQTT topic：pubaction

Message Name: dAck

Example

{

"msg": "dAck",

"mac": "625692F98100",

"rslt": "succ",

"seq": 8,

"cause": 0

}

Mac: ESL ID;

Seq: the sequence number of the corresponding request command;

Rslt:succ means success, fail means failure;

Cause: error code representation

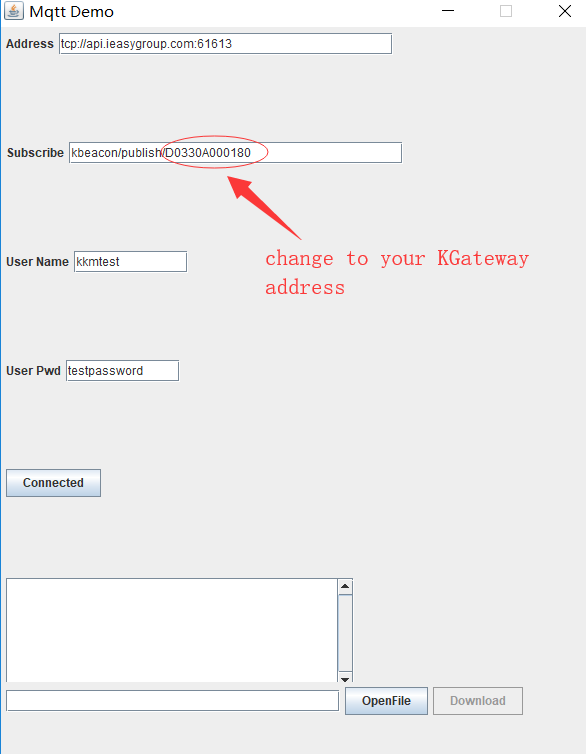
## Error Code for Ack message

|  |  |  |
| --- | --- | --- |
| rslt | value | meanning |
| succ | 1 | Gateway has received command, now start execute |
| 0 | Command execute success |
| fail | 65 | The buffered message exceed gateway capilibity |
| 129 | Scan ESL device failed |
| 130 | Gateway does not support the device |
| 131 | Gateway connect to device failed |
| 133 | Other internal error |
| 257 | Password error |
| 258 | Authentication algorithm not support |
| 259 | JSON data format invalid, it means the data field is invalid |
| 289 | Request to refresh image size and actual ink screen size definition does not match |
| 290 | The length of the image data delivered does not match the definition of the image header. |
| 291 | screen error |
| 298 | Unknown message |

# Gateway MQTT client demo

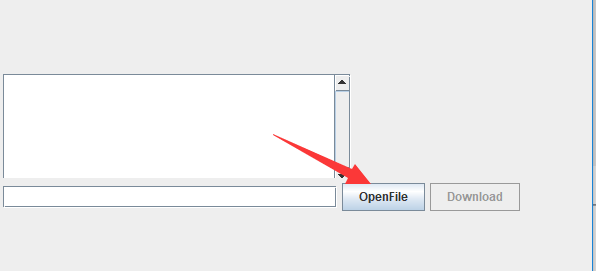
We have develop an MQTT client demo to connect to Gateway to recevie data. The project is baseon JDK1.7.

You can download the demo on: https://github.com/kkmhogen/GatewayClientDemo.git



Open the JSON command script file to download command. We have two example JSON file in bin directory.

You should modify the mac address in JSON file to you device before running download command.



# MQTT Test Guidance

## Configuring Gateway MQTT Parameters

We provide an test mqtt server, and the default mqtt parameter about Gateway is set to the test mqtt server.

Address: tcp://api.ieasygroup.com:61613

User name: kkmtest

Password: testpassword

For other parameter configuration of the gateway, see " Gateway Configuration Introduction.docx "

## Using third part MQTT client to verify Gateway

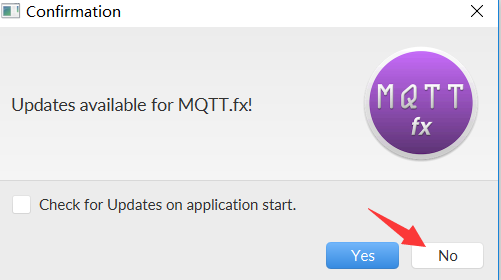
There are many third-party MQTT clients. We recommended to use Mqtt.fx. The client supports windows, linux, and mac.

Following example was test on Win10 and the mqtt.fx version is 1.5.0.

### Setup mqtt.fx

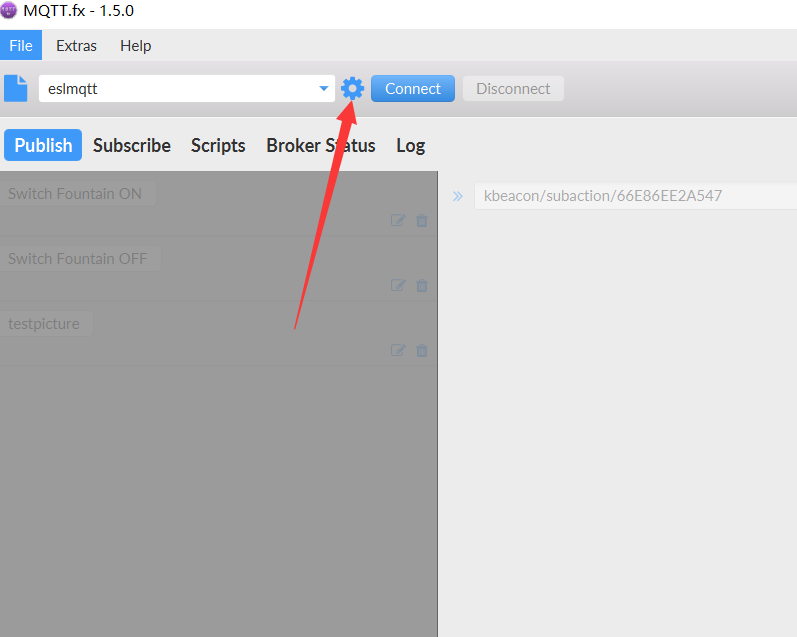
Download the software from following link, then setup it on your windows PC.  
<http://www.jensd.de/apps/mqttfx/1.5.0/>

### Running mqtt.fx

1. When you open it, it will prompt an update message. Click No directly, otherwise it will report an error.  
 

2. After opening the app, enter the setting page.

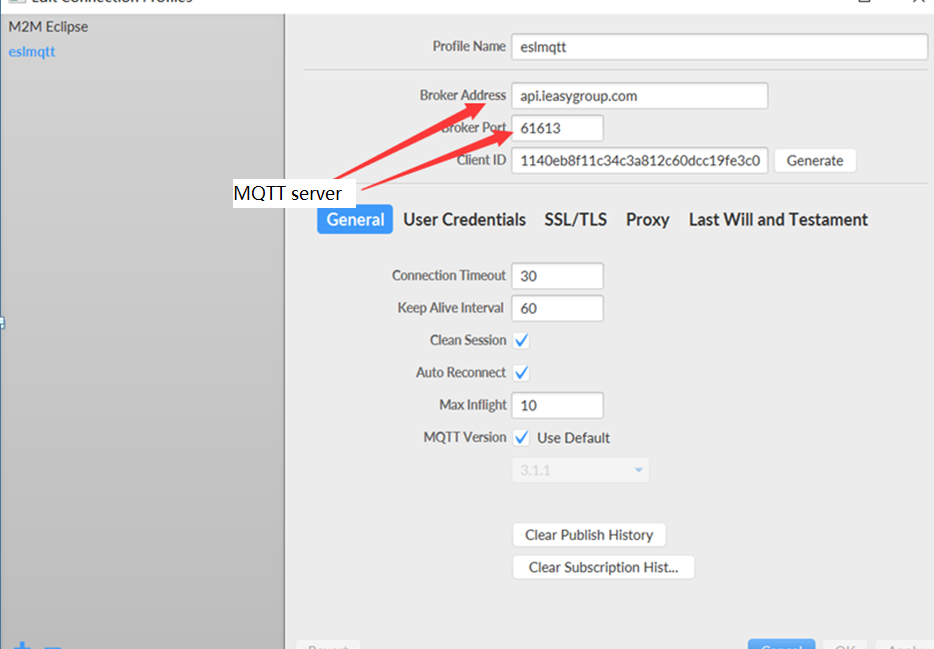
Step1. Tap on setting button.



Step2. Input the address of the MQTT server, as well as the username, IP address, and click OK.

* + Broker Address: api.ieasygroup.com
  + Broker Port: 61613
  + Client ID: Select Random Generate, click Generate

 Other parameters refer to the screenshot below.

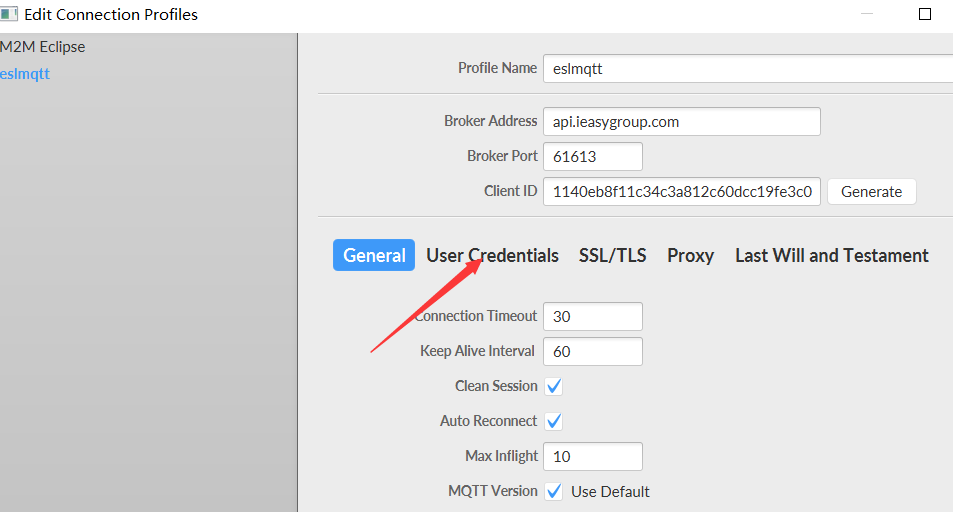


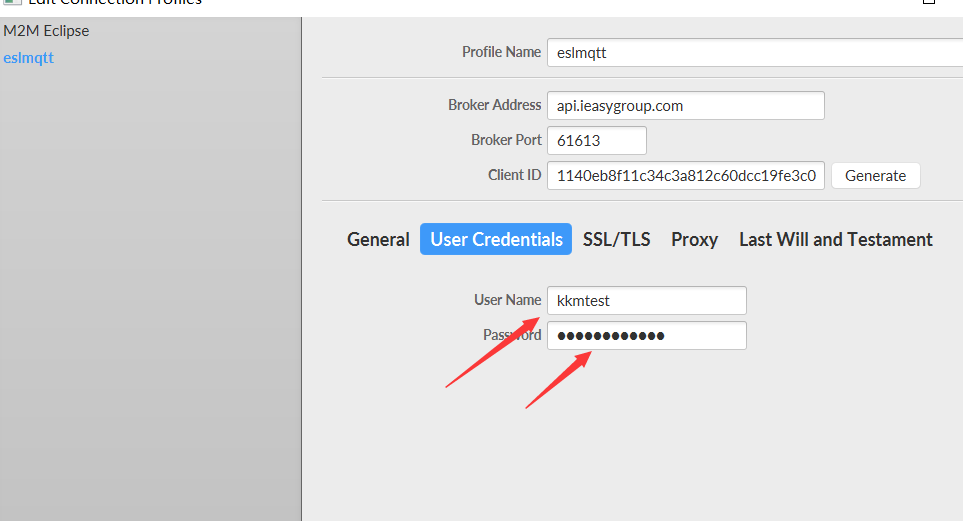
Step3. set the username and password

* Click “User Credentials” to switch to the username and password input interface.
* Enter your username and password.

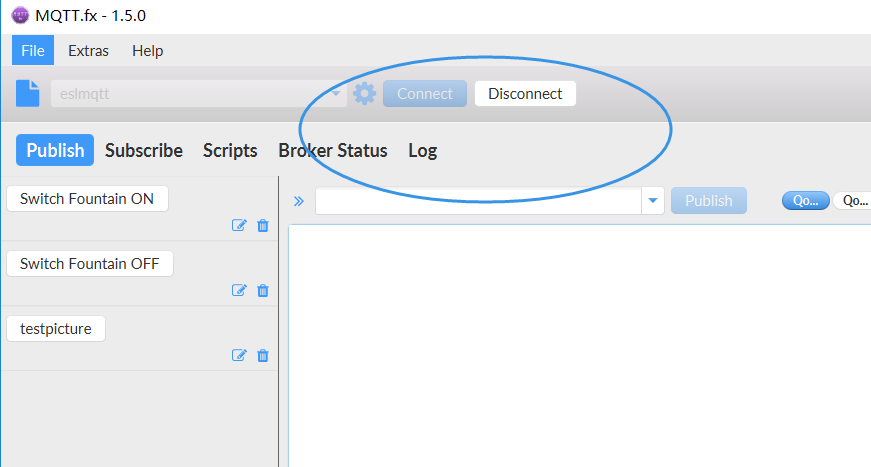
The default username of the test server is kkmtest.

The default password is: testpassword.





Step4. After setting, tap on connect. The connect button will become disable if connect success.



## Mqtt message test

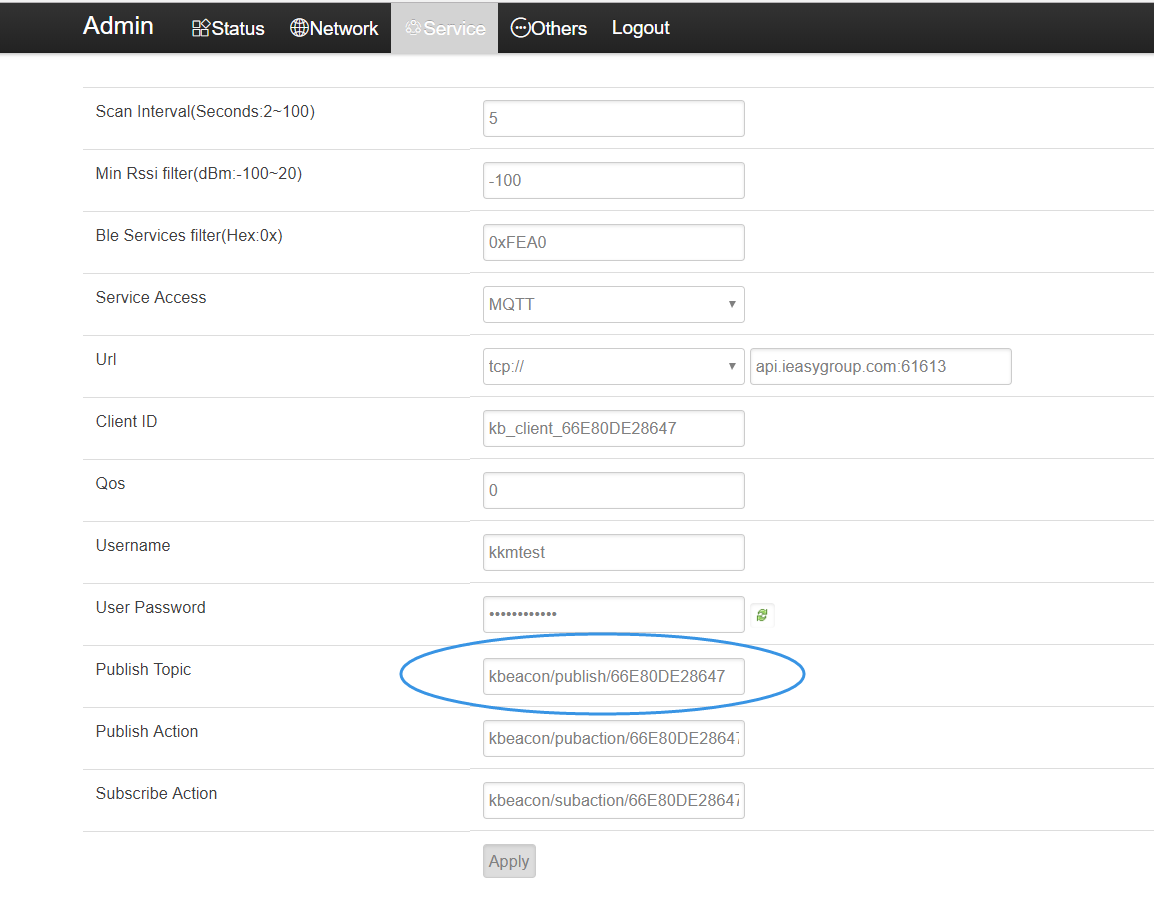
### Monitoring ESL advertisment Data

1) Get the gateway publish topic name.

 Method 1: If you have not logged in to the gateway and modified the mqtt publish topic name, the default name is: ESL/publish/mac address, for example, ESL/publish/66E80DE28647

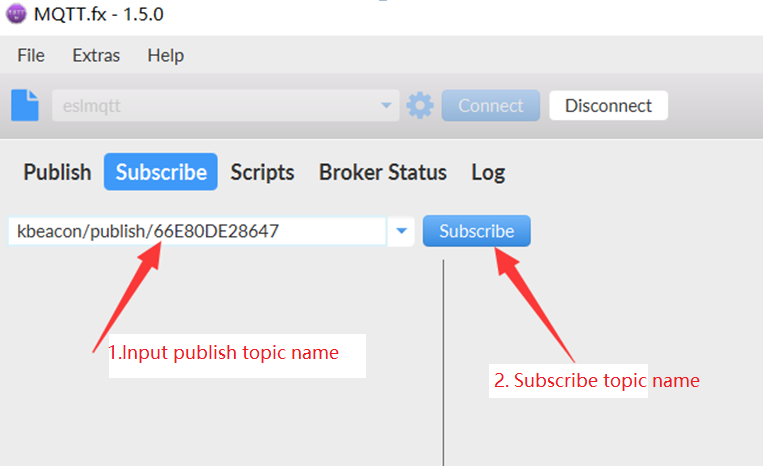
Note: the ID (66E80DE28647) in the subscription topic is the device ID of the Gateway, not the device ID of the ESL.

 Method 2: If you have modified the name of the published topic, you can log in to the gateway through the chrome browser. For the login method, refer to the <<Gateway Configruation Introduction. Docx>>. You can see the name of the published topic name on the Services page.



2）Subscribe Gateway publish topic.

The topic name is: ESL/publish/66E80DE28647；



3) Observe the ESL status data report:

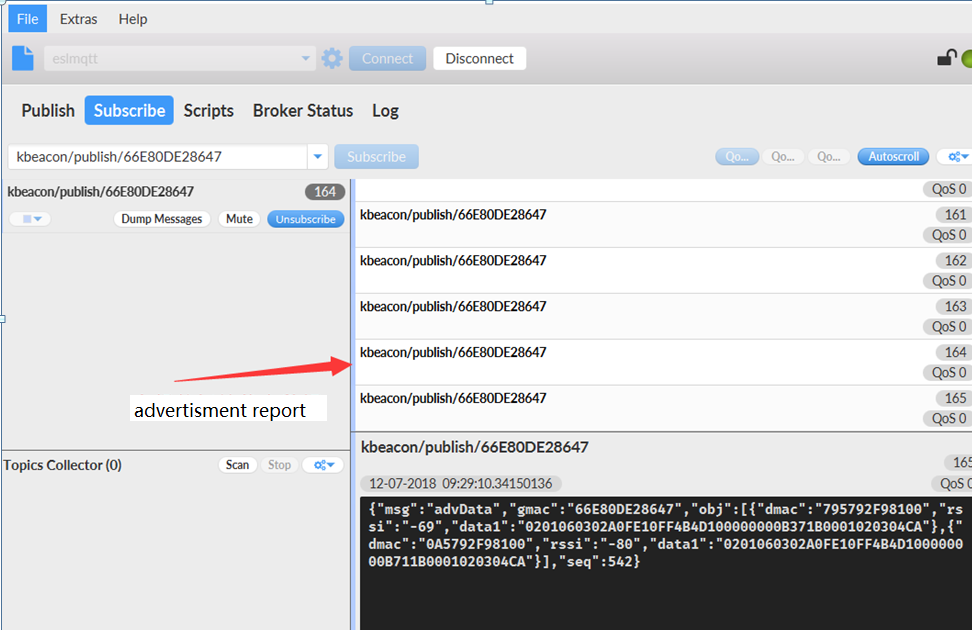
If there is a ESL next to the gateway, after waiting for serveral seconds, you will see the reported data.

As can be seen from the figure below, the gateway monitors the status of two ESL, and the ESL ID (mac address) is:

* ESL 1: 795792F98100, rssi signal strength is -60dBm
* ESL 2: 0A5792F98100, rssi signal strength is -66dBm,

Note: The ESL ID in the status report message is not the same as the QR code MAC address on the tag shell, but the reverse order of the QR code MAC address. The relationship between the two can be found in the 3.3 ESL ID Address Definition Rules.

And include status report data in data1

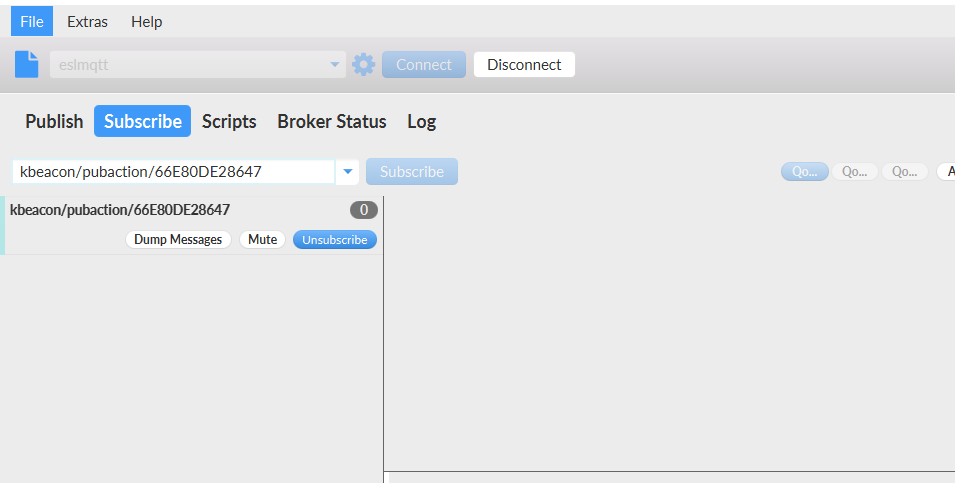


### Updating ESL paramaters

Step1. Get pubaction topic name

* If you have not logged in to the gateway and modified the mqtt pubaction topic name, the default name is: ESL/pubaction/mac address, for example, ESL/pubaction/66E80DE28647

Step2. Switch to the Subscribe page of the MQTT.fx software and start the subscription to the pubaction topic.

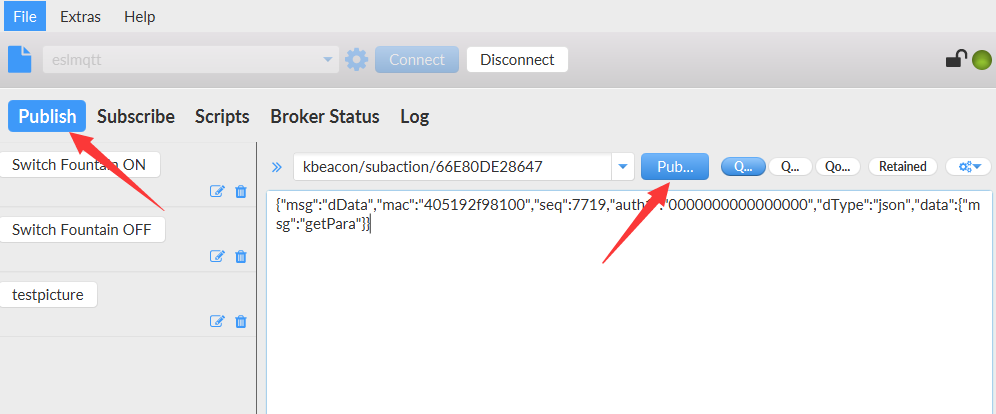


Step3. Obtain the Gateway subaction topic name. The Gateway using this topic to receives command which send from MQTT client.

* If you have not logged in to the gateway and modified the mqtt subaction subject name, the subject name defaults to: ESL/subaction/mac address, for example, ESL/subaction/66E80DE28647

Step4. Switch to the Publish page of MQTT.fx and send the command request.

* Input the name of the subaction topic as: ESL/subaction/66E80DE28647; then click Publish to post the message;



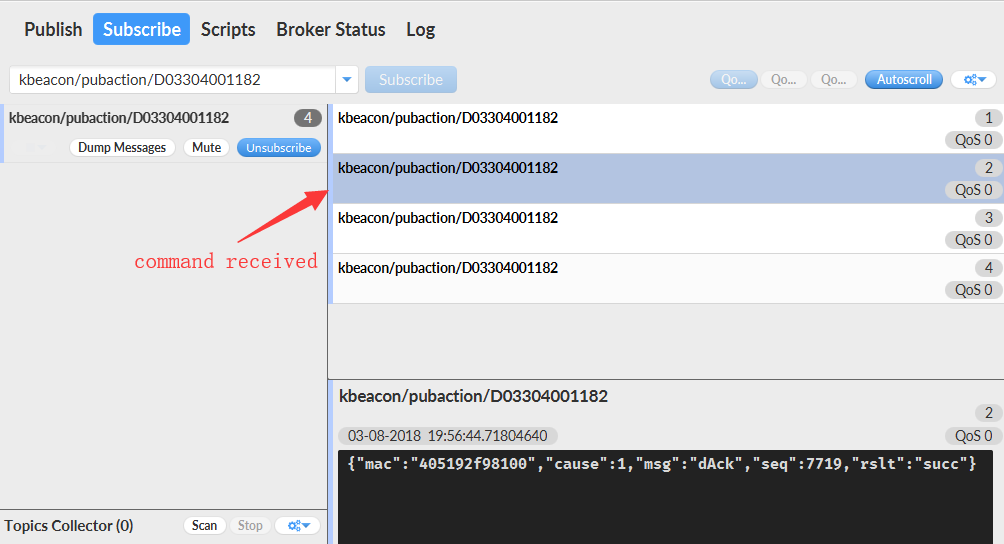
Json example:

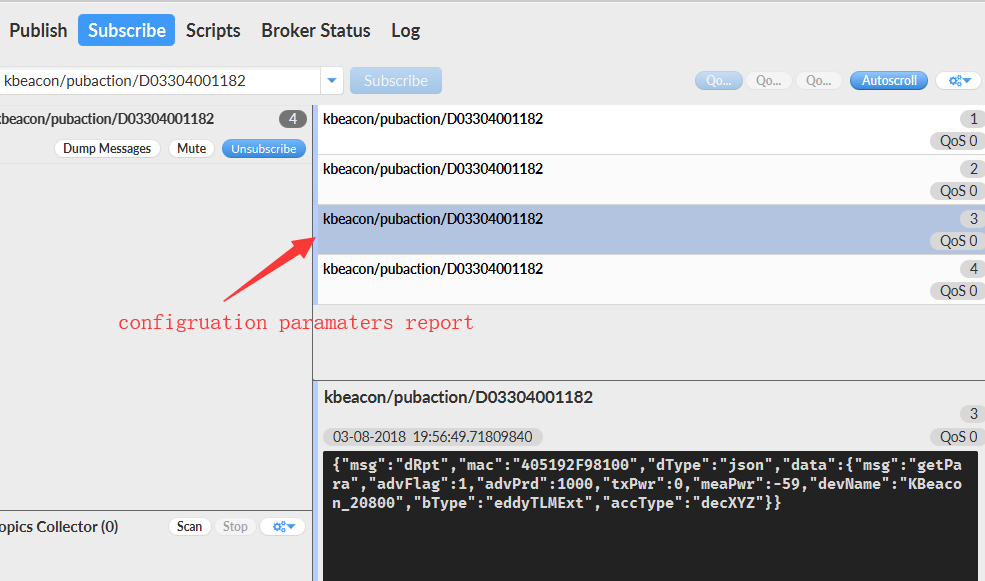
{"msg":"dData","mac":"405192f98100","seq":7719,"auth1":"0000000000000000","dType":"json","data":{"msg":"getPara"}}

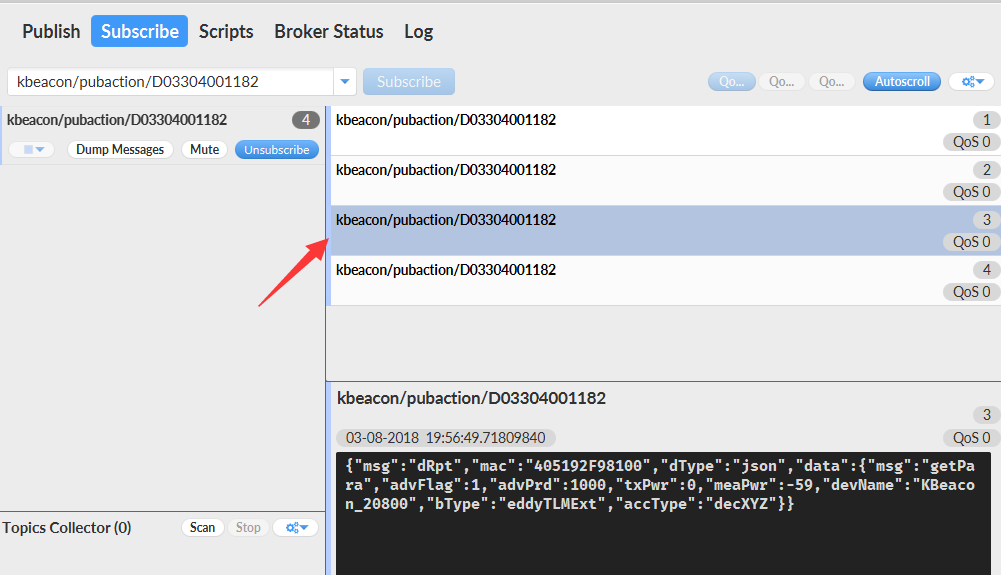
Please replace the mac address to you ESL ID.

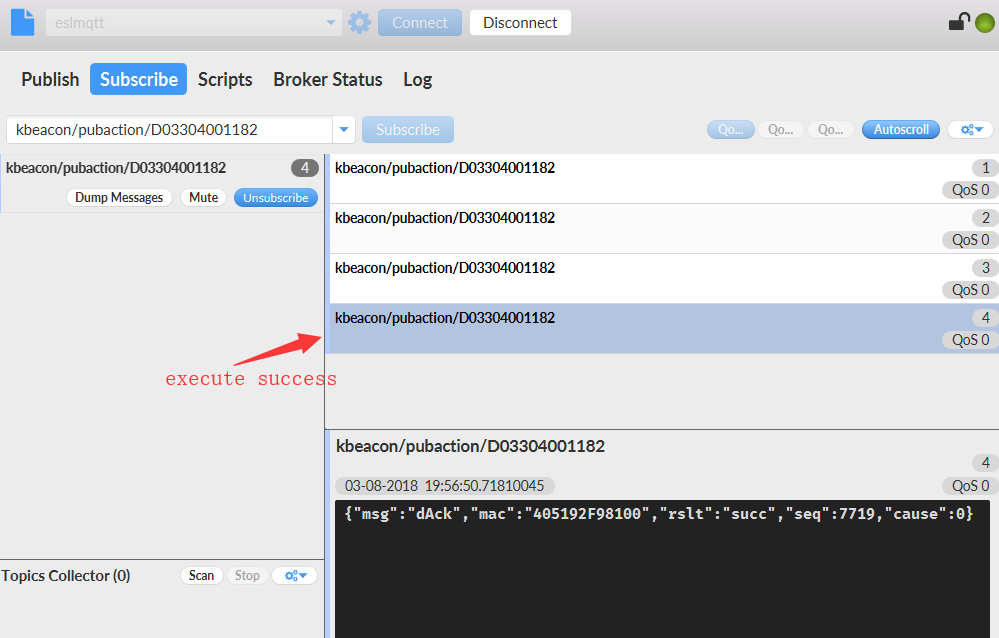
Step 5. View the command execution result

If the execution fails, you can find the corresponding reason based on the returned cause value. The reason value here corresponds to decimal, and the corresponding reason can be obtained according to the error code in the section 4.6.









## Installing the MQTT Server

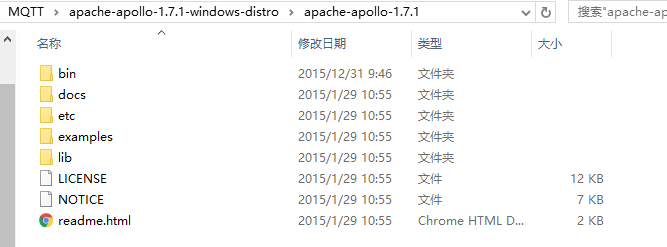
There are several third-party MQTT server software. Following example uses apache-apollo-1.7.1 as an example which test in windows10 environment.

### Download software

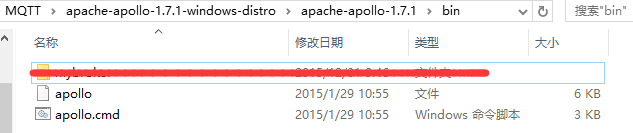
<http://activemq.apache.org/apollo/download.html>

1.Select windows version to download

2.unpacket the software to computer

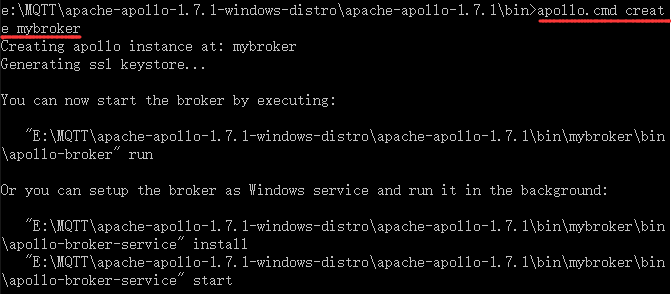


Bin dierctory contains two files apollo and apollo.cmd:

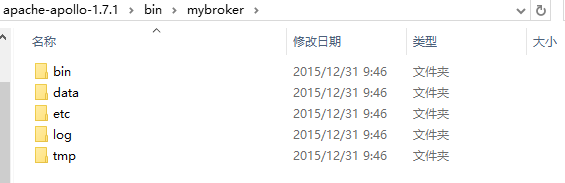


### Install MQTT service

Run apache-apollo-1.7.1\bin\apollo.cmd, enter create mybroker (any name is taken, here is based on the official website) to create a server instance, the server instance contains all the configuration, runtime data, etc. And associated with a server process. If you double-click apollo.cmd and it will be closed when you flash it, you need to type the command on the command line:



Then you can see new directory mybroker was created.



### Configuring User Name and IP Address

After the above step, it will generate mybroker folder in the bin directory

1) Add users:

The etc\users.properties file contains the username and password used to connect to the MQTT server. As you will see later, you can modify the original admin=password, and then add a new username and password.

The following example adds a username named test and the password is "testpassword"

* open users.properties file(\apache-apollo-1.7.1\bin\mybroker\etc\users.properties)

　　## ---------------------------------------------------------------------------  
　　## Licensed to the Apache Software Foundation (ASF) under one or more  
　　## contributor license agreements. See the NOTICE file distributed with  
　　## this work for additional information regarding copyright ownership.  
　　## The ASF licenses this file to You under the Apache License, Version 2.0  
　　#  
　　# admin=ENC(Cf3Jf3tM+UrSOoaKU50od5CuBa8rxjoL)  
　　#

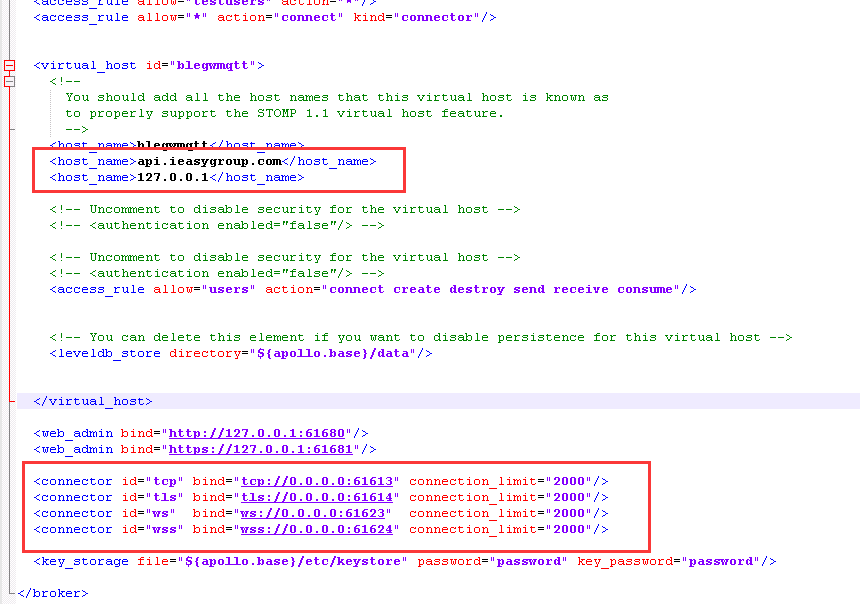
admin=password

test=testpassword

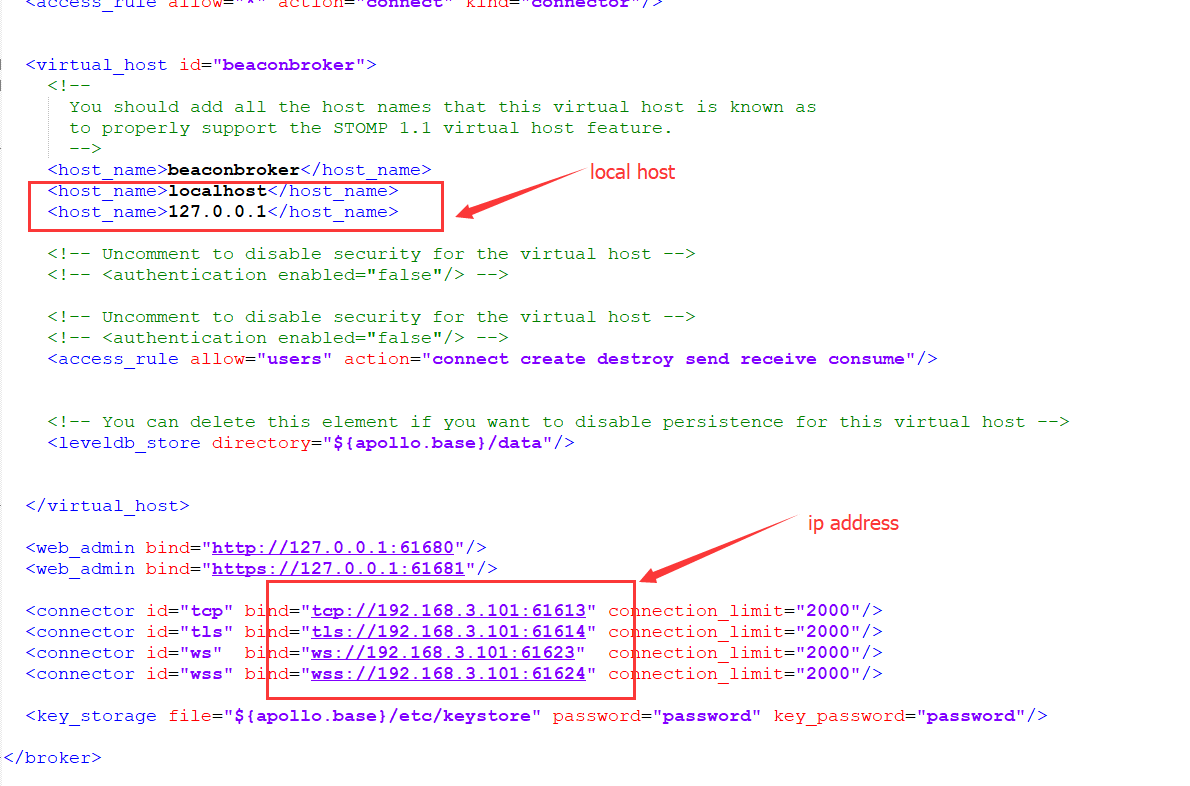
**2) Modify the server IP address**

Open etc\apollo.xml

* If using DNS name, please ref:



* If using ip address, please ref:

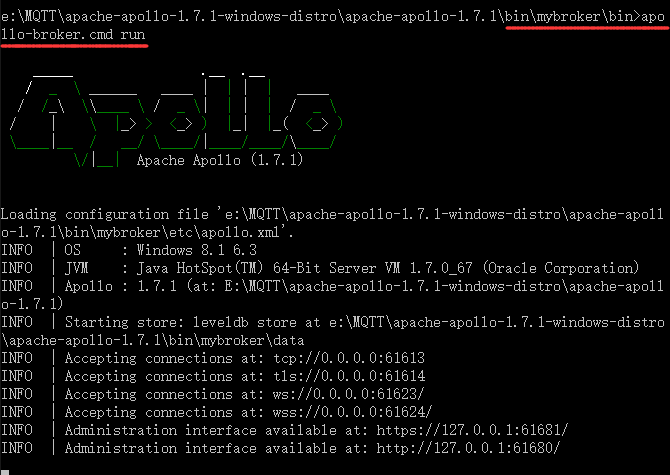


### Running MQTT service

1. Run cmd，

2. execute command:

apache-apollo-1.7.1\bin\mybroker\bin\apollo-broker.cmd run



### Verify MQTT service running

You can enter http://127.0.0.1:61680/ in the browser, it will be automatically transferred to: http://127.0.0.1:61680/console/index.html, apollo login page. Following image means MQTT service install complete.

