Alibaba Cloud IoT Platform

Best Practices

Issue: 20190125

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Connect Android Things to Alibaba Cloud IoT Platform

This article uses an indoor air test project as an example to explain how to connect Google Android Things to Alibaba Cloud IoT Platform.

Hardware

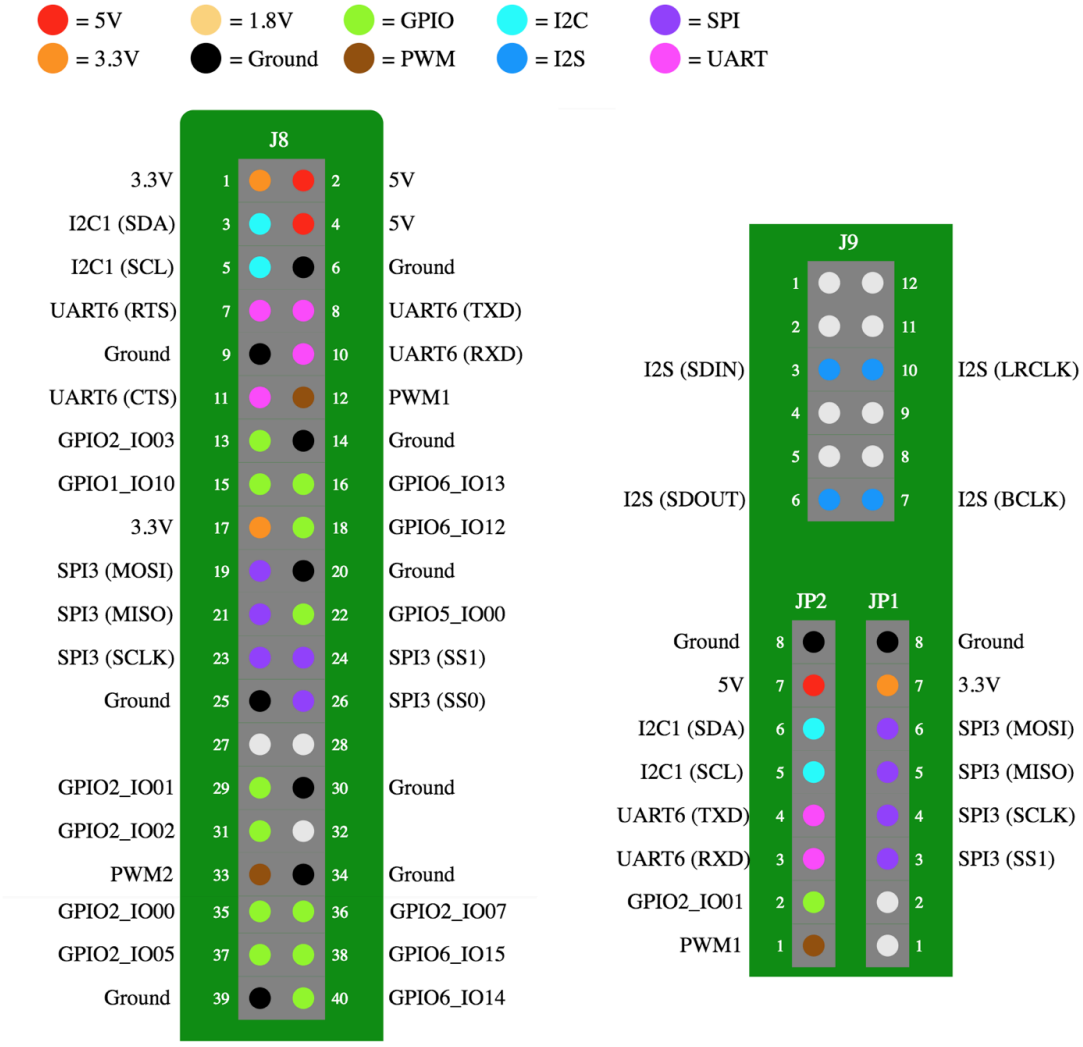
- Hardware list for the project

  The following table lists the hardware required by the indoor air test project.

Hardware	Picture	Remarks
NXP Pico i.MX7D Development board		Android Things 1.0 Note: You can also use Raspberry Pi instead.
DHT12 Temperature and humidity sensor		Supports I2C data communication method.

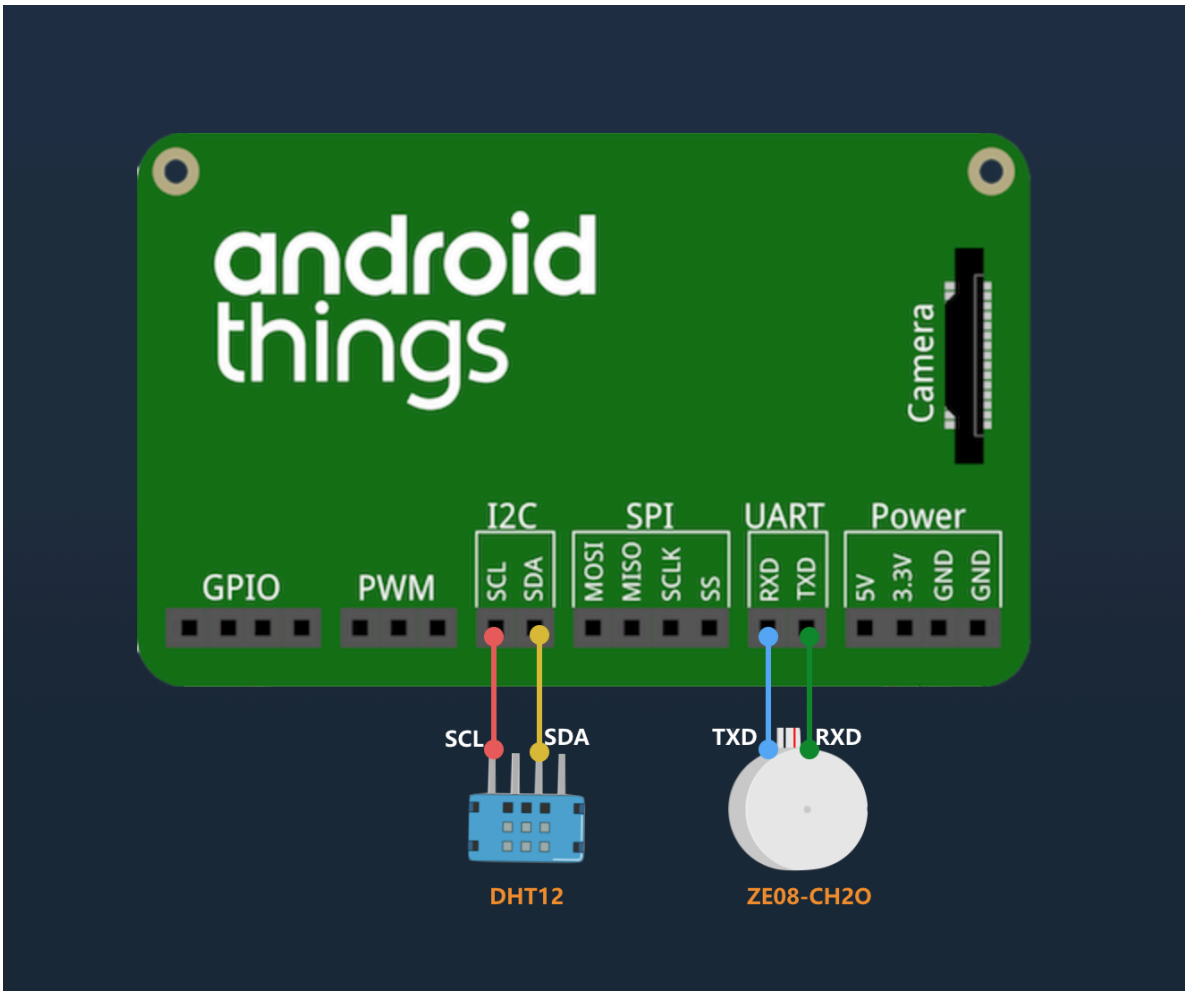
Hardware	Picture	Remarks
ZE08-CH2O Formaldehyde detection sensor		Supports UART data communication method.

- Layout diagram of the NXP i.MX7D development board pins



For more information about NXP Pico i.MX7D, see *https://developer.android.com/things/hardware/imx7d-pico-io*.

- Diagram of the hardware connection

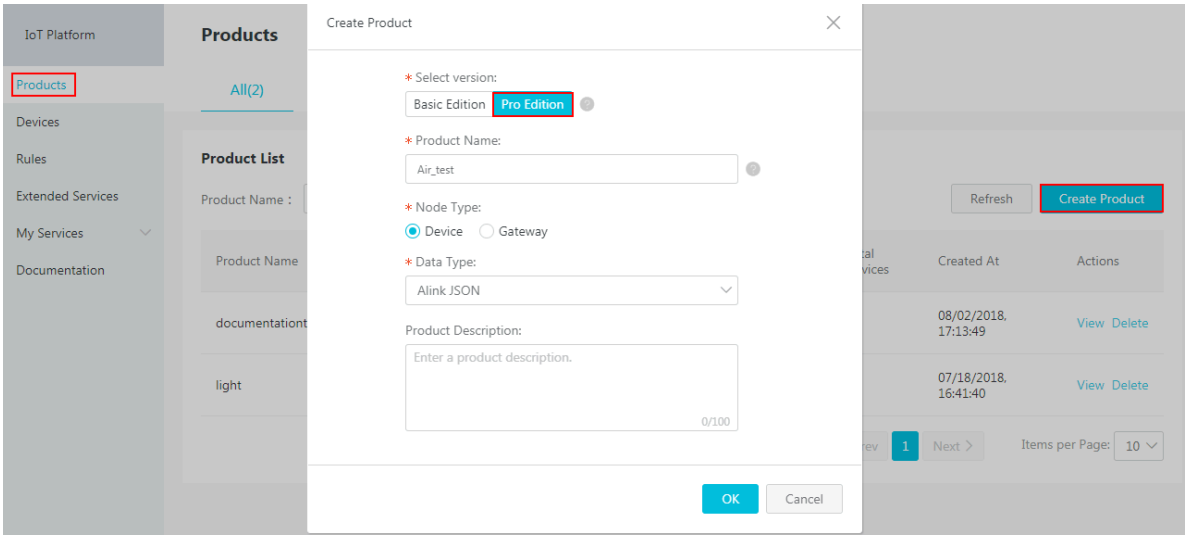


  - Connect the SCL (clock line) and SDA (data line) pins of the temperature and humidity sensor (DHT12) with the I2C SCL and SDA pins of the development board.
  - Connect the TXD (transmit data) pin of the formaldehyde detection sensor (ZE08-CH2O) with the RXD (receive data) pin of the development board, and connect the RXD pin of ZE08-CH2O with the TXD pin of the development board.

Create a product and device in the Alibaba Cloud IoT Platform console

1. Log on to the *IoT Platform console*.
2. Create a product in IoT Platform Pro.

   On the Products page, click Create Product. Select Pro Edition as the version when you are creating the product. For more information, see *Create a product (Pro Edition)*.

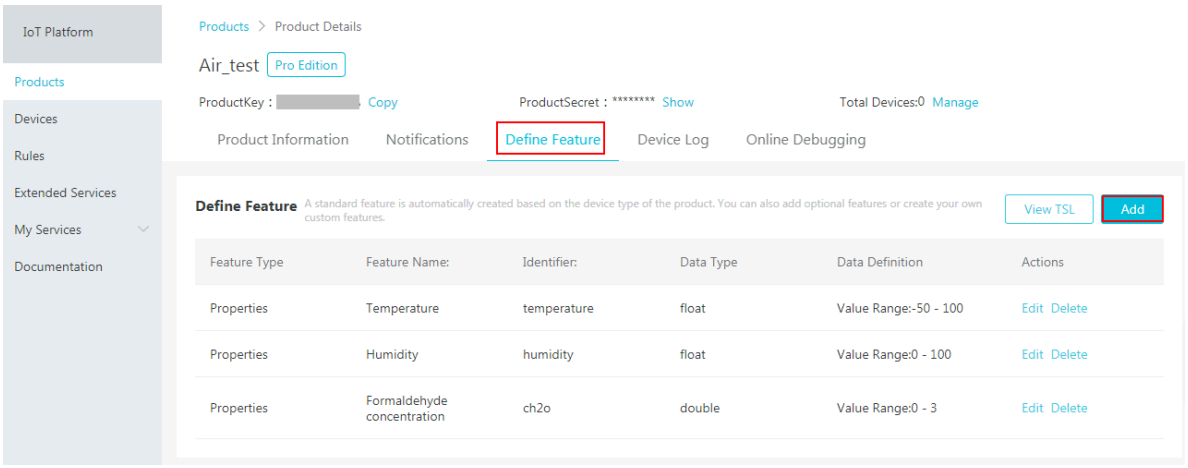


3. Define features for the newly created product.

   On the Product Details page, click Define Feature > Add, and then define properties for the product. For more information, see *What is Thing Specification Language (TSL)?*.

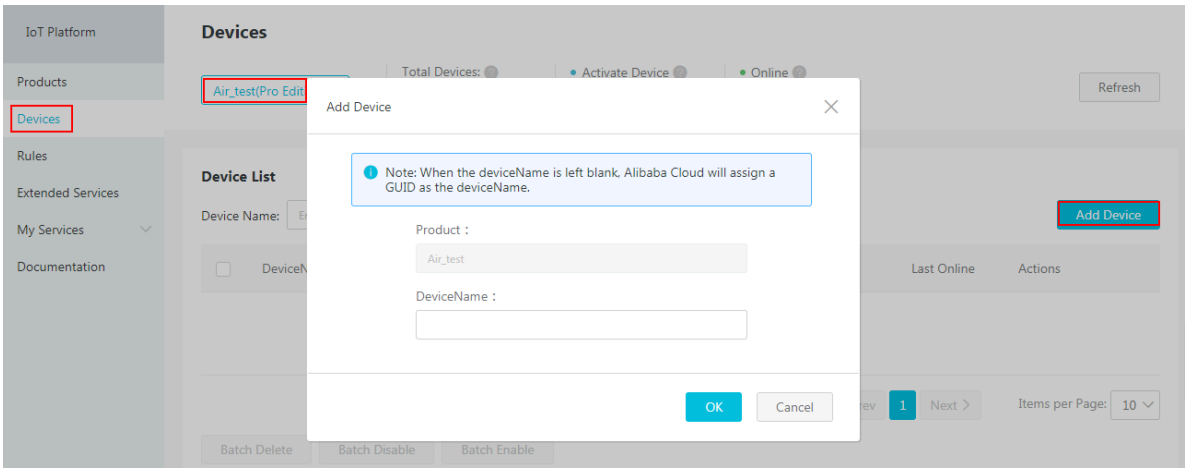
   Table 1-1: Properties required for the indoor air test project

Property name	Identifier	Data type	Value range	Description
Temperature	temperature	float	-50~100	Detected by DHT12.
Humidity	humidity	float	0~100	Detected by DHT12.
Formaldehyde concentration	ch2o	double	0~3	Detected by ZE08.



4. Create a device

On the Devices page, select the name of the newly created product, click Add Device, and then create a device. For more information, see *Create a device*.



Develop the Android Things client

1. Use Android Studio to create an Android Things project, and add the permission for Internet.

```
<uses-permission android:name="android.permission.INTERNET" />
```

2. Add eclipse.paho.mqtt to Gradle file.

```
implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.
0'
```

3. Set that data of DHT12 is read through I2C.

```
private void readDataFromI2C() {

        try {

            byte[] data = new byte[5];
            i2cDevice.readRegBuffer(0x00, data, data.length);

            // check data
            if ((data[0] + data[1] + data[2] + data[3]) % 256 ! =
data[4]) {
                humidity = temperature = 0;
                return;
            }
            // humidity data
            humidity = Double.valueOf(String.valueOf(data[0]) + "."
 + String.valueOf(data[1]));
            Log.d(TAG, "humidity: " + humidity);
            // temperature data
            if (data[3] < 128) {
                temperature = Double.valueOf(String.valueOf(data[2])
 + "." + String.valueOf(data[3]));
            } else {
                temperature = Double.valueOf("-" + String.valueOf(
data[2]) + "." + String.valueOf(data[3] - 128));
            }
```

```
            Log.d(TAG, "temperature: " + temperature);

        } catch (IOException e) {
            Log.e(TAG, "readDataFromI2C error " + e.getMessage(), e
);
        }

    }
```

4. Set that data of Ze08-CH2O is read through UART.

```
try {
                // data buffer
                byte[] buffer = new byte[9];

                while (uartDevice.read(buffer, buffer.length) > 0) {

                    if (checkSum(buffer)) {
                        ppbCh2o = buffer[4] * 256 + buffer[5];
                        ch2o = ppbCh2o / 66.64 * 0.08;
                    } else {
                        ch2o = ppbCh2o = 0;
                    }
                    Log.d(TAG, "ch2o: " + ch2o);
                }

            } catch (IOException e) {
                Log.e(TAG, "Ze08CH2O read data error " + e.
getMessage(), e);
            }
```

5. Connect Alibaba Cloud IoT Platform and the client, and report data.

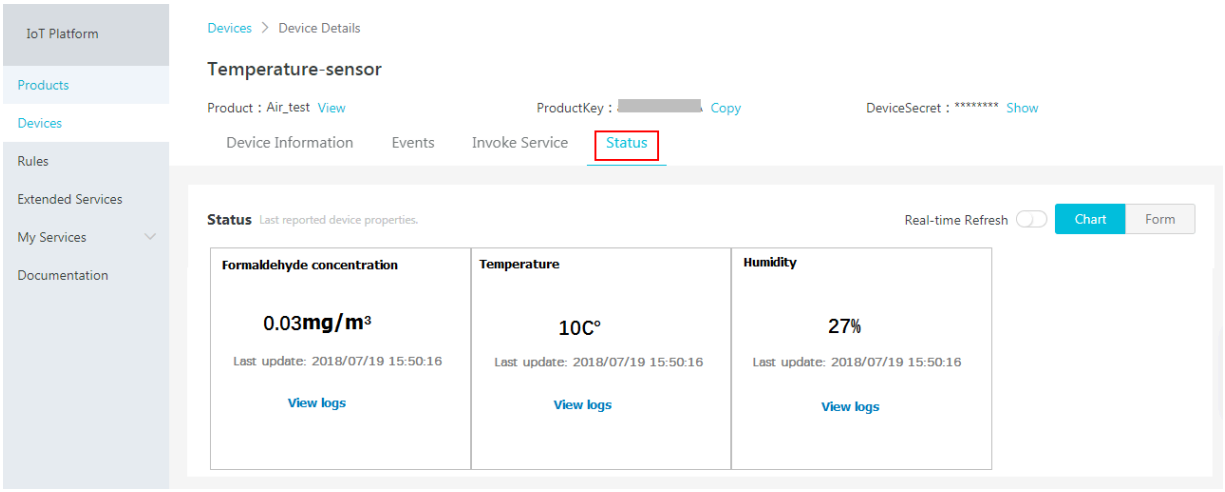
```
/*
Payload format
{
  "id": 123243,
  "params": {
    "temperature": 25.6,
    "humidity": 60.3,
    "ch2o": 0.048
  },
  "method": "thing.event.property.post"
}
*/
MqttMessage message = new MqttMessage(payload.getBytes("utf-8"));
message.setQos(1);

String pubTopYourPc = "/sys/${YourProductKey}/${YourDeviceName}/
thing/event/property/post";

mqttClient.publish(pubTopic, message);
```

View real-time data of the device

After the device is enabled, you can view the real-time data of the device from the Status column on the Device Details page in IoT Platform console.


IoT Platform
Products
Devices
Rules
Extended Services
My Services
Documentation
Devices > Device Details
Temperature-sensor
Product：Air_test View
ProductKey： Copy
DeviceSecret：******** Show
Device Information
Events
Invoke Service
Status
Status Last reported device properties.
Real-time Refresh
Chart
Form
Formaldehyde concentration
0.03mg/m³
Last update: 2018/07/19 15:50:16
View logs
Temperature
10C°
Last update: 2018/07/19 15:50:16
View logs
Humidity
27%
Last update: 2018/07/19 15:50:16
View logs

2 Connect to IoT Platform using MQTT.fx

This article uses MQTT.fx as an example to describe the method for using a third-party MQTT client to connect to IoT Platform. MQTT.fx is a MQTT client that is written in Java language and based on Eclipse Paho. It supports subscribing to messages and publishing messages through topics.

Prerequisites

You have created products and devices in the *IoT Platform console*, and have got the ProductKey, DeviceName, and DeviceSecret of the devices. When you set the connection parameters for MQTT.fx, you will use the values of the ProductKey, DeviceName, and DeviceSecret. See *Create a product (Basic Edition)*, *Create a product (Pro Edition)*, *Create a device*, and *Create multiple devices at a time*for help when creating products and devices.

Procedure

1. Download and install the MQTT.fx software.

   Download the MQTT.fx software for Windows from *http://mqtt-fx.software.informer.com/download/*.

   Download the MQTT.fx software for Mac from *http://macdownload.informer.com/mqtt-fx/*.

2. Open MQTT.fx, and click the settings icon.

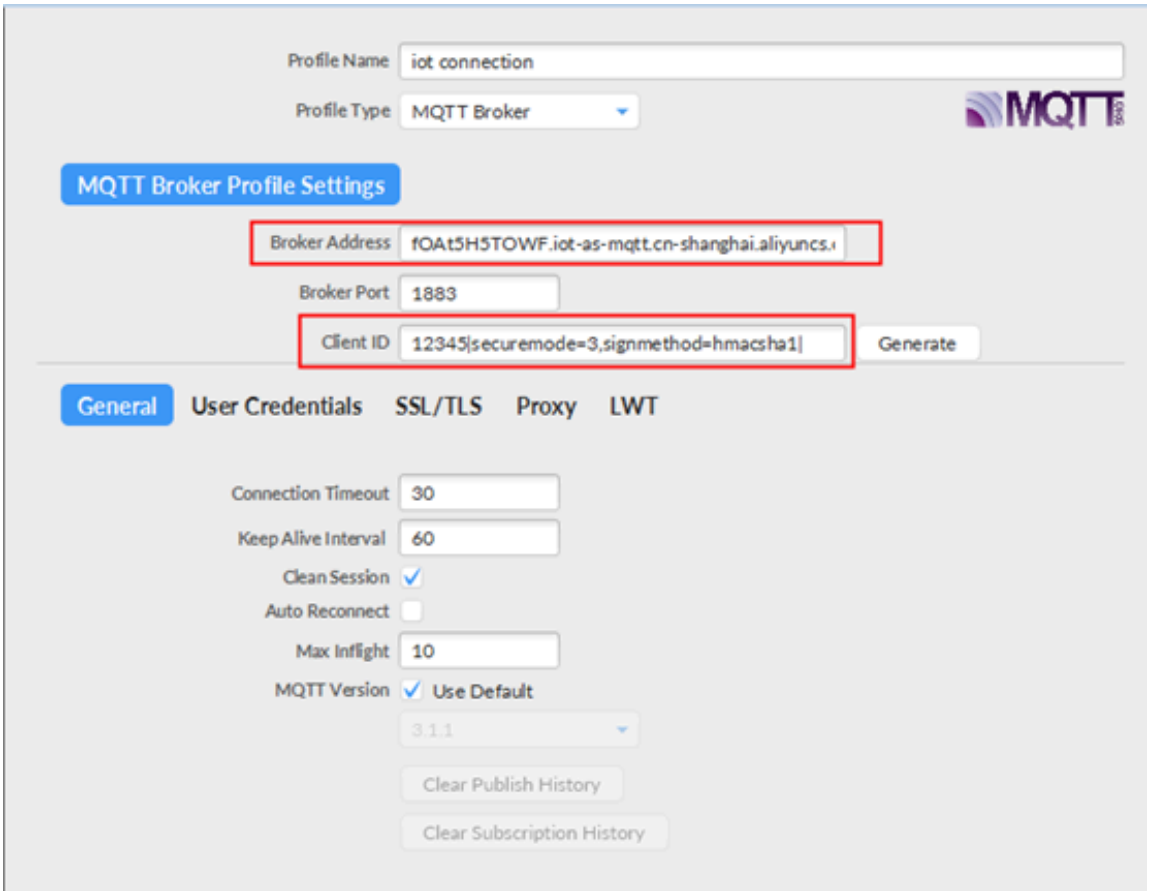
3. Set the connection parameters.

   Currently, two types of connection modes are supported: TCP and TLS. These two modes only differ in settings of Client ID and SSL/TLS.

   The procedure is as follows:

   a. Enter basic information. See the following table for parameter descriptions.

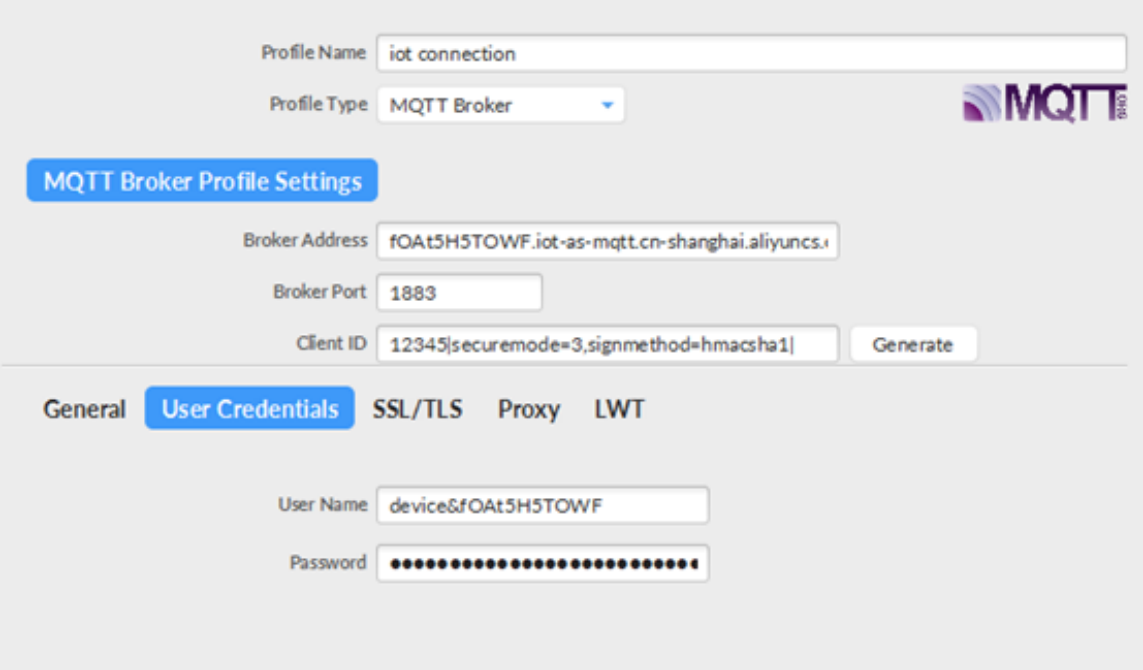
      You can keep the default parameters for General, or set the values according to your needs.



Parameter	Description
Profile Name	Enter a custom profile name.
Profile Type	Select MQTT Broker.
Broker Address	Enter the connection domain in the format of `${YourProductKey}.iot-as-mqtt.${region}.aliyuncs.com`. In this format, variable *`${region}`* indicates the region ID of your IoT Platform service region. For region IDs, see *Regions and zones*. Example: alPUPCoxxxx.iot-as-mqtt.cn-shanghai.aliyuncs.com.
Broker Port	Set to `1883`.

Parameter	Description
Client ID	Enter a value in the format of `${clientId}\|securemode=3,signmethod=hmacsha1\|`. Example: `12345\|securemode=3,signmethod=hmacsha1\|`. The parameters are described as follows: · *`${clientId}`* is a custom client ID. It can be any value within 64 characters. We recommend that you use the MAC address or SN code of the device as the value of clientId. · *`securemode`* is the security mode of the connection. If you use the TCP mode, set it as securemode=3; if you use the TLS mode, set it as securemode=2. · *`signmethod`* is the signature method that you want to use. IoT Platform supports `hmacmd5` and `hmacsha1`. Note: Do not click Generate after you enter the Client ID information.

b. Click User Credentials, and enter your User Name and Password.

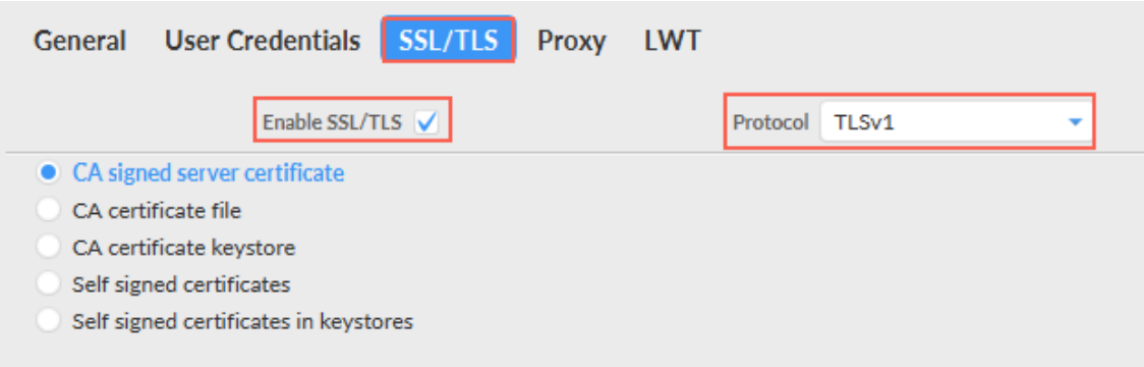


Parameter	Description
User Name	It must be the device name directly followed by the character "&" and the product key. Format: `${YourDeviceName}&${YourPrductKey}`. For example, device&fOAt5H5TOWF.

Parameter	Description
Password	You must enter an encrypted value of the input parameters. IoT Platform provides a *Password Generator* for you to generate one easily. You can also encrypt one by yourself. · Parameters in the password generator: - productKey: The unique identifier of the product to which the device belongs. You can view this information on the device details page in the console. - deviceName: The name of the device. You can view this information on the device details page in the console. - deviceSecret: The device secret. You can view this information on the device details page in the console. - timestamp: (Optional) Timestamp of the current system time. - clientId: The custom client ID, which must be the same as the value of `${clientId}` in Client ID. - method: The signature algorithm, which must be the same as the value of `signmethod` in Client ID. · Generate a password manually: A. Sort and join the parameters. Sort and join the parameters `clientId`, `deviceName`, `productKey`, and `timestamp` in a lexicographical order. (If you have not set a timestamp, do not include timestamp in the string.) Joint string example: `clientId12345deviceNamedevicep roductKeyfOAt5H5TOWF` B. Encrypt. Use the `deviceSecret` of the device as the secret key to encrypt the joint string by the signature algorithm defined in Client ID. Suppose the `deviceSecret` of the device is abc123, the encryption format is `hmacsha1(abc123, clientId12345deviceNamedeviceproductKeyf OAt5H5TOWF)`.

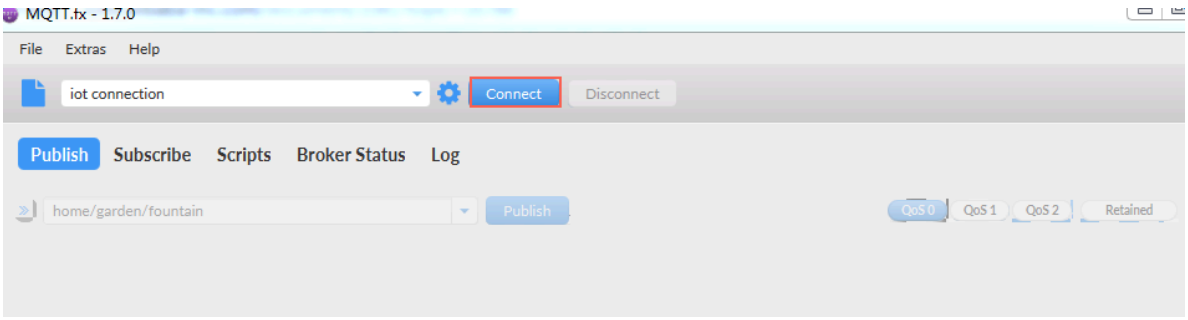
c. If you use TLS connection mode, you are required to set information for SSL/TLS. SSL/TLS settings are not required when the connection mode is TCP.

Check the box for Enable SSL/TLS, and select TLSv1 as the protocol.



d. Enter all the required information, and then click OK.

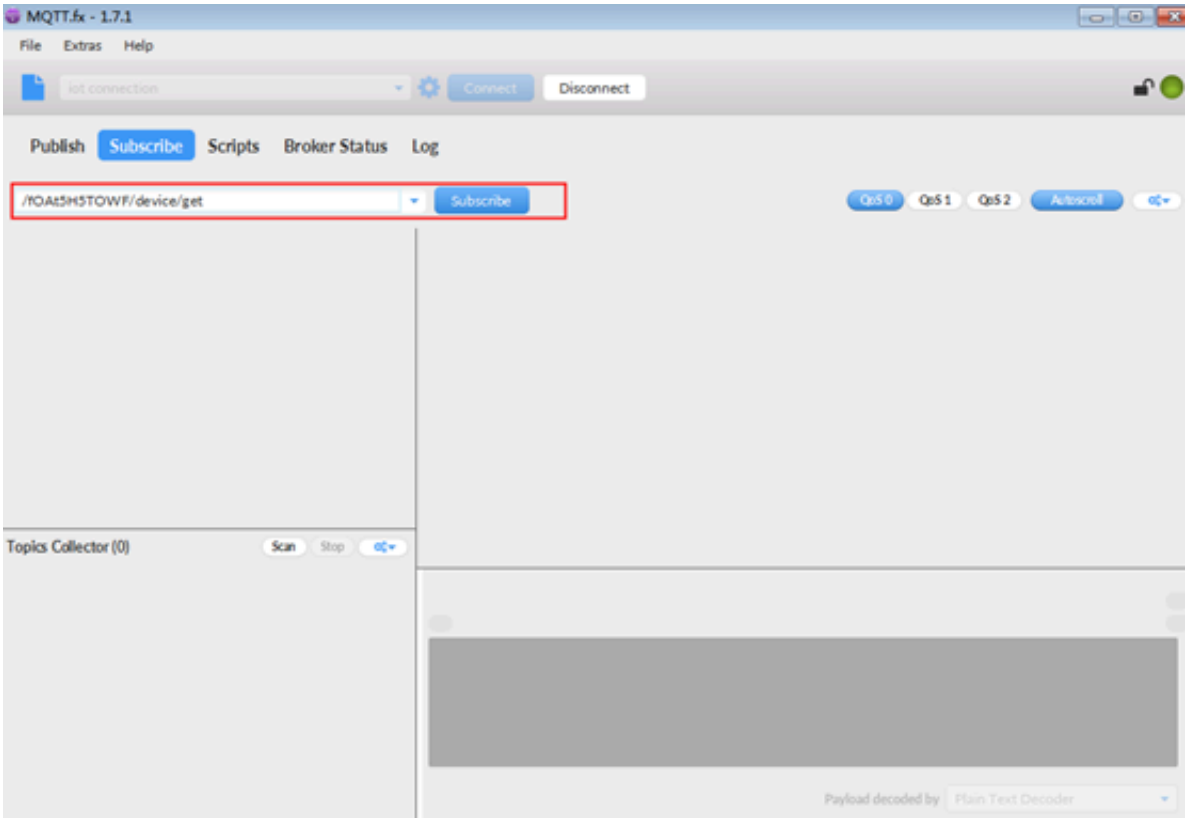
4. Click Connect to connect to IoT Platform.



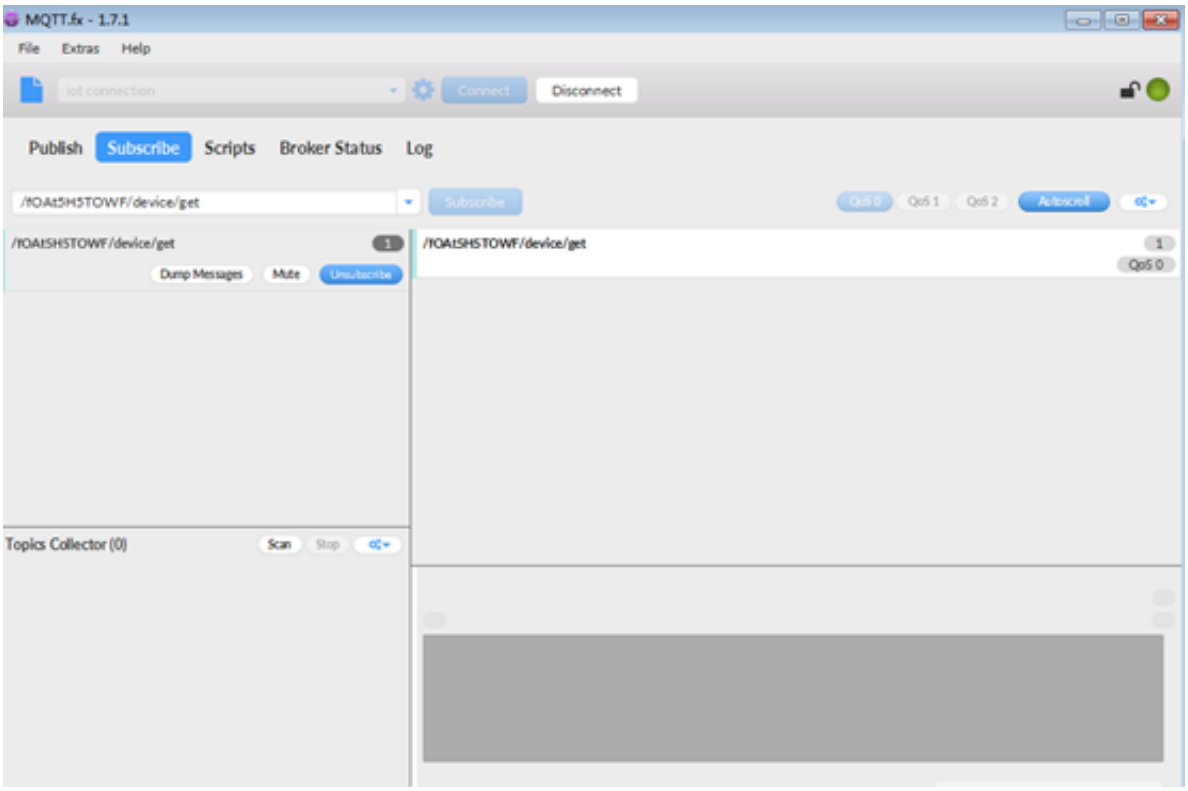
Message communication test

Test whether MQTT.fx and IoT Platform are successfully connected.

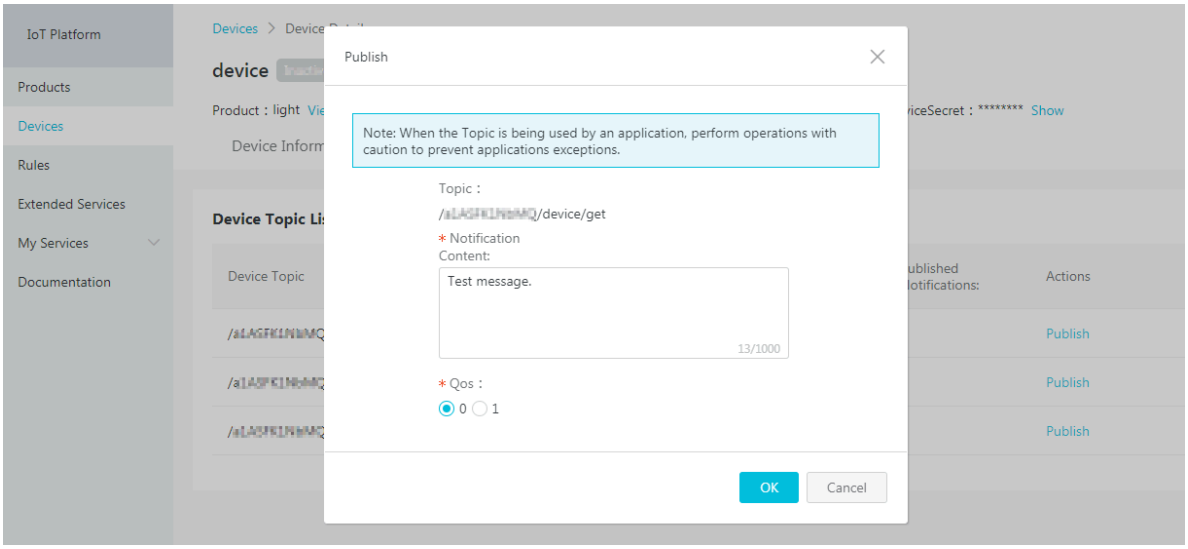
1. In MQTT.fx, click Subscribe.
2. Enter a topic of the device, and then click Subscribe.



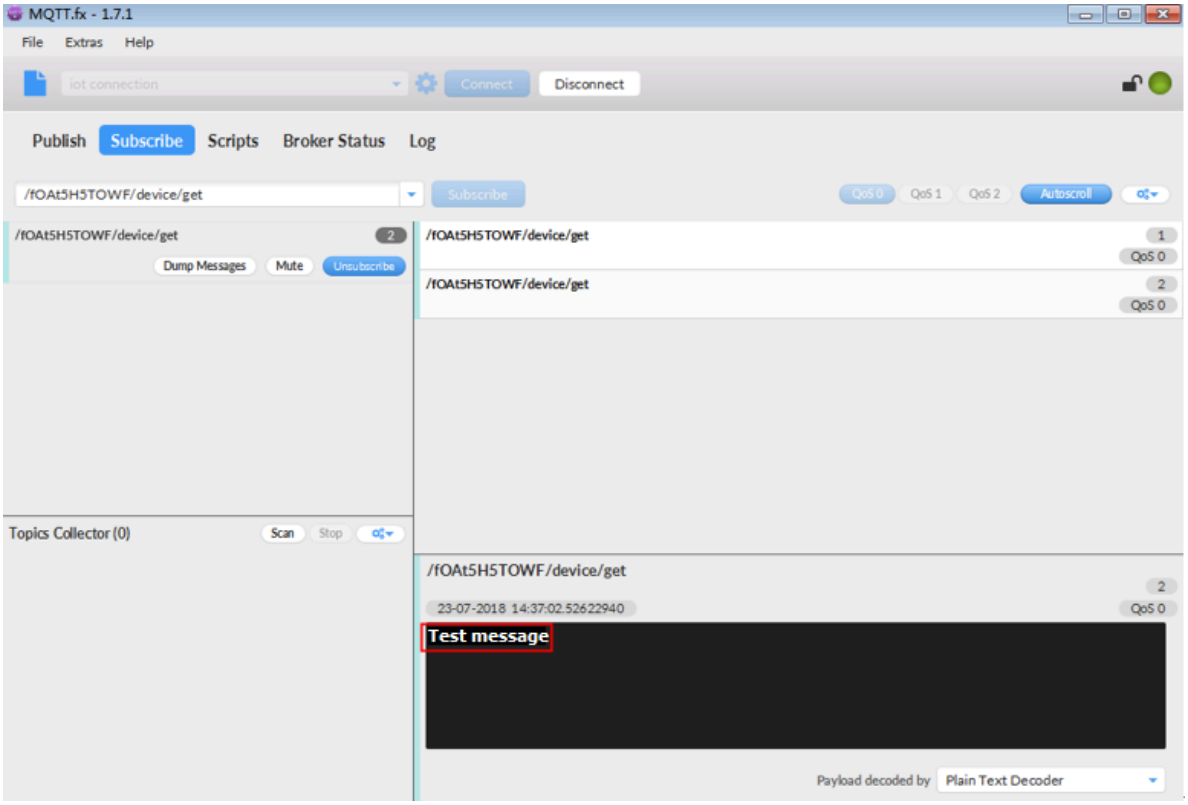
After you have successfully subscribed to a topic, it is displayed in the topic list.



3. In the *IoT Platform console*, in the Topic List of the Device Details page, click the Publish button of the topic that you have subscribed to.
4. Enter message content, and click OK.

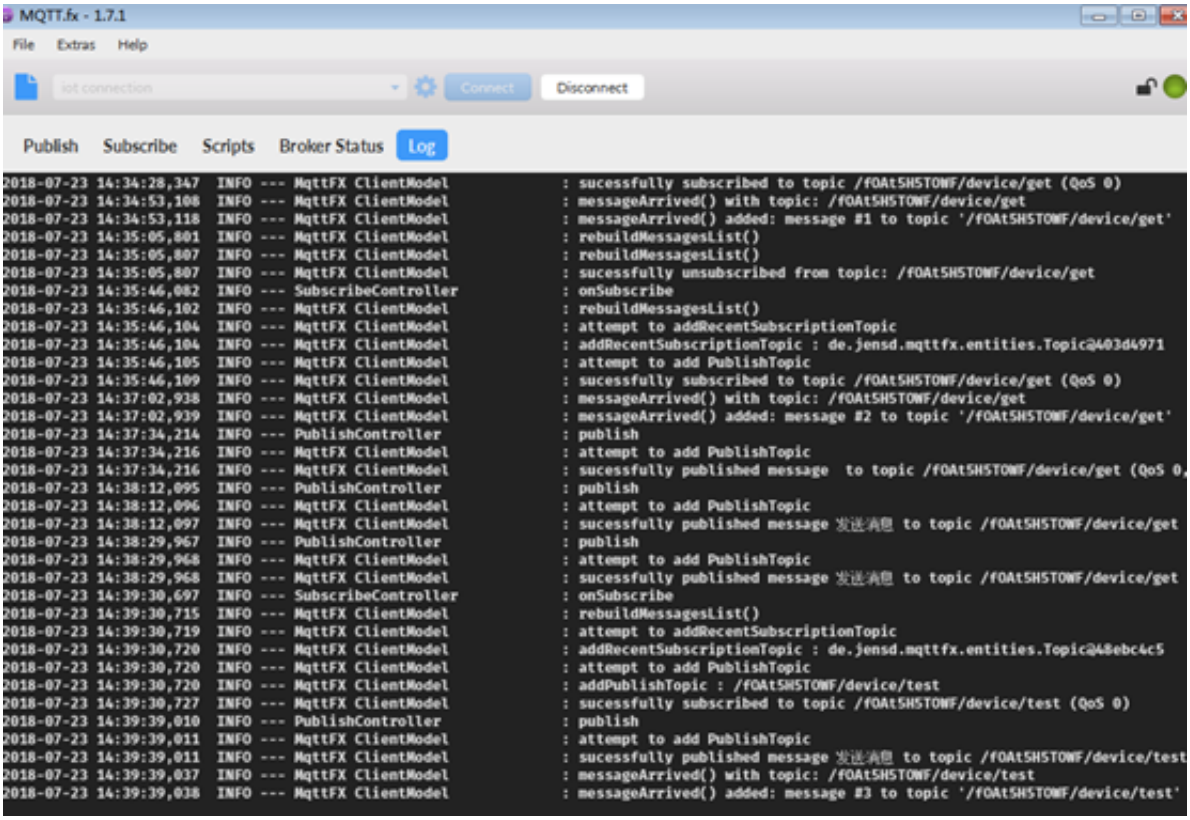


5. Go back to MQTT.fx to check if the message has been received.



View logs

In MQTT.fx, click Log to view the operation logs and error logs.


MQTT.fx - 1.7.1
File
Extras
Help
iot connection
Connect
Disconnect
Publish
Subscribe
Scripts
Broker Status
Log
2018-07-23 14:34:28,347 INFO --- MqttFX ClientModel : sucessfully subscribed to topic /fOAt5H5TOWF/device/get (QoS 0)
2018-07-23 14:34:53,108 INFO --- MqttFX ClientModel : messageArrived() with topic: /fOAt5H5TOWF/device/get
2018-07-23 14:34:53,118 INFO --- MqttFX ClientModel : messageArrived() added: message #1 to topic '/fOAt5H5TOWF/device/get'
2018-07-23 14:35:05,801 INFO --- MqttFX ClientModel : rebuildMessagesList()
2018-07-23 14:35:05,807 INFO --- MqttFX ClientModel : rebuildMessagesList()
2018-07-23 14:35:05,807 INFO --- MqttFX ClientModel : sucessfully unsubscribed from topic: /fOAt5H5TOWF/device/get
2018-07-23 14:35:46,082 INFO --- SubscribeController : onSubscribe
2018-07-23 14:35:46,102 INFO --- MqttFX ClientModel : rebuildMessagesList()
2018-07-23 14:35:46,104 INFO --- MqttFX ClientModel : attempt to addRecentSubscriptionTopic
2018-07-23 14:35:46,104 INFO --- MqttFX ClientModel : addRecentSubscriptionTopic : de.jensd.mqttfx.entities.Topic@403d4971
2018-07-23 14:35:46,105 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:35:46,109 INFO --- MqttFX ClientModel : sucessfully subscribed to topic /fOAt5H5TOWF/device/get (QoS 0)
2018-07-23 14:37:02,938 INFO --- MqttFX ClientModel : messageArrived() with topic: /fOAt5H5TOWF/device/get
2018-07-23 14:37:02,939 INFO --- MqttFX ClientModel : messageArrived() added: message #2 to topic '/fOAt5H5TOWF/device/get'
2018-07-23 14:37:34,214 INFO --- PublishController : publish
2018-07-23 14:37:34,216 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:37:34,216 INFO --- MqttFX ClientModel : sucessfully published message to topic /fOAt5H5TOWF/device/get (QoS 0,
2018-07-23 14:38:12,095 INFO --- PublishController : publish
2018-07-23 14:38:12,096 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:38:12,097 INFO --- MqttFX ClientModel : sucessfully published message 发送消息 to topic /fOAt5H5TOWF/device/get
2018-07-23 14:38:29,967 INFO --- PublishController : publish
2018-07-23 14:38:29,968 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:38:29,968 INFO --- MqttFX ClientModel : sucessfully published message 发送消息 to topic /fOAt5H5TOWF/device/get
2018-07-23 14:39:30,697 INFO --- SubscribeController : onSubscribe
2018-07-23 14:39:30,715 INFO --- MqttFX ClientModel : rebuildMessagesList()
2018-07-23 14:39:30,719 INFO --- MqttFX ClientModel : attempt to addRecentSubscriptionTopic
2018-07-23 14:39:30,720 INFO --- MqttFX ClientModel : addRecentSubscriptionTopic : de.jensd.mqttfx.entities.Topic@48ebc4c5
2018-07-23 14:39:30,720 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:39:30,720 INFO --- MqttFX ClientModel : addPublishTopic : /fOAt5H5TOWF/device/test
2018-07-23 14:39:30,727 INFO --- MqttFX ClientModel : sucessfully subscribed to topic /fOAt5H5TOWF/device/test (QoS 0)
2018-07-23 14:39:39,010 INFO --- PublishController : publish
2018-07-23 14:39:39,011 INFO --- MqttFX ClientModel : attempt to add PublishTopic
2018-07-23 14:39:39,011 INFO --- MqttFX ClientModel : sucessfully published message 发送消息 to topic /fOAt5H5TOWF/device/test
2018-07-23 14:39:39,037 INFO --- MqttFX ClientModel : messageArrived() with topic: /fOAt5H5TOWF/device/test
2018-07-23 14:39:39,038 INFO --- MqttFX ClientModel : messageArrived() added: message #3 to topic '/fOAt5H5TOWF/device/test'

3 Upload temperature and humidity data to DingTalk chatbots

Context

- Scenario:

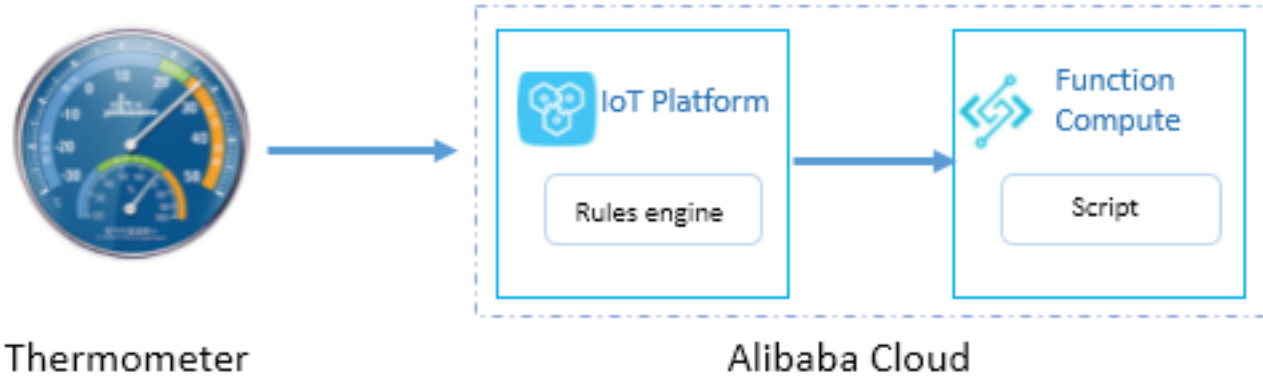
  Upload the data that is collected by the temperature and humidity sensor to DingTalk chatbots.

- Business logic:

  Connect the temperature and humidity sensor to IoT Platform through MQTT . Configure the rules engine to send the temperature and humidity data to the pushData2DingTalk function in Function Compute. The function processes the data and posts the results to the Webhook address of the specified DingTalk chatbot. The DingTalk group can then receive the temperature and humidity data sent by this DingTalk chatbot.

  The data flow diagram is shown in *Figure 3-1: Data flow diagram*.

  Figure 3-1: Data flow diagram



Procedure

1. Configure the DingTalk chatbot.
   a) Log on to the desktop version of DingTalk.
   b) In the upper-right corner of the DingTalk group page, click `...`, and then select ChatBot.

c) Click Add Robot, select Custom, then click Add.

d) Enter a name for the robot, click Next, and then click Finish.

2. Create a function.

a) Activate Alibaba Cloud Function Compute service first.

Function Compute is an event-driven, fully managed computing service. Currently supported languages include Java, Node.js and Python. For more information, see *How to use Function Compute*.

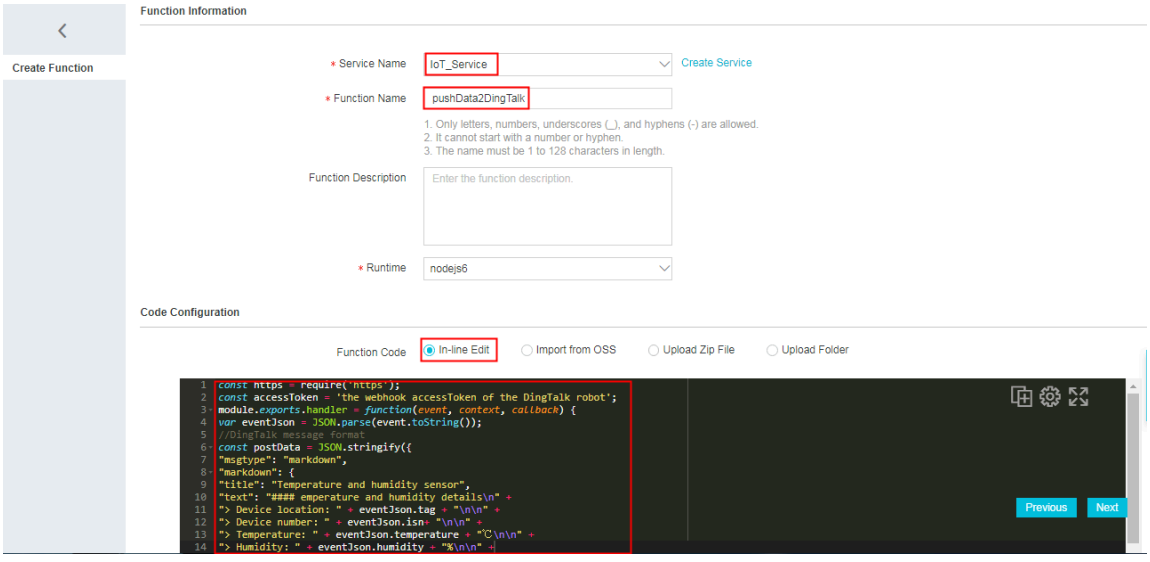
b) Write the function script code. In this example, Node.js is used. The function obtains the device location, device number, temperature, humidity, and the time of recording from IoT Platform, and splices the data according to the specified DingTalk message format. Data will be sent to the Webhook address of the specified DingTalk chatbot using HTTPS Post methods.

c) Log on to the Function Compute console and create a service named IoT_Service.

d) Click Create Function, and select the Empty Function template.

e) Select No Trigger and specify the basic configurations as shown in the following picture.

Figure 3-2: Basic configurations



The function pushData2DingTalk is declared as follows:

```
const https = require('https');
const accessToken = 'Specify the webhook accessToken of the
DingTalk robot';
module.exports.handler = function(event, context, callback) {
```

```
var eventJson = JSON.parse(event.toString());
//DingTalk message format
const postData = JSON.stringify({
"msgtype": "markdown",
"markdown": {
"title": "Temperature and humidity sensor",
"text": "#### Temperature and humidity details\n" +
"> Device location: " + eventJson.tag + "\n\n" +
"> Device number: " + eventJson.isn+ "\n\n" +
"> Temperature: " + eventJson.temperature + "℃\n\n" +
"> Humidity: " + eventJson.humidity + "%\n\n" +
"> ###### " + eventJson.time + " published by [Alibaba Cloud IoT
Platform](https://www.aliyun.com/product/iot) \n"
},
"at": {
"isAtAll": false
}
});
const options = {
hostname: 'oapi.dingtalk.com',
port: 443,
path: '/robot/send? access_token=' + accessToken,
method: 'POST',
headers: {
'Content-Type': 'application/json',
'Content-Length': Buffer.byteLength(postData)
}
};
const req = https.request(options, (res) => {
res.setEncoding('utf8');
res.on('data', (chunk) => {});
res.on('end', () => {
callback(null, 'success');
});
});
// When an exception returns
req.on('error', (e) => {
callback(e);
});
// Write the data
req.write(postData);
req.end();
};
```

3. **Configure IoT Platform.**
   a) **In the IoT Platform console, select Products and then create a product for the temperature and humidity sensor.**
   b) **On the Topic Categories tab page of the product details page, create a topic category `/productKey/${deviceName}/user/data` whose Device Operation Authorizations is Publish.**
   c) **On the Define Feature tab page, define two properties: temperature and humidity.**
   d) **Select Devices and then register a device.**

e) Click View next to the device name. On the Device Tag tab, click Add to add two device tags.

Tag Key	Tag Value	Description
tag	Room 007S, 3rd Floor, Building 2, Cloud Town	Device location
deviceISN	T20180102X nbKjmoAnUb	Device number

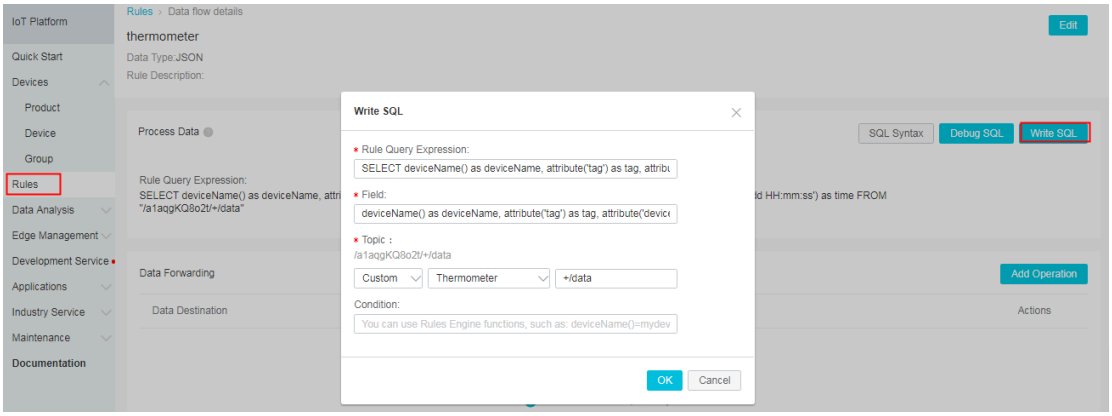
f) Select Rules to create and enable a rule. A complete rule contains three parts: basic information, data processing SQL, and data forwarding. You can specify multiple forwarding actions for a rule.

A. Configure the data processing script.

The rules engine supports *SQL statements*.

The device name (deviceName) and attributes (tag and deviceISN) are required.

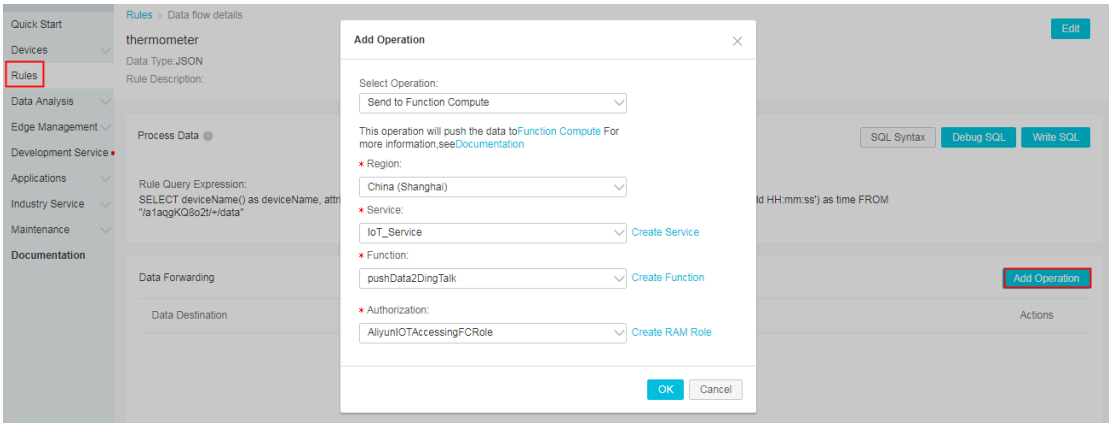
This SQL is to obtain the temperature and humidity values from the payload of the messages that are sent from the temperature and humidity sensor.



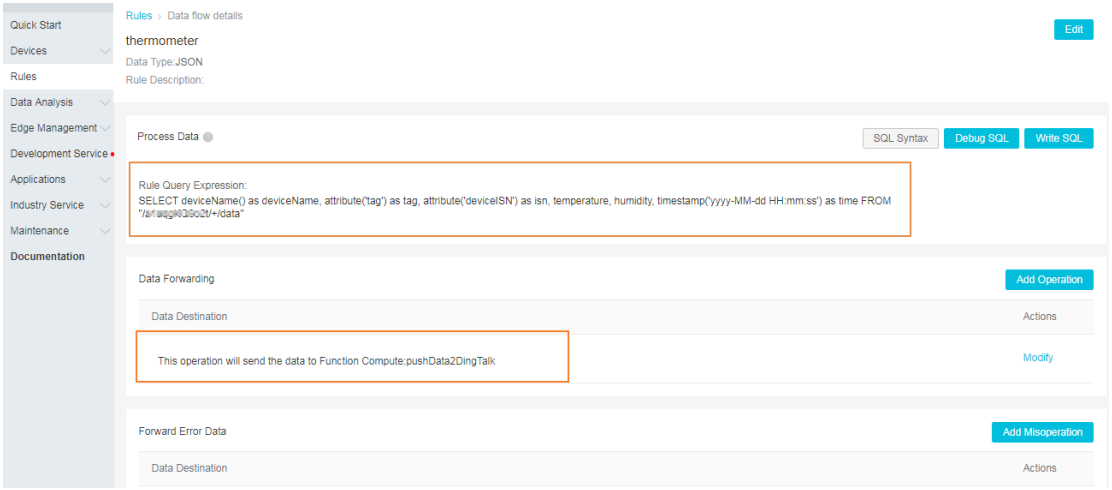
The SQL statements are as follows:

```
SELECT
deviceName() as deviceName,
attribute('tag') as tag,
attribute('deviceISN') as isn,
temperature,
humidity,
timestamp('yyyy-MM-dd HH:mm:ss') as time
FROM
"/Specify the productKey/+/data"
```

B. Configure the data forwarding action.



The complete rule is as follows:



g) On the rules page, click Enable to enable the rule.

4. Temperature and humidity sensor.

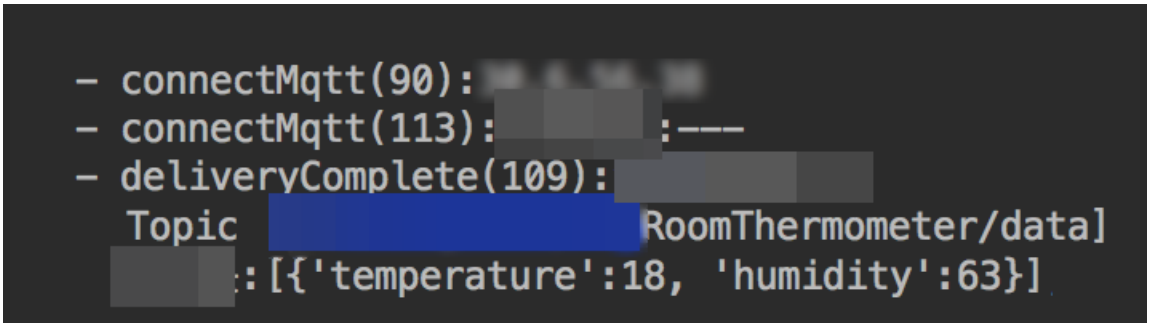
   To facilitate testing, a Node.js program is used to simulate the temperature and humidity sensor and send the temperature and humidity data. The *aliyun-iot-mqtt library* is used in this example. The complete code is as follows:

```
const mqtt = require('aliyun-iot-mqtt');
const client = mqtt.getAliyunIotMqttClient({
productKey: "Specify the productKey",
deviceName: "Specify the deviceName",
deviceSecret: "Specify the deviceSecret"
});
const topic = 'Specify the topic with forwarding actions';
const data = {
temperature: 18,
humidity: 63,
};
client.publish(topic, JSON.stringify(data));
```

5. DingTalk robot receives messages.

a) The program sends the temperature and humidity data.

```
$ npm install
$ node demo.js
```



b) DingTalk robot receives the data, and sends a message to the DintTalk group.