**IoT Device Access** 

## **Best Practices**

 Issue
 1.0

 Date
 2024-12-31





HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

#### Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

#### **Trademarks and Permissions**

NUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

#### Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

### **Contents**

1 Introduction	1
2 Device Access	6
2.1 Developing an MQTT-based Simulated Smart Street Light Online	6
2.2 Developing a Smart Street Light Using NB-IoT BearPi	25
2.3 Developing a Smart Smoke Detector Using NB-IoT BearPi	41
2.4 Connecting and Debugging an NB-IoT Smart Street Light Using a Simulator	
2.5 Developing a Protocol Conversion Gateway for Access of Generic-Protocol Devices	83
2.6 Connecting a Device That Uses the X.509 Certificate Based on MQTT.fx	86
2.7 Connecting to IoTDA Based on the BearPi-HM_Nano Development Board and OpenHarmo	ony 3.0106
2.8 Testing MQTT Performance Using JMeter	115
3 Device Management	122
3.1 Automatically Adjusting Air Conditioner Temperature Through Device Shadow	
3.2 Managing Indoor Air Conditioners Using Custom Topics	126
3.3 Performing OTA Firmware Upgrade for MQTT Devices	129
4 Data Forwarding	141
4.1 Pushing Metric Data to DMS for Kafka	141
5 Device Linkage	144
5.1 Triggering Alarms and Sending Email or SMS Notifications	144
5.2 Automatic Device Shutdown Upon High Temperature	152
5.3 Automatically Opening the Window upon High Gas Concentration	157
5.4 Monitoring Device Status Changes and Sending Notifications	

# Introduction

After you have a basic understanding of IoTDA, you may wonder how the platform can create value for you, in which business scenarios the platform can be used, and how you can access the platform. The following scenario examples are used to describe the service process and product model as well as the platform functions and benefits.

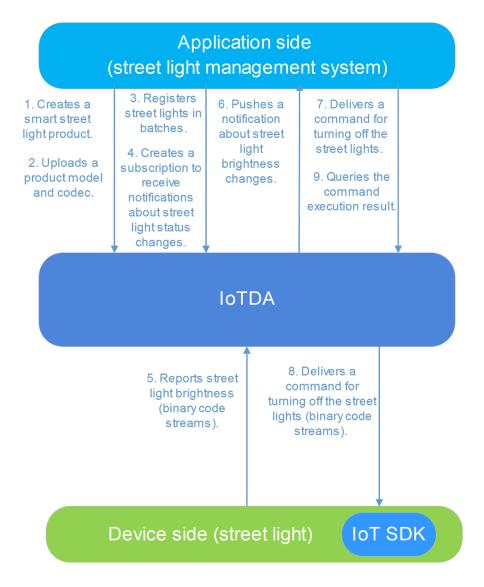
#### Scenario Example: Smart Street Light

The street light management system connects to the platform to monitor street lights that are integrated with the NB-IoT module and turn on/off these street lights.

In this scenario, a device interacts with the platform using LwM2M. The application subscribes to device change notifications on the platform and delivers commands to the device.

**Key points**: product model, codec, subscription and push, property reporting, and command delivery

Reference: Developing a Smart Street Light Using NB-IoT BearPi



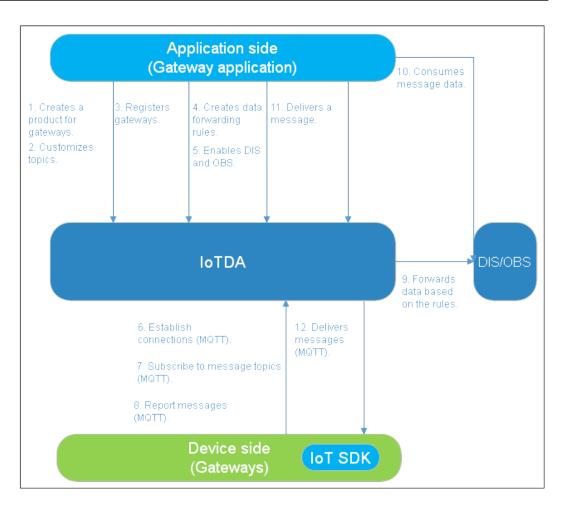
#### Scenario Example: Smart Gateway

Using gateways, you can manage existing devices under the gateways without migration and add new devices to the gateways.

In this scenario, devices (gateways) interact with the platform using the MQTT protocol. You can create topics on the product details page of the console and create data forwarding rules using application APIs or the console to forward device messages to other Huawei Cloud services for consumption.

**Key points**: product model, message reporting, message delivery, MQTT, data forwarding rules, and topic customization

Reference: Managing Indoor Air Conditioners Using Custom Topics



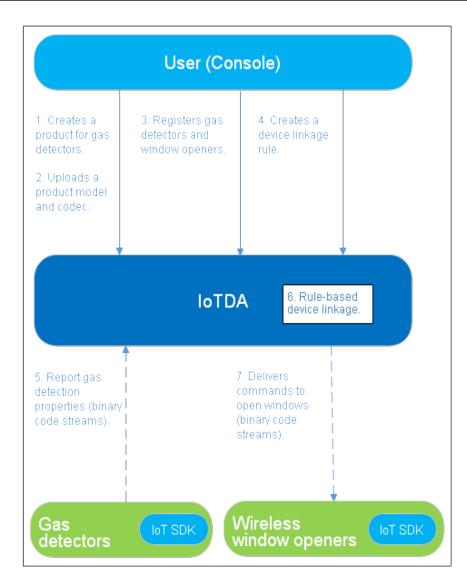
#### Scenario Example: Smart Home Gas Detection

If a gas detector detects excessive gas, the wireless window opener associated with the gas detector automatically opens the window for ventilation.

In this scenario, devices interact with the platform over MQTT to report properties. You can create device linkage rules on the console or by calling APIs to convert the reported properties into commands and deliver the commands to other specific devices.

**Key points**: product model, property reporting, command delivery, MQTT, and device linkage rules

For details about this scenario, see **Automatically Opening the Window upon High Gas Concentration**.



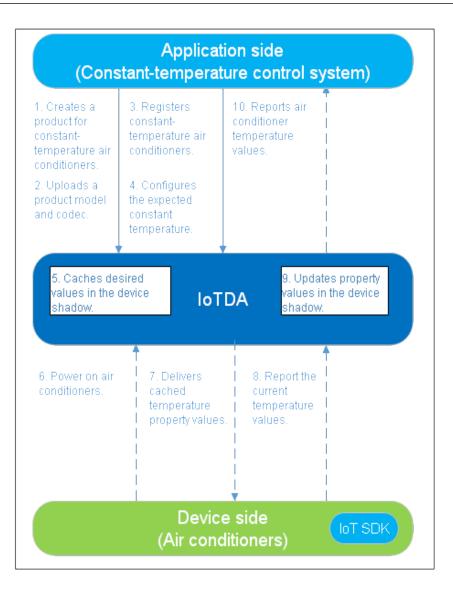
#### Scenario Example: Constant-Temperature Air Conditioner

Using a constant-temperature control system, you can adjust the default temperature of air conditioners (regardless of whether they are powered on). After being powered on, the air conditioners automatically run at the default temperature.

In this scenario, the application or console delivers property pending commands to offline devices. If devices go online and report different properties, the console automatically delivers commands to modify device properties until they are the same as the desired values.

**Key points**: product model, codec, device shadow, property reporting, and property modification

Reference: Automatically Adjusting Air Conditioner Temperature Through Device Shadow



# **2** Device Access

### 2.1 Developing an MQTT-based Simulated Smart Street Light Online

#### **Scenarios**

This section uses a smart street light as an example to describe how to use device simulators to experience data reporting and command delivery. You can use MQTT.fx (recommended) or MQTT\_Simulator.

A street light reports the light intensity (luminance) in JSON format. The command (switch) can be used to remotely control the street light status.

#### Prerequisites

- You have registered a Huawei Cloud account.
- You have subscribed to IoTDA. If you have not subscribed to the service, go to the IoTDA service page, and click **Access Console** to subscribe to the service.

#### **Service Flow**

The MQTT.fx simulator is used as an example to describe data reporting and command delivery.

- Step 1 Create an MQTT product.
- **Step 2 Develop a product model.** Define a product model to develop a street light that supports light intensity reporting and status control commands.
- Step 3 Register an MQTT device to experience data reporting.
- **Step 4 Perform connection authentication.** Use MQTT.fx to activate the device registered on IoTDA.
- **Step 5 Report data.** Use MQTT.fx to report data to IoTDA.
- **Step 6 Deliver a command** on the console to remotely control a device.

----End

#### **Creating a Product**

**Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.

Figure 2-1 Creating a product

-		elance voi hour v O Running					R Details & Modity ∙
Overview Products		Products					Description 📋 Quick Links
Devices	~						
Rules	~	Description					
08M	~	On the IoT platform, a product is a collection of devi		o develop a product model (a profile). The product model i	informs the piotform of the prope	tion and commands that are connected by devices. Yes	u can also define product model defails based on the
Resource Spaces		access protocol and data format. Learn more	anage and control the Genice on the planamit, you need a	o develop a product model (a prome). The product model i	invine the patronic of the prope	nes and commands may are supported by derices. Too	a can also denne product model detaile based on the
Documentation	c	1 Create Products	2 Defining Product I	Models	3 Register Devices		4 Device-side development
		A product is a set of devices that have the same	a canabilities or fashuras Dafina product moda	is to describe the capabilities and features of devices.	The registered desig	e obtains the identity information required to connect	Integrates device SDKs to connect devices to
		Product Development Guide	capabilities of resisters.		to the platform.		Davice Development Guide
	<	Create Product Delete					
		<ul> <li>Select a property or enter a keyword.</li> </ul>					0.0
		Product Name	Product ID \ominus	Resource Spaces $\ominus$	Device Type	Protocol  Created 🖯	Operation
			Bullin Car Tan Million Person	Defaultion_Midtood	8407	M077 Aug 28, 2004 20 88 16 087 48	Wew Copy More ~

**Step 2** Create a product whose protocol type is MQTT and device type is **StreetLamp**, set parameters as prompted, and click **OK**.

 $\times$ 

#### Figure 2-2 Creating a product - MQTT

Create Product

eroutorroudot		
* Resource Space  ?		~
	To create a new resource spa details page.	ace, you can go to the instance
★ Product Name		
Protocol 🧿	MQTT	~
🗙 Data Type	JSON	~
Device Type Selection	Standard profile	Custom
★ Device Type ⑦		
Advanced Settings $\land$	Custom Product ID   Descrip	tion
Product ID 💿		
Description		
		0/128 🛷
		Cancel OK

#### ----End

#### **Developing a Product Model**

- **Step 1** Click the created product. The product details page is displayed.
- **Step 2** On the **Model Definition** tab page, click **Customize Model** to add services of the product.

#### Figure 2-3 Custom model - MQTT

IoTDA Instances /	Yoducts / aaasssss	
<	Registered devices: 0	Culck Links
Basic Information	Codec Deployment Online Debugging Topic Management	
Product Det	a	
Product Name	₫ Persure Soure	
Device Type	Protocol MQTT	
Data Type	ton Created	
Industry	- Description - d	
	Aproduct model describer product desider and service capability. The device product medic larger major time. Larger         Water Water Water Visioner         Visioner	he data.

#### Step 3 Add the BasicData service.

Figure 2-4 Adding a service - BasicData

1. On the Add Service page, specify Service ID, Service Type, and Description, and click OK.

Add Service			>
* Service ID	BasicData		
Service Type	BasicData		0
Description	Reports street light data.		
		26/128 //	
		Cancel	ОК

2. In the **BasicData** service list on the right, click **Add Property**, enter related information, and click **OK**.

Add Property	×
* Property Name	luminance
Description	light intensity.
	16/128 🥠
* Data Type	Integer v
* Access Permissions	Read Write
★ Value Range	0 – 65535
Step	
Unit	
	Cancel

#### Figure 2-5 Adding a property - luminance

**Step 4** Add the **LightControl** service.

- 1. Click **Add Service** on the **Model Definition** tab page, set parameters as prompted, and click **OK**.
  - Service ID: Enter LightControl.
  - Service Type: You are advised to set this parameter to the same value as Service ID.
  - Description: Enter Controls the street light.
- 2. Choose LightControl, click Add Command, and enter the command name Switch.

Figure 2-6 Addi	ng a command -	Switch
-----------------	----------------	--------

Add Command		$\times$
* Command Name	Switch	
Command Parameters	Add Command Parameter	
	Parameter Name $\Leftrightarrow$ Data Type $\Leftrightarrow$ Description $\Leftrightarrow$ Operation	
	No table data available.	
	No Command Parameters data available. Add Command Parameter first.	
Response Parameters	Add Response Parameter	
	Parameter Name \ominus Data Type 😔 Description 😔 Operation	
	No table data available.	
	No Response Parameters data available. Add Response Parameter first.	
		_
	Cancel	

3. Click Add Command Parameter, enter related information, and click OK.

Add Parameter		
★ Parameter Name	value	
Description		
		0/128 4
★ Data Type	String	~
★ Length	15	
Enumerated Values	ON,OFF	
		6/1,024 4
		Cancel

Figure 2-7 Adding a command parameter - -value



#### **Registering a Device**

**Step 1** On the IoTDA console, choose **Devices** > **All Devices** in the navigation pane, and click **Register Device** in the upper right corner.

Figure 2-8 Registering a device

<ul> <li>Usermessa;</li> </ul>	periore at	fandert frankert	<ul> <li>O Running</li> </ul>							दि Details & Modif
view	All D	evices Total de	vices 0   • Activated devi	ces 0   🗢 Online de	wices 0					🗂 Quick
ucts	D	evice List Br	atch Registration	Batch Update	Batch Deletion Batch	Add Devices To Group File Upload	28			
All Devices		Register Device	Delete Unit	eeze Frees	re .					
Roups		Q. Search by node	ID by default.							(Advanced Search v)
Policies		Status O	Device Nan	10 8	Node ID 😔	Device ID ()	Resource Space ()	Product @	Node Type 😣	Operation
loftware/Firmware Jpgrades										
Device Certificates										
Device Praxy						<u> </u>	, T->			
Self-Registration Template	<									
Sustem						No table data	a available.			
Authentication						No Devices data available	a. Register Device first.			
s ~						Register I	Device			
· · ·										

**Step 2** Set the parameters as prompted and click **OK**.

Parameter	Description
Resource Space	Ensure that the device and its associated product belong to the same resource space.
Product	Select a corresponding product.

Parameter	Description
Node ID	Customize a unique physical identifier for the device. The value consists of letters and digits.
Device Name	Customize the device name.
Authenticatio n Type	Select <b>Secret</b> .
Secret	If you do not set this parameter, IoTDA automatically generates a value.

#### Figure 2-9 Registering a device - MQTT

* Resource Space (	Carlandings, M. Sanat	~
+ Product	Test_1	~
	Mqtt devices have subscribed to the platform preset topic by topics 🕑	default. Subscribed
* Node ID 🧿	Test_1	
Device ID 💿		
Device Name		
Description		
		0/2,048 //
Authentication Type ⑦	Secret X.509 certificate	
Secret		Ø
Confirm Secret		1

**Step 3** After the device is registered, the platform automatically generates a device ID and secret. Save the device ID and secret for device access.

Download

#### Figure 2-10 Device registered

Device Registered	×
The system automatically allocated the follow	ving device information.
For security reasons, the secret will not be you forget the secret, click Reset Secret of secret.	
Device ID	
Device Secret	
Next, you can use the IoT Device SDK to con SDK Development Guide (2)	nnect devices to the platform.

----End

#### **Performing Connection Authentication**

Use MQTT.fx to activate the device registered on IoTDA.

- Step 1 Download MQTT.fx (64-bit OS) or MQTT.fx (32-bit OS) and install it.
- **Step 2** Go to the **IoTDA client ID generator page**, enter the device ID and secret generated after registering a device to generate connection information (including **ClientId**, **Username**, and **Password**).

	IOT MQTT Client ID Generator Client IDs. For details about the algorithm of device connection authentication, click the button below.			
Learn More				
Device ID				
5fe	*******			
Device Secret				
1 33333333399				
	Generate			
Client ID	5/6			
Sername 5f6.000000000000000000000000000000000000				
Password	814000000000000000000000000000000000000			

Parame ter	Mand atory	Туре	Description
ClientId	Yes	String(2 56)	<ul> <li>The value of this parameter consists of a device ID, device type, password signature type, and timestamp. They are separated by underscores (_).</li> <li>Device ID: A device ID uniquely identifies a device and is generated when the device is registered with IoTDA. The value usually consists of a device's product ID and node ID which are separated by an underscore (_).</li> <li>Device type: The value is fixed at 0, indicating a device ID.</li> <li>Password signature type: The length is 1 byte, and the value can be 0 or 1.</li> <li>O: The timestamp is not verified using the HMAC-SHA256 algorithm.</li> <li>Timestamp: The UTC time when the device was connected to IoTDA. The format is YYYYMMDDHH. For example, if the UTC time is 2018/7/24 17:56:20, the timestamp is 2018072417.</li> </ul>
Userna me	Yes	String(2 56)	Device ID.
Passwo rd	Yes	String(2 56)	Encrypted device secret. The value of this parameter is the device secret encrypted by using the HMAC-SHA256 algorithm with the timestamp as the key. The device secret is returned by IoTDA upon successful device registration.

Each device performs authentication using the MQTT CONNECT message, which must contain all information of the client ID. After receiving a CONNECT message, IoTDA checks the authentication type and password digest algorithm of the device.

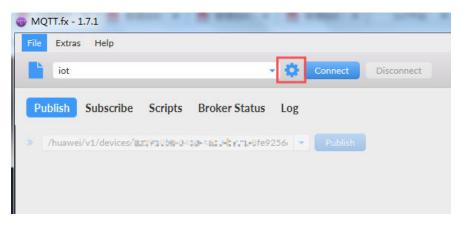
The generated client ID is in the format *Device ID\_0\_0\_Timestamp*. By default, the timestamp is not verified.

- If the timestamp needs to be verified using the HMAC-SHA256 algorithm, the platform checks whether the message timestamp is consistent with the platform time and then checks whether the password is correct.
- If the timestamp does not need to be verified using the HMAC-SHA256 algorithm, the timestamp must also be contained in the CONNECT message,

but the platform does not check whether the time is correct. In this case, only the password is checked.

If the authentication fails, the platform returns an error message and automatically disconnects the MQTT connections.

**Step 3** Open the MQTT.fx tool and click the setting icon.



Step 4 Configure authentication parameters and click Apply.

Edit Connection Profiles			– 🗆 X
iot local mosquitto		Name iot le Type MQTT Broker	
	Bro	tings           ddress         110000000           er Port         1883           lient ID         s8rrrr000000000000000000000000000000000	Generate
	Usi	als SSL/TLS Proxy LWT	
+ -	Revert		Cancel OK Apply

Parameter	Description
Broker Address	Enter the <b>device access address</b> (domain name) obtained from the IoTDA console. For devices that cannot be connected to the platform using a domain name, run the <b>ping</b> <i>Domain name</i> command in the CLI to obtain the IP address. The IP address is variable and needs to be set using a configuration item.
Broker Port	The default value is <b>1883</b> .

Parameter	Description
Client ID	Enter the device client ID obtained in 2.
User Name	Enter the device ID obtained in <b>2</b> .
Password	Enter the encrypted device secret obtained in 2.

**Step 5** Click **Connect**. If the device authentication is successful, the device is displayed online on the platform.

Figure 2-11 Device online status

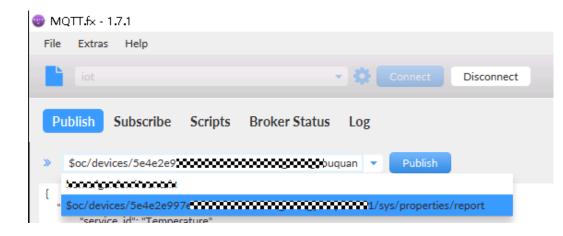
< 👩 usemessage	😧 🔤 🔤 🔤 🔤 🔤 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓 🖓							
Overview	All Devices Total devices 2	All Devices Total devices 21  Activated devices 11  Online devices 1 C Quiot Linus C Quiot Linus						
Products Devices ^	Device List Batch Re	Device List Batch Registration Batch Update Batch Deletion Batch Add Devices To Group File Uploads						
All Devices	Register Device Dowls Universe Freeze							
Groups	Q Search by node ID by de	stault.						(Advanced Search V)
Policies Software/Firmware	Status Θ	Device Name \varTheta	Node ID	Device ID (e)	Resource Space $\Theta$	Product @	Node Type \varTheta	Operation
Upgrades	Online	Held .	-	Ball/1048790108440701.1	Defaultion_ICINers	Strains	Drack consider	View Debug More ~

----End

#### **Reporting Data**

Use MQTT.fx to report data to IoTDA. If a device reports data through the MQTT channel, the data needs to be sent to a specific topic in the format **\$oc/devices/** *{device\_id}/sys/properties/report*. For devices that each has a different secret, set *device\_id* to the device ID returned upon successful device registration.

Step 1 Enter the API address in the format of "\$oc/devices/{device\_id}/sys/properties/ report", for example, \$oc/devices/5e4e2e92ac-164aefa8fouquan1/sys/ properties/report.



**Step 2** Enter the data to report in the blank area in the middle of the tool and click **Publish**.

 Table 2-1
 Service data list

Field Name	Mandat ory	Data Type	Description
services	Yes	List <servicepr operty&gt;</servicepr 	Service data list. (For details, see the <b>ServiceProperty</b> structure below.)

Table 2-2 ServiceProperty structure

Field Name	Manda tory	Data Type	Description
service_id	Yes	String	Service ID.
propertie s	Yes	Object	Service properties, which are defined in the product model associated with the device.
eventTim e	No	String	UTC time when the device reports data. The format is yyyyMMddTHHmmssZ, for example, <b>20161219T114920Z</b> .
			If this parameter is not carried in the reported data or is in incorrect format, the time when IoTDA receives the data is used.

#### Example request

```
{
    "services": [{
        "service_id": "BasicData",
        "properties": {
            "luminance": 30
        }
     }
}
```

**Step 3** Check whether the device successfully reports data on the device details page.

	Devices / Device Details						
< 1 On	nine ③					C	Quick Links
Device Info Clou	ud Run Logs Cloud Delivery Der	vice Shadow Message Trace Device Mo	nitoring Child Devices Tags Groups				
Device Name		Resource Space	Selacitya, Skilt-of	Product			
Device ID	merchanizer and 2	Node ID		Authentication Type	Secret Reset Secret		
Node Type	Directly connected	Firmware Version		Software Version			
Description		Registered	Stept 23, 2024 10:10:44 (047-08.08)	Activated	Stept 23, 2024 17 12:00 OM7-00.00		
Last Online	Nam 12, 2024 19 19 19 00 007-00 00	MQTT	View				
		Connection					
		Parameter					
Product Model	Data					View All Properties	
	orted by the device based on the product model ad property name is not contained in the product		abols (\$), or empty char (the hexadecimal ASCII code is 00), the property data ca	nnot be updated.		Channelland	00
Enter the service	e name. Q	Latest Reported Time	047-08.00			luminance	XIQ
BasicData							
LightControl		luminance luminance					
		30					
		Total Records: 1 16 V C 1 >					

Figure 2-12 Viewing reported data - MQTT



#### **Delivering a Command**

Deliver a command on the console to remotely control a device.

- Step 1 In the navigation pane, choose Devices > All Devices, locate the target device, and click View to access its details page.
- **Step 2** Click the **Commands** tab and deliver a synchronization command.

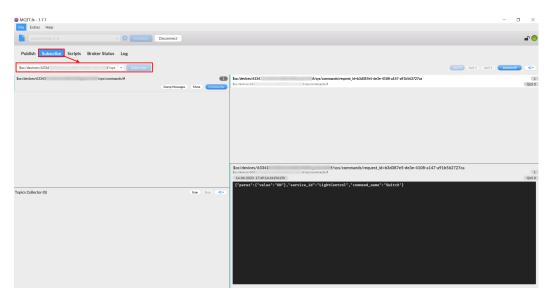
Figure 2-13 Command delivery - MQTT

IoTDA Instances / AI Devices / Device Details	
C Instant for the Collins and Second and Collins. O	🖸 Quick Links
Device Info Cloud Run Logs Cloud Delivery Device Shadow Message Trace Device Monitoring Child Devices Tags	
Message Delivery Command Delivery	
If the product that the device belongs to has commands configured, you can call the platform APIs or clicic Deliver Command to deliver a command. Currently, INQ11 devices support synchronous command delivery, and NB-in1 devices support asynchronous command delivery.	
Synchronous Command Delivery Note Hotorcal record query in nel available for trynchronously delivered commands.           Deliver Command	

#### **NOTE**

MQTT devices support only synchronous command delivery. NB-IoT devices support only asynchronous command delivery.

**Step 3** In the MQTT.fx simulator, select **Subscribe** and enter the command delivery topic. After the subscription, you can view the delivered command parameters.



#### **NOTE**

- Use the MQTT.fx simulator to view the delivered command parameters. The command delivery topic is in the format of **\$oc/devices**/{*device\_id*}/**sys/commands/#**, where {*device\_id*} indicates the value of device ID returned after the device is registered successfully.
- If the system displays a message indicating that the command delivery fails, the device needs to respond to the synchronization command in a timely manner. For details about the response content, see **Platform Delivering a Command**.

----End

#### Using MQTT\_Simulator for Access

**Step 1** Download the **MQTT\_Simulator simulator** (for 64-bit operating system by default) and start it.



**Step 2** Perform operations on the UI.

🖳 MQTT Device Ad	ccess Simulator	— (		×
SSL Conne	ction Enable Backoff Reconnect QoS 0 Connect Disconnect	t		
Server Address	iot=mqtts.cn=north=4.myhuaweicl Device ID Device Secret			
Topic to Subscribe	\$oc/devices//sys/commands/#	S	ubscrib	e
Log			<u>Clear</u>	Log
				Ŧ
Topic to Publish	\$oo/devices//sys/properties/report			
{"services":[{"pr {"alarm":1, "temper	operties": rature":92.670784, "humidity":78.37673, "zmokeConcentration":19.97906}, "service_id":"zmokeDetector", "event	t_time":r	null}]}	
	Publish			

- 1. On the simulator UI, enter the server address, device ID, and device secret. Set the parameters based on the actual device information.
  - Server Address: domain name. Obtain it by referring to Platform Connection Information.
  - **Device ID** and **Device Secret**: Obtain them from here.
- 2. Use the corresponding **certificates** together with different server addresses during SSL-encrypted access. Obtain certificates by referring to **Obtaining Resources** and replace certificates in the **certificate** folder.

📜 certificate
en-US
📜 zh-CN
device_demo_mqttnet.exe
🟳 device_demo_mqttnet.exe.config
🔒 device_demo_mqttnet.pdb
MQTTnet.dll
MQTTnet.Extensions.ManagedClient.dll

3. Select SSL encryption or no encryption when establishing a connection on the device side and set the QoS mode to **0** or **1**. Currently, QoS 2 is not supported. For details, see **Constraints**.

#### **Step 3** Establish a connection.

To connect a device or gateway to the platform, upload the device information to bind the device or gateway to the platform. Click **Connect**. If the domain name, device ID, and secret are correct, a device connection success is displayed in the log. Check the device status on IoTDA, as shown in the following figure.

#### Figure 2-14 Device list - Device online status

K [] usemessag	R Details & Mooth									
Overview	All Devices Total devices 2 @ Activated devices 1 (@ Oxfore devices 1									
Products Devices	Device Lat Batch Registration Batch Update Batch Deletion Batch Add Devices To Group File Uptrads									
All Devices	Register Device De	lete Untresse	Freeze							
Groups Policies	Q. Search by node ID by defa	suit.						(2) (Advanced Search v) (3)		
	Status 😣	Device Name 😣	Node ID 😣	Device ID 😣	Resource Space 😔	Product 😣	Node Type 😣	Operation		
Software/Firmware Upgrades	<ul> <li>Online</li> </ul>	1412	-	Matti 1448/1921/Material (Ma. 1	Delectrice, Hillings	Strates	Drafty contacted	View Debug More ~		

#### **Step 4** Subscribe to a topic.

Only devices that subscribe to a specific topic can receive messages about the topic published by the broker. For details on the preset topics, see **Topics**.

After the connection is established and a topic is subscribed to, the following information is displayed in the log area on the home page of the demo.

💀 MQTT Device A	ccess Simulator		_	
🗌 SSL Conne	otion 🗌 Enable Backoff Reconnect QoS 0	✓ Connect	Disconnect	
Server Address	Device ID		Device Secret	
Topic to Subscribe	\$oc/devices,	/sys/commands/#		Subscribe
Log				Clear Log
	05 - try to connect to server 05 - connect to mqtt server success, deviceId is			•
Topic to Publish	\$oc/devices	/sys/properties/report		
to rubiran		, ., ., p p		
{"services":[{"pr {"alarm":1, "tempe:	rature":92.670784, "humidity":78.37673, "smokeConcent	_	nokeDetector", "event_tim	e":null}]}
	Publish			

#### **Step 5** Publish a topic.

Publishing a topic means that a device proactively reports its properties or messages to the platform. For details, see the API **Device Reporting Properties**.

The simulator implements the property reporting topic and property reporting.

Enter the JSON message to be reported. The value of **luminance** is **30**. After a topic is published, the following information is displayed on the demo page.

🛃 MQTT Device Ac	ccess Simulator			– 🗆 X	
SSL Conne	ction 🗌 Enable Backoff Reconnect (	QoS 0 ~	Connect	Disconnect	
Server Address	Device	ID	Device	Secret *******	
Topic to Subscribe	\$oc/devices/	/sys/comma	nds/#	Subscribe	
	27 - mqtt server is disconnected			<u>Clear Lo</u>	s
2024-05-06 11:19:2 2024-05-06 11:27:0	null			properties/report service_id": "BasicData",	
Topic to Publish	\$oc/devices,	/sys/prope	rties/report		
{"services" : [ { "luminance": } event_time" : }]		ł			
		Publish			

If the reporting is successful, the reported device properties are displayed on the device details page.

#### Figure 2-15 Viewing reported data - MQTT

10	DA Instances / All D	Devices / Device Details							
	On	ine 🕤						🗋 Quid	Links
D	vice Info Clou	ud Run Logs Cloud Delivery De	vice Shadow Message Tra	ce Device Mon	itoring Child Devices Tags Groups				
	Device Name			Resource Space	Defaultings_Hidle-out	Product	Manual Victoria		
	Device ID	mechanicmeder. 7		Node ID		Authentication	Secret Reset Secret		
						Type			
	Node Type	Directly connected		Firmware Version		Software Version	-		
	Description			Registered	Teage 23, 2424 10 10 ee (367-68.08)	Activated	Stept 25, 2624 17 12 08 (367-08.08)		
	Last Online	Nav 12, 2024 18 18 18 08/7 -08 08		MQTT	Vew				
				Connection					
				Parameter					
	Product Model								
		used by the device based on the product model						View All Properties	a
				ns dots (.), dollar symb	ools (5), or empty char (the hexadecimal ASCII code is 00), the property data car	not be updated.			
	Enter the service	e name. Q	Latest Reported Time	12, 2024 19 21 48	QM/T =08.00			luminance × 0	2
	BasicData								
	LightControl								
			luminance						
			30						
			Total Records: 1 16 🗸						

**Step 6** Receive a command.

The simulator can receive commands delivered by the platform. After an MQTT connection is established and a topic is subscribed, you can deliver a command on the device details page of the IoTDA **console**. After the command is delivered, the MQTT callback receives the command delivered by the platform.

For example, if you want to remotely turn on the light, deliver the **LightControl: switch** command with the parameter value **ON**.

Figure 2-16 Command delivery - Synchronous command delivery

IoTDA Instances / All Devices / Device Details		
Device Info Cloud Run Logs Cloud Delivery () Device Shadow	Message Trace Device Monitoring Child Devices Tags	
Message Delivery Command Delivery	Deliver Command	×
() If the product that the device belongs to has commands configured, you can c	For synchronously delivered command, device should send response within 20 seconds after the command is sent. Otherwise, the status of this command will be set as "Timed Out". Learn more I2	very, and NB-IoT
Synchronous Command Delivery Note: Historical record query is not Deliver Command	Command     E     value	
	Cancel OK	0

After the synchronous command is delivered, the following information is displayed on the demo page.

💀 MQTT Device A	ccess Simulator	– 🗆 X
🗌 SSL Conne	ection 🗌 Enable Backoff Reconnect QoS 0 🗸 Connect	Disconnect
Server Address	Device ID	Device Secret
Topic to Subscribe	\$oc/devices//sys/commands/#	Subscribe
Log		Clear Log
2024-05-06 11:30: \$oc/devices. properties: { "luminance" is published an ivalue: "ON", s 2024-05-06 11:31: 2024-05-06 11:31: soc/devices.	56 - publish messageId 69b42b78-e6c4-4eed-8e7e-5ac5c2e0f7ed, topic: payload: {"services" : [ {' ': 30	ras": {"result": "success"}} 39-bf84-e540acf1bbcc. pavload:
Topic to Publish	\$oc/devices/sys/properties/report	
<pre>{"services" : [ {</pre>		

----End

#### **Advanced Experience**

After using the simulator to connect a simulated MQTT device to the platform, you may understand how the MQTT device interacts with the platform through open APIs.

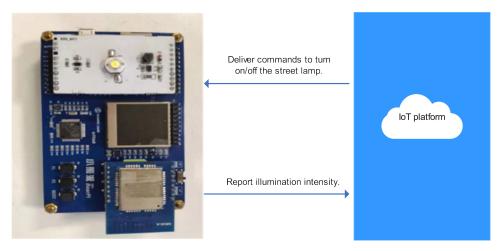
To better experience the IoTDA service, develop real-world applications and devices and connect them to the platform. For details, see **Developer Guide**.

# 2.2 Developing a Smart Street Light Using NB-IoT BearPi

#### **Scenarios**

Smart street lights play an important role in the intelligent transformation of city roads. They save energy in public lighting, reduce traffic accidents caused by poor lighting, and contribute to many other aspects in our community. As a common public facility, street lights can well exemplify how intelligence is transforming the world and implemented in our daily lives.

This section describes how to build a smart street light solution in just 10 minutes based on Huawei one-stop development tool platform (the IoT Link plug-in on Visual Studio Code), covering the device (BearPi development kit) and Huawei Cloud IoTDA. A smart street light detects and reports the illumination intensity to the IoTDA console. The LED light switch can be remotely controlled on the IoTDA console.

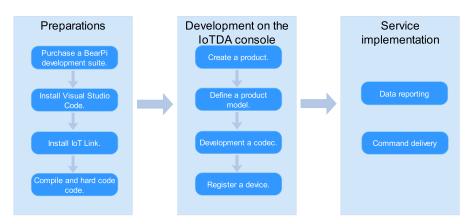


#### **Development Environment**

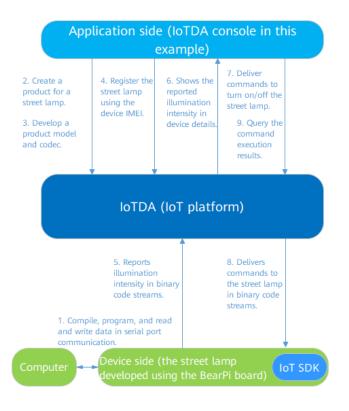
- Hardware: BearPi-IoT development suite (including NB-IoT cards, NB-IoT modules, smart street light function modules, and USB data cables)
- Software: Visual Studio Code, the IoT Link plug-in, Huawei Cloud IoTDA service, and 64-bit Windows 7 or later (64-bit Windows 10 is used in the following demonstration.)

#### **Development Process**

The following figure shows the end-to-end process of developing a smart street light.



In this scenario, a device interacts with the platform using LwM2M (NB-IoT card). The application displays property changes of the device and delivers commands to the device.

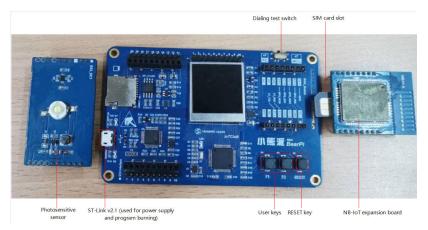


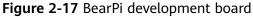
#### Introduction to the BearPi Development Board

The development board is a sensing device in the IoT architecture. This type of device usually includes a sensor, communications module, chip, and operating system. To improve scalability of the development board, the BearPi development board does not use a conventional onboard design. Instead, it uses replaceable sensor and communications module expansion boards. The communications module is an entrance and exit of data transmission. Common communications modules include NB-IoT, Wi-Fi, and 4G ones. A chip controls a device. The development board has a built-in low-power STM32L431 chip as the main control chip (MCU). The operating system is Huawei LiteOS, which provides various device-cloud interworking components.

To facilitate development and debugging, the development board uses the onboard ST-Link of the 2.1 version, as shown in **Figure 1**. It provides functions such as online debugging and programming, drag-and-drop download, and virtual serial port. An LCD screen with a resolution of 240 x 240 is installed at the center of the board to display sensor data and other debug logs. Below the LCD screen is the MCU.

There is a DIP switch in the upper right corner of the development board. When you set the switch to the AT-PC mode, use the serial port assistant on the computer to send AT commands to debug the communication module. When you set it to the AT-MCU mode, use the MCU to send AT commands to interact with the communication module and sends the collected sensor data to the cloud through the communication module.





#### Hardware Connection

- 1. Insert the NB-IoT card into the SIM card slot of the NB-IoT expansion board. Ensure that the **notch-end faces outwards**, as shown in **Figure 2**.
- 2. Insert the photosensitive sensor and NB-IoT expansion board into the development board. Ensure they are inserted in the correct direction. Use a USB data cable to connect the development board to the computer. If the screen displays information and the power indicator is on, the development board is powered on.



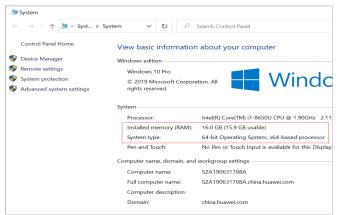


#### Installing the IoT Link Studio Plug-in

IoT Link Studio is an integrated development environment (IDE) developed for IoT devices. It provides one-stop development capabilities, such as compilation, programming, and debugging, and supports multiple programming languages, such as C, C++, and assembly language.

**Step 1** Obtain the operating system information. For example, on Windows 10, enter **pc** in the **Run** window, and click **Properties** to view the system information.

Figure 2-19 Obtaining the system configuration



**Step 2** Click **here** to download and install a Visual Studio Code version that suits your computer system. This section uses 64-bit Windows 10 as an example. Download version 1.49. Other versions do not support IoT Link.

💐 Visual Studio	Code Docs Updates Blog API Extensions FAQ Learn P Search Do	cs <u>↓</u> Download
	Learn. Connect. Code. Join us on May 25-27 at Microsoft Build! Register today	
UPDATES	August 2020 (version 1.49)	IN THIS UPDATE
April 2021	Update 1.49.1: The update addresses these issues.	Editor
March 2021	Update 1.49.2: The update addresses these issues.	Workbench Source Control
February 2021	Update 1.49.2: The update addresses these issues.	Debugging
January 2021		Languages
November 2020	Downloads: Windows User system ARM   Mac: 64 bit   Linux: snap deb rpm tarball	Contributions to extensions
October 2020	Welcome to the August 2020 release of Visual Studio Code. There are a number of updates in this version	Extension authoring Proposed extension APIs
September 2020	that we hope you will like, some of the key highlights include:	Engineering
August 2020	Format modified text - Limit formatting to just the code you've changed.	Notable fixes
July 2020	<ul> <li>Change casing on Search and Replace - Change text casing during global Search and Replace.</li> </ul>	Thank you
	Source Control repositories view - Display pending changes per repository.	y Tweet this link
June 2020	<ul> <li>Filter Debug Console output - Quickly filter and find debugging output.</li> <li>Improved JS debugger Auto Attach - "Smart" Auto Attach to Node is scripts or test runners.</li> </ul>	Subscribe
May 2020	<ul> <li>TypeScript optional chaining refactoring - Convert multiple checks to a concise optional chain.</li> </ul>	<ul> <li>Ask questions</li> <li>Follow @code</li> </ul>
April 2020	<ul> <li>JSDoc @deprecated tag support - IntelliSense clearly shows APIs marked as deprecated.</li> </ul>	Request features
March 2020	Notebook UX updates - Cell Status bar contributions, enhanced notebook diff editor.	Report issues
February 2020	If you'd like to read these release notes online, go to Updates on code.visualstudio.com.	Watch videos

Figure 2-20 Downloading Visual Studio Code

Note: Visual Studio Code does not support macOS.

- **Step 3** After Visual Studio Code is installed, in its plug-in store, search for IoT Link and install it.
- **Step 4** Perform the initial startup configuration.

When the IoT Link Studio is started for the first time, it automatically downloads the latest SDK package and GCC dependency environment. Ensure that the network is available. Do not close the window during the installation. After the installation is complete, restart the Visual Studio Code for the plug-in to take effect.

**NOTE** 

If a proxy is required, click in the lower left corner of the Visual Studio Code home page and choose **Settings** > **Application** > **Proxy**, and set **Use the proxy support for extensions** to **on**.

Step 5 If the automatic downloading failed, manually download SDK package, change the file name to IoT\_LINK, and save it to the C:\Users\\${User name}\.iotlink\sdk directory. Open the Visual Studio Code again. The following figure shows the directory format.

----End

#### Configuring an IoT Link Studio Project

**Step 1** Click **Home** on the toolbar at the bottom of Visual Studio Code.

- Home is used to manage the IoT Link project.
- **Serial** is used to enter AT commands to check the status of the development board.
- **Build** is used to compile the sample code (displayed after Step 3).
- **Download** is used to hard code to the MCU (displayed after Step 3).
- Step 2 Configure the cross compilation toolchain. On the displayed page, click IoT Link Settings and select a toolchain. If the GCC tool directory or file does not exist, download and install it.

#### 

The version of the compilation toolchain downloaded by BearPi STM32431 is win32.zip.

- **Step 3** On the displayed page, click **Create IoT Project**, enter the project name and project directory, and select the hardware platform and sample project template of the developer board.
  - Project Name: Enter a project name, for example, QuickStart.
  - **Project Path**: You can use the default installation path or select a path in a disk other than the system disk, for example, **D:**\.
  - **Platform**: Currently, the demo applies only to the STM32L431\_BearPi hardware platform. Select **STM32L431\_BearPi**.
  - Create based on examples: In this example, select oc\_streetlight\_template. Otherwise, the programmed demo does not match the product model defined on the console and data cannot be reported. If you need to adapt to other scenarios such as smart smoke sensors, select the oc smoke template demo.

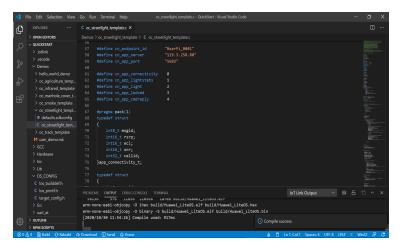
Step 4 Click OK.

----End

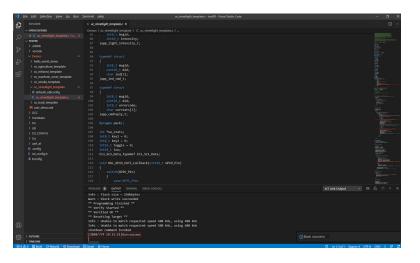
#### **Compiling and Burning Code**

In the provided demo, the information for connecting to the Huawei Cloud IoTDA has been configured. You can directly compile code without modifying code and burn it to the development board MCU.

**Step 1** Click **Build** on the toolbar at the bottom of Visual Studio Code and wait until the compilation is complete. A message is then displayed, indicating that the compilation is successful.



- Step 2 Use a USB data cable to connect the BearPi development board to the computer. Set the dialing test switch in the upper right corner of the board to the AT-MCU mode on the right.
- **Step 3** Click **Download** on the toolbar at the bottom of Visual Studio Code and wait until the burning is complete. A message is then displayed, indicating that the burning is successful.



#### **NOTE**

If the burning fails, the possible cause is that the development board does not have a driver and cannot communicate with the computer through the serial port. In this case, perform 2 to check whether the ST-Link driver is installed. If the driver is not installed, download and install the ST-Link driver by following **Step 4.1**.

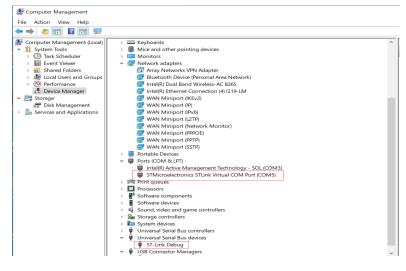
#### Step 4 (Optional) Install the ST-Link driver.

 Visit the ST website, download the ST-Link driver, and double-click the stlink\_winusb\_install.bat file to start automatic installation. This section uses Windows 10 64-bit ST-Link 2.0.1 as an example.

Name	Size	Packed	Туре	Modified	CRC32
II.			File folder		
📕 amd64			File folder	2/8/2018 1:24	
📕 x86			File folder	2/8/2018 1:24	
📧 dpinst_amd64.exe	680,440	242,269	Application	2/8/2018 1:24	E0BABB1A
dpinst_x86.exe	552,328	231,230	Application	2/8/2018 1:24	99D60074
📄 readme.txt	391	250	Text Document	9/11/2018 4:15	ABBCDABC
🚮 stlink_bridge_winusb.inf	2,853	1,093	Setup Information	2/8/2018 1:25	B797E7D3
stlink_dbg_winusb.inf	4,373	1,347	Setup Information	2/8/2018 1:26	83C18B24
stlink_VCP.inf	2,467	871	Setup Information	2/8/2018 1:29	16A7D847
🖲 stlink_winusb_install.bat	412	259	Windows Batch File	6/23/2017 10:1	F9AE8CDC
stlinkbridgewinusb_x64.cat	11,004	5,890	Security Catalog	2/8/2018 1:33	14FD19D4
stlinkbridgewinusb_x86.cat	11,004	5,892	Security Catalog	2/8/2018 1:33	410257F4
stlinkdbgwinusb_x64.cat	10,997	5,891	Security Catalog	2/8/2018 1:33	2B3C3CC5
stlinkdbgwinusb_x86.cat	10,998	5,892	Security Catalog	2/8/2018 1:33	9D9979E2
stlinkvcp_x64.cat	9,248	5,474	Security Catalog	2/8/2018 1:33	E33A16BA
stlinkvcp_x86.cat	9,247	5,470	Security Catalog	2/8/2018 1:33	829A23AB

Note: You can also use an EXE file that adapts to your system version to install the ST-Link driver.

2. Open the device manager on the computer to check whether the driver is installed. If the information shown in the following figure is displayed, the driver is installed.



----End

#### Locating Module Communication Problems Using AT Commands

When IoT Link is connected to the platform, you can use AT commands to quickly locate the connectivity problem between the module and the cloud. This section describes how to use AT commands to detect common problems of the communications module, for example, the device fails to go online or data fails to be reported.

- 1. Connect the BearPi development board to the computer and ensure that the driver has been installed. Set the dialing test switch in the upper right corner of the board to the **AT-PC** mode.
- 2. Click **Serial** on the toolbar at the bottom of Visual Studio Code.

Build ↔ Rebuild ↔ Download > Serail Home

3. Select the port number obtained in 2, set **Baudrate** to **9600**, and click **Open**.

C oc_streetli		≣ Ser	ial Monitor ×			□ …
				Module Test	OMIP Mode	
Port	COM5 -STMicro	D V				
Baudrate	9600	~				
Databit	8	~				
Parity	None	~				
Stopbit	1	~				
Flowcontrol	None	~				
		Open				Hex 🔟 Clea
				_		
					🗹 Add CR 🦳	Hex 🗅 Sen

4. Enter AT+CGATT? and click Send. If +CGATT:1 is returned, the network attach is successful, indicating that the NB-IoT network is normal. If +CGATT:0 is returned, the network attach fails, indicating that the NB-IoT network is abnormal. In this case, check whether the SIM card is correctly inserted or contact the carrier to check the network status.

	≣ Ser	ial Monitor ×					□ …
		Standard Mode	Module Test	OMIP Mode			
COM5 -STMicro	$\sim$						
9600	$\sim$						
8	$\sim$	OK					
None	$\sim$						
1	$\sim$						
None	$\sim$						
Cit	ose					- Hex	🗇 Clea
		AT+CGATT?	_		_		
						Add CR 🔵 Hex	Seni
	COM5 -STMicro 9600 8 None 1 None	COM5-STMicro > 9800 > 8 > None > 1 > Kons > Close	COM5 -STMcro	Standard Mode     Module Test       0600     CCATT: 1       0Kms     C       1     C       Code     At+reactT?	COMS-STMicro  COMS-S	COMS-STMicro V BECO COMS-STMicro V BECO CCAPTT: 1 OK CCAPTT: 1 OK CCAPTT: 2 CCAPTT:	Standard Mode Module Test OMIP Mode

#### D NOTE

After using the AT commands to detect the module communication, set the dialing test switch to the **AT-MCU** mode so that the collected data can be sent to the platform through the communication module after the console configuration.

In the **AT-PC** mode, the development board communicates with the serial port of the computer, and AT commands are used to read and write data such as the status of the development board. In the **AT-MCU** mode, the development board connects to the network through the SIM card inserted into the module to implement NB-IoT communications.

5. The AT+CSQ<CR> command is used to check the network signal strength and SIM card status. Enter AT+CSQ<CR> and click Send. +CSQ:\*\*,## is returned. In the preceding output, \*\* ranges from 10 to 31. A larger value indicates better signal quality. ## indicates the bit error rate, which ranges from 0 to 99. If the returned values are not within these ranges, check whether the antenna or SIM card is correctly installed.

Note: This section lists only two common AT commands for detecting the network status of the module. For more AT commands, see the instructions of the BearPi module.

#### **Operations on the Console**

After connecting the physical device and compiling and programming code, go to the IoTDA console to create a product, define a product model, develop a codec, and register the device.

- Creating a product: Specify the protocol type, data format, manufacturer name, and device type of a product on the platform. In this example, create a smart street light product on the console based on the product features.
- Defining a product model: A product model is a JSON file that describes device capabilities. It defines basic device properties and message formats for data reporting and command delivery. Defining a product model is to construct an abstract model of a device in the platform to enable the platform to understand the device properties. In this example, define a street light product model with light switch control, illumination intensity, and signal quality properties on the console.
- Developing a codec: A codec is called by the platform to convert data between the binary and JSON formats. The binary data reported by a device is decoded into the JSON format for the application to read, and the commands delivered by the application are encoded into the binary format for the device

to understand and execute. Since the data format of smart street lights is binary, a codec is needed to enable the platform to understand the data reported by the smart street light and to enable the smart street light to understand the commands delivered by the platform.

• Registering the device: Register the BearPi smart street light with the platform.

### **Creating a Product**

A product is a collection of devices with the same capabilities or features. In addition to physical devices, a product includes product information, product models, and codecs generated during IoT capability building. In this example, create a smart street light product on the IoTDA console.

**Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.

< 🖸 🖿 Details & Modify ... Description 📋 Quick Link Products Rules O&M If you want to view data reported by a device and manage and control the device on the plat access protocol and data format. Learn more Resource 2 Defining Product Mo 1 Create Products -3 Register Device 4 Device-side der A product is a set of devices that have the same canal The registered dev Create Product Delete 0 Resource Spaces  $\hat{\Theta}$  Device Type  $\hat{\Theta}$  Protocol  $\hat{\Theta}$  Created  $\hat{\Theta}$  Operation uct Name 🖯 Product ID 🖯 Defaultings, History ward ( 1000 Aug. 28, 2024 (2018) 18, 287-38, 28 Total Records: 1 10 V < 1 >

Figure 2-21 Creating a product

**Step 2** Create a product whose protocol type is LwM2M or CoAP and device type is **StreetLamp**, set parameters as prompted, and click **OK**.

Х

Create Product		
★ Resource Space ⑦		~
	To create a new resource s details page.	pace, you can go to the instance
* Product Name		
Protocol 📀	MQTT	~
★ Data Type ⑦	JSON	~
Device Type Selection	Standard profile	Custom
★ Device Type ⑦		
Advanced Settings \wedge	Custom Product ID   Descri	ption
Product ID (?)		
Description		
		0/128 //
		Cancel OK

Figure 2-22 Creating a product - CoAP

----End

# **Uploading a Product Model**

A product model is a JSON file that describes device capabilities. It defines basic device properties and message formats for data reporting and command delivery. Defining a product model is to construct an abstract model of a device in the platform to enable the platform to understand the device function. A developed product model is provided for you to quickly experience the cloud migration process. If you want to go through the process of developing a product model, go to **Developing a Product Model**.

### Procedure

**Step 1** Click the created product. The product details page is displayed.

### **Step 2** On the **Model Definition** tab page, click **Import from Local**.

Figure 2-23 Uploading a product model - CoAP

IoTDA Instances / F		Quick Links
× 1 mm ( ) 1	Registered devices: 0	QUICK DINS
Basic Information	Codec Deployment Online Debugging	
Product Deta	r	
Product Name	Resource Space	
Device Type	Protocol LV/M2M/CoAP	
Data Type	Binary Created	
Industry	Description - 2	
	Basic Service Battery Electron Set PPESSURE READ PERIOD Water Water Voltage Level	
	et i pressone neuropreno Visione visione visione voltage Leven Period Value Result	
	A product model describes product details and service capabilities. You can define a product model using multiple methods, typu do not define a product model for a device, the partition only forwards the data reported by the device and does not parts the r	sata.
	Customize Model Import from Local Import from Excel Import from Likrary Learn more	

Step 3 On the dialog box displayed, upload the product model provided and click OK.

Figure 2-24 Uploading a model file - CoAP

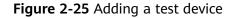
Import	from Local	$\times$
	e service definition in the imported file will replace the original service inition of the product.	
	evelop a product model based on the format standards, you can pack and earn about product models.	
★ File	StreetLight_58f383040b1e2 (1.93KB) X Select File	
-	re the model file by referring to the template. uct Template	
	Cancel OK	



### **Registering a Device**

This section describes how to register a device integrated with the NB-IoT module, the BearPi smart street light in this example, to the platform.

- **Step 1** On the product details page, click the **Online Debugging** tab, and click **Add Test Device**. This section use a non-security NB-IoT device as an example.
- Step 2 In the dialog box displayed, set the parameters and click OK.



Add Test Device	•		×
Device Type	Physical device	Virtual device	
Device Name			
* Node ID			
Authentication Type	Secret		
Secret			8
Confirm Secret			8
		Cancel	ОК

- **Device Name**: Customize a name.
- Node ID: Enter the IMEI of the device. The node ID will be carried by the device for device access authentication. You can view the node ID on the NB-IoT module. You can also set the dialing test switch to the AT-PC mode, select the STM port, set the baud rate to 9600, and run the AT+CGSN=1 command to obtain the IMEI.

Note: After obtaining the IMEI and registering the device, set the dialing test switch of the development board to the AT-MCU mode because the development board connects to the network through the NB-IoT card only in MCU mode.

C oc_streetli	ght_template.c	≡ Ser	ial Monitor ×			•
			Standard Mode	Module Test	OMIP Mode	
Port	COM5 -STMicro	- V				
Baudrate	9600	~	+CGSN:86	42		l
Databit	8	~	ок			
Parity	None	$\sim$				
Stopbit	1	$\sim$				
Flowcontrol	None	~				l
		Close				l
			AT+CGSN=1			
					🖉 Add CR 🕕 Hex 主 Sen	
			4			Ľ

- Registration Mode: Select Unencrypted.
- **Step 3** The device is created. You can view the created device on the console.

----End

# **Data Reporting**

After the connection between the platform and the development board is set up, the BearPi smart street light reports the light sensor data every 2 seconds according to the code burnt to the development board. The reporting frequency can be customized in the demo based on service requirements. You can block the light with your hand to change the light intensity and view the real-time change of the light intensity data reported to the platform.

Note: Ensure that the dialing test switch of the development board is set to the AT-MCU mode.

- **Step 1** Log in to the **IoTDA** console and click the target instance card. In the navigation pane, choose **Devices** > **All Devices**.
- **Step 2** Select the target device and click **View** to view the data reported to the platform.

Device time     Cloud Delvery     Device Shadow     Message Trace     Device Monitoring     Child Devices     Tags     Groups       Device Name     Reterrol Space     Product     Authentication     Servet     Reterrol Space       Device Name     Note 10     Authentication     Servet     Reterrol Space       Note Type     Directly connected     Replaned     Stream Servet       Device Joint     Replaned     Advertication     Servet	IoTDA Instances / All Devices / Device Details								
Device Name Product Device ID Node ID Adheritation Secret Revet Deput Tope Tope Decoption Decoption Decoption Reported Revet Revison Decoption Reported Revet Revison Revet Re	< 📻 👝 🕐 🗋 Qualitas								
Device D Node 10 Authentication Server Read Device Node Type Deedly convected Emmany Version Device Version Device Version Emmany Version Activated Activated Activated Emmany Version Device Version De	Device Info Cloud Run Logs Cloud Delivery Dev	ze Shadow Message Trace Device Monitoring Child Devices Tags Groups							
Lat Online	Device ID Node Type Directly connected Description	Node ID Firmware Version	Authentication Secret Reset Secret Type Software Version						
Product Model Data Project Model Data User AIP reperts Verv AI Properts Verv AIP Properts V									
Enter the service name     Q.     Latest Reported Time     Noninance     Noninance       Button     Life     Noninance     Noninance       Sensor     206       Total Records:     1     15	Button LED Sensor	luminance Jannance 206		[kminance X   Q_]					

Figure 2-26 NB-IoT device data reporting - Viewing data

----End

# **Delivering a Command**

- **Step 1** Log in to the **IoTDA console**. Click the target product to go to its details page.
- **Step 2** On the **Online Debugging** tab page, click the target device to access the debugging page.
- **Step 3** After setting the command parameters, click **Send**.

Figure 2-27 Command delivery debugging - SmokeDetector

Debug output	Real-Time Refresh X Clear Lo	Application Simulator Device Simulator
Application Simulator	Command Delivery rm Device Simulator Deta Reporting	Commands Sent The application simulator can issue commands to the device according to the product definition. If you use agraphically developed codec plug-in, please carry all the fields defined in the plug-in
All Received Sent	All Commands Received Data Reported	when issuing a command to obtain correct coding results and the length of each command must be less than 512 bytes.
Sent Jul 06, 2024 14:56:53 GMT+08:00 Message Body: {"service_jd": ", "command_name": ", "paras": { }	Ocmmands Received Jul 05, 2024 14:56:53 GMT+08:00 Received Message Body: {"paras"; { , , "service_idt" , "command_name"; " }	Service v
		sea Send

**Step 4** The light on the BearPi board is on. Deliver the **OFF** command. The light is turned off.



----End

This is the end-to-end development of a smart street light by using the NB-IoT BearPi development board.

### Reference

### • Developing a Product Model

On the **Basic Information** tab page, click **Customize Model** to add services of the product.

Table 2-3 describes the service defined in the product model.

Table 2	2-3 I	Device	service	list
---------	-------	--------	---------	------

Service ID	Description
Button	Real-time button detection
LED	LED control
Sensor	Real-time light intensity detection
Connectivity	Real-time signal quality detection

The following table lists the service capabilities.

### Table 2-4 Button

Capability Description	Property Name	Data Type	Data Range
Property	toggle	Integer	0 to 65535

Enumerat

ON,OFF

ON,OFF

ion

3

	Capabil ity Descrip tion	Com mand Nam e	Comm and Param eter	Paramet er Name	Туре	Data Length
	Comma	Set_L	Comm	LED	String	3

Light\_st

ate

Table 2-5 LED command list

ED

and

se

Respon

### Table 2-6 Sensor

nds

Capability Description	Property Name	Data Type	Data Range
Property	luminance	Integer	0 to 65535

String

### Table 2-7 Connectivity

Capability Description	Property Name	Data Type	Data Range
Properties	SignalPower	Integer	–140 to –44
	ECL	Integer	0 to 2
	SNR	Integer	-20 to 30
	CellID	Integer	0 to 65535

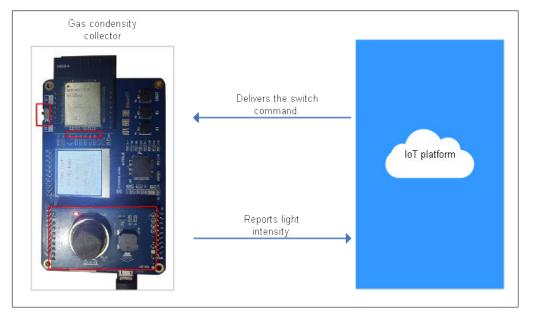
# 2.3 Developing a Smart Smoke Detector Using NB-IoT BearPi

### **Scenarios**

Fires have caused great loss of lives and properties each year. As more independent smoke detectors have been put to use, their limitations become obvious. For example, users cannot monitor the working status of smoke detectors in real time or receive alarm information when they are absent.

On the contrary, NB-IoT smart smoke detectors overcome their disadvantages like difficult cabling, short battery lifespan, high maintenance costs, and inability to interact with property owners and firefighting departments. These smart smoke detectors use wireless communication and feature plug-and-play, no cabling, and easy installation.

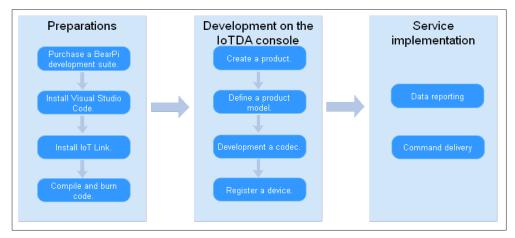
This topic describes how to build a smart smoke detector solution in just 10 minutes based on Huawei one-stop development tool platform (the IoT Link plugin on Visual Studio Code), covering the device (BearPi development kit) and Huawei Cloud IoTDA. A smart smoke detector detects and reports smoke density to the IoTDA console. The beep switch can be remotely controlled on the IoTDA console.



# **Development Environment**

- Hardware: BearPi-IoT development suite (including NB-IoT cards, NB-IoT modules, smart street light function modules, and USB data cables)
- Software: Visual Studio Code, the IoT Link plug-in, Huawei Cloud IoTDA service, and 64-bit Windows 7 or later (64-bit Windows 10 is used in the following demonstration.)

# **Development Process**



# Introduction to the BearPi Development Board

The development board is a sensing device in the IoT architecture. This type of device usually includes a sensor, communications module, chip, and operating

system. To improve scalability of the development board, the BearPi development board does not use a conventional onboard design. Instead, it uses replaceable sensor and communications module expansion boards. The communications module is an entrance and exit of data transmission. Common communications modules include NB-IoT, Wi-Fi, and 4G ones. A chip controls a device. The development board has a built-in low-power STM32L431 chip as the main control chip (MCU). The operating system is Huawei LiteOS, which provides various device-cloud interworking components.

To facilitate development and debugging, the development board uses the onboard ST-Link of the 2.1 version, as shown in **Figure 1**. It provides functions such as online debugging and programming, drag-and-drop download, and virtual serial port. An LCD screen with a resolution of 240 x 240 is installed at the center of the board to display sensor data and other debug logs. Below the LCD screen is the MCU.

There is a DIP switch in the upper right corner of the development board. When you set the switch to the AT-PC mode, use the serial port assistant on the computer to send AT commands to debug the communication module. When you set it to the AT-MCU mode, use the MCU to send AT commands to interact with the communication module and sends the collected sensor data to the cloud through the communication module.

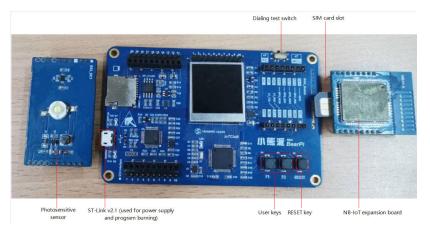


Figure 2-28 BearPi development board

# **Hardware Connection**

- 1. Insert the NB-IoT card into the SIM card slot of the NB-IoT expansion board. Ensure that the notch-end faces outwards, as shown in Figure 2-29.
- 2. Insert the smoke density collection control board and NB-IoT expansion board into the development board. Ensure they are inserted in the correct direction. Use a USB data cable to connect the development board to the computer. If the screen displays information and the power indicator is on, the development board is powered on.

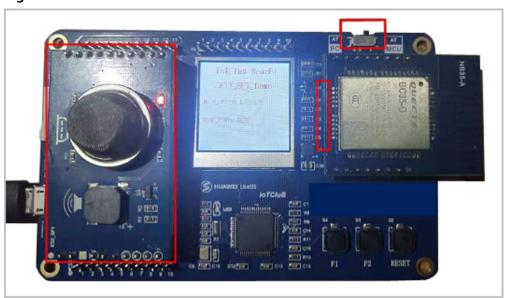


Figure 2-29 Hardware connection

# Installing the IoT Link Studio Plug-in

IoT Link Studio is an integrated development environment (IDE) developed for IoT devices. It provides one-stop development capabilities, such as compilation, programming, and debugging, and supports multiple programming languages, such as C, C++, and assembly language.

**Step 1** Obtain the operating system information. For example, on Windows 10, enter **pc** in the **Run** window, and click **Properties** to view the system information.

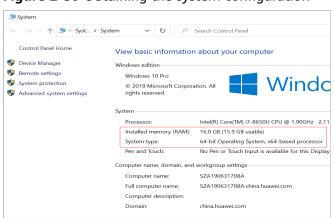


Figure 2-30 Obtaining the system configuration

**Step 2** Click **here** to download and install a Visual Studio Code version that suits your computer system. This section uses 64-bit Windows 10 as an example. Download version 1.49. Other versions do not support IoT Link.

刘 Visual Studio	Code Docs Updates Blog API Extensions FAQ Learn P Search Do	cs <u>↓</u> Download
	Learn. Connect. Code. Join us on May 25-27 at Microsoft Build! Register today	
UPDATES	August 2020 (version 1.49)	IN THIS UPDATE
April 2021	Update 1.49.1: The update addresses these issues.	Editor
March 2021	Update 1.49.2: The update addresses these issues.	Workbench Source Control
February 2021	Update 1.49.2: The update addresses these issues.	Debugging
January 2021	Downloads: Windows User System ARM   Mac: 64 bit   Linux: snap deb rpm tarball	Languages
November 2020	Downloads, windows oser bystem Akwi (Mac. of bit   Endx, shap deb fpm tarbail	Contributions to extensions
October 2020	Welcome to the August 2020 release of Visual Studio Code. There are a number of updates in this version	Extension authoring Proposed extension APIs
September 2020	that we hope you will like, some of the key highlights include:	Engineering
August 2020	<ul> <li>Format modified text - Limit formatting to just the code you've changed.</li> </ul>	Notable fixes
July 2020	Change casing on Search and Replace - Change text casing during global Search and Replace.	Thank you
June 2020	<ul> <li>Source Control repositories view - Display pending changes per repository.</li> <li>Filter Debug Console output - Quickly filter and find debugging output.</li> </ul>	Tweet this link
	<ul> <li>Improved JS debugger Auto Attach - "Smart" Auto Attach to Node.js scripts or test runners.</li> </ul>	Subscribe
May 2020	<ul> <li>TypeScript optional chaining refactoring - Convert multiple checks to a concise optional chain.</li> </ul>	y Follow @code
April 2020	<ul> <li>JSDoc @deprecated tag support - IntelliSense clearly shows APIs marked as deprecated.</li> </ul>	Request features
March 2020	<ul> <li>Notebook UX updates - Cell Status bar contributions, enhanced notebook diff editor.</li> </ul>	Report issues
February 2020	If you'd like to read these release notes online, go to Updates on code visualstudio.com.	Watch videos

Figure 2-31 Downloading Visual Studio Code

Note: Visual Studio Code does not support macOS.

- **Step 3** After Visual Studio Code is installed, in its plug-in store, search for IoT Link and install it.
- **Step 4** Perform the initial startup configuration.

When the IoT Link Studio is started for the first time, it automatically downloads the latest SDK package and GCC dependency environment. Ensure that the network is available. Do not close the window during the installation. After the installation is complete, restart the Visual Studio Code for the plug-in to take effect.

**NOTE** 

If a proxy is required, click in the lower left corner of the Visual Studio Code home page and choose **Settings** > **Application** > **Proxy**, and set **Use the proxy support for extensions** to **on**.

Step 5 If the automatic downloading failed, manually download SDK package, change the file name to IoT\_LINK, and save it to the C:\Users\\${User name}\.iotlink\sdk directory. Open the Visual Studio Code again. The following figure shows the directory format.

----End

# Configuring an IoT Link Studio Project

**Step 1** Click **Home** on the toolbar at the bottom of Visual Studio Code.

- **Home** is used to manage the IoT Link project.
- **Serial** is used to enter AT commands to check the status of the development board.
- **Build** is used to compile the sample code (displayed after Step 3).
- **Download** is used to hard code to the MCU (displayed after Step 3).
- Step 2 Configure the cross compilation toolchain. On the displayed page, click IoT Link Settings and select a toolchain. If the GCC tool directory or file does not exist, download and install it.

### D NOTE

The version of the compilation toolchain downloaded by BearPi STM32431 is win32.zip.

- **Step 3** On the displayed page, click **Create IoT Project**, enter the project name and project directory, and select the hardware platform and sample project template of the developer board.
  - **Project Name**: Set this parameter as required, for example, **Smoke**.
  - **Project Path**: You can use the default installation path or select a path in a disk other than the system disk, for example, **D**:\.
  - **Platform**: Currently, the demo applies only to the STM32L431\_BearPi hardware platform. Select **STM32L431\_BearPi**.
  - **Create based on examples**: In this example, select **oc\_smoke\_template demo**. Otherwise, the demo does not match the product model defined on the console and data cannot be reported. Click **OK**.

Step 4 Click OK.

----End

### **Compiling and Burning Code**

In the provided demo, the information for connecting to the Huawei Cloud IoTDA has been configured. You can directly compile code without modifying code and burn it to the development board MCU.

- **Step 1** Click **Build** on the toolbar at the bottom of Visual Studio Code and wait until the compilation is complete. A message is then displayed, indicating that the compilation is successful.
- **Step 2** Use a USB data cable to connect the BearPi development board to the computer. Set the dialing test switch in the upper right corner of the board to the **AT-MCU** mode on the right.
- **Step 3** Click **Download** on the toolbar at the bottom of Visual Studio Code and wait until the burning is complete. A message is then displayed, indicating that the burning is successful.

### **NOTE**

If the burning fails, the possible cause is that the development board does not have a driver and cannot communicate with the computer through the serial port. In this case, perform 2 to check whether the ST-Link driver is installed. If the driver is not installed, download and install the ST-Link driver by following **Step 4**.

- Step 4 (Optional) Install the ST-Link driver.
  - Visit the ST website, download the ST-Link driver, and double-click the stlink\_winusb\_install.bat file to start automatic installation. This section uses Windows 10 64-bit ST-Link 2.0.1 as an example.

Note: You can also use an EXE file that adapts to your system version to install the ST-Link driver.

2. Open the device manager on the computer to check whether the driver is installed. If the information shown in the following figure is displayed, the driver is installed.

----End

# **Creating a Product**

A product is a collection of devices with the same capabilities or features. In addition to physical devices, a product includes product information, product models, and codecs generated during IoT capability building.

**Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.

Figure 2-32 Creating a produ	uct
------------------------------	-----

< 🙆 Heelitandar	Instance wood boost V O Running					R Details & Modify ····
Overview Products	Products					Description 📋 Quick Links
Devices ~ Rules ~ Oddal ~ Resource Spaces Documentiation 【2	Description On the lef plattern, a product is a collection of devices with the Psysical transformation of the second second second second access protocol and data binding. Learn thread () Cristian Products Aproduct is a set of devices that have the same capabilitie Product Development Quele	control the device on the platform, you need to develop a	product model (a profile). The product model informs be the capabilities and features of devices.	3 Register Devices	mmands that are supported by devices. You ca e identity information required to connect	an also define product model defails based on the all Device-site development Integrates device SDKs to context devices to the Device Development Guide
		roduct ID 🖗	Resource Spaces 🕀	Device type () Protocol	θ Created θ	Q Operation View Copy More ~

**Step 2** Create a product whose protocol type is LwM2M/CoAP and device type is **SmokeSensors**, set parameters as prompted, and click **OK**.

Х

Create Product		
★ Resource Space 🧿		~ )
	To create a new resource s details page.	space, you can go to the instance
★ Product Name		
Protocol 🕥	MQTT	~
★ Data Type  ?	JSON	~
Device Type Selection	Standard profile	Custom
* Device Type 🧿		
Advanced Settings \land	Custom Product ID   Descr	ription
Product ID (?)		
Description		
		0/128 //
		Cancel OK

Figure 2-33 Creating a product - CoAP

----End

# **Defining a Product Model**

On the **Basic Information** tab page, click **Customize Model** to add a service for the product.

Table 2-8 describes the service defined in the product model.

Table 2-8 Device service list

Service ID	Description	
Enter <b>Smoke</b> .	Detects smoke density in real time.	

### Table 2 and Table 3 describe the service capabilities.

### Table 2-9 Smoke

Capability Description	Property Name	Data Type	Data Range
Properties	Smoke value	int	0 to 65535

### Table 2-10 Smoke commands

Capabi lity Descri ption	Command Name	Comma nd Paramet er	Paramet er Name	Data Type	Data Length	Enumer ation
Comm ands	Smoke control beep	Comman d	beep	string	3	ON,OFF
		Respons e	beep_sta te	int	/	/

### Adding the Smoke Service

- 1. On the Add Service page, configure Service ID, Service Type, and Description, and click OK.
  - Service ID: Enter Smoke.
  - Service Type: You are advised to set this parameter to the same value as Service ID.
  - Description: Enter Detects smoke density in real time.

Figure 2-34 Adding a service - Smoke

Add Service		×
* Service ID	Smoke	)
Service Type	Smoke	0
Description	Detects smoke density in real time	
	34/128 //	J
	Cancel	ок

Х

- 2. Choose **Smoke**, click **Add Property**, enter related information, and click **OK**.
  - Property Name: Enter Smoke\_Value.
  - Data Type: Select Integer.
  - Access Permissions: Select Read and Write.
  - Value Range: Set it to 0 to 65535.
  - Step: Enter 0.
  - Unit: Leave it blank.

### Figure 2-35 Adding a property - Smoke\_value

Property Name	Smoke_Value	
Description		
	0.	128 🗸
🖈 Data Type	Integer	$\sim$
Access Permissions	Read Write	
k Value Range	0 - 65535	
Step		
Unit		

3. Choose **Smoke**, click **Add Command**, and enter the command name **Smoke\_Control\_Beep**.

Add Command		$\times$
* Command Name	Smoke_Control_Beep	
Command Parameters	Add Command Parameter	
	Parameter Name $\Leftrightarrow$ Data Type $\Leftrightarrow$ Description $\Leftrightarrow$ Operation	
	No table data available.	
	No Command Parameters data available. Add Command Parameter first.	
Response Parameters	Add Response Parameter	
	Parameter Name         Data Type         Description         Operation	
	No table data available.	
	No Response Parameters data available. Add Response Parameter first.	
	Cancel	ж

Figure 2-36 Adding a command - Smoke\_Control\_Beep

4. Click Add Command Parameter and Add Response Parameter respectively, enter related information, and click OK.

Figure 2-37 Adding a command parameter - Beep

* Parameter Name	Веер	
Description		
		0/128 %
★ Data Type	String	~
★ Length	3	
Enumerated Values	ON,OFF	
		9/1,024 🕢
		Cancel

# Add Parameter

 $\times$ 

Add Parameter	r		×
* Parameter Name	Beep_State		
Description			
			0/128 %
★ Data Type	Integer		~
★ Value Range	0	- 1	
Step			
Unit			
			Cancel OK

Figure 2-38 Adding a command parameter - Beep\_State

# **Registering a Device**

This section describes how to register a non-secure NB-IoT module.

- **Step 1** On the product details page, click the **Online Debugging** tab, and click **Add Test Device**. This section use a non-security NB-IoT device as an example.
- Step 2 In the dialog box displayed, set the parameters and click OK.

### Figure 2-39 Adding a test device

Add Test Device			×
Device Type	Physical device	Virtual device	
Device Name			
* Node ID			
Authentication Type	Secret		
Secret			2
Confirm Secret			Ø
		Cancel	ОК

- **Device Name**: Customize a name.
- Node ID: Enter the IMEI of the device. The node ID will be carried by the device for device access authentication. You can view the node ID on the NB-IoT module. You can also set the dialing test switch to the AT-PC mode, select the STM port, set the baud rate to 9600, and run the AT+CGSN=1 command to obtain the IMEI.

Note: After obtaining the IMEI and registering the device, set the dialing test switch of the development board to the AT-MCU mode because the development board connects to the network through the NB-IoT card only in MCU mode.

C oc_streetle	ight_template.c	≡ Ser	ial Monitor ×		۵	
			Standard Mode	Module Test	OMIP Mode	^
Port	COM5 -STMicro	. ~				
Baudrate	9600	$\sim$	+CGSN:86	42		
Databit	8	~	OK			
Parity	None	$\sim$				
Stopbit	1	~				
Flowcontrol	None	$\sim$				
	C	Close				Clea
			AT+CGSN=1			
					🗹 Add CR 🔵 Hex 🚺	Seni
						+ *

• **Registration Mode**: Select **Unencrypted**.

**Step 3** The device is created. You can view the created device on the console.

----End

# **Reporting Data**

After the development board is connected to the platform, it reports the smoke density.

- **Step 1** Log in to the **IoTDA** console and click the target instance card. Choose **Devices** > **All Devices**.
- **Step 2** Select the target device and click **View** to check the data reported to the platform.

Figure 2-40 NB-IoT device data reporting - Viewing data (Smoke)

IoTDA Instances / All Devices / Device Details					
< I ()					🖸 Quick Links
Device Info Cloud Run Logs Cloud Delivery Device	Shadow Message Trace Device Monitor	ring Child Devices Tags Groups			
Danta Name Divita 15 Node Type Directly contracted Discostran Latt Online	Node ID Firmware Version	Mandrag, Milliouri 1999 - J. Mar D. 2011 U. U. H. (1997-1911)	Product Authentication Type Software Version Activated	Secret Reset Secret	
Smoke			data cannot be updated.		Were All Properties



# **Delivering a Command**

**Step 1** Log in to the **IoTDA console**. Click the target product to go to the product details page.

- **Step 2** On the **Online Debugging** tab page, click the target device to access the debugging page.
- **Step 3** After setting the command parameters, click **Send**.

Figure 2-41 Command delivery debugging - Smoke

Debug output				Real-Time Refresh X Clear Log	Application Simu	lator		
Application Simulator	Command Delivery	IoT Platform	Command Delivery	Physical device	Commands The application sin	Sent mulator can issue commands to the device according to the product definition.		
All Received	Sent				If you use a graphically developed code plugin, please carry all fields defined in the plug in when issuing a command to obtain correct coding results and the length of each command must be less than 512 bytes.			
					Service	Smoke		
					Command	Smoke_Control_Beep ~		
		x 1.7			Beep	ON V		
		_				Pending Period (8) 2880 Cache Send		
		No data available.						

----End

# (Optional) Configuring a Device Linkage Rule

- **Step 1** In the navigation pane, choose **Rules** > **Device Linkage**, and click **Create Rule** in the upper right corner.
- **Step 2** Create a device linkage rule based on the table below.

Figure 2-42	Creating a	linkage rule -	Smoke_Beep
-------------	------------	----------------	------------

Set Basic Informa	tion		
Resource Space			
* Rule Name	Smoke_Beep  Activate upon creation		
Rule Type	Cloud Device Side		
	Triggers will be activated on the cloud, and commands delivered to devices or notifications sent.		
Effective Period	Aways effective Specific time		
Description	The smoke detector buzzes when the smoke density is higher than 600.		
	68/256 ×		
	00/230 %)		
Set Triggers Triggers to be met: all Device Property	✓ ✓ ✓ ∫ Smale ✓ ) Assign Device ✓ ) Smale Reselved (	sincker v ) (Sincker, Value v ) (> v ) (500 ) Tropper loging 3006 (3	Detete
Add Trigger		Configure Parameter	
Set Actions		Command Smake_Control_Beep	
Deliver commands	✓ <u>Smoke</u> Reselect Smoke ✓ Smoke_Control ✓ Configure Parameter	Smoke_Control_Beep ON V	Delete
Add Action		Cancel OK	

Parameter	Description
Rule Name	Specify the name of the rule to create, for example, <b>Smoke_Beep</b> .
Activate upon creation	Select Activate upon creation.
Effective Period	Select <b>Always effective</b> .

Parameter	Description
Description	Enter a description of the rule, for example, "The smoke detector buzzes when the smoke density is higher than 600."
Set Triggers	<ol> <li>Click Add Trigger.</li> <li>Select Device Property, and select the device added in Registering a Device.</li> <li>Select smoke for Service Type, Smoke_Value for Property, &gt; for Operation, and 600 for Value. Click Trigger Mode. In the dialog box displayed, set Trigger Strategy to Repetition suppression and Data Validity Period (s) to 300, and click OK.</li> </ol>
Set Actions	<ol> <li>Click Add Action.</li> <li>Select Deliver Commands, and select the device created in Registering a Device.</li> <li>Select Smoke for Service Type, and Smoke_Control_Beep for Command. Click Configure Parameter. In the dialog box displayed, set Beap to ON, and click OK.</li> </ol>

### ----End

# (Optional) Verification

Spray cooling agents around the smoke detector to simulate the smoke density. When the smoke density is greater than 600, the smoke detector buzzes.

# Locating Module Communication Problems Using AT Commands

When IoT Link is connected to the platform, you can use AT commands to quickly locate the connectivity problem between the module and the cloud. This section describes how to use AT commands to detect common problems of the communications module, for example, the device fails to go online or data fails to be reported.

- 1. Connect the BearPi development board to the computer and ensure that the driver has been installed. Set the dialing test switch in the upper right corner of the board to the **AT-PC** mode.
- 2. Click Serial on the toolbar at the bottom of Visual Studio Code.

Build ↔ Rebuild ↔ Download > Serail Home

3. Select the port number obtained in **Step 4.2**, set **Baudrate** to **9600**, and click **Open**.

ight_template.c	≡ Ser	ial Monitor ×				□ …
		Standard Mode	Module Test	OMIP Mode		
COM5 -STMicro	) 🗸					
9600	~					
8	~					
None	~					
1	~					
None	~					
	Open				Hex	🗊 Clea
					🗹 Add CR 🔵 Hex	⊥ Sen
	COM5 -STMicro 9600 8 None 1	COM5-STMicro v 9600 v 8 v None v 1 v None v Cpen	Standard Mode	Standard Mode Module Test COM5-STMicro ~ 9600 ~ 8 ~ None ~ 1 ~ None ~ Open	Standard Mode Module Test OMIP Mode	Sandard Mode Module Test OMIP Mode

4. Enter AT+CGATT? and click Send. If +CGATT:1 is returned, the network attach is successful, indicating that the NB-IoT network is normal. If +CGATT:0 is returned, the network attach fails, indicating that the NB-IoT network is abnormal. In this case, check whether the SIM card is correctly inserted or contact the carrier to check the network status.

C oc_streetlig	ght_template.c	≡ Ser	ial Monitor ×				۵	
_			Standard Mode	Module Test	OMIP Mode			*
Port	COM5 -STMicro	) 🗸						
Baudrate	9600	~	+CGATT:1					
Databit	8	~	ОК					
Parity	None	~						
Stopbit	1	~						
Flowcontrol	None	~						
		Close					<b> _</b>	
			AT+CGATT?			 	Hex 💼	Clea
						🗹 Add C	CR 🕖 Hex 🗖	Sen
			•					+ -

#### **NOTE**

After using the AT commands to detect the module communication, set the dialing test switch to the **AT-MCU** mode so that the collected data can be sent to the platform through the communication module after the console configuration.

In the **AT-PC** mode, the development board communicates with the serial port of the computer, and AT commands are used to read and write data such as the status of the development board. In the **AT-MCU** mode, the development board connects to the network through the SIM card inserted into the module to implement NB-IoT communications.

# 2.4 Connecting and Debugging an NB-IoT Smart Street Light Using a Simulator

# **Scenarios**

This section uses a smart street light as an example to describe how to use the device and application simulators provided by the IoTDA console to experience data reporting and command delivery.

Assume that:

A street light reports a data message carrying the light intensity (light\_intensity) and status (light\_status). The data message is in binary format. The command (SWITCH\_LIGHT) can be used to remotely control the street light status.

# Prerequisites

- You have registered a Huawei Cloud account.
- You have subscribed to IoTDA. If you have not subscribed to the service, go to the IoTDA service page, and click **Access Console** to subscribe to the service.

# (Optional) Creating a Resource Space

A resource space is the commissioning space provided by the platform for applications and devices. You can create different resource spaces for different scenarios.

The system provides a preset resource space, where you can develop product models and codecs for devices online. To create a resource space, perform the following steps:

- Step 1 Visit the IoTDA service page and click Access Console.
- **Step 2** In the navigation pane, choose **IoTDA Instances**, and click the target instance card.
- Step 3 In the navigation pane, choose Resource Spaces and click Create Resource Space. In the displayed dialog box, specify Space Name and click OK.

----End

### **Creating a Product**

You can develop product models and codecs online. The platform provides device and application simulators for you to debug the product models and codecs.

**Step 1** Visit the **IoTDA** service page and click **Access Console**. Click the target instance card.

**Step 2** In the navigation pane, choose **Products**.

### Figure 2-43 Creating a product

< 🖸 teella	endand	National Contract V O Running					🖟 Details 🗞 Modify ···
Overview Products		Products					Description <a>Clutck Links</a>
Devices	~						
Rules O&M Resource Spaces	ž	Description On the IoT platform, a product is a collection of der If you want to view data reported by a device and i access protocol and data format. Learn more		reed to develop a product model (a profile). The product model in	forms the platform of the proper	ies and commands that are supported by devices.	You can also define product model details based on the
Documentation	C	Create Products     A product is a set of devices that have the sar     Product Development Quide	Defining Pro     Define product i     Define product i	duct Models	3 Register Devices The registered device to the platform.	a obtains the identity information required to conne	Device-side development     Integrates device SDKs to connect devices to the     Device Development Guide
		Create Product  Delete  O Select a property or enter a keyword.  Product Name	Product ID (9)	Resource Spaces $\Theta$	Device Type 😔	Protocol @ Created @	Qperation Q
			Statistical Contractory of the	Information, Stational	water:	M271 Aug.20.2024.2018.10.007	View Copy More ~
		Total Records: 1 10 V < 1 >					

**Step 3** Click **Create Product** to create a product using LwM2M over CoAP. Set the parameters and click **OK**.

Create Product	>
★ Resource Space ⑦	
	To create a new resource space, you can go to the instance details page.
★ Product Name	
Protocol 💿	MQTT ~
🗙 Data Type (	JSON ~
Device Type Selection	Standard profile Custom
* Device Type (	
Advanced Settings \wedge	Custom Product ID   Description
Product ID 💿	
Description	
	0/128 //
	Cancel OK

Figure 2-44 Creating a product - CoAP

Basic Informat	ion
Resource Space	Select the resource space to which the product to create belongs.
Product Name	Customize a name, for example, <b>Test_1</b> .
Protocol	Select LwM2M over CoAP.
Data Type	Select <b>Binary</b> . <b>NOTE</b> If <b>Data Type</b> is <b>Binary</b> , codec development is required for the product. If <b>Data Type</b> is <b>JSON</b> , codec development is not required.

Industry	Select <b>Default</b> .
Device Type	streetlamp

----End

# Defining a Product Model

- **Step 1** Click the name of the product created in **3** to go to the product details page.
- **Step 2** On the **Model Definition** tab page, click **Customize Model** to add a service for the product, and click **OK**.
  - Service ID: Enter StreetLight.
  - Service Type: You are advised to set this parameter to the same value as Service ID.
  - **Description**: Provide a description, for example, "Define the properties of light intensity and status."
- **Step 3** Click the ID of the service added in **2**. On the displayed page, click **Add Property** to define a light intensity property collected by the street light.
  - **Property Name**: Enter light\_intensity.
  - Data Type: Select Integer.
  - Access Permissions: Select Read and Write.
  - Value Range: Set it to 0 to 100 (light intensity range).

r

★ Property Name	light_intensity
Description	
	0/128
★ Data Type	Integer ~
* Access Permissions	Read Write
★ Value Range	0 - 100
Step	
Unit	

Figure 2-45 Adding a property - light\_intensity

**Step 4** Click **Add Property** to define a property as the status of the street light.

- **Property Name**: Enter light\_status.
- Data Type: Select Integer.
- Access Permissions: Select Read and Write.
- Value Range: Set it to 0 or 1. 0 indicates that the light is turned off. 1 indicates that the light is turned on.

Add Property		×
* Property Name	light_status	
Description		
	0/128 //	
★ Data Type	Integer ~	
* Access Permissions	Read Write	
★ Value Range	0 - 1	
Step		
Unit		
	Cancel OK	

### Figure 2-46 Adding a property - light\_status

**Step 5** Define the command used to remotely control the street light switch status.

- 1. Click Add Command. In the dialog box displayed, set Command Name to SWITCH\_LIGHT.
- 2. Click Add Command Parameter. Set Parameter Name to SWITCH\_LIGHT, Data Type to String, Length to 3, and Enumerated Value to ON,OFF. Click OK.

Property Name	SWITCH_LIGHT	
Description		
		0/128 //
k Data Type	String	~
Access Permissions	Read Write	
k Length	3	
Enumerated Values	ON.OFF	
		6/1,024 //

Figure 2-47 Adding a command parameter - SWITCH\_LIGHT

3. Click Add Response Parameter. Set Parameter Name to result and Data Type to Integer.

Parameter Name	result	
Description		
		0/128 //
Data Type	Integer	~
Value Range	0	- 65535
Step		
Unit		

#### Figure 2-48 Adding a response parameter - result

**Step 6** Click **OK**. The product model of the street light is created.

----End

# **Developing a Codec Online**

For purposes of power conservation, devices generally report binary data. The codec converts the binary data into JSON data based on properties defined in the product model so that the platform and application can identify the data. When an application remotely delivers a command, the platform converts the command in JSON format into binary and delivers the binary data to the device.

### **NOTE**

If a device reports data in JSON format, you do not need to define a codec.

- **Step 1** On the product details page of the street light, click the **Codec Development** tab and click **Develop Codec**.
- **Step 2** Click **Add Message** and configure a data reporting message as follows to report street light data:
  - Message Name: Enter LightData.
  - Message Type: Select Data reporting.
  - Add Response Field: Select this option. After a response field is added, the platform delivers the configured response data to the device when receiving data reported by the device.

• **Response**: Retain the default value **AAAA0000**.

Add Mes	ssage			
Basic Inform	nation			
Message Na	ame		Description	
LightData				
Message Ty	pe			
🔵 Data rep	orting 🔵 Command de	livery		
🗸 Add Res	sponse Field			0/1,024 %
Fields				Add Field
Offset	Field Name	Description	Data Type Length	h Tagged as Address Fi Operation
		N	lo table data available.	
		No F	ïelds data available. Add Field firs	st.
Response			Add Field	
	AAAA0000			
response				

# Figure 2-49 Adding a message

1. Click **Add Field** to add the **messageId** field, which indicates the message type.

# Figure 2-50 Adding a field - messageId

Add Field		×
	ed as address field, the field name is fixed at messageld alds cannot be set to messageld.	1.
✓ Tagged as address field	0	
★ Field Name	messageld	
Description	Enter	
	0/1,024 %	
Data Type (Big Endian)	int8u ~	
Offset	0-1	0
★ Length	1	0
Default Value	0x0	?
	Cancel	ж

2. Add the **LightIntensity** field to indicate the light intensity. Set **Data Type** to **int8u** (8-bit unsigned integer) and **Length** is **1**.

Add Field		×
Tagged as address field	0	
★ Field Name	LightIntensity	
Description	Enter	
	0/1,024 %	
Data Type (Big Endian)	int8u ~	
Offset	1-2	0
★ Length	1	0
Default Value		0
	Cancel	ок

# Figure 2-51 Adding a field - LightIntensity

3. Add the LightStatus field to indicate the street light switch status. Set Data Type to int8u (8-bit unsigned integer) and Length is 1.

Add Field		^
Tagged as address field	0	
★ Field Name	LightStatus	
Description	Enter	
	0/1,024	11
Data Type (Big Endian)	int8u ~	·
Offset	2-3	0
★ Length	1	0
Default Value		0
	Cancel	ок

### Figure 2-52 Adding a field - LightStatus

- **Step 3** Click **Add Message** again to develop a codec for command delivery messages.
  - Message Name: Enter SwitchStatus.
  - Message Type: Select Command delivery.
  - Add Response Field: Select this option. After a response field is added, the device reports the command execution result when receiving the command.

Basic Informa	tion			
Message Nar	ne		Description	
SwitchStatu	3			
Message Typ	e			
Data repo		lelivery		
Add Res	oonse Field			0/1,024 //
ields				Add Field
Offset	Field Name	Description	Data Type Length	Tagged as Address Fi Operation
Offset	Field Name	Description	Data Type Length	Tagged as Address Fi Operation
Offset	Field Name			Tagged as Address Fi Operation
Offset	Field Name	Νο		Tagged as Address Fi Operation
		Νο	table data available.	Tagged as Address Fi Operation
Offset		Νο	table data available.	Tagged as Address Fi     Operation

Figure 2-53 Adding a message - SwitchStatus

1. Click **Add Field** to add the **messageId** field, which indicates the message type.

# Figure 2-54 Adding a field - messageId

Add Field		×
	ed as address field, the field name is fixed at messagel Ids cannot be set to messageld.	d.
✓ Tagged as address field	0	
★ Field Name	messageld	
Description	Enter	
	0/1,024 ⁄⁄	
Data Type (Big Endian)	int8u V	
Offset	0-1	?
★ Length	1	0
Default Value	0x0	0
	Cancel	ок

2. Add the **mid** field to associate the delivered command with the command execution result.

# Figure 2-55 Adding a command field - mid

Add Field		X			
	When the field is tagged as response ID field, the field name must be fixed at mid. The names of other fields cannot be set to mid.				
Tagged as address field (	2				
✓ Tagged as response ID fie	ld ③				
★ Field Name	mid				
Description	Enter				
	0/1,024 //				
Data Type (Big Endian)	int16u ×				
Offset	1-3	0			
★ Length	2	?			
Default Value		?			
	Cancel	ок			

3. Add the SwitchStatus field. Set Data Type to string and Length is 3.

Add Field		×
Tagged as address field	0	
Tagged as response ID fie	eld 💿	
★ Field Name	SwitchStatus	
Description	Enter	
		0/1,024 1/
Data Type (Big Endian)	string	~
Offset	3-6	0
★ Length	3	0
Default Value		0
		Cancel OK

## Figure 2-56 Adding a command field - SwitchStatus

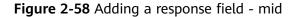
- 4. Click Add Response Field to configure command delivery responses.
  - Add the messageld field to indicate the message type. The command execution result is an upstream message, which is differentiated from the data reporting message by the messageld field.

# Figure 2-57 Adding a response field - messageId

Add Field		×
	ed as address field, the field name is fixed at message elds cannot be set to messageld.	ld.
✓ Tagged as address field	0	
Tagged as response ID fi	eld (?)	
Tagged as command exe	cution state field 🧿	
★ Field Name	messageld	J
Description	Enter	
	0/1,024 4	
Data Type (Big Endian)	int8u ~	)
Offset	0-1	0
★ Length	1	0
Default Value	0x2	0
	Cancel	ок

- Add the **mid** field to associate the delivered command with the command execution result.

.



Add Field		×
	ed as response ID field, the field name must be fixed at ner fields cannot be set to mid.	
<ul> <li>Tagged as address field</li> <li>Tagged as response ID fi</li> <li>Tagged as command exe</li> </ul>	eld (?)	
★ Field Name	mid	
Description	Enter 0/1,024 //	
Data Type (Big Endian)	int16u 🗸	
Offset	1-3	0
★ Length	2	0
Default Value		0
	Cancel	ок

Add the errcode field to indicate the command execution status. 00 indicates success and 01 indicates failure. If this field is not carried, the command is executed successfully by default.

# Figure 2-59 Adding a response field - errcode

Add Field		×
_	ed as command execution state field, the field name is names of other fields cannot be set to errcode.	
Tagged as address field	0	
Tagged as response ID field	eld (?)	
✓ Tagged as command exe	cution state field 🧿	
★ Field Name	errcode	
Description	Enter	
	0/1,024 %	
Data Type (Big Endian)	int8u ~	
Offset	3-4	0
★ Length	1	0
Default Value		0
	Cancel	ок

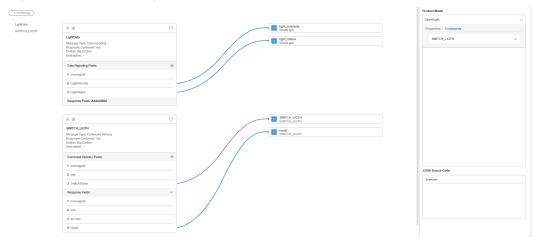
- Add the **result** field to indicate the command execution result.

Figure 2-60 Adding	а	response	field -	· result
--------------------	---	----------	---------	----------

Add Field		
Tagged as address field	0	
Tagged as response ID fi	eld 🧿	
Tagged as command exe	ecution state field (?)	
★ Field Name	result	
Description	Enter	
		0/1,024 //
Data Type (Big Endian)	int8u	~
Offset	4-5	
★ Length	1	

**Step 4** Drag the property and command fields (defined in the product model) in the **Product Model** area on the right to map the fields in the data reporting and command delivery messages defined by the codec.

Figure 2-61 Developing a codec online



**Step 5** Click **Save** in the upper right corner and click **Deploy** to complete codec deployment.

----End

# Debugging the Codec Online Using a Physical Device

Simulators debug the functions of devices and applications. You can debug the defined product model and codec by simulating data reporting and command delivery.

- Step 1 On the product details page of the street light, click Online Debugging and click Add Test Device.
- **Step 2** In the dialog box displayed, set the parameters and click **OK**.
  - **Device Type**: Select **Physical device**.
  - **Device Name**: Customize a name.
  - **Node ID**: Enter the IMEI of the device. The node ID will be carried by the device for device access authentication. You can view the node ID on the NB-IoT module.
  - Secret: If DTLS is used for access, keep the secret secure.

Figure 2-62 Adding a test device

Add Test Device			×
Device Type	Physical device	Virtual device	
Device Name			
* Node ID			
Authentication Type	Secret		
Secret			8
Confirm Secret			8
		Cancel	ОК

#### D NOTE

The newly added device is in the inactive state. In this case, online debugging cannot be performed. For details, see **Device Connection Authentication**. After the device is connected to the platform, perform the debugging.

**Step 3** Click **Debug** to access the debugging page.

#### Figure 2-63 Entering debugging

IoTDA Instances / Products /						
<   mmm.220	Registered devices:					Quick Links
Basic Information Code	Conline Debugging					
Add Test Device						
Q Search by node ID by o	lefault.					00
Status ⊖	Device Name \ominus	Node ID \ominus	Device ID $\ominus$	Device Type $ \Theta $	Operation	
Online	approximation of the continuance	10000000	And the country of the second sector of the	Virtual	Debug Delete	
Total Records: 1 10 V						

Step 4 Simulate a scenario where a control command is remotely delivered. In Application Simulator, set Service to StreetLight, Command to SWITCH\_LIGHT, and Command Value to ON, and click Send. The street lamp is turned on.

----End

## Debugging the Codec Online Using a Virtual Device

- **Step 1** On the product details page of the street light, click **Online Debugging** and click **Add Test Device**.
- **Step 2** In the dialog box displayed, select **Virtual device** and click **OK**. The virtual device name contains **DeviceSimulator**. Only one virtual device can be created for each product.

Figure 2-64 Creating a virtual device

Add Test De	vice		×
Device Type	Physical device	Virtual device	
Device Name	21,48171871171486.2	DeviceSimulator	
* Node ID	1726253780179		
		Cancel	ж

#### **Step 3** Click **Debug** to access the debugging page.

Figure 2-65 Entering debugging

IelDA htstores / Peduds /								
Basic Information Code: Deployment Online Debugging								
Add Test Device								
C. Search by rode 1D by default.						00		
Status ⊖	Device Name \ominus	Node ID \ominus	Device ID (e)	Device Type \ominus	Operation			
Online	aparter on's development	121424703	60000c/0010c/10ca466c_1001401000	Virtual	Debug Delete			
Total Records: 1 10 V (1)								

Step 4 Simulate a data reporting scenario. Specifically, under Device Simulator, enter the hexadecimal code stream 002000, and click Send. (The first byte 00 indicates the message ID, the second byte 20 indicates the light intensity, and the third byte 00 indicates the switch status.) Under Application Simulator, "Light\_Intensity": 32, "Light\_Status": 0 (converted JSON data) should be displayed.

Figure 2-66 Simulating data reporting

Debug output	✓ Real-Time Refresh × Clear Application Simulator Log	Application Simulator Device Simulator
Application Simulator	Platform Delivery Device Simulator	Property Report The equipment simulator can report data to the platform according to the product definition
As Received Sent     C. Received Jul Do. 2024 15:02:09:080 CMT1-09:00     (serviceld: StreetLight, data: ("light_intensity":32,"light_istatus":0))	Commands Received     Data Reported     Commands Received Jul 08, 2024 15:02:09 GMT+98:09     AAAA000     Data sees Jul 08, 2024 15:02:08 GMT+98:00     062000	
		Period (s): 5 Auto-Send Send

Step 5 Simulate a command delivery scenario. Specifically, under Application Simulator, select StreetLight for Service, SWITCH\_LIGHT for Command, ON for SWITCH\_LIGHT, and click Send. Under Device Simulator, 0100014F4E (hexadecimal number converted from the ASCLL code) should be displayed.

Figure 2-67 Simulating command delivery

Debug output	Real-Time Refresh X Clear Application Simulator Log	Application Simulator Device Simulator
Application Simulator	tform	Commands Sent The application simulator can issue commands to the device according to the product definition. If you use applicably deviceped codec play-in, please carry all the fields defined in the play-in
All Received Sent	AI Commands Received Data Reported	when issuing a command to obtain correct coding results and the length of each command must be less than 512 bytes.
<ul> <li>Sent Jul 06, 2024 16.08:02 GMT+08:00</li> <li>Message Body: ("service_id": "StreeLlight," command_name": "SWITCH_UGTH", "pares"; ("SWITCH_UGTH: "ON"), "send_strategy": "Immediated": "scope imme": 0 1</li> </ul>	Commands Received Jul 06, 2024 16:08:02 GMT+08:00 0100014F4E	Service StreetLigth  Command SWITCH LIGTH
Received Jul 06, 2024 16:04:24.813 GMT+08:00     (serviceid: StreetLigth, data: {"light_intenaity"32,"light_status".0})	AAAA0000  Data Sent Jul 06, 2024 16:04:23 GMT+08:00	SWITCH_LI ON
	002000	Pending Period (s) 2880 Cache Sand

----End

# **Debugging Using an Offline Simulator**

The **NB-IoT device simulator** is used to simulate the access of NB-IoT devices (devices using LwM2M over CoAP) to the platform for data reporting and command delivery.

- **Step 1** Obtain the node ID and secret generated during device registration in **2**.
- **Step 2** Download and decompress the NB-IoT device simulator, and double-click **NB-IoTDeviceSimulator\_zh.jar** to run the simulator.

## D NOTE

- The simulator can run on Windows, but not macOS.
- Ensure that the JDK has been installed. Otherwise, the JAR file cannot be executed.

Size	Packed	Туре	Modified	CRC32
		File folder		
		File folder	10/26/2020 9:5	
4,464,721	4,062,301	JAR File	8/7/2017 2:07	B47CBF61
1,203	607	PROPERTIES File	7/29/2017 12:3	01A69483
111	108	PROPERTIES File	8/7/2017 3:21	3CEB9C9C
	4,464,721 1,203	4,464,721 4,062,301 1,203 607	File folder           File folder           File folder           4,464,721         4,062,301           J,203         607           PROPERTIES File	File folder           File folder         10/26/2020 9:5           4,464,721         4,062,301         JAR File         8/7/2017 2:07           1,203         607         PROPERTIES File         7/29/2017 12:3

The package contains the following files:

- NB-IoTDeviceSimulator\_en.jar: simulator
- Californium.properties: simulator configuration file
- **setting.properties**: configuration file for connecting the simulator to the platform
- Step 3 When a message is displayed requesting you to confirm whether to enable DTLS encrypted transmission, click No if a secret is not entered during device registration, or Yes if a secret is entered.

Confirm	×
?	Are you sure you want to enable DTLS encryption transmission?
	Yes(Y) No(N)

**Step 4** Enter **IP address**, **VerifyCode**, and **psk**, and click **Register Device** to bind the simulator to the platform.

Note: If DTLS encryption transmission is disabled, you do not need to enter a PSK.

Set the three parameters as follows:

- IP address: access domain name displayed on the IoTDA console. (You can also use the IP address. You can run the ping [domain name] command to obtain the IP address.)
- VerifyCode: device ID, for example, aaaaa11111.
- **psk**: secret set during device registration, for example, **aaaaa11111aaaaa**.



On the IoTDA console, choose **Devices** > **All Devices** and view the device status. If the device is displayed as **Online**, the simulator is bound to the platform.

Step 5 Simulate a data reporting scenario. Specifically, under the NB-IoT device simulator, enter the hexadecimal code stream 001400, and click Send Data. (The first byte 00 indicates the message ID, the second byte 14 indicates the light intensity, and the third byte 00 indicates the switch status.)

NB-IoT Device Simulator	_	Х
IP address: Jaweicloud.com VerifyCode: PSK:	Register Device	
Log:		
The data reported by the simulator is as follows:001400		1
CIG IP: 119.3.250.80, PORT: 5684		
The data reported by the simulator is as follows:001400	=	
CIG IP: 119.3.250.80, PORT: 5684	=	
The data delivered by the CIG is as follows:□□	=	
CIG IP: 119.3.250.80, PORT: 5684	=	
The data delivered by the CIG is as follows:□□		
	=	-
Hexadecimal Code Stream JSON		
Hexadecimal message:		
001400		
Send Data		
Enable command listening:	O No	

The data is reported successfully. You can return to the console and view the latest reported data on the device details page of **aaaaa11111**. The latest reported data is **"Light\_Intensity": 20, "Light\_Status": 0**.

**Step 6** Simulate the remote command delivery scenario. On the console, locate the target product and click the product to go to the product details page. Click the **Online Debugging** tab and click **Debug** on the right of the **aaaaa11111** device.

Under **Application Simulator**, select **StreetLight** for **Service**, **SWITCH\_LIGHT** for **Command**, **ON** for **SWITCH\_LIGHT**, and click **Send**.

The data delivered by the CIG is **ON** in the log area, and the simulator displays a message indicating that the downstream message from the platform is received and asking whether to respond. Click **Yes**. On the **Application Simulator** page of the IoTDA console, you can see that the command is in the **Delivered** state.

Log:		
The data delivered by the CIG is as follows: OFF		-
CIG IP: 119.3.250.80, PORT: 5684		
The data delivered by the CIG is as follows: ON		
===================Confirm	×	
CIG IP: 119.3.250.80, The downlink message from the platform is rece	ived Respond?	
The data delivered by th	wear respond	
Yes(Y) No(N)		
CIG IP: 119.3.250.80,		
The data delivered by the CIG is as follows: ON		
		=
		•

## 

Strings are parsed using ASCII in the codec. You need to deliver printable characters. Non-printable characters are not displayed in the simulator.

----End

# 2.5 Developing a Protocol Conversion Gateway for Access of Generic-Protocol Devices

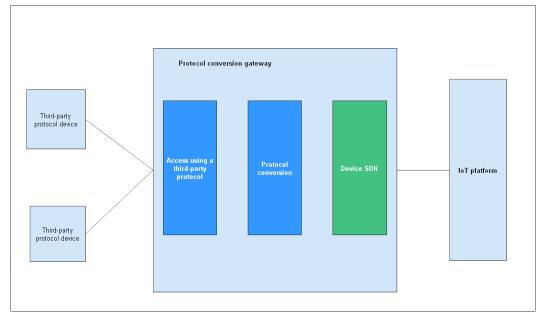
# Scenarios

Currently, the IoT platform supports only standard protocols such as MQTT, HTTP, and LwM2M. If a device uses other protocols (referred to as third-party protocols), it cannot access the platform directly.

To address this issue, protocol conversion must be performed outside the platform. It is recommended that a gateway be used to convert third-party protocols into MQTT. This gateway is called the protocol conversion gateway.

## **Implementation Principle**

The figure below shows the overall architecture of the solution.

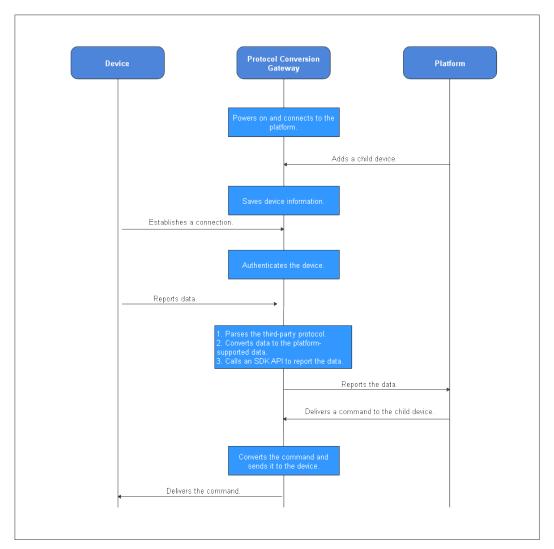


The protocol conversion gateway can be deployed in the cloud or locally. A thirdparty protocol device is connected to the platform as a child device of the protocol conversion gateway.

The protocol conversion gateway consists of the following components:

- 1. Third-party protocol access component: This component parses and authenticates third-party protocols.
- 2. Protocol conversion component: This component converts between third-party protocol data and platform data.
  - In the upstream direction, the component converts third-party protocol data into platform-supported data and invokes the device SDK APIs to report the data.
  - In the downstream direction, the component, after receiving data from the platform, converts the data into third-party protocol data and forwards the data to third-party protocol devices.
- 3. Device SDK component: The component is the device access SDK provided by the platform and provides common gateway functions. You can implement your own gateways based on the SDK.

# **Service Flow**



- 1. Register a gateway with the platform. For details, see **Device Authentication**.
- 2. Power on the gateway and connect it to the platform. Obtain the authentication parameters required for connection during gateway registration.
- 3. Register a child device on the IoTDA console. The platform delivers a child device addition event to the gateway. The gateway saves the child device information and makes the information persistent. (The SDK provides the default persistent implementation.)
- 4. The child device connects to the gateway, and the gateway authenticates the child device.
- 5. The child device reports data to the gateway. The gateway converts the data to the format supported by the platform and then calls an SDK API for reporting child device properties or messages to the platform.
- 6. The platform delivers a command to the child device. The gateway converts the command into a command compliant with the third-party protocol and forwards it to the child device. The child device processes the command.

# Implementation of Protocol Conversion Gateway

For details on the implementation and usage of the gateway, see **IoT Device SDK** (Java) and **IoT Device SDK** (C).

# 2.6 Connecting a Device That Uses the X.509 Certificate Based on MQTT.fx

This section uses MQTT.fx as an example to describe how to connect devices to IoTDA using the native MQTT protocol. MQTT.fx is a widely used MQTT client. Using MQTT.fx, you can easily verify whether devices can interact with IoTDA to publish or subscribe to messages.

An X.509 certificate is a digital certificate used for communication entity authentication. IoTDA allows devices to use their X.509 certificates for authentication. The use of X.509 certificate authentication protects devices from being spoofed.

# Prerequisites

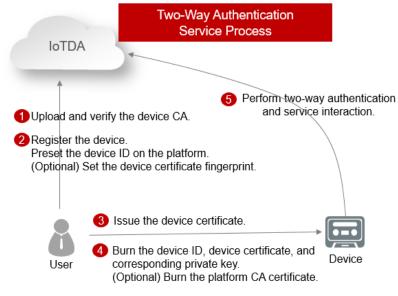
- You have registered a Huawei Cloud account.
- You have subscribed to IoTDA. If you have not subscribed to the service, go to the IoTDA service page, and click Access Console to subscribe to the service.

# Constraints

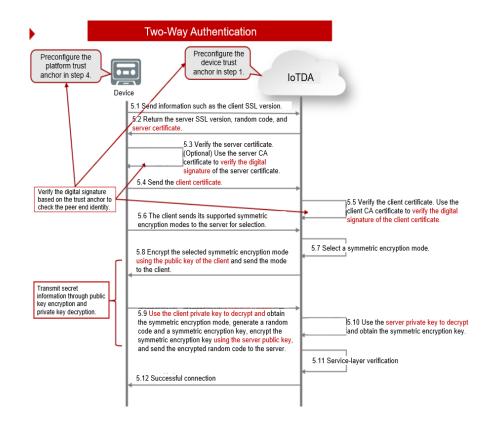
- Only MQTT devices can use X.509 certificates for identity authentication.
- You can upload a maximum of 100 device CA certificates.

# X.509-based Authentication

• The complete two-way certificate authentication is as follows:



• Two-way certificate authentication:



# **Obtaining Device Access Information**

Perform the following procedure to obtain device access information on the IoTDA console:

- **Step 1** Log in to the **IoTDA console**.
- **Step 2** In the navigation pane, choose **Overview** and click **Access Details** in the **Instance Information** area to check the device access information and record domain names and ports.

Figure 2-68 Obtaining access information



## 

For devices that cannot be connected to the platform using a domain name, run the **ping** *Domain name* command in the CLI to obtain the corresponding IP address. Then you can connect the devices to the platform using the IP address. The IP address is variable and needs to be set using a configuration item.

----End

# **Creating a Product**

**Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.

Figure	2-69	Creating	а	product
--------	------	----------	---	---------

< 🙆 feel	Nandard	Running					R Details ∂⊭ Modify ···
Overview Products		Products					Description
Devices Rules O&M Resource Spaces Documentation	× ×	access protocol and data format. Learn more  1 Create Products	age and control the device on the platform, you need to 2 Defining Product Mi		3 Register Devices		4 Device-side development
		A product is a set of devices that have the same of Product Development Gate	uppointer of Hasures. Lettine product modes	to describe the capabilities and features of devices.	Ine registered device to the platform.	obtains the identity information required to connect	Integrates device SDKs to connect devices to the Device Devicement Guide
		Product Name θ	Product ID $\Theta$	Resource Spaces $\Theta$	Device Type $\Theta$	Protocol 🖯 Created 🖯	Operation

#### Step 2 Set the parameters as prompted and click OK.

Basic Information					
Resource Space	The platform automatically allocates the created product to the default resource space. You can also select an existing resource space from the drop-down list or <b>create one</b> .				
ProductCustomize the value. The name can contain letters, numbers, underscores (_), and hyphens (-).					
Protocol	Select <b>MQTT</b> .				
Data Type	Select <b>JSON</b> .				
Industry	Set this parameter as required.				
Device Type	Set this parameter as required.				
Advanced Settings					
Product IDSet a unique identifier for the product. If this parameter is specified, the platform uses the specified product ID. If this parameter is not specified, the platform allocates a product					

Х

#### Figure 2-70 Creating a product - MQTT

Create Product	
★ Resource Space ⑦	To create a new resource space, you can go to the instance
★ Product Name	details page.
Protocol 🧿	(MQTT ~
🗙 Data Type	JSON V
Device Type Selection	Standard profile Custom
★ Device Type ⑦	
Advanced Settings \land	Custom Product ID   Description
Product ID 💿	
Description	
	0/128 ⁄⁄
	Cancel OK

----End

# **Developing a Product Model**

- **Step 1** Click the created product. The product details page is displayed.
- Step 2 On the Basic Information tab page, click Customize Model to add services of the product.
- **Step 3** Add the **Connectivity** service.
  - 1. On the Add Service page, configure Service ID, Service Type, and Description, and click OK.
    - Service ID: Enter Connectivity. \_
    - Service Type: You are advised to set this parameter to the same value as \_ Service ID.

 $\times$ 

#### - **Description**: Enter **Connectivity**.

Figure 2-71	Adding a	a service -	Connectivity
-------------	----------	-------------	--------------

Add	Se	rvice

* Service ID	Connectivity	
Service Type	Connectivity	0
Description	Connectivity	
	12/128 /	5
	Cancel	ок

- 2. Choose **Connectivity**, click **Add Property**, enter related information, and click **OK**.
  - Property Name: Enter dailyActivityTime.
  - Data Type: Select Integer.
  - Access Permissions: Select Read.
  - Value Range: Set it to 0 to 65535.
  - Step: Enter 1.
  - Unit: Enter s.

Add Property		×
* Property Name	dailyActivityTime	
Description		
	0/128 %	
* Data Type	Integer V	
* Access Permissions	Read Write	
★ Value Range	0 – 65535	
Step		
Unit		
	Cancel OK	

Figure 2-72 Adding a property - dailyActivityTime

- Step 4 Add the Battery service.
  - 1. On the **Basic Information** tab page, click **Add Service**, configure **Service ID**, **Service Type**, and **Description**, and click **OK**.
    - Service ID: Enter Battery.
    - Service Type: You are advised to set this parameter to the same value as Service ID.
    - **Description**: Enter **Battery**.

1

#### Figure 2-73 Adding a service - Battery

Add Service		×
* Service ID	Battery	
Service Type	Battery	0
Description	Battery	
	7/128 //	
	Cancel	ок

- 2. Choose **Battery**, click **Add Property**, enter related information, and click **OK**.
  - Property Name: Enter batteryLevel.
  - Data Type: Select Integer.
  - Access Permissions: Select Read.
  - Value Range: Set it to 0–100.
  - Step: Enter 1.
  - **Unit**: Enter %.

Add Property	×
* Property Name	batteryLevel
Description	
	0/128 //
★ Data Type	Integer ~
* Access Permissions	Read Write
★ Value Range	0 – 100
Step	1
Unit	%
	Cancel

Figure 2-74 Adding a property - batteryLevel (Battery)

----End

# Uploading a Device CA certificate

- Step 1 In the navigation pane, choose Devices > Device Certificates. On the Device CA Certificates tab page, select a resource space and click Upload Certificate.
- Step 2 In the displayed dialog box, click Select File to add a file, and then click OK.

Figure 2-75 Device CA certificate - Uploading a certificate

Upload Certific	ate	×
★ CA Certificate 🧿	rootCA.pem (3.56KB)	× Select File
		Cancel OK

## D NOTE

- Device CA certificates are provided by device vendors. You can prepare a commissioning certificate during commissioning. For security reasons, you are advised to replace the commissioning certificate with a commercial certificate during commercial use.
- CA certificates cannot be used to verify server certificates upon expiration. Replace these certificates before expiration dates to ensure that devices can connect to the IoT platform properly.

----End

# Making a Device CA Commissioning Certificate

This section uses the Windows operating system as an example to describe how to use OpenSSL to make a commissioning certificate. The generated certificate is in PEM format.

- 1. Download and install **OpenSSL**.
- 2. Open the CLI as user **admin**.
- 3. Run **cd c:\openssl\bin** (replace **c:\openssl\bin** with the actual OpenSSL installation directory) to access the OpenSSL view.
- 4. Generate a public/private key pair. openssl genrsa -out rootCA.key 2048
- 5. Use the private key in the key pair to generate a CA certificate. openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem

The system prompts you to enter the following information. All the parameters can be customized.

- Country Name (2 letter code) [AU]: country, for example, CN
- State or Province Name (full name) []: state or province, for example, GD
- Locality Name (for example, city) []: city, for example, SZ
- Organization Name (for example, company) []: organization, for example, Huawei
- Organizational Unit Name (for example, section) []: organization unit, for example, IoT
- Common Name (e.g. server FQDN or YOUR name) []: common name, for example, zhangsan
- Email Address []: email address, for example, 1234567@163.com

Obtain the generated CA certificate **rootCA.pem** from the **bin** folder in the OpenSSL installation directory.

# **Uploading a Verification Certificate**

If the uploaded certificate is a commissioning certificate, the certificate status is **Unverified**. In this case, upload a verification certificate to verify that you have the CA certificate.

 Device Certificates

 Device Certificates

 Or or replace file device OA certificates to the publitem for device access authentication.

 To can uplace a maximum of 100 certificate

 Output

 Uplace Certificates

 Output

 Output</td

Figure 2-76 Device CA certificate - Unverified certificate

The verification certificate is created based on the private key of the device CA certificate. Perform the following operations to create a verification certificate:

**Step 1** Obtain the verification code to verify the certificate.

Figure 2-77 Device CA certificate - Verifying a certificate

Device Certificates						
Device CA Certificates Device Ce	rtificates					
You can upload device CA certificates to the You can upload a maximum of 100 certification						
Upload Certificate						С
Verification Status	Certificate ID	Certificate Owner	Created	Valid Till	Operation	
Unverified	100010-0-0040-011-005-0011	Drofala and Draws 2-DC	Sec. 8, 201 1 (0.4) (0.4"-0.10	$M_{\rm H} > 10.111 \pm 0.011 \pm 0.07 \pm 0.01$	Delete Download	
Certificate ID	1000-0015	Certificate Owner	On of all as and in the second second	In France Court Concentration Product Co.		
Valid From	17-18-18	Verification Certificate ③	Upload Verification Certificate			

Figure 2-78 Device CA certificate - Obtaining the verification code

	103-4644-1475-0887888147	/erification Code 🧿
Select File		/erification Certificate
Select File		/erification Certificate

- **Step 2** Generate a key pair for the verification certificate. openssl genrsa -out verificationCert.key 2048
- **Step 3** Create a certificate signing request (CSR) for the verification certificate. openssl req -new -key verificationCert.key -out verificationCert.csr

The system prompts you to enter the following information. Set **Common Name** to the verification code and set other parameters as required.

- Country Name (2 letter code) [AU]: country, for example, CN
- State or Province Name (full name) []: state or province, for example, GD
- Locality Name (for example, city) []: city, for example, SZ
- Organization Name (for example, company) []: organization, for example, Huawei

1

- Organizational Unit Name (for example, section) []: organization unit, for example, IoT
- Common Name (e.g. server FQDN or YOUR name) []: verification code for verifying the certificate. For details on how to obtain the verification code, see Step 1.
- Email Address []: email address, for example, 1234567@163.com
- Password[]: password, for example, 1234321
- Optional Company Name[]: company name, for example, Huawei

```
Step 4 Use the CSR to create a verification certificate.
openssl x509 -req -in verificationCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out verificationCert.pem -days 500 -sha256
```

Obtain the generated verification certificate **verificationCert.pem** from the **bin** folder of the OpenSSL installation directory.

**Step 5** Locate the target certificate, click , and click **Upload Verification Certificate**.

#### Figure 2-79 Device CA certificate - Verifying a certificate

Device Certificates						
Device CA Certificates Device Certifi	cates					
You can upload device CA certificates to the pi You can upload a maximum of 100 certificates						
Upload Certificate						C
Verification Status	Certificate ID	Certificate Owner	Created	Valid Till	Operation	
Unverified	4703015-1618-4815-1866-1875	(N-10 <sup>1</sup> al-area) 2-have 2-040.	14 10, 201 1 (0.4) (0. <sup>4</sup> - 0.1)	$M_{\rm H} > 10.211 \pm 0.011 \pm 0.07 \pm 0.01$	Delete Download	
Certificate ID	NATION 10171	Certificate Owner	Christiana and Dimana 2-CM	In France Deal Dev Mercel Product Dr		
Valid From	-10.00	Verification Certificate ③	Upload Verification Certificate			

**Step 6** In the displayed dialog box, click **Select File** to add a file, and then click **OK**.

Figure 2-80 Device CA certificate - Uploading a verified certificate

Upload Verification	on Certificate	×	
Verification Code 🕥	1986 (1771) dise (a <sup>1</sup> 1) (1871) a <sup>1</sup>		
★ Verification Certificate	verificationCert.pem (3.56KB)	X Select File	
		Cancel	

After the verification certificate is uploaded, the certificate status changes to **Verified**, indicating that you have the CA certificate.

----End

# Presetting an X.509 Certificate

Before registering an X.509 device, preset the X.509 certificate issued by the CA on the device.

## D NOTE

The X.509 certificate is issued by the CA. If no commercial certificate issued by the CA is available, you can **create a device CA commissioning certificate**.

#### Creating an X.509 Commissioning Certificate

- Run cmd as user admin to open the CLI and run cd c:\openssl\bin (replace c:\openssl\bin with the actual OpenSSL installation directory) to access the OpenSSL view.
- 2. Generate a public/private key pair. openssl genrsa -out deviceCert.key 2048
- 3. Create a CSR. openssl req -new -key deviceCert.key -out deviceCert.csr

The system prompts you to enter the following information. All the parameters can be customized.

- Country Name (2 letter code) [AU]: country, for example, CN
- State or Province Name (full name) []: state or province, for example, GD
- Locality Name (for example, city) []: city, for example, SZ
- Organization Name (for example, company) []: organization, for example, Huawei
- Organizational Unit Name (for example, section) []: organization unit, for example, IoT
- Common Name (e.g. server FQDN or YOUR name) []: common name, for example, zhangsan
- Email Address []: email address, for example, 1234567@163.com
- Password[]: password, for example, 1234321
- Optional Company Name[]: company name, for example, Huawei
- 4. Create a device certificate using CSR. openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out deviceCert.pem -days 500 -sha256

Obtain the generated device certificate **deviceCert.pem** from the **bin** folder in the OpenSSL installation directory.

## Registering a Device Authenticated by an X.509 Certificate

- Step 1 Log in to the IoTDA console.
- **Step 2** In the navigation pane, choose **Devices** > **All Devices**, click **Register Device**, set parameters based on the table below, and click **OK**.

Register Device			×
* Resource Space 💿			~
* Product			~
* Node ID 💿			
Device ID 💿			
Device Name			
Description			
			0/2,048 🕢
Authentication Type (	Secret	X.509 certificate	
Fingerprint			
			Cancel OK

# Figure 2-81 Device - Registering an X.509 device

Parameter	Description
Resource Space	Select the resource space to which a device belongs.
Product	Select the product to which the device belongs. If no product is available, <b>create a product</b> first.
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the device name.
Authenticatio n Type	<b>X.509 certificate</b> : The device uses an X.509 certificate for identity verification.
Fingerprint	This parameter is displayed when <b>Authentication Type</b> is set to <b>X.509 certificate</b> . Import the fingerprint corresponding to the <b>preset device certificate on the device side</b> . You can run <b>openssl x509 -fingerprint -sha256 -in deviceCert.pem</b> in the OpenSSL view to query the fingerprint. <b>Note: Delete the colon</b> (:) from the obtained fingerprint when filling it.

----End

# **Performing Connection Authentication**

You can activate the device registered with the platform by using MQTT.fx. For details, see **Device Connection Authentication**.

Step 1 Download MQTT.fx (64-bit OS) or MQTT.fx (32-bit OS) and install it.

**NOTE** 

- Install the latest MQTT.fx tool. Download it.
- MQTT.fx 1.7.0 and earlier versions have problems in processing topics containing \$. Use the latest version for test.
- **Step 2** Go to the **IoTDA client ID generator page**, enter the device ID and secret generated after **registering a device** to generate connection information (including **ClientId**, **Username**, and **Password**).

#### **NOTE**

You can set **DeviceSecret** to any value, for example, **12345678**.

#### Figure 2-82 Obtaining ClientId

5free300000000000000000000000000000000000	5/0000000000000000000000000000000000000				
Device Secret	Device Secret				
18888888899	120000009				
Generale					
Client ID	566-00000000000000000000000000000000000				
Username	5/6				
Password	81-000000000000000000000000000000000000				

Parame ter	Mand atory	Туре	Description
ClientId	Yes	String(2 56)	<ul> <li>The value of this parameter consists of a device ID, device type, password signature type, and timestamp. They are separated by underscores (_).</li> <li>Device ID: A device ID uniquely identifies a device and is generated when the device is registered with IoTDA. The value usually consists of a device's product ID and node ID which are separated by an underscore (_).</li> <li>Device type: The value is fixed at 0, indicating a device ID.</li> <li>Password signature type: The length is 1 byte, and the value can be 0 or 1.</li> </ul>
			<ul> <li>D: The timestamp is not verified using the HMAC-SHA256 algorithm.</li> </ul>
			<ul> <li>- 1: The timestamp is verified using the HMAC-SHA256 algorithm.</li> </ul>
			<ul> <li>Timestamp: The UTC time when the device was connected to IoTDA. The format is YYYYMMDDHH. For example, if the UTC time is 2018/7/24 17:56:20, the timestamp is 2018072417.</li> </ul>
Userna me	Yes	String(2 56)	Device ID.

Each device performs authentication using the MQTT CONNECT message, which must contain all information of the client ID. After receiving a CONNECT message, IoTDA checks the authentication type and password digest algorithm of the device.

The generated client ID is in the format *Device ID\_0\_0\_Timestamp*. By default, the timestamp is not verified.

- If the timestamp needs to be verified using the HMAC-SHA256 algorithm, the platform checks whether the message timestamp is consistent with the platform time and then checks whether the password is correct.
- If the timestamp does not need to be verified using the HMAC-SHA256 algorithm, the timestamp must also be contained in the CONNECT message, but the platform does not check whether the time is correct. In this case, only the password is checked.

If the authentication fails, the platform returns an error message and automatically disconnects the MQTT connections.

**Step 3** Open the MQTT.fx tool and click the setting icon.

# Figure 2-83 Settings

ile Extras	; Help					
iot				- 🔅	Connect	Disconnect
Publish	Subscribe	Scripts	Broker Status	Log		
/huawe	i/v1/devices/ <b>a</b>	<b>2793056-</b> 34	2 <b>0-450 0</b> 1-0fe9	256		

**Step 4** Enter **Connection Profile** information.

Figure 2-84 Using	default settings	for parameters o	n the General tab page
-------------------	------------------	------------------	------------------------

Profile Name	local mosquitto	
Profile Type	MQTT Broker	
MQTT Broker Profile Settings		
Broker Address	a1604	
Broker Port	8883	
Client ID	61	Generate
General User Credentials	SSL/TLS Proxy LWT	
Connection Timeout	30	
Keep Alive Interval	60	
Clean Session	✓	
Auto Reconnect		
Max Inflight	10	
- (	✓ Use Default	
MQ11 Version		
	3.1.1	
	Clear Publish History	
	Clear Subscription History	
Revert	Cancel	OK Apply

Parameter	Description
Broker Address	Enter the <b>device access address</b> (domain name) obtained from the IoTDA console. If the device cannot be connected using a domain name, enter the IP address obtained in <b>2</b> .
Broker Port	Enter <b>8883</b> .
Client ID	Enter the device client ID obtained in <b>2</b> .

#### Step 5 Click User Credentials and specify User Name.

## Figure 2-85 Entering the device ID

Profile Name	local mosquit	to			
Profile Type	MQTT Broke	r	•		
MQTT Broker Profile Settings					
Broker Address	••••••	~~~~		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
Broker Port	8883				
Client ID		*****			Generate
General User Credentials	SSL/TLS F	roxy	LWT		
User Name					
Password				J	
Revert				Cancel	OK Apply

Parameter	Description
User Name	Enter the device ID obtained in <b>2</b> .
Password	Leave it blank when the X.509 certificate is used for authentication.

Step 6Click SSL/TLS, set authentication parameters, and click Apply. Select Enable SSL/<br/>TLS, select Self signed certificates, and enter the certificate information.

	· ·	
Profile Name	local mosquitto	
Profile Type	MQTT Broker	
MQTT Broker Profile Settings		
Broker Address	*******	
Broker Port	8883	
Client ID		Generate
General User Credentials	SSL/TLS Proxy LWT	
Enable SSL/TLS	Protocol TLSv1.2	•
<ul> <li>CA signed server certificate</li> </ul>		
CA certificate file		
<ul> <li>CA certificate keystore</li> </ul>		
Self signed certificates (2)		
CA File		ra\DigiCt 3
Client Certificate File		
Client Key File	Cert.key	5
Client Key Password		
PEM Formatted	✓ 🚯	
Self signed certificates in keystore	5	
Revert	Cancel	ОК Apply

#### Figure 2-86 Setting SSL/TLS parameters

### **NOTE**

**CA File**: corresponding CA certificate. Download the certificate from **Obtaining Resources** and load the PEM certificate.

Client Certificate File: device certificate (deviceCert.pem).

**Client Key File**: private key (deviceCert.key) of the device.

**Step 7** Click **Connect**. If the device authentication is successful, the device is displayed online on the platform.

#### Figure 2-87 Device list - Device online status

< 🖸 usernessag	K 🚺 unemensagelinit years taken v O Running 🛛 R Details & Monty …							
Overview	All Devices Total devices	All Devices Tetal devices 2.   Activated devices 11   Chine devices 1  C Quick Links  C Quick Links						
Products Devices	Device List Batch i	Device List Batch Registration Batch Update Batch Deletion Batch Add Devices To Group File Uptoads						
All Devices	Register Device	Register Device Diside Definition Frence						
Groups	Q. Search by node ID by	C Search by node 10 by default.						
Policies	Status 🖯	Device Name 😣	Node ID 😔	Device ID (B)	Resource Space 😔	Product O	Node Type 😣	Operation
Software/Firmware Upgrades	Online	1412	-	1947/1947/1019au 751	Delectrice, Millions	Strains	Drafty consolid	View Debug More ~



## **Reporting Data**

Use MQTT.fx to report data to the platform. For details, see **Reporting Device Properties** in the *API Reference*.

If the device reports data through the MQTT channel, the data must be sent to the specified topic. The format of the topic for reporting the data is **\$oc/devices/** 

*{device\_id}***/sys/properties/report**, where *{device\_id}* is the device ID returned after device registration.

Step 1 Enter the API address in the format of "\$oc/devices/{device\_id}/sys/properties/ report", for example, \$oc/devices/5e4e2e92ac-164aefa8fouquan1/sys/ properties/report.

Figure 2-88 Entering the API address

File Extras Help
iot - Connect Disconnect
Publish Subscribe Scripts Broker Status Log
» \$oc/devices/5e4e2e9
* \$oc/devices/5e4e2e997e *service_id***Temperature*

**Step 2** Enter the data to report.

#### **Request parameters**

Paramet er	Mandat ory	Туре	Description
services	Yes	List <servicepr operty&gt;</servicepr 	Service data list. (For details, see the <b>ServiceProperty</b> structure below.)

#### ServiceProperty structure

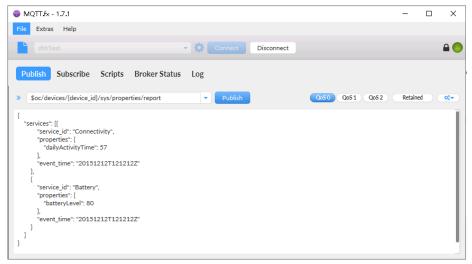
Paramet er	Manda tory	Туре	Description
service_id	Yes	String	Service ID.
propertie s	Yes	Object	Service properties, which are defined in the product model of the device.
event_tim e	No	String	UTC time when the device reports data. The format is yyyyMMddTHHmmssZ, for example, <b>20161219T114920Z</b> .
			If this parameter is not carried in the reported data or is in incorrect format, the time when IoTDA receives the data is used.

#### **Example request**

. "services": [{

```
"service_id": "Connectivity",
    "properties": {
        "dailyActivityTime": 57
     },
     "event_time": "20151212T121212Z"
     },
     {
        "service_id": "Battery",
        "properties": {
            "batteryLevel": 80
        },
        "event_time": "20151212T121212Z"
     }
     ]
}
```

Figure 2-89 Example request



**Step 3** Click **Publish**. Then you can check whether the device successfully reports data on the platform.

ces / All D < | ffffffff Online ① C Quick Link Device Info Cloud Run Logs Cloud Delivery oring Child De ce Device M mm 2 66d83258a86c20541c8e47.... Device ID Node ID Authenti Type Directly connected Sept 23, 2024 10:19:44 GMT+08:00 Product Model Data View All Properties Q Property data reported by the d Note: If the reported property na ASCII code is 00), the pro betteryLevel X Q Q Latest Reported Time: Nov 12, 2024 19:51:14 GMT+08:00 Enter the service name. batteryLevel (Unit: %) batteryLevel 80 Total Records: 1 16 🗸 🧃 🗧 ----End

Figure 2-90 Viewing reported property - batteryLevel

# 2.7 Connecting to IoTDA Based on the BearPi-HM\_Nano Development Board and OpenHarmony 3.0

This section uses BearPi-HM\_Nano development board as an example to describe how to connect an OpenHarmony 3.0 device to IoTDA using the huaweicloud\_iot\_link SDK.

# Prerequisites

- You have registered a Huawei Cloud account.
- You have subscribed to IoTDA. If you have not subscribed to the service, go to the IoTDA service page, and click **Access Console** to subscribe to the service.

# **Hardware Environment**

You need to prepare a BearPi-HM\_Nano development board, E53\_IA1 expansion module, Type-C data cable, and PC.

# Software Environment

Set up the environment in IDE-based mode or CLI-based mode. The BearPi-HM\_Nano main control chip is Hi3861. Install the corresponding environment.

# **NOTE**

If you install **gcc\_riscv32** in the environment of the Hi3861 development board, download the **gcc\_riscv32** image. Otherwise, some plug-ins may fail to be downloaded or installed.

# NOTICE

It may take a long time to download a large amount of open-source code. Therefore, reserve sufficient time.

# **Creating a Product**

- **Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.
- Step 2 Set the parameters as prompted and click OK.

Basic Informa	Basic Information				
Resource Space	The platform automatically allocates the created product to the default resource space. You can also select an existing resource space from the drop-down list or <b>create one</b> .				
Product Name	Customize the value. The product name must be unique in the same resource space. The value can contain up to 64 characters. Only letters, digits, and special characters (_?'#().,&%@!-) are allowed.				

Protocol	Select MQTT.					
Data Type	elect JSON.					
Industry	this parameter as required.					
Device Type	Set this parameter as required.					
Advanced Settings						
Product ID	Set a unique identifier for the product. If this parameter is specified, the platform uses the specified product ID. If this parameter is not specified, the platform allocates a product ID					

 $\times$ 

# Figure 2-91 Creating a product - MQTT

Create Product	
* Resource Space 🧿	
	To create a new resource space, you can go to the instance details page.
★ Product Name	
Protocol 🧿	(MQTT ~
* Data Type	JSON ~
Device Type Selection	Standard profile Custom
* Device Type	
Advanced Settings $ \wedge $	Custom Product ID   Description
Product ID 📀	
Description	
	0/128 1⁄2
	Cancel OK

----End

# **Developing a Product Model**

**Step 1** Click the created product. The product details page is displayed.

**Step 2** On the **Basic Information** tab page, upload the model file **BearPi\_Agriculture.zip**.

Registered devices: 0				Cuick Links 🕲 Adding to model library
CT Reposite verses. V				
Basic Information Codec Deployment Online Debugging T	pic Management			
Product Detail				
Product Name d		Resource		
Product Name 22		Space	Selection, 2000-con	
Device Type		Protocol	MQTT	
Data Type json		Created	Art 20, 2024 (0.27 3) (207-010)	
Industry		Description	- 2	
A product model describes product details and	SET_PRESSURE_READ_PERIOD Period Value Result Provide a product model using multiple methods. If	Water Water Volt Tmperature Usage	Battery Page Lovel	nd parte lhe dela
英 力 *, 简 😳 🕸	Customize Model Import from Local	Import from Excel Import from Library	Learn more	

# Figure 2-92 Uploading a product model - MQTT

# **NOTE**

- 1. In the product list, click the name of a product to access its details page. On the displayed page, you can view basic product information, such as the product ID, product name, device type, data format, manufacturer name, resource space, and protocol type.
- 2. You can click **Delete** to delete a product that is no longer used. After the product is deleted, its resources such as the product models and codecs will be cleared. Exercise caution when deleting a product.

### ----End

# **Registering a Device**

- **Step 1** Log in to the **IoTDA console**.
- **Step 2** In the navigation pane, choose **Devices** > **All Devices**, click **Register Device**, set parameters based on the table below, and click **OK**.

Resource Space 💿	Selecting, Milleron	~
Product	Test_1	~
	Mqtt devices have subscribed to the platform preset topic topics 🖒	by default. Subscribed
Node ID 💿	Test_1	
Device ID (?)	807600-00080-47667-88, No. 1	
Device Name		
Description		
		0/2,048 4/
Authentication Type ③	Secret X.509 certificate	
Secret		Ø
Confirm Secret		Ø

# Figure 2-93 Registering a device - MQTT

Parameter	Description					
Resource Space	Select the resource space to which a device belongs.					
Product	Select the product to which the device belongs. If no product is available, <b>create a product</b> first.					
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string of 4 to 64 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed.					
Device Name	Customize the device name.					
Authenticatio n Type	Select <b>Secret</b> .					
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.					

# Figure 2-94 Device registered

~	Device Registered	×
•	The system automatically allocated the following device information.	
	For security reasons, the secret will not be available on the device details page. If you forget the secret, click Reset Secret on the Overview tab page to reset the secret.	
I	Device ID	
I	Device Secret	
	Next, you can use the IoT Device SDK to connect devices to the platform.	
	Download	

# **NOTE**

- 1. Save the device ID and secret. They are used for authentication when the device attempts to access IoTDA.
- 2. If the secret is lost, you can **reset the secret**. The secret generated during device registration cannot be retrieved.
- 3. You can **delete a device** that is no longer used from the device list. Deleted devices cannot be recovered. Exercise caution when performing this operation.

----End

# Using the Huaweicloud\_iotlink SDK

- Step 1 Download the source code oh3.0\_hwcloud\_iotlink-master.zip.
- Step 2 Copy the preceding source code to the src > third\_party directory of the OpenHarmony source code. Note that the third-party libraries of OpenHarmony and Huaweicloud\_iotlink SDK use the OpenHarmony library files, such as cJSON and Mbed TLS.
- Step 3 Add the following code to the OpenHarmony 3.0 source code file device\bearpi \bearpi\_hm\_nano\app\BUILD.gn and select the demo using the comment symbol (#).

"//third\_party/hwcloud\_iotlink/demos/mqtt\_tiny\_v5\_agriculture\_demo:mqtt\_tiny\_v5\_agriculture\_demo",

# Figure 2-95 Selecting a demo

	··· 🗲 BUILD	.gn\app M )	BUILD.gn\mqtt_tiny_v5_agriculture_demo				F BUILD.gn\at
>	device >	bearpi > bearpi_h	nm_nano>app> IF BUILD.gn				
V GITWORKSPACE							
> .deveco	12						
> .repo							
> .vscode		import("/	/build/lite/config/component/	lite_component.	gni")		
> applications							
> ark		ite comp	onent("app") {				
> base			ires = 🗊				
> build							
> developtools		1 1	<pre># "//third party/hwcloud iotl</pre>	ink/demos/test	iotlink test	iotlink".	
✓ device	• 20	- I I I I I I I I I I I I I I I I I I I	# // entra_parey/interoda_roer	.intra demosy cese_	rocrimercese_	locitink j	
✓ bearpi∖bearpi_hm_nano		1 D I	W / / bl. for the sector / bring the sector for the table	1.1.1			A A STATE OF A STATE O
> .vscode	21		"//third_party/hwcloud_iotlin	ik/demos/mqtt_ti	ny_v5_agricul	ture_demo:mqτ	t_tiny_v5_agriculture_demo",
✓ app	22						
> A1_kernal_thread							
> A2_kernel_timer			"A2_kernel_timer:timer_examp				
/ Ac_kemei_umei							

# **NOTE**

**2** in the preceding figure is the demo of connecting an MQTT device to Huawei Cloud. Check the **BUILD.gn** file, as shown in **Figure 2-96**. **A** contains the library files related to the development board hardware and Wi-Fi. **B** contains the library files required for connecting an MQTT device to Huawei Cloud, including cJSON, MQTT- and OSAL-related files, and configuration library files. **C** indicates that the hwcloud\_iotlink library needs to be compiled when the file is compiled. Necessary libraries and C files of the file are located based on the specified path for compilation.

# Figure 2-96 Code compilation file

	E BUILD.gn\app M) E BUILD.gn\mqtt tiny_v5_agriculture_demo M × C wifi_connect.c U C iot_config.h 🖏
	A third_party > hwcloud_iotlink > demos > mqtt_tiny_v5_agriculture_demo >
GITWORKSPACE	4 # You may obtain a copy of the License at
> grpc	5 #
> gstreamer	
> harfbuzz	<pre># http://www.apache.org/licenses/LICENSE-2.0</pre>
hwcloud_iotlink	↓ #
> at	8 # Unless required by applicable law or agreed to in writing, softw
> JSON -	9 # distributed under the License is distributed on an "AS IS" BASIS
> crc	10 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
💙 demos 🔹	11 # See the License for the specific language governing permissions
> hello_world •	12 # limitations under the License.
✓ mqtt_tiny_v	13 Generated library name
> include	<pre>14 static_library("mott_tiny_v5_agriculture_demos") {</pre>
> src 4	15 sources = [
E BUILD.gn M	16
> oc_agricultu •	17 Page (552 TAL - "
> oc_mqtt_de =	17 Src/E55_1ALC, 18 Src/wifi connect.c", C file to compile
<pre>&gt; oc_mqtt_de =</pre>	19
E BUILD.gn M	20
C test_main.c M	20 21 include dirs = [
C app_demo M	
M demos.mk M	22 "//base/iot_hardware/peripheral/interfaces/kits",
✓ driver ●	23 A "//device/bearpi/bearpi_hm_nano/iot_hardware_hals/include",
E BUILD.gn M	24 "//foundation/communication/wifi lite/interfaces/wifiservice",
C dev fs test.c M	25 "//third_party/cJSON",
C dev_fs.c M	<pre>26 "//third_party/hwcloud_iotlink/oc/oc_mqtt/oc_mqtt_al",</pre>
C dev_test.c M	<pre>27 "//third_party/hwcloud_iotlink/oc/oc_mqtt/oc_mqtt_profile_v5",</pre>
C driver.c M	28 B "//third_party/hwcloud_iotlink/inc",
C driver.h M	29 📕 "//third_party/hwcloud_iotlink/os/osal",
M driver.mk M	30 "//third_party/hwcloud_iotlink/os/ohos",
≯ fs ●	31 "include",
🗠 hwcloud_iotli 😐	32 "//third_party/hwcloud_iotlink/hwcloud_iotlink_config",
🖩 BUILD.gn 🛛 M	33 1
C iot_config.h M	34
C iot_link_con M	35 deps = [
C link_main.c M	36 [ "//third_party/hwcloud_iotlink:hwcloud_iotlink",
C link_main.h M	
> inc •	57 J 38 }
> link_log ==	20 }

•	
> crc •	22
🗸 demos 🛛 🔍	23 #include <dtls al.h=""></dtls>
> hello_world 🔍	24 #include <mgtt al.h=""></mgtt>
✓ mqtt_tiny_v ●	2A #include <oc al.h="" mgtt=""></oc>
> include	26 #include <oc mgtt="" profile.h=""></oc>
> src •	27 #include "E53 IA1.h"
EBUILD.gn M	_
C mqtt_v5 1, M	28 #include "wifi_connect.h"
> oc_agricultu •	355 C
> oc_mqtt_de •	356 static void IotMainTaskEntry void)
✓ test_iotlink ●	357 {
≡ BUILD.gn M	358 D link main task entry()
C test_main.c M	
C app_demo M	(
M demos.mk M	360 (void) osal_task_create("SelsorTaskEntry",SensorTaskEntry,NULL
<ul> <li>✓ driver</li> </ul>	361 }
	362 <b>B</b>
≣ BUILD.gn M	
C dev_fs_test.c M	363 APP_FEATURE_INIT IotMainTaskEntry;
C 1 C	

Figure 2-97 Main function file of the demo

# **NOTE**

A in Figure 2-97 shows the library files and DTLS library files required for connecting an MQTT device to Huawei Cloud. The link\_main\_task\_entry() function must be called in the entrypoint function IoTMainTaskEntry() first to ensure the OSAL installation and initialization of other configurations.

# **Step 4** Set parameters.

Figure 2-98 Modifying parameters



# 

(1) To connect to the cloud, you need to modify the network configuration information, Wi-Fi hotspot account and password, and device ID and secret registered on the cloud based on your device. Note that this device supports only 2.4 GHz Wi-Fi.

(2) Change the interconnection address to the MQTT access address displayed in the **Access Details** dialog box on the **Overview** page of the **console**.

(3) Compilation and burning can be performed in IDE mode or CLI mode.

(4) After burning, press the **RESET** button on the development board to start the device.

(5) The preceding code is based on OpenHarmony 3.0. For other versions, modify the OpenHarmony source code path introduced in the **BUILD.gn** file as required.

----End

# Connecting the Device to the Platform

After the code is burnt to the device, restart the device. You need to restart the device twice for the first time. During the first burning, you may need to configure the internal information. After the device is restarted twice, it can connect to Huawei Cloud.

On the platform, you can view details about the reported data and deliver commands to control devices, as shown in the following figures.

# Figure 2-99 Viewing reported data - MQTT

IoTDA Instances / All	oTDA Instances / All Devices / Device Details						
< 1 0	Courte D						
Device Info Cic	www.eMM6 Cloud Ruin Logis Cloud Delivery Device Bhadrow Message Trace Device Monitoring Child Devices Tags Groups						
Device Name		Resour	rce Space	Selectings_Midle-out	Product	Trees .	
Device ID	mechanicment. 7	Node IE	D I		Authentication	Secret Resel Secret	
					Туре		
Node Type	Directly connected	Firmwa	are Version		Software Version	-	
Description		Registe	ered	Supt 23, 2024 10:10 as (367-48:00	Activated	Tage 21, 2024 17 12 00 (347-00.00	
Last Online	Nas 12, 2024 18 18 18 087-08 08	MQTT	· · · · · ·	View			
		Connec	ction				
		Parame	ieter				
Product Mode	Product Model Data						
	Property data regorited by the drive based on the product model deformant.						
Note: If the report	ed property name is not contained in the product	model, or the property name contains dots (.).	), dollar symbols	(b), or empty char (the nexadecimal ASCII code is 00), the property data cannot	d be updated.		
Enter the service	ce name. Q	Latest Reported Time	4 19 21 45 04	(*-08.00			luminance × Q
BasicData							
LightControl		luminance					
		luminance					
		30					
		Total Records: 1 16 Y	1.5				

#### Figure 2-100 Simulating command delivery

Debug output	Real-Time Refresh X Clear Application Simulator Log	Application Simulator Device Simulator
Application Simulator	atform Command Delivery Dela Reporting Device Simulator	Commands Sent The application simulator can issue commands to the device according to the product definition. If you use a graphicaby developed codec play-in, please carry all the fields defined in the play-in
All Received Sent	All Commands Received Data Reported	when issuing a command to obtain correct coding results and the length of each command must be less than 512 bytes.
⊙ Sent Juli 06, 2024 16:08:02 GMT+08:00 Message Body: [14:wick_jdf, "StreetLight," "command_name". "SWTCH_LIGTH.;"paras". ["SWITCH_LIGTH: "ON"], "send_strategy". "immediately", "expire_time". 0 }	Commands Received Jul 66, 2024 16 08 02 GMT+08:00     100014F4E     Commands Received Jul 66, 2024 16:04:23 GMT+08:00	Service StreetLigth  Command SWITCH_LIGTH  V
Received Jul 06, 2024 10 04 24 813 OMT-06.00  [serviceid StreetLight, data: ['tight_intensity"-32; 'tight_status"-0])	AAA0000  Deta Sent Jul 06, 2024 16:04:23 GMT+08:00  002000	SWITCH_U ON Pending Period (uf 2880 Cache Send

•	•								
lients <1> :									
mac idx mac	addr	state	lease	tries	rto				
0 201131081a50	and the second second	10	0	1	4				
EBUG][3690][hub step] hub	b step:enter								
DEBUG][3690][dmp_connect]	oc_mqtt_connect;	server:	and the second second	po	rt:				
EBUG] [3700] [dmp_connect]	oc_mqtt_connect;	client_	id:			the second second	in the local	a si parten	
DEBUG] [3710] [dmp_connect]	oc_mqtt_connect:	user:		1000		and a lot	10.00		
asswd:	and the second se	1.00			1.00				
DEBUG] [3720] [dtls_ssl_new]									
EBUG] [3730] [dtls_ssl_new]									
EBUG] [3730] [dtls_shakeha									
EBUG][3930][dtls_shakeham	nd] performing th	ne SSL/T	LS hands	hake					
EBUG][4370][dtls_shakeham	nd] handshake suc	cceed							
EBUG][4470][dmp_connect]				SS					
DEBUG][4470][dmp_subscribe									
EBUG][4480][dmp_subscribe									
EBUG][5590][dmp_subscribe					uccess				
BUG][5590][dmp_subscribe									/sys/me
EBUG][6680][dmp_subscribe									
EBUG][6680][dmp_subscribe									/sys/pr
EBUG] [7810] [dmp_subscribe									
EBUG] [7810] [dmp_subscribe									'sys/pr
EBUG][8940][dmp_subscribe									
EBUG][8940][dmp_subscribe									/sys/sh
EBUG][10070][dmp_subscrib									
EBUG][10070][dmp_subscrib									/sys/e
EBUG][11090][dmp_subscrib		cribe:re	tcode:0:	success					
EBUG][11090][hub_step] hu	ub_step:ok exit								
t = 0									
mgtt profile connect suc	.cced!								

# Figure 2-101 Log information

# 2.8 Testing MQTT Performance Using JMeter

# Scenarios

The number of global IoT device connections is increasing with the development of IoT technologies. The access and management of devices at scale pose great challenges to the network bandwidth, communications protocols, and platform architecture. It is important to test the platform performance during IoT architecture selection. This section describes how to use JMeter to perform a performance pressure test on the MQTT access capability of the platform.

The test plan is described as follows:

Test scenario:

- Simulate 10,000 concurrently online devices to verify the stability of platform persistent connections.
- Simulate a scenario where devices initiate 100 message reporting requests per second to verify the message processing capability of the platform.

Test environment:

- Test object: Huawei Cloud IoTDA SU2 (10,000 online devices and 100 TPS upstream and downstream messages)
- Test executor: One JMeter executor. The specifications are as follows:

Instanc e Type	Flavor	Number of vCPUs	Memory
General computi ng S6	s6.xlarge.2	4 vCPUs	8 GiB

# Table 2-11 Test executor

# 

A single JMeter executor can simulate up to 50,000 online devices. To simulate more online devices, use **Huawei Cloud CPTS** and deploy multiple JMeter executors.

# Prerequisites

- You have registered a Huawei ID and enabled Huawei Cloud services.
- You have subscribed to IoTDA. If not, access IoTDA and buy an SU2 unit (10,000 online devices and 100 TPS upstream and downstream messages).

# Preparations

- Install the Java runtime environment on the JMeter executor. Visit the Java website, and download and install the Java runtime environment.
- **Download** and install JMeter 5.1.1 or later.
- **Download** the **mqtt-jmeter** plug-in package and store it in the **lib/ext** directory of the JMeter installation directory.

# **Service Flow**

The process of using JMeter to perform an MQTT performance pressure test on the platform is as follows:

- Step 1 Create an MQTT product.
- Step 2 Register 10,000 devices in batch import mode.
- Step 3 Import the test plan created for the IoT performance test.
- Step 4 Initiate a platform performance pressure test based on service specifications.
- **Step 5** View the test result. Check whether the test result meets the expectation based on the monitoring metrics displayed on the IoT platform.

----End

# **Creating a Product**

- **Step 1** Log in to the **console**, choose **Products** in the navigation pane, and click **Create Product** on the left.
- **Step 2** Set the parameters as prompted and click **OK**.

### Table 2-12 Parameters

Basic Informa	Basic Information		
Resource Space	The platform automatically allocates the created product to the default resource space. You can also select an existing resource space from the drop-down list or <b>create one</b> .		
Product Name	Customize the value. The name can contain letters, numbers, underscores (_), and hyphens (-).		

Protocol	Select <b>MQTT</b> .		
Data Type	Select <b>JSON</b> .		
Industry	Set this parameter as required.		
Device Type	Set this parameter as required.		
Advanced Set	ttings		
Product ID	Set a unique identifier for the product. If this parameter is specified, the platform uses the specified product ID. If this parameter is not specified, the platform allocates a product ID.		

----End

# **Registering a Batch of Devices**

- **Step 1** Visit the **IoTDA** service page and click **Access Console**.
- Step 2 In the navigation pane, choose Devices > All Devices, click the Batch Registration tab, and then click Batch Register.
- **Step 3** Download and fill in the batch device registration template based on the following table. You can download the **sample file**.

Parameter	Description
node_id	Device identifier. Set this parameter in ascending order, for example, <b>10001</b> , <b>10002</b> , and <b>10003</b> .
product_id	Product ID generated in Creating a Product.
app_id	Resource space. For details about how to obtain the resource space, see <b>Resource Spaces</b> .
device_name	Device name, which can be the same as the value of <b>node_id</b> .
secret	Device secret. You can enter a fixed secret during a performance test.

Table 2-13 Parameters

**Step 4** In the **Batch Registration** dialog box, click **Select File** to upload the prepared batch registration template, and click **OK**.

# Figure 2-102 Device - Registering devices in batches

Batch Reg	istration	1
★ Task Name	batchstreetdevice	
★ File	BatchCreateDevices_Templa (16.02KB) X Select File	
	e template, enter the content in text format, and upload the file.	
	Cancel	ок

**Step 5** After the batch registration is successful, save the device IDs and secrets.

----End

# **Importing a Test Plan**

- **Step 1 Download** the JMeter test plan.
- **Step 2** Open JMeter and click **Open** to import the downloaded test plan.
- **Step 3** In the JMeter directory on the left, choose **User Defined Variables**. On the **User Defined Variables** page, configure the following parameters:

 Table 2-14 Parameters

Parameter	Description
server	MQTT server address. For details about how to obtain the value, see <b>Obtaining Resources</b> .
port	MQTT port number. Set this parameter to 8883.
productId	Product ID generated in Creating a Product.
password	MQTT connection password, which is the value of the device secret encrypted by using the HMAC-SHA256 algorithm with the timestamp as the key. <b>secret</b> is the secret entered during <b>batch device registration</b> . You can use this <b>tool</b> to obtain the encrypted value.
timeStamp	Timestamp used for encrypting the password. The time format is YYYYMMDDHH.

# Figure 2-103 Example



----End

# **Initiating a Pressure Test**

Step 1 In the JMeter directory on the left, choose Thread Group, set Number of Threads to 10000. (A thread corresponds to an online device. 10000 indicates that 10,000 devices are online on the IoT platform.)

Figure 2-104 Configuring devices



Step 2 In the JMeter directory on the left, choose Thread Group > Delay between sampler. Set Thread Delay (in milliseconds) to 100000 (indicating that each device publishes a message every 100 seconds).

Figure 2-105 Configuring devices

<u>File Edit Search Run Options Tools H</u> elp		
📰 🍪 🍓 🗒   👗 🐚 🧻   + 🏷   + - 6 🍩	🗢 💐 🎬 🚓 🍾 🚍 👔	00:00:00 🛕 0 0/0 🕄
<ul> <li>A Test Plan</li> <li>Wer Defined Variables</li> </ul>		
Y 🐯 Thread Group		
Counter		
> Ecop Controller		
👀 Delay between sampler		
🦽 Summary Report		
View Results Tree		
View Results in Table		

**Step 3** Click **D** on the JMeter toolbar to start the performance test.

# Figure 2-106 Performance test

Eile Edit Search Bun Options Tools Help		
📑 🛱 🚔 🚍  🐚 🖄 🕂 — 🍫 💽 🖕 🍩	🖲 💐 🏙 🚓 🏷 🚍 👔	00:00:24 🛕 0 0/5 🚱
<ul> <li>Test Plan</li> <li>User Defined Variables</li> </ul>		
Conset     Conset     Conset     Conset     Conset     Conset     Dispondence     Simply Apport     Wre Results in Bale		
	Ramp-up period (seconds): 0 Loop Count: Infinite 10	

Step 4 In the JMeter directory on the left, choose Summary Report. The throughputs of Connect and Publish Message are displayed. You can modify the values of Number of Threads and Thread Delay (in milliseconds) to adjust the throughputs.

# Figure 2-107 Performance test



**Step 5** After the JMeter test plan is debugged, import the test plan to **CodeArts PerfTest** for distributed deployment to meet requirements of higher-level performance tests.

----End

# Viewing the Pressure Test Result

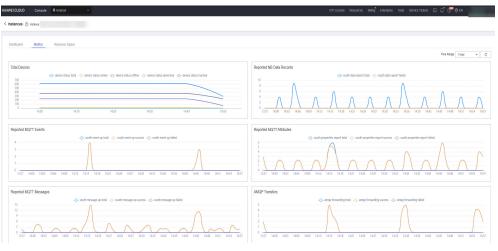
1. Log in to the **IoTDA console** and choose **O&M** > **Reports** in the navigation pane to view statistical metrics of the platform.

HUAWEI CLOUD   슮	Console	В	illing & Costs <sup>®</sup> Resources Enterprise Develop	per Tools ICP License Support Service Tickets English
IoT Device Access	Reports 🗳 Resource space: 🔹 🗸			
Basic Change Default	Device Statuses (?) (Total: 9)	Export Device Messages ③		Export         Month         •         Mar 07, 2023 1128:03 – Apr 07, 2023 112         Image: Compared
Overview Products Devices •	$\bigcirc$	0 (0%) Online 3 3(333%) Enactive 6 (66.67%) Office		0 Downstream messaged (this month) 0
Rules • Storape Management O&M New •		Olifine     Olifine     Abnormal	No data available.	
Reports	General Device Trends ⑦	Export Hour v Mar 31, 2023 11:27:18 - Apr 07, 2023 11:2	2  Device Online Trends ⑦	Export Hour • Mar 31, 2023 11:27:18 - Apr 07, 2023 11:2 📋
Online Debug Message Trace Device Alarms	10 8 6	-@- All -@- Online	80	
Anomaly Detection New Run Logs Remote Login	4 2 0 03/311200 04/0110:00 04/02 08:00	04/03 0650 04/04 0430 04/05 02:00 04/06 00:00 04/06 22:0	40 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	04/02 08:00 04/03 06:00 04/04 04:00 04/05 02:00 04/06 00:00 04/06 22:00

Figure 2-108 Viewing statistical reports

2. For more reports, log in to the AOM console, choose Monitoring > Cloud Service Monitoring, and select IoT Device Access (IoTDA).





# **3** Device Management

# 3.1 Automatically Adjusting Air Conditioner Temperature Through Device Shadow

# Scenarios

Using a constant-temperature control system, you can adjust the default temperature of air conditioners (regardless of whether they are powered on). After being powered on, the air conditioners automatically run at the default temperature. This is achieved by using the device shadow of the IoT platform. For any connected air conditioner, you can set the device shadow on the application or IoTDA console to deliver the preset temperature to the air conditioner. The air conditioner automatically adjusts the temperature after receiving the property modification request.

# **Developing a Product**

- **Step 1** Visit the **IoTDA** service page and click **Access Console**. Click the target instance card.
- **Step 2** In the navigation pane, choose **Products**. In the search box, select the resource space to which the new product belongs.
- **Step 3** Click **Create Product** on the left to create a constant-temperature air conditioner product, set the parameters, and click **OK**.

Basic Inform	nation
Product Name	Enter a value, for example, <b>aircondition</b> .
Protocol	Select <b>MQTT</b> .
Data Type	Select <b>JSON</b> .
Industry	Customize the values.

Device Type				
----------------	--	--	--	--

- **Step 4** After the product is created, click the corresponding product to access its details.
- **Step 5** On the **Basic Information** tab page, click **Customize Model** and configure the product model based on the table below.

Service data	
Service	Service ID: Enter temperature.
	<b>Service Type</b> : You are advised to set this parameter to the same value as <b>Service ID</b> .
Property	Property Name: Enter temperature. Data Type: Select Integer. Access Permissions: Select Read and Write.
	Length: Enter 1.

**Step 6** In the navigation pane, choose **Devices** > **All Devices**, and click **Register Device**. In the dialog box displayed, set the parameters based on the table below.

Register Device		
* Resource Space  ?		)
* Product		)
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics $\textcircled{2}$	
* Node ID 🧿	Test_1	)
Device ID 🧿	an term of construct and term. Term	)
Device Name		)<
Description		1
	0/2,048 //	J
Authentication Type 💿	Secret X.509 certificate	
Secret	Ø	)
	() () ()	1

Figure 3-1	Registering	a device -	MQTT
------------	-------------	------------	------

Parameter	Description
Product	Select the product created in <b>Step 3</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticatio n Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

----End

# Configuring a Device Shadow

You can set a device shadow on the IoTDA console or by calling the API for **configuring desired properties in a device shadow** on the application side. This section describes how to set a device shadow on the IoTDA console.

- Step 1 Log in to the IoTDA console, choose Devices > All Devices in the navigation pane, and click View in the row of the device registered in Step 6 to access its details.
- Step 2 On the Device Shadow tab page, click Configure Property.
- **Step 3** In the dialog box displayed, enter the desired value of a property. In this example, the value of **temperature** is set to **25**.

Figure 3-2 Configuring a device shadow

Configure Prope	erty	×
Only writable prop	erties can be configured.	
Service	Property	Desired Value
temperature	temperature	Parameter type: int
		Cancel OK

----End

# **Verifying Configurations**

Method 1:

You can use MQTT.fx to simulate device verification.

- Use MQTT.fx to simulate a constant-temperature air conditioner and connect it to the platform. For details, see Developing an MQTT-based Simulated Smart Street Light Online.
- On the Subscribe tab page, enter topic=\$oc/devices/{device\_id}/sys/ shadow/get/response/# of the device shadow (replace {device\_id} with the device ID in Step 6), and click Subscribe.

Publish	Subscribe	Scripts	Broker Status	Log	3
\$oc/devices	/	-	/sys/sh	•	Subscribe

- On the Publish tab page, enter Topic=\$oc/devices/{device\_id}/sys/ shadow/get/request\_id={request\_id} of the device shadow.
- 4. Enter a request for obtaining the device shadow and click **Publish**.

Example: {

	object_device_id :
	"service_id": temperature"
}	

	ublish Subscribe	Scripts	Broker Status	Log		
2 »	\$oc/devices/		/sys	s/sh	Publish	0
{	"object_device_id": " "service_id": "temperate	ure"		· · ·		
	3					

5. Click the **Subscribe** tab. The device shadow data delivered by the platform is displayed.

Publish Subscribe	Scripts Broker Status Log		
\$oc/devices/	/sys/sh 💌 Subscribe	Qo50 C	Qo51 Qo52 Autoscroll OST
\$oc/devices/	//sys/shadow/get/response/ 2 Dump Messages Mute Unsubscribe	\$oc/devices/ 7/sys/shadow/get/response/request_id=001 Soc/devices/5 /sys/shadow/get/response/#	1 QoS 0
	Comp mosages mate Croadence	\$oc/devices/ //sys/shadow/get/response/request_id=001 Soc/devices/* //sys/shadow/get/response/#	2 QoS 0
opics Collector (0)	Scan Stop Or		
		\$oc/devices/ \$oc/devices/ \$oc/devices/ \$14:38:37.52717346 \$25717346	2 QoS 0
		<pre>{"shadow":[{"desired":{"properties":{"temperature":"25"},"event_time ted":{"properties":{"temperature":26},"event_time":"20200525T0615012 "temperature"}],"object_device_id":</pre>	"},"version":7,"service_id":

# Method 2:

You can connect a physical device to the platform. The device will receive the device shadow configuration delivered by the platform and change the preset temperature accordingly.

# 3.2 Managing Indoor Air Conditioners Using Custom Topics

# Scenarios

Custom topics apply to MQTT devices connected to IoTDA. Topics for **message reporting** and **message delivery** can be customized. Applications can implement different service logic processing based on topics. Custom topics can also be used in the scenario where a device cannot report properties or receive delivered commands defined in the product model.

In this example, an application receives data reported by a device and determines whether to turn on or off the indoor air conditioner.

# Prerequisites

- You have registered a Huawei ID and enabled Huawei Cloud services.
- You have subscribed to IoTDA.
- You have created an MQTT product, developed a product model, and registered a device on the IoTDA console. For details, see Products, Developing a Product Model Online, Registering an Individual Device, or Registering a Batch of Devices.
- The connection between the device and platform has been established. For details, see **Device Connection Authentication**.

# **Customizing a Topic**

For details, see Custom Topic Communications.

# **Message Reporting**

- 1. Visit the **IoTDA** service page and click **Access Console**. Click the target instance card.
- 2. In the navigation pane, choose **Devices** > **All Devices**, locate the target device, and click **View** to access its details page.
- 3. Click the **Message Trace** tab, click **Start Trace**, and set the trace duration as required.
- 4. Use the MQTT.fx simulator to simulate a device to report a custom topic message. For details, see **Quick Device Access**.

🕲 MQTT.fx - 1.7.1					_		×
File Extras Help							
local mosquitto	- Q	Connect	Disconnect				
	Scripts Broker Status	s Log Publish		Qo Qo	Qo ]	Retained	0;*
{     "object_device_id": "tet     "name": "currentTempe     "id": "00000001",     "content": "{'temperatu }							

# D NOTE

For devices connected using the IoT Device SDK or native MQTT protocol, you need to set the custom topic name reported by the device in the device program.

5. On the **Message Trace** page, view the custom topic messages reported by the device.

# Figure 3-3 Viewing message trace

oTDA Instances / All Devices / Device Details	
Contine (3)	🗋 Quick Links
Device Info Cloud Run Logs Cloud Delivery Device Shadow Message Trace Device Monitoring Child Devices Tags	
Traced messages help you quickly locate and identify failure causes. Learn more To ensure data validity and prevent the platform from occupying too many read and write compute and storage resources, the platform can only trace messages for up to 10 devices at a time for a single user, and for no more than three days.	
Implementation [Running] Start time: Jul 06, 2024 16:25:30 GMT+08:00 End time: Jul 06, 2024 17:25:29 GMT+08:00 Stop Trace	Clear Data
Edit Configuration Export Data	
Search by service deta     IoTDA has received the properties reported from the device data ("services" (["properties":     ("AI1"10";REG20003":267";REG20014":499";REG20130":322; [REG20015":657], "service.jd":"basicData", "event_time":null()),	0
Service Type $\ominus$ app_ld: device_ld: topic: Messag $\ominus$ Operation	
Socidevices/u	
Device to platform Reporting properties f IoTDA has received the properties reported from the device data.["ser Jul 05, 2024 16 25 45 GMT+08 • Successful View	
Device to platform Device authentication The authentication is successful, device_id: Jul 06, 2024 16 25 30 GMT+08 • Successful View	
Device to platform Device authentication IoTDA has received an authentication message from the device, devi Jul 06, 2024 16 25 30 GMT+08 • Successful View	
Total Records: 4 10 v (1)	

6. An application obtains the custom topic message reported by the device through data forwarding. For details, see **Data Forwarding**. You can also refer to **Forwarding Device Data to OBS for Long-Term Storage**.

# Message Delivery

In this example, the Postman is used to deliver an instruction for starting the indoor air conditioner.

1. Use the MQTT.fx simulator to subscribe to a custom topic.

🐵 MQTT.fx - 1.7.1								$\times$
File Extras Help								
local mosquitto	•	Connect	Disconnect					<b>A</b>
Publish Subscribe	Scripts Broker S	status Log						
\$oc/devices/		<ul> <li>Subscribe</li> </ul>	] 1		Qo Qo	Qo Au	toscroll	0;*
\$oc/devices/		\$oc/devices/888			WYSSE /user/te	mperature		
Topics Collector (0)	Scan Stop 😋							
		\$oc/devices/		a an	<u>teriteri energi</u>	/user/	temperat	ture
								QoS 0
	2		ommand","id":" Conditioner':'				88","co	onte

# **NOTE**

- Ensure that the device operation permissions contain the subscription function when you create a custom topic. For details, see **Custom Topic Communications**.
- For devices connected using the IoT Device SDK or native MQTT protocol, you need to set the name of the custom topic to which the device subscribes in the device program.

2. Use the Postman to simulate the application to call the API **Delivering a Message to a Device** to deliver a command for starting the indoor air conditioner.

end_mqtt_messag	
POST 🔻	https://iotda.cn-north-4.myhuaweicloud.com/v5/iot/{{project_id}}/devices/{{device_id}}/messages
uthorization	Headers (2) Body  Pre-request Script Tests
form-data	x-www-form-urlencoded 💿 raw 🔍 binary JSON (application/json) 🔻
3 "message	<pre>"command", " : "{'airConditioner':'start','mode':'cold'}", : "temperature<sup>t</sup></pre>

3. Call the API **Querying Device Messages** to check whether the command is delivered. If the command is delivered, the indoor air conditioner will be started.

query_mqtt_message × + •••	
▶ query_mqtt_message	
GET • https://iotda.cn-north-4.myhuaweicloud.com	/v5/iot/{{project_id}}/devices/{{device_id}}/messages
Authorization Headers (2) Body Pre-request Script	Tests
Кеу	Value
Content-Type	application/json
X-Auth-Token	{{X-Subject-Token}}
New key	Value
Body     Cookles     Headers (6)     Test Results       Pretty     Raw     Preview     JSON ▼     ⇒       1 ▼ {	
5         "message_id": "\$\$\$           6         "name": "command",           7         "message": "{'airConditioner': 'start',           8         "topic": \$\$\$           9         "status": "OELIVERED",           10         "created_time": "\$	

# 3.3 Performing OTA Firmware Upgrade for MQTT Devices

# **Scenarios**

Message Queuing Telemetry Transport (MQTT) is a publish/subscribe messaging protocol that transports messages between clients and a server. It is suitable for remote sensors and control devices (such as smart street lights) that have limited computing capabilities and work in low-bandwidth, unreliable networks through persistent connections. Firmware upgrade is a basic function that network devices must support. Online upgrades are important especially when the firmware needs

to be upgraded due to security vulnerabilities, software bugs, function optimization, and device performance improvement. This section describes how to upgrade the firmware in Huawei Cloud IoTDA using MQTT.fx to simulate a device.

# **NOTE**

The software upgrade process is the same as that of the firmware upgrade. The only difference is that the parameters for reporting version numbers are different. For firmware upgrade, the parameter that specifies the version number is **fw\_version**. For software upgrade, the parameter is **sw\_version**. For details, see **Device Reporting the Software and Firmware Versions**.

# **Development Process**

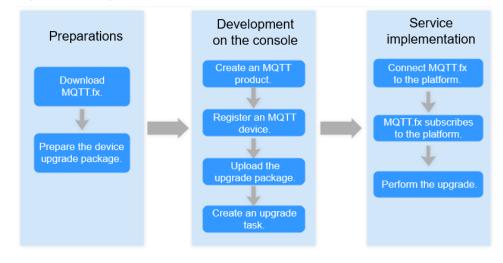


Figure 3-4 Upgrade process

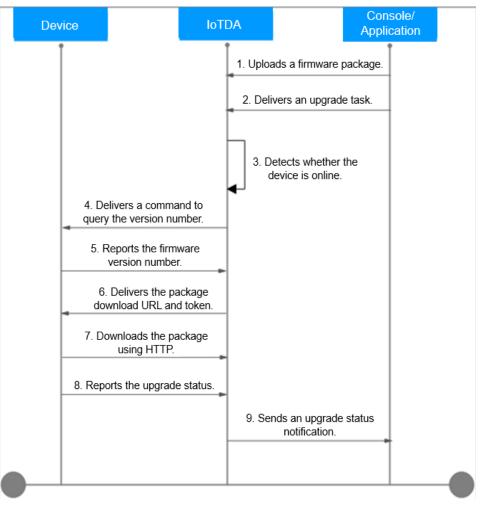


Figure 3-5 MQTT device upgrade process

# **Development Environment**

• Software: Huawei Cloud IoTDA, 64-bit Windows 7 or later (64-bit Windows 10 is used in the following demonstration), and MQTT.fx simulator.

# Prerequisites

- You have registered a Huawei Cloud account.
- You have completed real-name authentication. Otherwise, IoTDA cannot be used.
- You have subscribed to IoTDA. If you have not subscribed to the service, go to the IoTDA service page, and click **Access Console** to subscribe to the service.

# Preparations

- 1. Download and install MQTT.fx (version 1.7.1 or later).
- Prepare the upgrade package. Obtain the firmware upgrade package and its version number from the module vendor. Temporary files are used in this demonstration.
- Create an MQTT product. (If an MQTT product already exists, skip this step.)

- Register an individual device.
- Add the registered device to a **device group**, which will be used during upgrade task creation.

# Uploading an Upgrade Package

- **Step 1** Visit the **IoTDA** service page and click **Access Console**. Click the target instance card.
- **Step 2** In the navigation pane, choose **Devices** > **Software/Firmware Upgrades**.
- Step 3 Click the Manage Resource Package tab and click Firmware List.
- **Step 4** Click **Upload**. On the displayed page, upload a firmware package from OBS or your local PC.

Figure 3-6 Uploading the upgrade package - OBS file

< Upload Pack	sge				
Upload Mode	OBS Local				
* OBS Region					
* OBS Bucket	Q No buckets available? Go to the OBS concole to Create Bucket (2)				
* Object	1 Currently, only the GBS region that is the same as the IoTDA region is supported.     2 Only MOTT device timeme their can be reported from OBS Tru need to configure a new every, type value on the device side. For details, see Dativo     1/ you use OBE has a support perception, you will be allow using OBS Te example. If to MB package is used for upgrading 10,000 devices per	ng an Upgrade Event nonth, you will be charged CNYS12.15 on a pay-per-use basis. For details, see Software/Firmware P	Package Upload		
		U	ipload Object 🛙	Enter the object name.	Q
	Name	Size			
	Test_package_1.bin	38			
	Test_package_1.bin	38			
	10 🗸 (1)				
					Ne

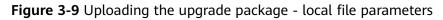
Figure 3-7 Uploading the upgrade package - local file



**Step 5** After the upgrade package is uploaded, configure package parameters and click **OK**.

Upload Pack	age			
Upload Mode	OBS Local			
Package List	~ Test_package_1.bin			Delete
	* Firmware Version	V1.0	* Product Text_1 V	
	Source Versions 💿	Version can be separated by semicolons Add	Signature ①	
		V0.1 ×		
	Description	01,024	Customized Information puthed to 0/4,055	
	<ul> <li>Test_package_2.bin</li> </ul>		devices	Delete
	* Firmware Version	V2.0	★ Product Test_1 ✓	
	Source Versions 💮	Version can be separated by semicolons Add	Signature (3)	
		V0.2 ×		
	Description	01,024	Customized Information pushed to 0/4,005	
		W1,024)	devices Unit, Units	
			Previ	ious Cancel OK

Figure 3-8 Uploading the upgrade package - OBS file parameters



+ Upgrade Package	Note: When you upload multipl	e files, file names must be unique and cannot contain slathes (/) or backstathes ().	6	
			Drog files here. Or Select File fou can uptiod up to 10 files at a time. Max, size of each file: 20 MB	
Package List	~ Test_package_3.bin			Delete
	* Firmware Version	V3.0	* Product Test_1 ~	
	Source Versions ③	Version can be separated by semicolons Add	Signature 💿	
	Description	V0.3 × 01,024	Customized information putted to devices 04,000	

**Step 6** The uploaded upgrade package is displayed in the firmware list.

Figure 3-10 Software/Firmware upgrade - Firmware list

Manage Resource Package			2 Create Upgrade Tasks			
Firmware List Software List						
Upload Delete	Firmware ⇔	Version A	Product ID \varTheta	Product Name 😔	Uploaded 😔	Queration
	Test_package_3.bin	V3.0	0x0710x0710210xxx70	Test_1	AV 31, 2004 15:27 28 (MP-48).	View Download Delete
		V2.0	0140710400700108em/702	Test_1	AN 10, 2024 10:27 10:047-08.	View Download Delete
	Test_package_2.bin	V2.0				

**Step 7** Click **Create Upgrade Task**. On the **Firmware Upgrades** tab page, click **Create Task**. On the displayed page, configure parameters, select upgrade packages, select devices to upgrade, and click **Create Now**.

<   Create Firmware Upgrad							
Set Basic Information							
★ Task Name	Test_upgrade_1						
Execution Time	Now Scheduled						
Single Device Task Timeout ③	- 30 + day						
Retry							
Single Device Upgrade Timeout ③							
Select Upgrade Package Select firmware and a target version.							Advanced Search 🗸
Firmware ID	Firmware	Version	Source	Product	Description	Source version	Customized information pushed
e15be46a1937ba68233dce53	Test_package_3.bin	V3.0	IoTDA	Test_1			
						V0.3	
9b7eb4d4d8c519d0229ab23d	Test_package_2 bin	V2.0	OBS	Test_1		V0.3 V0.2	
<ul> <li>967eb4d4d8c619d0229ab23d</li> <li>≤ 511591557c85e8ef8e96a025</li> </ul>	Test_package_2 bin Test_package_1.bin						
	Test_package_1.bin	V2.0	OBS	Test_1		V0.2	

Figure 3-11 Creating a firmware upgrade task - Test\_upgrade\_1

# **NOTE**

When creating an upgrade task, you can select up to 10 upgrade packages. Supported source versions of the upgrade packages must be unique. If no source version is specified for an upgrade package, the package can be used to upgrade devices of all source versions by default.

Step 8 View the created task in the task list. You can click View to view the task details. On the task details page, you can stop executing upgrade tasks for up to 100 devices in batches or an upgrade task for a single device. You can retry failed upgrade tasks for up to 100 devices in batches or an upgrade task for a single device. You can click All Retry to retry the upgrade task for all failed devices.

Basic Information				
Task Name	Test_upgrade_1	Status		Running
Execution Time	Now	Task Typ	De	Firmware upgrade
Start Time	Jul 02, 2024 17:01:19 GMT+	08:00 Retry Po	licy	Retry attempts: 0   Ret interval (min): 0
Single Device Upgrade Timeout	30 minute	Single D	evice Task Timeout 🧿	30day
0	Total Devices to Upgrade 1	<ul> <li>Successful</li> <li>0</li> </ul>	<ul> <li>Failed</li> <li>0</li> </ul>	Other 1
All Retry Batch Retry	Batch Stop	All	✓ Enter the	e device ID. Q
Status	Device ID	Upgrade Descript	ion	Operation
Running	Tel.,1,26()8071.)	QueryingVersion		Retry Stop

Figure 3-12 Task details - Test\_upgrade\_1

# Service Implementation

- **Step 1** Use MQTT.fx to simulate device access to the platform. For details, see "Performing Connection Authentication".
- **Step 2** Use MQTT.fx to subscribe to the downstream message topic of the platform. MQTT.fx receives the version query command delivered by the platform.

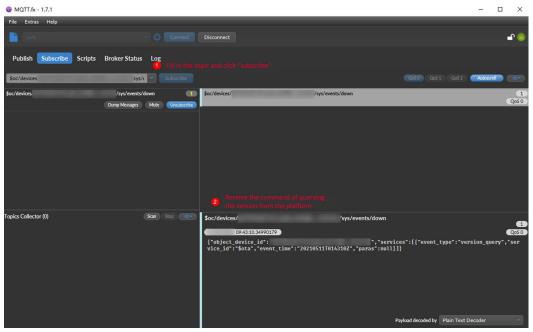
Topic

Downstream: **\$oc/devices**/{*device\_id*}/**sys/events/down** 

### Parameters

For details, see Platform Delivering an Event to Obtain Version Information.

Figure 3-13 Subscribe/Push



Step 3 Report the software and firmware version information using MQTT.fx.

### Topic

Upstream: **\$oc/devices**/{*device\_id*}/**sys/events/up** 

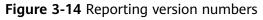
### Parameters

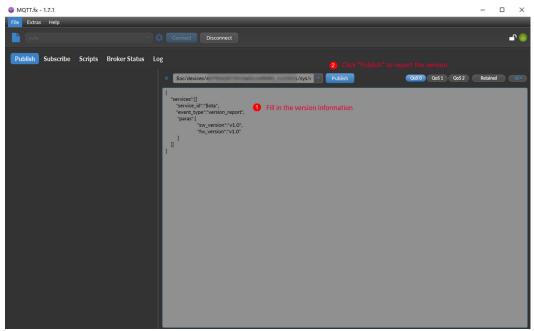
For details, see Device Reporting the Software and Firmware Versions.

# Example

```
Topic: $oc/devices/{device_id}/sys/events/up
Data format: {
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "version_report",
        "event_time": "20151212T121212Z",
        "paras": {
            "sw_version": "v1.0",
            "fw_version": "v1.0"
```

} }] }





**Step 4** After the version numbers are reported, the simulator receives an upgrade notification from the platform. The notification is as follows:

# Торіс

Upstream: **\$oc/devices**/{*device\_id*}/**sys/events/down** 

# Parameters

For details, see **Platform Delivering an Upgrade Event**.

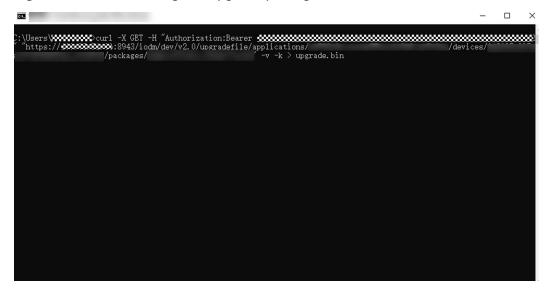
WQTT.fx - 1.7.1		- 🗆 X
File Extras Help		
2-6-ota0615 💌	Connect Disconnect	<b>-</b>
Publish Subscribe Scripts Broker Status	Log	
\$oc/devices/i/sys/ev	Subscribe	QoSO QoS1 QoS2 Autoscroll
\$oc/devices/( i/sys/events 2 Dump Messages Mute Unsubscribe	\$oc/devices/ sys/events/down	1 QoS 0
Dunip messages mule Unbubling	\$oc/devices/ /sys/events/down	2 Qo5 0
Topics Collector (0) Scan Stop @~	<pre>{"object_device_id":"\$ota","event_time" "ware_upgrade","service_id":"\$ota","event_time" :"V3.23","url":"https://wirces, //devices, ""file_size":32,"access_toke ","expires":86400 {</pre>	dev/v2.0/upgradefile/applications/ //packages/ ,"sign":" fo":"12121212"}}]}
	Pa	yload decoded by Plain Text Decoder

Figure 3-15 Obtaining the upgrade notification

**Step 5** After the device receives the upgrade notification, send an HTTP request to download the upgrade package.

A cURL command is used in this demonstration.

Figure 3-16 Downloading the upgrade package



# Example

# 

- Add Authorization to the HTTP request header. The value of Authorization is Bearer {access\_token}, where the value of {access\_token} is the token in the upgrade notification. Leave a space between Bearer and {access\_token}.
- If event\_type is set to firmware\_upgrade\_v2 or software\_upgrade\_v2, the request header does not need to be carried in the request for downloading the software or firmware package. An example request is as follows:

GET https://\*\*\*\*\*.obs.cn-north-4.myhuaweicloud.com:443/test.bin? AccessKeyId=DX5G7W\*\*\*\*\*\*\*

Step 6 Enable the device to report the upgrade status.

# Topic

Upstream: **\$oc/devices**/{*device\_id*}/**sys/events/up** 

# **Parameters**

For details, see Device Reporting the Upgrade Status.

### Example

```
Topic: $oc/devices/{device_id}/sys/events/up
Data format:
    "object_device_id": "{object_device_id}",
    "services": [{
        "service_id": "$ota",
        "event_type": "upgrade_progress_report",
        "event_time": "20151212T1212122",
        "paras": {
            "result_code": 0,
            "progress": 50,
            "version": "V1.0",
            "description": "upgrade processing"
        }
    ] }
```

The following figure shows that the upgrade progress is 50%, which is displayed on the platform.

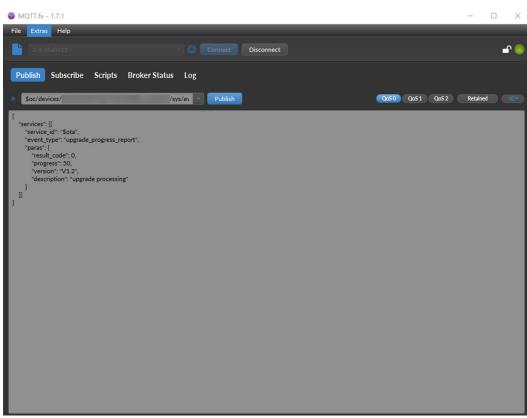


Figure 3-17 Reporting the upgrade progress (50%)



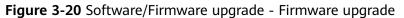
Task Details					>
Basic Information					
Task Name	Test_upgrade_1	S	Status	Run	ning
Execution Time	Now	1	ask Type	Firm	ware upgrade
Start Time	Jul 02, 2024 17:01:19 GMT+08:00	F	Retry Policy		y attempts: 0   Retry val (min): 0
Single Device Upgrade Timeout 🧿	30 minute	ç	Single Device Task Timeout	③ 30da	ay
Execution Details Selected I	Upgrade Package				
Total	Devices to Upgrade	Successful O	<ul> <li>Failed</li> <li>O</li> </ul>		Other 1
All Retry Batch Retry	1 Batch Stop	0	0 All ~ Enter	the device	1 ID. Q Q
0	1	0 A Upgrade D	0 All ~ Enter	the device Ope	1

**Step 7** Complete the upgrade.

If the upgrade progress is 100% and the current version is the target version, the upgrade task success will be displayed on the platform.

WQTT.fx - 1.7.1			– 🗆 X
File Extras Help			
2-6-ota0615	Connect Disconnect		<b>-</b>
Publish Subscribe Scripts Broker Statu	us Log		
» \$oc/devices/	/sys/ex - Publish	QoS 0 QoS 1 QoS 2	Retained 😽
<pre>{     "service_id": "Sota",     "event_type:"upgrade_progress_report",     "parast";     "result_code":0,     "porrest":100,     "version": "V3.23"     ] ] </pre>			

Figure 3-19 Reporting the upgrade progress (100%)



Software/Firmware Upgrades Resource Space	· • •			Culok Links
Manage Resource Package		Create Upgr	ade Tasks	
Firmware Upgrades Software Upgrades				
Max. ongoing tasks: 10.				
Create Task Delete				
< Q. Search by status by default.				00
Task Name/ID 😔	Status 🕀	Task Type $\ominus$	Start Time \ominus	Operation
HuawelDeviceUpgrade	Successful	Firmware upgrade	An 10, 2014 17 (0:10) (0:07-00.00	View copied. Delete
Total Records: 1 10 V ( 1 )				

----End

# **4** Data Forwarding

# 4.1 Pushing Metric Data to DMS for Kafka

# Introduction

IoTDA provides various data reports, which you can integrate into your O&M system. You can use the subscription function of Application Operations Management (AOM) to push metric data from IoTDA to DMS for Kafka, and then consume and display the data.

# Prerequisites

- You have registered a Huawei account.
- You have subscribed to IoTDA. Method: Visit the **IoTDA** service page, and click the button for free trial or the price calculator to purchase and subscribe to the service.
- You have subscribed to AOM. Method: Visit the AOM service page, click AOM **2.0 Console**, and click the button to enable the service for free and authorize.
- You have subscribed to DMS for Kafka. Method: Visit the DMS for Kafka service page, and click Buy Now.

# **Example Scenario**

In this example, the AOM subscription function is used to push metric data to DMS for Kafka. The number of online devices is used as an example. For details, see **IoTDA Metrics**.

The procedure is as follows:

- 1. Create a subscription on the AOM console.
- 2. Simulate device going online and verifying the result.

# **Creating an AOM Subscription**

**Step 1** Visit the **IoTDA** service page, and click **Access Console**.

Step 2 In the navigation pane, choose O&M > Reports, click Subscribe to Data, and click Access AOM.

Figure 4-1 Creating a data subscription - going to AOM

& Modify …
cribe to Data
*⊕⊞
ed (now) O
essages (now)
0
rte

Step 3 In the navigation pane, choose Management > Data Subscription, click Create Subscription Rule, select the subscription content, data type, and DMS instance, and click OK.

Figure 4-2 Creating a data subscription - AOM

Application O&M Management	Data Subscription ⑦					-	Create Subscription Rule
Log Analysis 💷 🔹	Max. subscription rules: 10; avail	lable for creation: 10.					Enter a rule name. Q
	Rule Name	Rule ID	Subscription Target Type Subscription Target Name	_	Subscription Content	Status	Operation
Application Insights			Create Subscription Rule	×			
Prometheus Monitoring			Rule Name				
Container Insights +			test Subscription Content	- 1			
Business Monitoring			Distributed Message Self-built Kafka				
Infrastructure Monitoring			Data Type Metric Alarm				
Process Monitoring +			Instance				
Intelligent Insights BETA			kafka-test	• 0			
Collection 🕑			After a rule is created, you cannot change the instance.				
Automation 🕑			OK Cancel	_			
Management +							
Global							
Data Subscription							
Back to 1.0							

- **Step 4** When configuring DMS subscription for the first time, you need to configure an IAM agency. For details, see **How Do I Create the apm\_admin\_trust Agency**?
- **Step 5** Click **Verify and Save DMS Configuration** to verify the connectivity between AOM and DMS. After the connection is successful, select the data sending topic and click **OK**.

Figure 4-3 Rule details - AOM

Basic Information		
Rule Name test 🖉	Rule ID	4fd66c69-83d7-433a-83d6-81ea643958d0
DMS Instance katka-dms-wf1-6	Prometheus Instance	kafka-dms-wf1-6kafka-dms-wf1-6kafka-dms-wf1-6kafka-dms-wf1-6kafka
Verify DMS Instance Connectivity To send subscribed data to DMS, ensure that you have created the apm_admin_trust agency in the IAM console.		
Verify and Save DMS Configuration		
Select Topic for Transmitting Data testhahatopic		
OK Cancel		

----End

# Stimulating Device Going Online and Verifying the Result

- **Step 1** Use the MQTT simulator to simulate the connection of a device to the IoT platform and its going online. (For details about how to use the simulator, see Using MQTT.fx for Commissioning.)
- Step 2 Wait for about 1 minute, go to the DMS for Kafka console, choose Message Query, set Topic Name to the topic configured in AOM, set Search For to the ID of the resource space where the device is located, and check whether the received device number metrics contain the number of online devices.

	Figure 4-4	Viewing	Kafka	messages	- AOM
--	------------	---------	-------	----------	-------

Basic Information					
Monitoring	* Topic Name	Partition Ent	ter a partition number.	Basic Information	
Topics	Search For	ource space id		Topic Name testkafkatopic Partition 1	
Consumer Groups					
Message Query	Query with content is limited to 10 results. Fa	ich search covers at most 10.000 records, or 200 MB. For large records (> 20	KB per message) or a long period, dump messa	Offset 1,011,026 Created Mar 27, 2024 00:00:08 GMT+08	1:00
Background Tasks				Message Size 997 Bytes Key	
Disk Usage Statistics	Topic Name 🙃	Partition A	Offset A	Message Body	
Parameters	Institution		1011026		Copy
Smart Connect	iestrainaright.		1011020		Copy
Tags	testkafkatopic	0	567140	("project_id":"d92d9c5eb8e347b5bb31ecfe5bc0c4e1", "metrics".[["metric": ("namespace":" <u>10TDA DEVICE_STATUS</u> ;"dimensions":	
Analysis & Diagnosis 🗸 👻	testkafkatopic	٥	567141	[["name":app","value";" ),["name":"region","value";"unknown_null"), ("name":"instance","val 7"),	
	testkafkatopic	0	567142	("name","appName","value"," )]), values". [("value"3,"type","int","unit",","statisticvalues",","metric_name","iotda_device_status_totalCount"),	
	testkafkatopic	1	1011044	('value'' 0, 'type'' inf', 'unit'', 'tatistic values'', 'metric, name'' india, device, status_onlineCount', ('value'', type'' inf', 'unit'', 'statistic values'', 'metric, name'' india, device, status_onlineCount', ('value'', type'' inf', 'unit'', 'statistic values'', ''metric, name'' india, device, status, abromacCourt').	
	testicaficatopic	1	1011074	<pre>('value'.1,'type''.'int','unit'.'','statisticvalues'.'','metric_name'.'iotda_device_status_activeCount'), ('value'.2,'type''.'int','unit'.'', 'statisticvalues'.'','metric_name'.'iotda_device_status_inactiveCount')).'collect_</pre>	tim
	testkafkatopic		567914	e~1711468200000)}}	
	testkafkatopic	1	1012546		



# **5** Device Linkage

# 5.1 Triggering Alarms and Sending Email or SMS Notifications

# Introduction

Many IoT devices run 24 hours a day. Device administrators do not need to know the real-time device status. They only need to be notified of certain statuses.

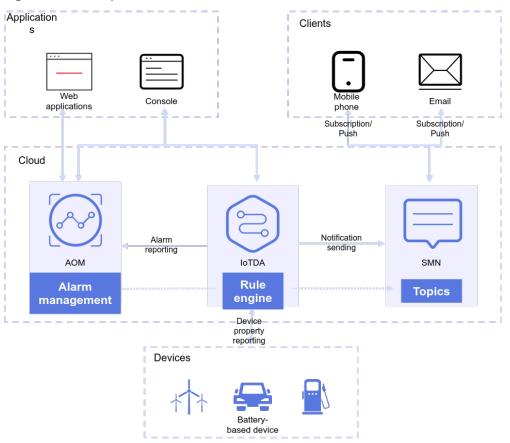
IoT Device Access provides the rule engine function to meet this requirement. You can set rules to enable the platform to send a notification when the data reported by a device meets a certain condition.

### **Example Scenario**

In this example, when the battery level reported by the device is lower than 20%, the IoT platform reports an alarm and sends an email or SMS notification to the specified mobile number.

The procedure is as follows:

- 1. Simple Message Notification (SMN) : Subscribe to notification channels.
- 2. IoTDA: Define product models, create devices, and configure linkage rules.
- 3. Perform verification.



#### Figure 5-1 Example

# **Configuring SMN**

On the Simple Message Notification (SMN) console, create a topic and add a subscription for the IoT Device Access service to invoke to send emails or SMS messages.

- 1. Log in to Huawei Cloud and access **Simple Message Notification (SMN)**.
- 2. Click Access Console. If you have not subscribed to SMN, subscribe to it first.
- 3. Choose **Topic Management** > **Topics** page, and click **Create Topic**.
- 4. Enter the topic name, for example, **Test\_1**, and click **OK**.

Figure 5-2	Creating a	i topic - SN	٨N
------------	------------	--------------	----

Create Topic		×
* Topic Name	Test_1 ⑦ The name cannot be changed after the topic is created.	
Display Name	⑦	
★ Enterprise Project	default    C ⑦ <u>Create Enterprise Project</u>	
CTS Log		
Tag	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags C To add a tag, enter a tag key and a tag value below.	
	Enter a tag key     Enter a tag value     Add	
	Tags you can still add: 20	
	OK Cancel	

- 5. Choose **Topic Management > Subscriptions**, and click **Add Subscription**.
- 6. Enter the subscription information and click **OK**.

Figure 5-3 Adding a subscription - SMN

	T
	•
Endpoint     Endpoints	Description
Add Endpoint     Batch Add Endpoints	

Parameter	Description
Topic Name	Select the topic created in <b>4</b> .
Protocol	<ul> <li>To send an email notification, select Email.</li> <li>To send an SMS notification, select SMS.</li> </ul>
Endpoint	• If <b>Protocol</b> is set to <b>Email</b> , enter the email address for receiving notifications.
	• If <b>Protocol</b> is set to <b>SMS</b> , enter the mobile number for receiving notifications.
	To add multiple endpoints, place one endpoint in a line. A maximum of 10 lines can be entered.

Table 5-1 Paramet	ers for	adding a	a subscription
-------------------	---------	----------	----------------

# **Configuring IoTDA**

Using IoT Device Access, you can create a product model, register a device, and set a device linkage rule. When the device reports specific data, an alarm is triggered and an email or SMS notification is sent.

- 1. Visit the **IoTDA** product page and click **Access Console**. Click the target instance card.
- 2. In the navigation pane, choose **Products**.

**Note**: The product model and device used in this document are only examples. You can use your own product model and device.

3. Click **Create Product** to create a product using MQTT. Set the parameters and click **OK**.

Basic Informa	tion
Product Name	Enter a value, for example, <b>MQTT_Device</b> .
Protocol	Select <b>MQTT</b> .
Data Type	Select <b>JSON</b> .
Industry	Set the parameters as required.
Device Type	

Table 5-2 Parameters for creating a product

- 4. Click here to download a sample product model.
- 5. Click the created product. The product details page is displayed.
- 6. On the **Basic Information** tab page, click **Import from Local**. In the displayed dialog box, load the local product model and click **OK**.

#### Figure 5-4 Uploading a model file

Import from Local	$\times$
A The service definition in the imported file will replace the original service definition of the product.	
After you develop a product model based on the format standards, you can pack and upload it. Learn about product models.	
* File Add a file and upload it. Select File	
Configure the model file by referring to the template.	
J. Product Template	
Cancel OK	

 In the navigation pane, choose Devices > All Devices. On the displayed page, click Register Device. On the displayed page, set device registration parameters. Click OK. Save the device ID and secret returned after the registration is successful.

* Resource Space 🧿	· · · · · · · · · · · · · · · · · · ·
* Product	Test_1 ~
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics I
* Node ID 💿	Test_1
Device ID	
Device Name	
Description	
	0/2,048 //
Authentication Type (?)	Secret X.509 certificate
Secret	
Confirm Secret	Ø

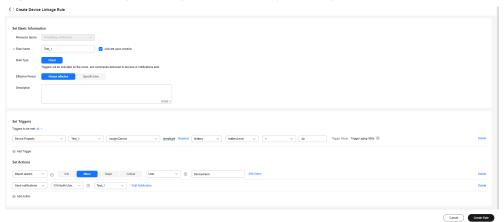
Figure 5-5 Registering a device - MQTT

**Table 5-3** Parameters for registering a device

Parameter	Description
Product	Select the product created in <b>3</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticati on Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

 In the navigation pane, choose Rules > Device Linkage, and click Create Rule. (Before creating a rule, select the resource space to which the rule will belong.) 9. Set the parameters based on the table below. The following parameter values are only examples. You can create your own rules by referring to **User Guide**. After setting the parameters, click **Create Rule**.

#### Figure 5-6 Creating a linkage rule - BatteryProperty



#### **Table 5-4** Parameters for creating a linkage rule

Parameter	Description
Rule Name	Enter a name, such as <b>Battery_Low_Mail</b> or Battery_Low_SMS.
Activate upon creation	Select Activate upon creation.
Effective Period	Select <b>Always effective</b> .
Description	Provide a description of the rule, for example, "When the battery level reported by a device is lower than 20%, an alarm is reported and a notification is sent."
Set Triggers	<ol> <li>Click Add Trigger.</li> <li>Select Device Property.</li> <li>Select the product added in 3, select Assign Device, and then select the device added in 7.</li> <li>Select Battery for Select service, batteryLevel for Select property, &lt; as the operation, and enter 20. Click Trigger Mode. In the dialog box displayed, set Trigger Strategy to Repetition suppression and Data Validity Period (s) to 3600, and click OK.</li> </ol>

Parameter	Description
Set Actions	Add an alarm.
	1. Click Add Action.
	2. Select Report alarms.
	<ol> <li>Set the alarm severity to Minor, alarm isolation level to Device, Alarm Name to Low battery level, and Description to The battery level is lower than 20%. Check and replace the battery in time. Click OK.</li> </ol>
	Add a notification.
	1. Click Add Action.
	2. Select Send notifications.
	3. Select the region where SMN is available, for example, cn-north-4. When you create a rule for connecting to SMN for the first time, a cloud service access authorization window will be displayed based on the cloud service to connect and region. Configure cloud service access authorization as prompted. (You can log in to the SMN console and view the information in the upper left corner.)
	<ol> <li>Select the topic created when configuring SMN for Topic Name.</li> </ol>
	<ul> <li>If Protocol corresponding to the topic is Email, set Message Title to an email title, for example, [Huawei IoT Platform] Low Battery Warning, and set Message Content to information similar to You have a device with less than 20% charge, please log in to the Huawei IoT Platform for details.</li> <li>If Protocol corresponding to the topic is SMS, left Message Title unspecified, and set Message Content to information similar to You have a device with less than 20% charge, please log in to the Huawei IoT Platform for details.</li> </ul>

# Verifying the Configurations

- You can use a registered physical device to access the platform and enable the device to report the battery level less than 20.
- You can also use a simulator to simulate a device to report the battery level less than 20. For details, see **Developing an MQTT-based Simulated Smart Street Light Online**.
- You can also use a virtual device for online debugging and enable the device to report the battery level less than 20.

Expected result:

 In the navigation pane on the left, choose O&M > Device Alarms. Click Application Operations Management (AOM) to go to the AOM console. A minor alarm is generated indicating that the device battery is low.

- If you have subscribed to email notification, the mailbox receives an email indicating that the device battery is low.
- If you have subscribed to SMS notification, the mobile phone receives an SMS notification indicating that the battery level is low.

# 5.2 Automatic Device Shutdown Upon High Temperature

# Introduction

The IoT platform supports device data reporting and command delivery. To associate the two, an application needs to provide corresponding logic.

However, with the rule engine function provided by IoT Device Access, the platform can automatically deliver specified commands when specific data is reported, reducing the application development workload.

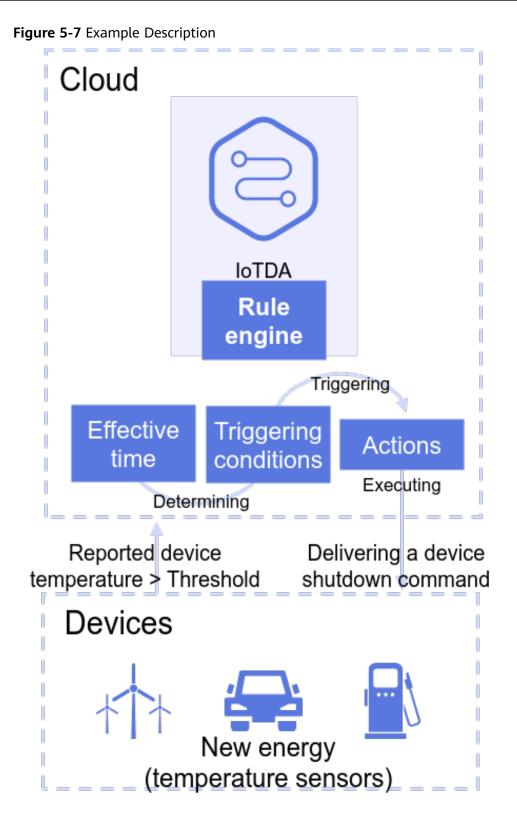
# **Example Scenario**

In this example, when the temperature reported by the temperature sensor of a device is higher than 80°C, the IoT platform automatically delivers a command to shut down the device.

The procedure is as follows:

1. **IoTDA**: Define product models, create devices, and configure linkage rules.

2. Verify the access.



# **Configuring IoTDA**

Using IoT Device Access, you can create a product model, register a device, and set a device linkage rule to enable the IoT platform to send a command when receiving specific data from the device.

- 1. Visit the **IoTDA** product page and click **Access Console**. Click the target instance card.
- In the navigation pane, choose **Products**.
   **Note**: The product model and device used in this document are only examples. You can use your own product model and device.
- 3. Click **Create Product** to create a product using MQTT. Set the parameters and click **OK**.

Basic Information	
Product Name	Enter a value, for example, <b>MQTT_Device</b> .
Protocol	Select MQTT.
Data Type	Select <b>JSON</b> .
Manufacture r	Customize the value.
Industry	Set the parameters as required.
Device Type	

#### Table 5-5 Parameters for creating a product

- 4. Click **Profile\_tempSensor.zip** to download a sample product model.
- 5. On the **Basic Information** tab page, click **Import from Local**. In the displayed dialog box, load the local product model and click **OK**.

Figure 5-8 Uploading a model file

Import from Local	$\times$
A The service definition in the imported file will replace the original service definition of the product.	
After you develop a product model based on the format standards, you can pack an upload it. Learn about product models.	d
★ File Add a file and upload it. Select File	$\supset$
Configure the model file by referring to the template.	
J. Product Template	
Cancel	ОК

 In the navigation pane, choose Devices > All Devices. On the displayed page, click Register Device. On the displayed page, set device registration parameters.

* Resource Space	
★ Product	Test_1 ~
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics 🖒
* Node ID 💿	Test_1
Device ID 🧿	
Device Name	
Description	
	0/2,048 2
Authentication Type ③	Secret X.509 certificate
Secret	Ø
Confirm Secret	Ø

Figure 5-9 Registering a device - MQTT

Parameter	Description
Product	Select the product created in <b>3</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticati on Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

Click **OK**. Save the device ID and secret returned after the registration is successful.

- In the navigation pane, choose Rules > Device Linkage, and click Create Rule. (Before creating a rule, select the resource space to which the rule will belong.)
- 8. Set the parameters based on the table below. The following parameter values are only examples. You can create your own rules by referring to **User Guide**. After setting the parameters, click **Create Rule**.

Figure 5-10 Creating a linkage rule - TemperatureProperty

Set Basic Informat	tion .	
Resource Space	Defaulting_Effectual v	
* Rule Name	Tec_11 C Attude upon creation	
Rule Type	Chuid Direkko Sikle	
Effective Period	Togen of the activate on the Court, and community delivered to devices or notifications sent.           Aways editation         Specific time	
Description		
Set Triggers		
Triggers to be met all		
Device Property	v Set_t v Anips Device v streetigt Restel tengSenser v tempentare v > v (o Tipper Node Tripper aging 3006 ⊗	Delete
Add Trigger		
Set Actions		
Deliver commands	✓ stitelight Reside: GenosSwith ∨ OILOFF ∨ Configure Plasmatur	Delete
Add Action		
		Cancel Create Rule

Parameter	Description
Rule Name	Specify the name of the rule to be created, for example, <b>Overheated</b> .
Activate upon creation	Select Activate upon creation.
Effective Period	Select <b>Always effective</b> .
Description	Provide a description of the rule, for example, "The device is automatically shut down when the device temperature is higher than 80°C."
Set Triggers	<ol> <li>Click Add Trigger.</li> <li>Select Device Property.</li> <li>Select the product added in 3, select Assign Device, and then select the device added in 6.</li> <li>Select tempSensor for Select service, temperature for Select property, &gt; as the operation, and enter 80. Click Trigger Mode. In the dialog box displayed, set Trigger Strategy to Repetition suppression and Data Validity</li> </ol>

#### Table 5-7 Parameters for creating a linkage rule

Parameter	Description
Set Actions	1. Click Add Action.
	2. Select <b>Deliver commands</b> , and select the device created in <b>6</b> .
	3. Select <b>deviceSwitch</b> for <b>Select service</b> , and <b>ON_OFF</b> for <b>Select command</b> . Click <b>Configure Parameter</b> . In the dialog box displayed, set <b>power</b> to <b>OFF</b> , and click <b>OK</b> .

# Verifying the Configurations

- You can use a registered physical device to access the platform and enable the device to report the temperature greater than 80.
- You can also use a simulator to simulate a device to subscribe to Topic: \$oc/ devices/{device\_id}/sys/properties/report (replace {device\_id} with the actual device ID) and report data whose temperature is greater than 80. For details, see Developing an MQTT-based Simulated Smart Street Light Online.
- You can also use a virtual device for online debugging and enable the device to report the temperature greater than 80.

Expected result:

- If you use a physical device to report data, the device receives an ON\_OFF command in which **power** is **OFF**.
- If you use a simulator to report data, you can view the ON\_OFF command in which **power** is **OFF** on the **Subscribe** tab page.

# 5.3 Automatically Opening the Window upon High Gas Concentration

# Introduction

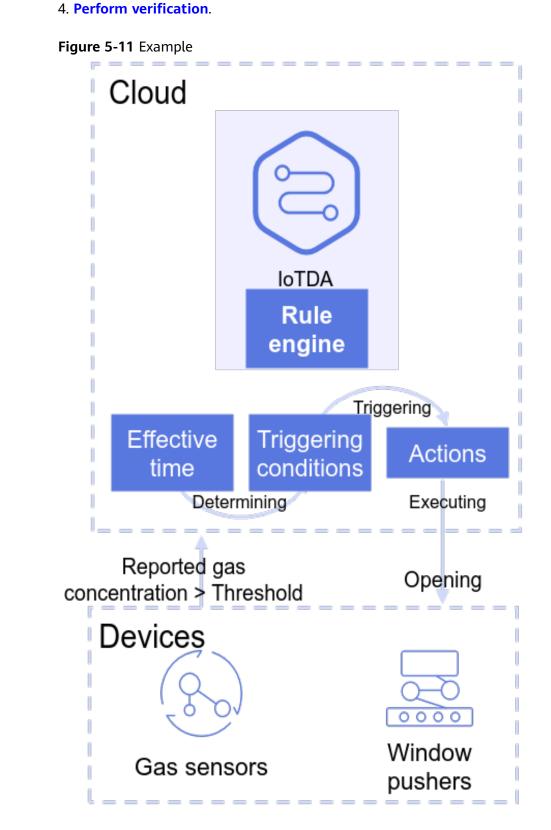
IoTDA can instruct a wireless window opener to open the window through a device linkage rule.

# **Example Scenario**

In this example, a gas detector reports the gas concentration value to the IoT platform. When the gas concentration exceeds a specific threshold, the preset device linkage rule is triggered. The platform delivers a window opening command to the wireless window opener, which then opens the window as instructed.

The procedure is as follows:

- 1. Create a gas monitoring product.
- 2. **Register the device** based on the gas monitoring product.
- 3. Configure a device linkage rule.



# **Creating a Gas Monitoring Product**

**Step 1** Visit the **IoTDA** product page and click **Access Console**. Click the target instance card.

- **Step 2** In the navigation pane, choose **Products**.
- **Step 3** Click **Create Product** on the left to create a gas monitoring product, set the parameters, and click **OK**.

Table 5-8 Parameters	s for cre	eating a	product
----------------------	-----------	----------	---------

Basic Information	
Product Name	Enter a value, for example, <b>gasdevice</b> .
Protocol	Select <b>MQTT</b> .
Data Type	Select <b>JSON</b> .
Industry	Customize the values.
Device Type	

**Step 4** On the **Basic Information** tab page, click **Customize Model** and configure the product model based on the table below. The **gaslevel** service monitors the gas concentration. The **windowswitch** service executes commands for opening and closing windows.

**Table 5-9** Parameters of the gas concentration monitoring product model

Service ID	Туре	Description	
gaslevel	Properties	Property Name: gaslevel	
		Data Type: int	
		Access Permissions: Read	
		Value range: 0–100	
windowswitch	Commands	Command Name: switch	
		Parameter Name: switch	
		Data Type: enum	
		Enumerated Values: on,off	

----End

# **Registering a Device**

Step 1 In the navigation pane, choose Devices > All Devices. On the displayed page, click Register Device. On the displayed page, set device registration parameters. Register the gas monitoring device and record the device ID and secret.

Register Device	
* Resource Space 💿	(1494-147) v
* Product	gasdevice ~
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics $\ensuremath{\mathbb{C}}$
* Node ID 💿	gasdevice
Device ID 📀	
Device Name	gasdevice
Description	
	0/2,048 1/2
Authentication Type (?)	Secret X.509 certificate
Secret	
Confirm Secret	Ø

Figure 5-12 Registering a device - gasdevice

Table 5-10 Parameters for registering a device

Parameter	Description
Product	Select the product created in step <b>3</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticatio n Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

Step 2 In the navigation pane, choose Devices > All Devices. On the displayed page, click Register Device. On the displayed page, set device registration parameters. Register a window opener device and record the device ID and secret.

Register Device	
* Resource Space	(1)
* Product	gasdevice ~
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics $\ensuremath{\mathbb{C}}$
* Node ID	windowswitch
Device ID 🕜	
Device Name	windowswitch
Description	
	0/2,048 //
Authentication Type 🧿	Secret X.509 certificate
Secret	Ø
Confirm Secret	
	Cancel

## Figure 5-13 Registering a device - windowswitch

Table 5-11 Parameters for registering a device

Parameter	Description
Product	Select the product created in <b>3</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticatio n Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

----End

# Configuring a Device Linkage Rule

- Step 1 In the navigation pane, choose Rules > Device Linkage, and click Create Rule. (Before creating a rule, select the resource space to which the rule will belong.)
- **Step 2** Set the parameters based on the table below. The following parameter values are only examples. You can create your own rules by referring to **User Guide**. After setting the parameters, click **Create Rule**.

Figure 5-14 Creating a linkage rule - GasLevel

Set Basic Informa	tion	
Resource Space		
* Rule Name	Test_01 Statusha legon creation	
Rule Type	Crowd Device Side	
	Triggers will be activated on the cloud, and commands delivered to devices or notifications sent.	
Effective Period	Always disclow Specific time	
Description		
	0256.4	
Set Triggers Triggers to be met: si Device Property (c) Add Trigger Set Actions Deliver commands (c) Add Action	v) ( pasterior v) (Asign Device v) pasterior Research ( pasteriel v) ( pasteriel v) ( * v) ( * v) ( * ) Trigger Moles Trigger aprig 106: ()	Delete
		ancel Create Rule

Parameter	Description
Rule Name	Specify the name of the rule to create, for example, <b>windowswitch</b> .
Activate upon creation	Select Activate upon creation.
Effective Period	Select <b>Always effective</b> .
Description	Enter a description of the rule, for example, "Automatically opens the window when the gas concentration is higher than 6".
Set Triggers	<ol> <li>Click Add Trigger.</li> <li>Select Device Property.</li> <li>Select the product added in 3, select Assign Device, and then select the device added in 1.</li> <li>Select gaslevel for Select service, gaslevel for Select property, &gt; as the operation, and enter 6. Click Trigger Mode. In the dialog box displayed, set Trigger Strategy to Repetition suppression and Data Validity Period (s) to 300, and click OK.</li> </ol>

Table 5-12 Parameters for creating a linkage rule

Parameter	Description
Set Actions	1. Click Add Action.
	<ol> <li>Select Deliver Commands, and select the device created in 2.</li> </ol>
	<ol> <li>Select windowswitch for Select service, and on_off for Select command. Click Configure Parameter. In the dialog box displayed, set switch to on, click OK.</li> </ol>

----End

# Verifying the Configurations

#### Method 1:

You can use MQTT.fx to simulate device verification.

- Use MQTT.fx to simulate a gas detector and a window opener, and connect them to the platform. For details, see Developing an MQTT-based Simulated Smart Street Light Online.
- 2. Open MQTT.fx that simulates the window opener to subscribe to commands delivered by the platform.
  - a. Click the **Subscribe** tab.
  - b. Enter **Topic=\$oc/devices**/{*device\_id*}/**sys/commands**/# of the command delivered by the subscription platform. (Replace {*device\_id*} with the device ID obtained in **2**.)
  - c. Click **Subscribe** to deliver the subscription.

Figure 5-15 Creating an MQTT subscription

Publish 2	Subscribe	Scripts	Broker Status	Lo	g 3
\$oc/devices	/		/sys/sh	•	Subscribe

- 3. Switch to MQTT.fx that simulates the gas detector to report properties.
  - a. Click the **Publish** tab.
  - b. Enter topic **\$oc/devices**/{*device\_id*}/**sys/properties/report** for property reporting. (Replace {*device\_id*} with the device ID obtained in **1**.)
  - c. Report the property **gaslevel** with a value greater than 6.

```
Example:
{
    "services": [{
        "service_id": "gaslevel",
        "properties": {
            "gaslevel": 45
            }
        }
}
```

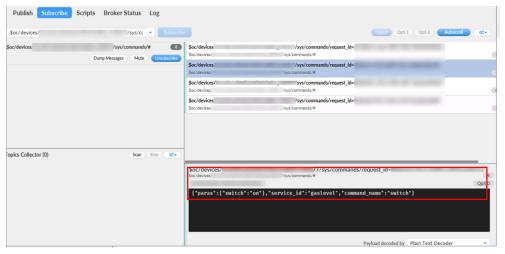
d. Click **Publish** to report the property value.

#### Figure 5-16 MQTT property reporting

Publish Subscribe	Scripts B	roker Status Log		
0	2	\$oc/devices/	*	Publish 4
	6	<pre>{     "services": [[         "service_id": "gaslevel",         "properties": {             "gaslevel":45             }         }     ] }</pre>		

4. Switch to MQTT.fx that simulates the window opener and click the **Subscribe** tab. The command carrying **switch** with the value set to **on** delivered by the platform is received.

#### Figure 5-17 Viewing delivered commands



#### Method 2:

You can use a registered physical device to access the platform and enable the device to report the **gaslevel** greater than 6. The device receives a command carrying **switch** with the value set to **on** and automatically opens the window.

# 5.4 Monitoring Device Status Changes and Sending Notifications

### Introduction

Device administrators need to know connection statuses of IoT devices, such as IoT gateways.

IoTDA provides the rule engine function to meet this requirement. You can easily enable the platform to send a notification when the device status meets a certain condition.

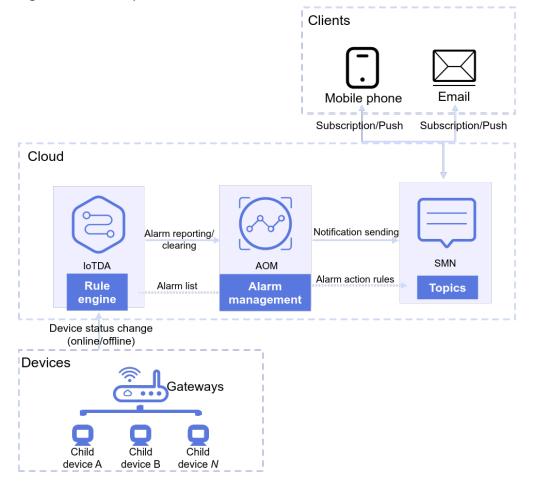
# **Example Scenario**

An enterprise has a batch of gateways under a gateway product. About 400 child devices are mounted to a gateway. Administrators need to check gateway statuses in real time to ensure that child devices report data properly. In addition, the gateways are connected to the IoT platform over the 4G network, frequent alarms are generated due to network jitter. Administrators consider that the scenario where devices go offline and quickly go online is normal and do not want to be notified of this scenario.

The following example shows how to monitor all gateways. When a gateway is offline for 5 minutes, an alarm is reported by IoTDA. When the gateway is back to online for 1 minute, the alarm is cleared and an email or SMS is sent to a specified address or mobile number.

The procedure is as follows:

- 1. **Configure IoTDA.** Create an IoT product and device and create a linkage rule so that alarms can be sent to Application Operations Management (AOM) when device status conditions are met.
- Configure Simple Message Notification (SMN). Create an SMS or email subscription.
- 3. **Configure AOM**. Create alarm rules to process alarms reported by IoTDA and send SMS or email notifications using SMN.



#### Figure 5-18 Example

# **Configuring IoTDA**

Create a product model, register a device, and configure a device linkage rule on IoTDA. When a device is offline for 5 minutes, an alarm is reported to AOM. This alarm is cleared 1 minute after the device goes online.

#### **NOTE**

The product model and device used are only examples. You can use your own product model and device.

- **Step 1** Visit the **IoTDA** service page and click **Access Console**. Click the target instance card.
- **Step 2** Choose **Products** in the navigation pane and click **Create Product** on the left to create an MQTT product. Set the parameters and click **OK**.

Basic Informat	Basic Information	
Product Name	Enter a value, for example, <b>MQTT_Device</b> .	
Protocol	Select <b>MQTT</b> .	
Data Type	Select <b>JSON</b> .	
Industry	Set the parameters as required.	
Device Type		

Table 5-13 Parameters for creating a product

- **Step 3** Click **here** to download a sample product model.
- **Step 4** Click the created product. The product details page is displayed.
- **Step 5** On the **Basic Information** tab page, click **Import from Local**. In the dialog box displayed, load the local product model and click **OK**.

#### Figure 5-19 Uploading a model file

Import from Local	$\times$
A The service definition in the imported file will replace the original service definition of the product.	
After you develop a product model based on the format standards, you can pack and upload it. Learn about product models.	
* File Add a file and upload it. Select File	
Configure the model file by referring to the template.	
Cancel OK	

Step 6 In the navigation pane, choose Devices > All Devices. On the displayed page, click Register Device. On the displayed page, set device registration parameters. Click OK. Save the device ID and secret returned after the registration is successful.

Resource Space 🧿		~
Product	MQTT_Device	~
	Mqtt devices have subscribed to the platform preset topic by defatopics 🖸	ult. Subscribed
Node ID 🕜	gateway_1	
Device ID (?)	10111-00000-01001-01_pm-nc_1	
Device Name	gateway_1	
Description		
		0/2,048 //
Authentication Type 💮	Secret X.509 certificate	
Secret		Ø
Confirm Secret		ر ک

Figure 5-20 Registering a device - gateway

Parameter	Description
Product	Select the product created in <b>4</b> .
Node ID	Set this parameter to the IMEI, MAC address, or serial number of the device. If the device is not a physical one, set this parameter to a custom character string that contains letters and digits.
Device Name	Customize the value.
Authenticatio n Type	Select <b>Secret</b> .
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates a secret.

Step 7 In the navigation pane, choose Rules > Device Linkage, and click Create Rule. (Before creating a rule, select the resource space to which the rule will belong.) **Step 8** Configure rule parameters based on the table below to create a rule for reporting offline gateway alarms. The following parameter values are only examples. You can create your own rules by referring to **User Guide**. After setting the parameters, click **Create Rule**.

Parameter	Description	
Rule Name	Customize a value, for example, Gateway_Offline_Monitor.	
Activate upon creation	Select Activate upon creation.	
Effective Period	Select <b>Always effective</b> .	
Description	Describe the rule, for example, <b>This alarm is generated when the gateway is offline for 5 minutes.</b>	
Set Triggers	1. Click Add Trigger.	
	2. Select Device Property.	
	<ol> <li>Select the MQTT_Device product created in 4, select All Devices, and select Offline as the trigger status.</li> </ol>	
	4. Set <b>Duration</b> to 5 minutes.	
Set Actions	1. Click Add Action.	
	2. Select Report alarms.	
	3. Set the alarm severity to <b>Major</b> , alarm isolation level to <b>Device</b> , <b>Alarm Name</b> to <b>GatewayStatusChange</b> , and <b>Description</b> to <b>The gateway is offline</b> . Check the issue and arrange personnel for maintenance in a timely manner., and click OK.	

**Table 5-15** Parameters for creating a linkage rule

#### Figure 5-21 Creating a linkage rule - GatewayOffline

Set Basic Informa	
Resource Space	
* Rule Name	Dateway_Offine_Monitor 🖉 Activate upon creation
Rule Type	Claud Device Side
	Triggers will be achivated on the cloud, and commands delivered to devices or notifications sent.
Effective Period	Always shocked Specific time
Description	This aliam is generated when the galaxies is offline for 5 minutes.
	<b>50</b> 256.4
Set Triggers Triggers to be met; all	
Device Status	√   M0TT, Device √   /M Devices √   Office √   Outline √   Database Semicability   Database Semica
Dence change	·) (mericana ·) (mericanaa ·) (mericana ·) (
Add Trigger	
Set Actions	
Report alarms	O Info Minor Minor Critical Device V O GatewayStatus/Change Edit/Airm Device
Add Action	
Q	
	Cancel Create Fule

**Step 9** Create a linkage rule for alarm clearance.

Set Basic Inform		
Resource Space		
resource opece		
* Rule Name	Galereisy_Chrite_Monitor Creation	
Rule Type	Cloud Device Side	
	Triggers will be activated on the cloud, and commands delivered to devices on notifications sent.	
Effective Period	Aways ethicitie Specific time	
Description	This aliam is cleared when the gateway is online for 1 minute.	
	\$2256.4	
Set Triggers		
Triggers to be met: a		
	al v v) (MOTLDexice v) (Al Devices v) (Dailee v) Daildon Tematro)	Delete
Triggers to be met: a		Delete
Triggers to be met: a Device Status  Add Trigger		Delete
Triggers to be met: a Device Status  Add Trigger Set Actions	v     MOTT_Device     v     Centre     v     Durinton     Triminde(s)	
Triggers to be met: a Device Status  Add Trigger Set Actions Clear alarms		Delete
Triggers to be met: a Device Status  Add Trigger Set Actions	v     MOTT_Device     v     Centre     v     Durinton     Triminde(s)	
Triggers to be met: a Device Status  Add Trigger Set Actions Clear elarms	v     MOTT_Device     v     Centre     v     Durinton     Triminde(s)	

Figure 5-22 Creating a linkage rule - GatewayOnline

### NOTICE

- The alarm name, alarm severity, and alarm isolation dimension together identify an AOM alarm. The three attributes of the alarm to clear must be the same as those specified during alarm reporting. Otherwise, the alarm clearance will fail.
- There is a flow control for device status monitoring. If a large number of devices are monitored, flow control is triggered. As a result, alarms cannot be reported. For details, see Limitations.

----End

### **Configuring SMN**

On the Simple Message Notification (SMN) console, create a topic and add a subscription for AOM to invoke to send emails or SMS messages.

- Step 1 Log in to Huawei Cloud and access Simple Message Notification (SMN).
- Step 2 Click Access Console. If you have not subscribed to SMN, subscribe to it first.
- Step 3 Choose Topic Management > Topics, and click Create Topic.
- Step 4 Enter a topic name, for example, Test\_1, and click OK.

Figure 5-23 Creating a topic - SMN

Create Topic		×
* Topic Name	Test_1 ⑦ The name cannot be changed after the topic is created.	
Display Name	•	
* Enterprise Project	default    C  (2)  Create Enterprise Project	
CTS Log		
Tag	It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. View predefined tags C To add a tag, enter a tag key and a tag value below.	
	Enter a tag key     Enter a tag value     Add       Tags you can still add: 20	
	OK Cancel	

**Step 5** Choose **Topic Management** > **Subscriptions**, and click **Add Subscription**.

**Step 6** Enter the subscription information. Click **OK**.

Figure 5-24 Adding a subscription - SMN

tion		
Test_1		
SMS	V	
Endpoints	Description	
Add Endpoint Batch Add Endpoints		
Batch Add Endpoints		ОК Салс
	Test_1 SMS Endpoints (+) Add Endpoint	Test_1 SMS  Endpoints Description  Add Endpoint

Parameter	Description	
Topic Name	Select the topic created in <b>Step 4</b> .	
Protocol	<ul> <li>To send an email notification, select Email.</li> <li>To send an SMS notification, select SMS.</li> </ul>	
Endpoint	<ul> <li>If Protocol is set to Email, enter the email address for receiving notifications.</li> </ul>	
	<ul> <li>If Protocol is set to SMS, enter the mobile number for receiving notifications.</li> </ul>	
	To add multiple endpoints, place one endpoint in a line. A maximum of 10 lines can be entered.	

#### Table 5-16 Parameters for adding a subscription

----End

# **Configuring AOM**

Create alarm rules and alarm action rules on AOM. When IoTDA alarms are reported, AOM handles the alarm and sends email or SMS notifications.

- **Step 1** Log in to Huawei Cloud and visit **AOM**.
- **Step 2** Click **Try Free** to access the AOM console. If you have not subscribed to AOM, subscribe to it first.
- **Step 3** Choose **Alarm Center > Alarm Action Rules** and click **Create**.
- **Step 4** Enter an alarm action rule name, for example, **Test\_1**, select the **Test\_1** topic created in **Configuring SMN**, and click the button to confirm the setting.

Figure 5-25 Creating an action rule - AOM

Create Alarm Act	tion Rule
* Rule Name	Test_1
* Enterprise Project	default •
Description ⑦	🖉
* Action Type	Metric/Event Log
* Action	Notification
* Topic	Test_1 💿 🔹 C
	If you do not see a topic you like, create one on the SMN console.
* Message Template	aom.built-in.template.en   C Create Template   View Template
	OK Cancel

#### **Step 5** Choose **Alarm Center > Alarm Rules** and click **Create Alarm Rule**.

Step 6 Enter a rule name, for example, Gateway\_Status\_Change\_Alarm\_Rule, select the event alarm and custom event options, enter IoTDA in Alarm Source, set Select Object to event\_name=GatewayStatusChange (GatewayStatusChange is the alarm name), set Triggering Policy to Immediate Triggering, set Alarm Mode to Direct Alarm Reporting, set Action Rule to the action rule created in 4, and click Create Now in the lower right corner.

Alarm Rule Settings	
Rule Type	
Metric alarm rule Event alarm rule	
Event Type	
System Custom	
Event Source	
IoTDA -	
Alarm Rule Details	
Click Custom to specify an event name. Then you can view the event on the Alarm List > Events page.	
Monitored Object	
Q Custom Attributes: event_name (2) If you select Event Name but do not specify any event, all events will be processed.	×
a Event Name All events - Trigger Mode Immediate Trigger - Atam Severity 🔇 Major -	
Edit	
Alarm Notification	
Alarm Mode 💿	
Direct alarm reporting Alarm noise reduction	
Action Rule 💽	
Test_1 - C 🗒	
	Confirm

Figure 5-26 Creating an alarm rule - AOM

----End

### **Verifying Configurations**

- You can use a registered physical device to access the platform.
- You can also use a simulator to simulate a device going online and offline. For details, see Developing an MQTT-based Simulated Smart Street Light Online.

Expected result:

- 1. After the device is offline for 5 minutes:
  - In the navigation pane, choose O&M > Device Alarms. Click Application Operations Management (AOM) to go to the AOM console. A major GatewayStatusChange alarm is generated.

#### Figure 5-27 Current alarms - Going to AOM

< 🖸 🚥 🚃	<ul> <li>O Running</li> <li>B</li> </ul>	C Details & Modify …
Overview	Device Alarms	🕒 Quick Links
Products Devices   Rules   OBM   Reports	Allem date is instructioned of Application Convertions Meansported (COM) (2)         The can handle adams, when haddwall adams, and calibrates adam rules as the ACM cancels.           Total 1         O Criticatio         Marcol 2         Marcol 2         Waters to a damage of the ACM cancels.           Total 1         O Criticatio         Marcol 2         Marcol 2         Waters to a damage of the ACM cancels.           Marcol 2         Marcol 2         Marcol 2         Marcol 2         Marcol 2           Marcol 2         Marcol 2         Marcol 2         Marcol 2         Marcol 2	00
Online Debugging Message Trace Device Alarms	Control         Servity ()         Resource Type ()         Resource Type ()         Resource Type ()         Allern Description ()         Created () <sup>4</sup> Genery Solutio/Darye         Major         Rule         (* clean/date/date/?sec*/rt/_s/M/_2/M/_2/M/_2/M/_2/M/_2/M/_2/M/_2/M	
Anomaly Detection Run Logs Remote Login Remote Configuration	blad Recoder t 1 19 v v < (€) >	

- If the alarm rule and email notification action are configured in AOM, the target email address will receive an email notifying that the gateway is offline.
- If the alarm rule and SMS notification action are configured in AOM, the target number will receive an SMS notifying that the gateway is offline.
- 2. After the device is back online for 1 minute:
  - The major alarm is cleared. You can view the alarm in the historical alarm list.
  - If the alarm rule and email notification action are configured in AOM, the target email address will receive an email notifying that the gateway goes online.
  - If the alarm rule and SMS notification action are configured in AOM, the target number will receive an SMS notifying that the gateway goes online.