IoT Device Access

Service Overview

 Issue
 1.0

 Date
 2024-11-06





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 Quick Device Access - Property Reporting and Command Receiving	1
1.1 Subscribing to IoTDA	1
1.2 Connecting a Smart Smoke Detector to the Platform (Quick Usage)	2
1.3 Registering a Simulated Smart Street Light Device	6
1.4 Using MQTT.fx to Simulate Communication Between the Smart Street Light and the Platform	15
1.5 Using a Virtual Smart Street Light to Communicate with the Platform (Java SDK)	22
1.6 Using a Virtual Smart Street Light to Communicate with the Platform (C SDK)	34
2 Quick Device Access - Message Sending and Receiving	45
2.1 Overview	45
2.2 Subscribing to IoTDA	47
2.3 Registering a Device	48
2.4 Using MQTT.fx to Send and Receive Messages	50
2.5 Using Device SDKs to Send and Receive Messages	56
3 Quick Application Access	60

1 Quick Device Access - Property Reporting and Command Receiving

1.1 Subscribing to IoTDA

This section describes how to subscribe to a free IoTDA instance of the standard edition in the CN-Hong Kong region.

- **Step 1** Access the **IoTDA** service page and click **Access Console**.
- **Step 2** In the navigation pane, choose **IoTDA Instances** and click the button for subscribing to a free instance.

Figure 1-1 Standard edition - Enabling free instances



Step 3 Configure as required (recommended: default value).

< IoTDA Stand	lard Edition			
Region Q Billing Mode	> > >			
Specificatio	ons			
Unit Type		Messages Per Day per Unit ③	Size of Each Message	
SUF		10,000	4 KB	
Selected Specification	ons SUF Maximum Messages per Day 10,000			
* Instance Name	freeStandardInstance			
Tags	TMS's predefined tags are recommended for adding the same tag to different cloud resources.	Create predefined tags 🕑 🔾		
Description	+ Add Tag Vex can el 20 more legis. Cotonut 6256 4			
Free				Cancel

Figure 1-2 Instance - Configuring a free instance

Step 4 Click **Create**, and the instance page is displayed. Refresh the page and wait until the instance status changes to **Running**, indicating that the free instance has been created.

Figure 1-3 Instance - Free instance created



----End

1.2 Connecting a Smart Smoke Detector to the Platform (Quick Usage)

Scenarios

You can use device data collection and command receiving with a Windows or Linux PC as a virtual device. This section uses a virtual smart smoke detector as an example to describe how to connect a device to the platform, enable the device to report data, and deliver commands from the platform to the device.

Prerequisites

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

Procedure

- **Step 1** Access the **IoTDA** service page and click **Access Console**. Click the target instance card.
- **Step 2** In the navigation pane, choose **Overview** and click **Start** to start the experience.

Figure 1-4 Start

< 📀	Standar	d-Standard Y O Running		R Details & Modify 🕲 Change to Pay-per-Use
Overview		IoT Device Access Start Your Journey to IoT		
Products	~	Start		(1) Hide
Rules	~	Quick Experience 5min	Development Using Virtual Devices 10 min	Development Using Sample Code 30 min
O&M Resource Spaces	~	A predefined smoke sensor is used to demonstrate the process from device access to device management.	Develop a product and debug product functions on the platform.	Develop a device and an application, and connect them to the platform.
Documentation	Ľ	Start	View Guide	View Guide

Step 3 A smart smoke detector model has been predefined for the quick usage. On the displayed page, check the properties and commands of the model and click **Create Product**.

IoTDA Experience Wizard	Define Product
	A product model defines product details and service capabilities. We will create a smoke detector based on a product model that has already been defined.
	Properties
	Property Data Type Mandatory Access Mode Description
Set data reporting period	alarm int true Readable, Writ Check whether there is a smoke alarm. 0: n
Alarms	smokeConcentration decimal true Readable Smoke density
Property I	temperature decimal true Readable Temperature
TATA	humidity int true Readable Humidity
Humidity	Commands
Temperature Property 4	Command Name Command Parameter Response Parameter Description
	setReportingFrequency value None Set data reporting frequency
Smoke density Property 3	Resource Space DefaultApp_667bmvk1 ()
	* Product Name smokeDetector
	* Device Type ③ smokeDetector ~
	Create Product

Figure 1-5 Quick experience - Creating a product

Step 4 Create a virtual smart smoke detector. Customize the node ID and device name. Click **Register Device**.



Figure 1-6 Quick experience - Registering a device

Step 5 Select a device demo package as required.

Figure 1-7 Quick experience - Selecting a device demonstration package



Step 6 Click **Download Demo Package** and decompress the package as prompted. Run the **huaweicloud-iot-device-quickstart.exe** command. If the device status changes from **Inactive** to **Online** and properties such as the temperature have reported values, the device has been connected to the platform.



Figure 1-8 Quick experience - Configuring the simulated device



Download the Demo Package and decompress it to hardcode the device information generated on the platform to the simulation device. Download Demo Package The demo package is used to simulate data reporting. The reported data includes the configured product information. Run huaweicloud-iot-device-quickstart exe to check the device connectivity status. (You must decompress the demo package before running the start bat script. Otherwise, an error will occur) Device Name myDeviceName Device Status Online Refresh	IoTDA Experience Wizard	Configure Simulation Device
After the device comes online, you can view the reported property data in the device shadow. Property Reported At Reported Value Description -Readable, Wirtable >atarm May 06, 2024 16:18:06 GMIT-08:00 Check whether there is a smoke al -Readable >Wirtable >atarm May 06, 2024 16:18:06 GMIT-08:00 Check whether there is a smoke al -Readable >temperature May 06, 2024 16:18:06 GMIT-08:00 Smoke density -Readable >temperature May 06, 2024 16:18:06 GMIT-08:00 64 Humidity Vou are advised to set the reporting frequency to 1 and click Deliver Command so that the device reports data every second. If you do specify a value, the default setting of reporting frequency Integer [1, 100] Deliver Command Previous Complete	DIDA Experience Wizard	Configure Simulation Device • Download the Demo Package and decompress it to hardcode the device information generated on the platform to the simulation device. Device Demo Package The demo package is used to simulate data reporting. The reported data includes the configured product information: • Device Name Device Status Device Name Device Status • After the device comes online, you can view the reported property data in the device shadow. Property Reported At • Readable > smokeConcent May 06, 2024 16:18:06 GMT+08:00 Check whether there is a smoke al • Readable > smokeConcent May 06, 2024 16:18:06 GMT+08:00 Check whether there is a smoke al • Readable > humidity May 06, 2024 16:18:06 GMT+08:00 Themperature • Vou are advised to set the reporting frequency to 1 and click Deliver Command so that the device reports data every second. If you do not prove of available statis stering or reporting data every the seconds will apply. • setReportingFrequency Integer (1, 100) Deliver Command

Step 7 Set setReportingFrequency (property reporting frequency). Click Deliver
 Command to deliver the new reporting frequency to the device. Check the update speed of the reported property values before and after the setting.



Figure 1-10 Quick experience - Setting device property reporting frequency

----End

Advanced Experience

By following the instructions in this section, you may have understood how to connect a device to the platform and related concepts.

You can customize a product and use a virtual device and simulated application to use the basic functions and perform development on the platform. For details, see **Registering a Simulated Smart Street Light Device**.

1.3 Registering a Simulated Smart Street Light Device

Scenarios

This section describes how to use the MQTT.fx tool to simulate a smart street light. You can connect the simulated light to IoTDA, report the light intensity data to the platform, and deliver a light-on command to the light.



Figure 1-11 Communications between a simulated smart street light and the platform

Prerequisites

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

Service Process

The MQTT.fx simulator is used as an example to describe data reporting and command delivery. Download MQTT.fx (64-bit OS) or MQTT.fx (32-bit OS) and install it. Overall service process:

- 1. Product creation: Create an MQTT smart street light product on the console. Define a product model to develop a street light that supports light intensity reporting and status control commands.
- 2. Device registration: Register an MQTT smart street light on the console.
- 3. Device connection establishment: Use MQTT.fx to simulate a smart street light, complete connection authentication, and activate the device registered on IoTDA.
- 4. Data reporting: Use MQTT.fx to simulate a smart street light to report light intensity data to IoTDA.
- 5. Command delivery: Deliver a street light switch command on the console to remotely control MQTT.fx to simulate a smart street light.

Creating a Product

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

< 🖸 Heelitanda	ndinal	ande Deces Deces						🖟 Details d	& Modify …
Overview Products	F	Products						Description [Ouick Links
Devices ~									
Rules V O&M V Resource Spaces		Description On the IoT platform, a product is a collection of devices v If you want to view data reported by a device and manag access protocol and data format. Learn more	with the same capabilities and features. e and control the device on the platform, you nee	td to develop a product model (a profile). The product model infor	ms the platform of the prope	erties and comman	ds that are supported by devices. You ca	n also define product model details b	ased on the
Documentation (2)		Create Products A product is a set of devices that have the same cap Product Development Quide	2 Defining Produ ablifies or features. Define product m	ct Models	3 Register Device The registered devi to the platform.	s	By information required to connect	4 Device-side development Integrates device SDKs to come Device Development Guide	ect devices to the
	¢	Create Product Delate Q. Select a property or enter a keyword.			L Borto Tora D	L Burrant G			00
		Product Name 😔	Product ID 😔	Resource Spaces 😔	Device Type 😣	Protocol 8	Created 😣	Operation	
			Statistical Technology	Delasting_Hitlesed	water?	10277	Aug 20, 2004 20 HI 10 OM7-48 HI	View Copy More ~	
		Total Records: 1 10 \checkmark < 1 \Rightarrow							

Figure 1-12 Creating a product

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

 \times

Figure 1-13 Creating a product - MQTT

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 \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 **Model Definition** tab page, click **Customize Model** to add services of the product.

Figure 1-14 Custom model - MQTT

IoTDA Instances / I	Products / aaasssss			
<	Registered devices			Culck Links
Basic Information	Codec Deployment Online Debugging	Topic Management		
Product Det	ai ai			
Product Name	2		Resource	
Device Type	-		Protocol MOTT	
Data Type	ison		Created	
Industry			Description - &	
	A product model describes product deta	Basic Serv SET_PRESSURE_READ_PERCO Period Value Result and service capabilities. Yes do not define Contracts Wood (The Service) and the service matching multiple methods. If yes do not define Contracts Wood (The Service) and the service matching multiple methods.	Votree Using Lovel a product model for a device, the platform only torwards the data reported by the device and does no come (import trans Losery) Learn more	form in the



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

Figure 1-15 Adding a service - BasicData

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

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

 \times

★ Property Name	luminance	
Description	light intensity.	
		16/128 //
★ Data Type	Integer	~
* Access Permissions	Read Write	
★ Value Range	0 - 65535	5
Step		
Unit		

Figure 1-16 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.

★ Command	Switch			
Name				
Command	Add Command Parameter			
Parameters	Parameter Name ⇔ Data Type ⇔	Description 🖨	Operation	
	No	table data available.		
	No Command Paramete	rs data available. Add Command	Parameter first.	
		dd Command Parameter		
	Total Records: 0 5 v < 1 >			
Response	Total Records: 0 5 v < 1 > Add Response Parameter			
Response Parameters	Total Records: 0 5 ∨ < 1 > Add Response Parameter Parameter Name ⊕ Data Type ⊕	Description 🔶	Operation	
Response Parameters	Total Records: 0 5 ✓ 1 > Add Response Parameter	Description 🔗	Operation	
Response Parameters	Total Records: 0 5 <	Description 😚	Operation	
Response Parameters	Total Records: 0 5 < (1) > Add Response Parameter Parameter Name () Data Type () No	Description ⇔	Operation	
Response Parameters	Total Records: 0 5 < 1 > Add Response Parameter Parameter Name Data Type No No Response Paramete	Description 🕀	Operation Parameter first.	

Figure 1-17 Adding a command - Switch

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

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

Figure 1-18 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 1-19 Registering a device

< 🖸 usemessag	R Details & Moorty
Overview	All Devices Total devices 0: @ Activated devices 0: @ Online devices 0 @ Online devices 0 @ Outre devices 0
Products Devices	Device List Batch Registration Batch Lipidate Batch Detelsion Batch Add Devices To Group File Lipidates
Groups Policies	
Software/Firmware Upgrades	Internet A Device Water A Mode ID A Device ID A Hereorice Parce A Homory A Mode UA A Obersoon
Device Certificates Device Proxy	
Self-Registration Template Custom	No table data available.
Rules ~	Register Device
08M ~	

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 Secret .
Secret	If you do not set this parameter, IoTDA automatically generates a value.

Figure 1-20 Registering a device - MQTT

* Resource Space (?)	Carlandinan, Art Senat	~
✤ Product	Test_1	~
	Mqtt devices have subscribed to the platform preset topic by d topics $\ensuremath{\mathbb{C}}^2$	efault. Subscribed
* Node ID 🧿	Test_1	
Device ID 💿		
Device Name		
Description		
		0/2,048 1/2
Authentication Type 💿	Secret X.509 certificate	
Secret		Ś
Confirm Secret		8

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.

Figure 1-21 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.	
Device ID	
Device Secret	
Next, you can use the IoT Device SDK to connect devices to the platform.	
SDK Development Guide 🖄	
Download	

----End

1.4 Using MQTT.fx to Simulate Communication Between the Smart Street Light and the Platform

Connecting the Simulated Street Light to the Platform

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 device details page, find **MQTT Connection Parameter**, and click **View** to check the clientId, username, password, and hostname.

Figure 1-22 Device - Device details



 \times



MQTT Cor	nnection Parameter	
clientId	100752 (artists (1997)),	
usemame	BRYCHARDS IN TRACK,	
password	D	
hostname	a televise or right or right	constantine on C
port	8883 🗇	
		Cancel Download

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

Figure 1-24 MQTT.fx Settings

e Extra	s Help					
iot					Connect	Disconnect
Publish	Subscribe	Scripts	Broker Status	Log		
/huawe	ei/v1/devices/8	ханасыв-он	od-Abdurb vitte6fe9:	2561 🝷	Publish	Ì

Step 4 Click the **User Credentials** tab and set authentication parameters by referring to the following table.

3 3 3		•				
Edit Connection Profiles						\times
iot		Deedle Marca	• •			_
local mosquitto		Profile Name	lot		B 400	
		Profile Type	MQTT Broker		IVQ	ORG
	MQTT Broker Prot	file Settings				
		Broker Address	1100000000			
		Broker Port	1883			
		Client ID	a8fffi0000000000000000000000000000000000	Generate		
	General User Cr	redentials	SSL/TLS Proxy LWT			
		User Name	5e8			
		Password				
+ -	Revert			Cancel	к	pply

Figure 1-25 Configuring authentication parameters

Table 1-1 Parameter description

Parameter	Description				
Broker Address	Host name, which is obtained in 2 . The access address is a domain name. For devices that cannot be connected to the platform using a domain name, run the ping <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	8883. In this example, port 8883 is used for secure connection.				
Client ID	Enter the device client ID obtained in 2.				
User Name	Enter the device ID obtained in 2 .				
Password	Enter the encrypted device secret obtained in 2.				

Step 5 Click the SSL/TLS tab and then select Enable SSL/TLS. Recommended: Set Protocol to TLSv1.2. Click CA certificate file, go to the certificate resources page to download the certificate file of the corresponding region and instance version, and enter the complete local path of the certificate file in the text box. Click Apply, and then click Cancel to exit the configuration page.

Edit Connection Profiles		o ×
M2M Eclipse		
iot	Profile Name iot	
	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	-
	CA signed server certificate	
	CA certificate file	
	CA Certificate File	em
	CA certificate keystore	
	Self signed certificates	
	 Self signed certificates in keystores 	
+ -	Revert	OK Apply

Figure 1-26 Setting SSL/TLS parameters

Step 6 Click **Connect**. If the icon in the upper right corner turns green, the device simulator has been authenticated and connected. The device status displayed on the platform is **Online**.

Figure 1-27 Device simulator connected

WQTT.fx - 1.7.1	– D X
File Extras Help	
iot - Connect Disconnect	
Publish Subscribe Scripts Broker Status Log	
» Publish	QoSO QoS1 QoS2 Retained OST



< 🙆 and the state of the state								
Overview	All Devices Total devices 2	II Devices Tetal devices 2						
Products Devices ^	Device List Batch Registration Batch Update Batch Deletion Batch Add Devices To Group File Uploads							
All Devices	Register Device Device United Treeze							
Groups	Q. Search by node ID by defa	C Search by rode (D by default.						
Policies	Status O	Device Name	Node ID \varTheta	Device ID (e)	Resource Space Θ	Product (e)	Node Type \varTheta	Operation
Sottware/Firmware Upgrades	 Online 	1412		Ball/16/87/82/8646/782.1	Delastrap, Hillions	Strains	Dracity corrected	View Debug More ~

----End

Reporting Light Intensity Data

Use MQTT.fx to report light intensity data to the IoT platform. If a device reports data through the MQTT channel, the data needs to be sent to a specific topic in the format **Soc/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 URL, for example, **\$oc/devices**/{*device_id*}/**sys/properties/report**.

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

Table 1-2 Service data list

Field	Mandat ory/ Optiona l	Туре	Description
services	Mandato ry	List <servicepr operty></servicepr 	Service data list. (For details, see the ServiceProperty structure below.)

 Table 1-3
 ServiceProperty structure

Field	Manda tory/ Option al	Туре	Description
service_id	Mandat ory	String	Service ID.
propertie s	Mandat ory	Object	Service properties, which are defined in the product model associated with the device.

Field	Manda tory/ Option al	Туре	Description
eventTim e	Optiona l	String	UTC time when the device reports data. The format is yyyyMMddTHHmmssZ, for example, 20161219T114920Z .
			If this parameter is not carried in the reported data or is in incorrect format, the time when IoTDA receives the data is used.

Request example:

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

Step 3 Check whether the device successfully reports data on the device details page. As shown in the following figure, the luminance is updated to 30.

Figure 1-30 Viewing reported data - MQTT

e Info	Cloud Run Logs Cloud Delivery	Device Shadow Message Trac	e Device Monitoring Child Devices Tags			_
wice Name	and the second second second	Resource	Space	Product	Test_1	
wice ID	107120-249-1-1892. J	Node ID	Conversion of	Authentication	Secret Reset Secret	
ide Type	Directly connected	Firmware	Version	Software Version	-	
scription	- a	Registere	ALC R. 2024 (\$19.10) 10 (\$87.40.10)	Activated	ACR. 2010/01/01 12:007-02:00	
st Offline	APR. 2014/03/2012 007-018	MOTT Co	nnection View			
		Paramete				
oduct Mode perty data rep e: If the repor	el Data ported by the device based on the product mod ried property name is not contained in the produ	t definition. ct model, or the property name contains dots	.), dollar symbols (5), or empty char (the hexadecimal ASCII code is	10), the property data cannot be updates		(Vew All Properties
oduct Mode operty data rep te: If the repor	el Data ported by the device based on the product mod ried property name is not contained in the produ- tion name.	t definition. cl model, or the property name contains dots Latest Reported Time	(), dotter symbols (3), or empty char (the hexadecimal ASCII code to	10), the property data cannot be updated		(View All Properties)
oduct Mode operly data rep ite: If the repor	el Data ported by the device based on the product mod red property name is not contained in the produ- tice name.	I definition. ct model, or the property name contains dots Latest Reported Time), dollar symbols (3), or empty char (the hexadecimal ASCII code is t	10), the property data cannot be updated	L.	View All Properties
oduct Mode operly data rep te: If the report infer the servis BasicData JohtControl	el Data ported by the device based on the product mod rised property name is not centained in the product con name. Q.	I definition. ct model, or the property name contains dots Latest Reported Time	1), defair symbols (3), or empty char (the hexadiscrinel AGCII code is in	10), the property data cannot be updated	L.	Werr All Properties
oduct Mode operly data rep le: If the repor inter the servi BasicData .ightControl	el Data ported by the device based on the product most inted property name is not contained in the produ- tice name. Q	I definition. cf model, or the property name contains dets Latest Reported Time Iuminance), datar symbols (3), or empty char (the hexadecimal ASCs) code to	20), the property data cannot be updated		User Al Properties
oduct Mode operty data rep te: If the report infer the servia BasicData JightControl	el Data control by the device based on the product mod property name in not contained in the produ- too name.	t definition. cl model, or the property name contains dots Latest Reported Time luminance 30	1), dellar symbols (3), or empty char (he hesadacinal ASCII code is 1	30), the property data cannot be update	ı.	View All Properties
oduct Mode operty data rep de: If the report Enter the servia BasicData LightControl	el Data goldel by the davice based on the product most end grapping name is not contained in the produ- see name. Q.)	i defession. ct model, or the property name contains dots Latest Reported Time luminance 30	1), defair lymbols (B), or empty char (the hexadiomal ABCII code is i	30), the property data cannot be updated	L. C.	User At Poperties
oduct Mode operty data rep te: If the report inter the servi- BasicData JightCentrol	el Data de la bite device hassed en life product mot ende property name is nel contained in the produ- rice name: Q	tedelation. Latest Reported Time Latest Reported Time 30 Table Records 1 19 ~ ()	1), defair tymbels (B), or energy char (the hexadicinal AGCI code is	bit, the property data cannot be updated		User At Properties

----End

Remotely Delivering Commands for Turning On the Light

Deliver a command on the console to remotely control smart street lights.

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 Cloud Delivery tab, click Deliver Command, set Command to LightControl: Switch, and set Value to ON to deliver a command for turning on the light.

Figure 1-31 Command delivery -	- Synchronous command	delivery
--------------------------------	-----------------------	----------

IoTDA Instances / All Devices / Device Details	
Conline	
Device Info Cloud Run Logs Cloud Delivery O 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 Synchropous Command Delivery Video Listense exercises and	• 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 (2)
Deliver Command	* Command (
	value 🧕
	Cancel OK 3



 For synch is sent. C 	ronously delivered command, device should send re therwise, the status of this command will be set as "	esponse within 20 seconds after the comman Timed Out'. Learn more 🕑
Command	LightControl: Switch	~
value	ON	~

NOTE

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

Step 3 In the MQTT.fx simulator, click the Subscribe tab and enter the command delivery topic. After the subscription, check the delivered command parameters. The format of the command delivery topic is Soc/devices/{device_id}/sys/commands/#. As shown in the following figure, the MQTT.fx simulator has received the command whose command name is Switch and value is ON.



Figure 1-33 Checking the delivered command parameters

NOTE

The device needs to respond to the delivered synchronous command in a timely manner. However, MQTT.fx does not automatically report the command response. Therefore, the console page may display a message indicating that the command request times out. For details about the command response, see **Platform Delivering a Command**.

----End

1.5 Using a Virtual Smart Street Light to Communicate with the Platform (Java SDK)

Overview

This section describes how to connect a device to Huawei Cloud IoTDA through MQTTS/MQTT using Java code, implement southbound data reporting and command delivery using **platform APIs**, and receive messages subscribed by the northbound server using the application-side sample code. Taking a smart street light as an example, the device reports information such as luminance to IoTDA, and an application receives device data pushed by IoTDA.

Prerequisites

- You have installed JDK 1.8 or later.
- You have installed IntelliJ IDEA. If you have not installed IntelliJ IDEA, visit the IntelliJ IDEA official website to download and install it.

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.

Procedure

- **Step 1** Access the **IoTDA** service page and click **Access Console**.
- **Step 2** Choose **Products** in the navigation pane and click **Create Product**.

Figure 1-34 Creating a product

< 🙆 📾	eltandar	finalance v O Ru	inning					🖟 Details 🖧 Modify 🚥
Overview Products		Products						Description Duick Links
Devices	\sim							
Rules	~	Description						
O&M	~	On the IoT platform, a product is a collection	on of devices with the same capabilities and features.					
Resource Spaces		If you want to view data reported by a devia access protocol and data format. Learn mo	ce and manage and control the device on the platform, you nee Ke	d to develop a product model (a profile). The product model in	forms the platform of the proper	ties and command	s that are supported by devices. You ca	n also define product model details based on the
Documentation	C	1 Create Products	2 Defining Produ	ct Models				4 Device-side development
		A product is a set of devices that have	the same capabilities or features. Define product me	dels to describe the capabilities and features of devices.	The registered devic	e obtains the iden	ity information required to connect	Integrates device SDKs to connect devices to the
		Product Development Guide			to the platform.			Device Development Guide
		< Create Product Delete						
		 Select a property or enter a keyword 						Q
		Droduct Name A	Broduct ID &	Percente Spaces	Device Type A	Brotocol A	Created A	Operation
		- Product manne (P	Productio O	Resource spaces ()	Dence type ()	PICKOLOI O	Citana Q	Operation
			Statistic day "In Cold Property	Defaultings_Hildhood	9407	MUTT	Aug 28, 2024 20 08 16 087 48 08	View Copy More ~
		Total Records: 1 10 V <	1 >					

Step 3 In the displayed dialog box, set parameters based on your requirements.

Х

Figure 1-35 Creating a product - MQTT

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 ⑦	
Advanced Settings \land	Custom Product ID Description
Product ID 💿	
Description	
	0/128 //
	Cancel OK

Step 4 Download the **model file**. For details about the development process, see **Developing a Product Model Online**.

Step 5 After the product is created, click the product, and then click **Import from Local** to upload the downloaded model file. The model file does not need to be decompressed, and the package name cannot contain brackets.

Figure 1-36 Uploading a product model - MQTT

Kegistered devices: 0		Cutck Links Em Adding to model library
Basic Information Codec Deployment Online Debugging Topic Ma	anagement	
Product Detail		
Product Name 2		Resource Space
Device Type		Protocol MQTT
Data Type Json		Created
Industry		Description - 2
A product model describes product details and service of 英文 가 简 ⓒ ⓒ	Bail: Genico Bail: Genico SET_PRESSURE_READ_PERIOD Period Value Result spacifices a product mode using institute methods. If you do not define a product mode Continues Mode (Import from Local) (more true Excel)	Butoy Butoy Votage Level of for a dwice, the pattern only browth: the data reported by the dwice and does not parse the data. mport than Learn V

----End

Creating a Device

Step 1 In the navigation pane, choose **Devices** > **All Devices**, and click **Register Device**.

Figure 1-37 Registering a device

< 🙆 usermensag	primit means means	 O Running 						@ Details	& Modify …
Overview	All Devices Total dev	rices 0 • Activated devices 0 • Onlin	r devices 0						Cuick Links
Products									
Devices ^	Device List Ba	atch Registration Batch Update	Batch Deletion Batch Ar	dd Devices To Group File Uploads					
All Devices	Register Device	Delete Untreeze F	0200						
Groups	Q. Search by node I	ID by default.						Q Advanced Searc	n ~) 🛞
Policies	Status O	Device Name O	Node ID . O	Davica ID -Q	Resource Space O	Broduct Q	Note Tupe O	Operation	
Software/Firmware Upgrades	0 00000	Conto Mario Q	1000 0	Control O	instance space (Product Q	1000 1997 0	oputation	
Device Certificates									
Device Praxy									
Self-Registration Template	<			=					
Contem				No table data av	ailable.				
Authentication				No Devices data available. Reg	ister Device first.				
Rules ~				Register Devic					
osm ~									
Resource Spaces									

Step 2 In the displayed dialog box, configure the parameters by referring to the following figure (select the created product), and click **OK**. If you do not specify **Secret**, a secret will be automatically generated by the platform. In this example, the secret is automatically generated.

Figure 1-38	Registering	a device	(test123)
-------------	-------------	----------	-----------

Register Device	
* Resource Space 💿	Calledings (RTable) V
* Product	
	Mqtt devices have subscribed to the platform preset topic by default. Subscribed topics \boxdot
★ Node ID 🧿	test123
Device ID 🕜	Helia (H.: /Helia) (** Tao Island), [Heli [*]] (1)
Device Name	
Description	
	0/2,048 2
Authentication Type 🧿	Secret X.509 certificate
Secret	Ø
Confirm Secret	Ø
	Cancel

Step 3 After the device is created, save the device ID and secret, which will be used for device connection.

Download

Figure 1-39 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.	
Device ID	
 Next, you can use the IoT Device SDK to connect devices to the platform. SDK Development Guide ⁽²⁾ 	

----End

Importing Sample Code

- **Step 1** Download the Java demo.
- Step 2 Open the IDEA developer tool and click Import Project.

Welcome to IntelliJ IDEA		_	×
	1		
	Intelli, LIDEA		
	Version 2016.3.2		
	┿ Create New Project		
	💕 Import Project		
	늘 Open		
	븆 Check out from Version Control 🗸		
		🏶 Configure 🗸	Get Help 👻

Step 3 Select the Java demo downloaded in 1 and click Next.

🖳 Import	Project					×
O Create	project from <u>e</u> xisting s	ources				
	project from external r	nodel				
	project from external <u>i</u>	nodel				
Gra	dle					
<i>m</i> Ma	ven					
			Previous	Next	Cancel	Help

Step 4 Import the sample code.



----End

Establishing a Connection

To connect a device or gateway to the platform, upload the device information to bind the device or gateway to the platform.

- Before establishing a connection, modify the following parameters: // MQTT connection address of IoTDA static String serverIp = "iot-mqtts.cn-north-4.myhuaweicloud.com"; // Device ID and secret obtained during device registration (Replace them with the actual values.) static String deviceId = "yourDeviceID"; // device_id obtained during device registration static String secret = "yourSecret"; // secret obtained during device registration
 - serverIp indicates the address used by devices to access IoTDA using MQTT. For details about how to obtain the address, see Obtaining Resources.
 - device_id and secret indicate the device ID and secret, which can be obtained after the device is registered.
- 2. Run the program. The device is displayed as online on the platform.

Figure 1-40 Device list - Device online status

K 👩 usernessag	: 👩 and and a constant of Running 🛛 Details & Moothy -					🛱 Details 🖧 Modify 😶	
Overview	All Devices Total devices 2 (# Activated devices 1 # Online devices 1						
Products Devices	A Device Lat Batch Registration Batch Update Batch Design Batch Add Devices To Group File Uploads						
All Devices	Register Device Delete Untreeze F	Register Device Delete Unitedas Freeze					
Groups	Q. Search by node 10 by default. Q) Advanced Search ~> (●)						
Policies	Status	Node ID 😣	Device ID 😣	Resource Space 😣	Product @	Node Type 😣	Operation
Upgrades	Online will	-	Matt/1608/102100000100.1	Defaulting, Millions	Stratio	Drach, consolid	View Debug More ~

Reporting Properties

A device reports its properties to IoTDA. (The sample code implements scheduled reporting. You can view the data reported by the device in IoTDA by referring to **Viewing Reported Data**.)

// Report JSON data. service_id must be the same as that defined in the product model.
String jsonMsg = "{\"services\":[{\"service_id\":\"BasicData\",\"properties\":{\"luminance\":32},\"eventTime
\":null]]}";

- The message body jsonMsg is assembled in JSON format, and service_id must be the same as that defined in the product model. properties indicates a device property.
- luminance indicates the street light brightness.
- **eventTime** indicates the UTC time when the device reports data. If this parameter is not specified, the system time is used by default.

After a device or gateway is connected to the platform, you can call **publish(String topic,MqttMessage message)** of **MqttAsyncClient** to report device properties to the platform.

Viewing Reported Data

After the **main** method is called, you can view the reported device property data on the device details page. For details about the API, see **Device Reporting Properties**.

Figure 1-41 Viewing reported data - luminance

NOTE

If no latest reported data is displayed on the device details page, modify the services and properties in the product model to ensure that the services and properties reported by the device are consistent with those in the product model. If they are inconsistent, the data reported by the device is not available on the historical data page. Alternatively, delete all services on the **Basic Information** page.

Delivering a Command

Step 1 In the navigation pane, choose **API Explorer**.

< 🖸 userne	C Details & Modely **								
Overview		All Devices Total devices	All Devices Total devices 21 Activated devices 2 Online devices 1						
roducts									
levices	~	Device List Batch F	Registration Batch Update	Batch Deletion Batch Add	Devices To Group File Uploads				
tules	~	Register Device	Delete Unfreeze Freeze						
38M	×	O. Search by node ID by	default.						Advanced Search v
esource Spaces		Status 🖯	Device Name 😣	Node ID \varTheta	Device ID 🖯	Resource Space \ominus	Product 🖯	Node Type \varTheta	Operation
ocumentation	C	· · mite	n/becatiane	matterned.	Hallshield Witchield Witchield	Detection, Millions	and address of the second	Draft, consider	View Debug More ~
T Device Provisioning PI Explorer	C	· · · · · · · ·	April 10 and 10 and 10	Name (Second V)	0x0756/0702/max/00_4	Defaultion_ICINers	Test, 1	Drafty contracted	View Debug More ~
		- • maine	Name Device 110	Name and Address of the	19407/1007/00/100/001/10.	Information, IClines	Test, 1	Dracky connected	View Debug More ~
		· · · · · · · ·	NavaDarce108	Name and Address and Ad	Rest Construction (Co.)	Defaultion, Millions	Text, 1	Draft consolid	View Debug More ~

Figure 1-42 Navigation pane -API retrieval and debugging

Step 2 Locate the row that contains the device command. For details about the delivered parameters, see the figure (consistent with those in the product model). Then, click **Debug** to send the command.

HUAWEI CLOUD	Console		Search	۵	Billing R	esource Service	Enterprise	ICP Lic
API Explorer	IoT Device Access		CLI Example	Debugging Result	Documentation	Mock Data	Scenarios	
Overview								
Debugging History								
Version History • d ^o								
All Products								
My Products								
Elastic Cloud Server		1 - 1						
Cloud Service Engine	No APIs available.	No request parameters are found.				No documen	ntation available	ole.

- **service_id** indicates the service ID, for example, **BasicData**.
- **command_name** indicates the command name, for example, **lightControl**.
- paras indicates a delivered parameter, for example, {"switch":"ON"}.

You can view the received commands on the device. (The sample code has implemented the subscription to the command receiving topic.)





Obtaining Data Reported by a Device from the Cloud

The following uses AMQP as an example to describe how to obtain data reported by a device to the cloud.

- **Step 1** Obtain the Java AMQP access **demo**.
- **Step 2** Log in to the **console**, choose **Rules** > **Data Forwarding**, and click **Create Rule** to create a data forwarding rule.

Figure 1-43 Data forwarding - Creating a rule

< 🖸 userness	append and a second of the sec	द्ध Details के Modify •••
Overview	Data Forwarding	Culck Links
Products		
Devices ~	ROUE LESI ANICIP CODEDS	
Rules ^	You can add and manarge rules that trigger corresponding actions on connected devices when specific conditions are met.	
Data Forwarding	Create Rule Import Rule Configure Policy Pinate Connection Configuration Diskte Exutite Disable	
Server Certificates	Q. Select a property or enter a keyword.	0
Device Linkage	Rule Name θ Rafe ID θ Resource Space θ Data Source θ Trigger θ Status θ Operation	
osm ~	c .	
Resource Spaces		
Documentation (2	3	
	No table data available.	
	No Ruleo data available. Cireate Rule first.	
	Crede Rule	

Step 3 On the **Set Forwarding Data** page, configure parameters, and click **Create Rule**.

-	• •	
< Create Data F	orwarding	Cutck Links
Set Forwarding Data	(2) Bet Forwarding Target (3) Enable Rule	
Basic Information		Specify the data source, trigger, and SQL statements. You can also click Quick Configuration to quickly configure specific data or Edit SQL for more complex queries.
* Rule Name	TestRulePropertiesUp	SUL Filter statements
Description	description	SELECT * FROM DEVICE_PROPERTY_REPORT
	18236 %	
* Data Source	Device property v ()	
* Tripger	Device property reported V	
* Resource Space	All resource spaces v	
		_

Figure 1-44 Data forwarding - Creating a property reporting rule

Parameter	Description
Rule Name	Customize a rule name.
Description	Describe the rule.
Data Source	Select Device property .
Trigger	Select Device property reported .
Resource Space	Select All resource spaces .

Step 4 Set the forwarding target. Note that you need to click **Preset Access Credential** to download the file.

Figure 1-45 Creating a forwarding target - to an AMQP push message queue



Parameter	Description
Forwarding Target	Select AMQP message queue.

Parameter	Description
Access Credential	Click Preset Access Credential and save the downloaded file, which includes access_key and access_code .
Message Queue	DefaultQueue is selected by default.

Step 5 Click Enable Rule.

Figure 1-46 Enabling a rule - Forwarding data to AMQP

Step 6 Modify the parameters in the AMQP sample code obtained in **Step 1**.



- **yourAccessKey**: access key of the access credential. For details about how to obtain it, see **Step 4**.
- **yourAccessCode**: access code of the access credential. For details about how to obtain it, see **Step 4**.
- yourAMQPUrl: AMQP domain name. You can log in to the console, choose Overview, and click Access Addresses to obtain the domain name, as shown in the following figure.

Previous OK

Figure 1-47 Access information - AMQP access address

Access Deta	ills			×
Select the corre	esponding address to complete the access. For	details, see Quick Application Access andQuick Device Access		
 For sec 	urity purposes, CoAP/CoAPS access addresse	is cannot be pinged.		
Access	Access Protocol (Port)	Access Address	Custom Domain Name	Access Control
	HTTPS (443)	rhuaweicloud.com		
Applicati	MQTTS (8883)	rhuaweicloud.com		
	AMQPUrI ->	huaweicloud.com		Preset Access Cred
	CoAP (5683) CoAPS (5684)	myhuaweicloud		
Device a	MQTT (1883) MQTTS (8883) MQ	myhuaweicloud	Details	
	HTTPS (443)	myhuaweicloud		

• yourQueue: queue name. Use the default queue DefaultQueue.

Step 7 AMQP data is received successfully.

AmgpDemo) src) main) java) com) demo) 🥥 HwiotAmąpiavaClientDemo			
g 🚍 Project 🕶 🚱			
		privat final static Encourservice social/Survice - and HeadPollExecter(Bottime optimized, cantaliat/orcossred), meanwork/dbs. Ratine pribation(), available/occessori() + implications ds, Tissuni. TATORNI, are (inceditor.inployee() (pasts 5889));	2,
x ™ con.demo			
HulotAmgsleveClientDemo		String accessMey =	
ing resources ► int test			
🕨 🖿 tasget			
HviotAmaplavaClientDeno.imi			
Im External Libraries			
 Because and the second s		<pre>trop constructioncom() //// // /// /// ////////////////////</pre>	
Run: 🚍 HwiotAmgplevaClientDemo 🗵			
G → onInboundHessage, JmsInboundHessageDispatch (■ ↓ receive an message, the content is (resource) ゆ 品 ☆ 註 ③ 音	equence = 2, messageId "device.property","even	- mill, genumeri - 10 (fanille-fif-adf-adf-adf-adf-file) Tit "report, "read, file" (MININGT-STATE), "safry,date" (fan,er" (fan,er" (fan,er" (fan,er" (fan)er" (fan)er	

----End

Additional Information

For more development guides, see **Using IoT Device SDKs for Access** and **Using MQTT Demos for Access**.

1.6 Using a Virtual Smart Street Light to Communicate with the Platform (C SDK)

Overview

This section describes how to connect a device to Huawei Cloud IoTDA through MQTTS/MQTT using C code, implement southbound data reporting and command delivery using platform APIs, and receive messages subscribed by the northbound server using the application-side sample code. Taking a smart street light as an example, the device reports information such as luminance to IoTDA, and an application receives device data pushed by IoTDA.

Prerequisites

You have installed Linux and GCC (4.8 or later).

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.

- **Step 1** Access 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**.

< 🙆 freeStandard	R Details & Modify
Overview	Products @ Description (*) Quick Links
Products	
Devices 🗸	
Rules 🗸	Description
OBM V	On the IoT platform, a product is a collection of devices with the same capabilities and features.
Resource Spaces	If you want to keek add and shorts. Leave and manage and costrol the device on the pattern, you need to develop a product model (a potio). The product model informs the plattern of the properties and commands that are supported by devices. You can also define product model (delais) based on the access protocal and data forms.
Documentation	1 Deale Products(2) Defining Product Models(3) Register Devices(4) Devices ide development
	A product is a set of devices that have the same capabilities or features. Define product models to describe the capabilities and features of devices. The registered device obtains the identity information required to connect integrates device SDKs to connect devices to the
	Product Development Guide to the platform. Device Development Guide
	Create Product Delete
	C Select a broberty or enter a segment.
	Product Νωπe θ Product D θ Resource Spaces θ Device Type θ Product θ Created θ Operation
	Capit Mar -
	Total Records: 1 10

Figure 1-48 Creating a product

Step 3 In the displayed dialog box, set parameters based on your requirements.

Х

Figure 1-49 Creating a product - MQTT

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 V
Device Type Selection	Standard profile Custom
* Device Type ᠀	
Advanced Settings $ \wedge $	Custom Product ID Description
Product ID 💿	
Description	
	0/128 //
	Cancel OK

Step 4 Download the **model file**. For details about the development process, see **Developing a Product Model Online**.

Step 5 After the product is created, click the product, and then click **Import from Local** to upload the downloaded model file. The model file does not need to be decompressed, and the package name cannot contain brackets.

Figure 1-50 Uploading a product model - MQTT

Registered devices: 0	Cuick Links En Adding to model library
Basic information Codec Deployment Online Debugging Topic Management	
Product Detail	
Product Name Z Resource Tipere	
Device Type Protocol MOTT	
indutty - Deception - 2	
Desice for vice Basic Basic	see the data.

----End

Creating a Device

Step 1 In the navigation pane, choose **Devices** > **All Devices**, and click **Register Device**.

Figure 1-51 Registering a device

< 👩 usernessage	elimit nood house – • • Rut	ning					ि Details के	à Modify •••
Overview	All Devices Total devices 0 • Act	ivated devices 0 Online devices 0					(Cuick Links
Products								
Devices ^	Device List Batch Registra	tion Batch Update Batch Deletio	Batch Add Devices To Group File Uploads					
All Devices	Register Device Delete	Untreeze Freeze						
Groups	Q. Search by node ID by default.						Q Advanced Search	~) (@)
Policies	Status A	Device Name R Node ID	B Device ID B	Resource Space A	Product A	Node Type (B)	Operation	
Software/Firmware Upgrades								
Device Certificates								
Device Proxy								
Self-Registration Termilate	<		=					
Custom			No table data avai	lable.				
Authentication			No Devices data available. Regis	ter Device first.				
Rules ~			Register Device)				
08M ~								
Resource Spaces								

Step 2 In the displayed dialog box, configure the parameters by referring to the following figure (select the created product), and click **OK**. If you do not specify **Secret**, a secret will be automatically generated by the platform. In this example, the secret is automatically generated.

Figure 1-52 Registering a device - test

Register Device		
★ Resource Space ⑦	Defaultrup_INTellul	~
* Product	Gateway	~
	Mqtt devices have subscribed to the platform pre topics $\ensuremath{\mathbb{C}}^3$	set topic by default. Subscribed
* Node ID 💿	test123	
Device ID 🧿	85x3025038113x3x48_3x4123	
Device Name		
Description		
		0/2,048 2
Authentication Type	Secret X.509 certificate	
Secret		1
Confirm Secret		ø

Step 3 After the device is created, save the device ID and secret, which will be used for device connection.

Figure 1-53 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.	
	Device ID	
	Device Secret	
	Next, you can use the IoT Device SDK to connect devices to the platform.	

Download

----End

Importing Sample Code

- **Step 1** Download the sample code quickStart(C).
- **Step 2** Copy the code to the Linux runtime environment. The following figure shows the code file hierarchy.
 - mqtt_c_demo
 mqtt_c_demo
 mqtt_c_demo.c
 mqtt_c_demo.c
 mqtt_c_demo.c
 conf
 rootcert.pem
 conf
 include
 copenssl
 copenssl
 com
 tib
 Makefile

Description of the directories:

- src: source code directory
 mqtt_c_demo: core source code of the demo util/string_util.c: utility resource file
- conf: certificate directory
 rootcert.pem: certificate used by the device to verify the platform identity. It is used for login authentication when the device connects to the platform.
- include: header files
 base: dependent Paho header files
 openssl: dependent OpenSSL header files
 util: header files of the dependent tool resources
- lib: dependent library file
 libcrypto.so*/libssl.so*: OpenSSL library file
 libpaho-mqtt3as.so*: Paho library file
- Makefile: Makefile

----End

Compiling Library Files

- Compiling the OpenSSL library
 - a. Download **OpenSSL**, upload it to any directory on the Linux compiler, and run the following command to decompress it: tar -zxvf openssl-1.1.1d.tar.gz
 - b. Generate a makefile.

Run the following command to access the OpenSSL source code directory:

cd openssl-1.1.1d

Create a directory (for example, **/home/test**) for OpenSSL compilation. mkdir /home/test

Create a directory for OpenSSL compilation. mkdir /home/test/openssl

Create a configuration file directory. mkdir /home/test/openssl/ssl

Run the following configuration command: ./config shared --prefix=/home/test/openssl --openssldir=/home/test/openssl/ssl

In this command, **prefix** is the installation directory, **openssldir** is the configuration file directory, and **shared** is used to generate a dynamic-link library (**.so** library).

If an exception occurs during the compilation, add **no-asm** to the configuration command (indicating that the assembly code is not used).

./config no-asm shared --prefix=/home/test/openssl --openssldir=/home/test/openssl/ssl

[root@server-1908071538 test]# cd openssl-1.1.1d [root@server-1908071538 openssl-1.1.1d]# ./config

c. Generate library files.

Run the following command in the OpenSSL source code directory:

make depend

Run the following command for compilation:

make

Install OpenSSL.

make install

Find the **lib** directory in **home/test/openssl** under the OpenSSL installation directory.

The library files **libcrypto.so.1.1**, **libssl.so.1.1**, **libcrypto.so**, and **libssl.so** are generated. Copy these files to the **lib** folder of **quickStart(C)** and copy the content in **/home/test/openssl/include/openssl** to **include/openssl** of **quickStart(C)**.

/home/test/openssl/lib		
	*	
<u>*</u>	1569672248319596.png	20
🌙 engines-1.1		20
🌛 pkgconfig		20
libcrypto.a	5,461 KiB	20
a libcrypto.so	16 B	20
libcrypto.so.1.1	3,276 KiB	20
libssl.a	996 KiB	20
a libssl.so	13 B	20
libssl.so.1.1	660 KIP	21

Note: Some compilation tools are 32-bit. If these tools are used on a 64bit Linux computer, delete **-m64** from the **makefile** before the compilation.

- Compiling the Eclipse Paho library file
 - a. Download the paho.mqtt.c source code.
 - b. Decompress the package and upload it to the Linux compiler.
 - c. Modify the **makefile**.
 - i. Run the following command to edit the **makefile**: vim Makefile
 - ii. Run the following command to display the number of lines: :set nu
 - iii. Add the following two lines (customized OpenSSL header files and library files) after line 129: CFLAGS += -I/home/test/openssl/include

LDFLAGS += -L/home/test/openssl/lib -lrt

120	
127	<pre>INSTALL_PROGRAM = \$(INSTALL)</pre>
128	$INSTALL_DATA = $ (INSTALL) -m 644
129	DOXYGEN COMMAND = doxygen
130	CFLAGS += -I/home/test/openssl/include
131	LDFLAGS += -L/home/test/openssl/lib -lrt
132	
133	$MAJOR_VERSION = 1$
134	$MINOR_VERSION = 0$
135	<pre>VERSION = \${MAJOR_VERSION}.\${MINOR_VERSION}</pre>
_	

iv. Change the addresses in lines 195, 197, 199, and 201 to the corresponding addresses.



- d. Start the compilation.
 - i. Run the following command: make clean
 - ii. Run the following command: make
- e. After the compilation is complete, you can view the libraries that are compiled in the **build/output** directory.

/home/test/paho.mqtt.c/build/output		
* []		2019/10/23 15:34:01
🍌 samples		2019/10/23 15:34:14
퉬 test		2019/10/23 15:34:16
e libpaho-mqtt3a.so	19 B	2019/10/23 15:34:10
a libpaho-mqtt3a.so.1	21 B	2019/10/23 15:34:10
libpaho-mqtt3a.so.1.0	477 KiB	2019/10/23 15:34:10
e libpaho-mqtt3as.so	20 B	2019/10/23 15:34:13
a libpaho-mqtt3as.so.1	22 B	2019/10/23 15:34:13
libpaho-mqtt3as.so.1.0	529 KiB	2019/10/23 15:34:13
a libpaho-mqtt3c.so	19 B	2019/10/23 15:34:03
a libpaho-mqtt3c.so.1	21 B	2019/10/23 15:34:03
libpaho-mqtt3c.so.1.0	446 KiB	2019/10/23 15:34:03
a libpaho-mqtt3cs.so	20 B	2019/10/23 15:34:07
a libpaho-mqtt3cs.so.1	22 B	2019/10/23 15:34:07
libpaho-mqtt3cs.so.1.0	498 KiB	2019/10/23 15:34:07
paho_c_version	13,768 B	2019/10/23 15:34:14

f. Copy the Paho library file.

Currently, only **libpaho-mqtt3as** is used in the SDK. Copy the **libpaho-mqtt3as.so** and **libpaho-mqtt3as.so.1** files to the **lib** folder of **quickStart(C)**. Go back to the Paho source code directory, and copy **MQTTAsync.h**, **MQTTClient.h**, **MQTTClientPersistence.h**, **MQTTProperties.h**, **MQTTReasonCodes.h**, and **MQTTSubscribeOpts.h** in the **src** directory to the **include/base** directory of **quickStart(C)**.

Establishing a Connection

To connect a device or gateway to the platform, upload the device information to bind the device or gateway to the platform.

- **Step 1** Configure the parameters. Change the values of **username** and **password** only. For details, see **Obtaining Resources**.
- **Step 2** Start the connection.

- 1. Run the **make** command for compilation. Delete **-m64** from the **makefile** in a 32-bit OS.
- 2. Run export LD_LIBRARY_PATH=./lib/ to load the library file.
- 3. Run ./MQTT_Demo.o.
- **Step 3** If the connection is successful, the message "connect success" is displayed. The device is also displayed as **Online** on the console.



Figure 1-54 Device list - Device online status

< 🙆 usernessag	ilimit period besider - • • Running						ଟି Details & Modify ···
Overview	All Devices Total devices 21 Activated devices 11 Online devices 1 Online						Cuick Links
Products							
Devices ^	A Device List Batch Registration Batch Deletion Batch Deletion Batch Add Devices To Group File Uploads						
All Devices	Register Davice Disks Untwerze Frenze						
Groups	Q. Search by node ID by default.						(Advanced Search v)
Policies	Status 😔 Device Name 🕀	Node ID 🕀	Device ID 😣	Resource Space 😔	Product (e)	Node Type \ominus	Operation
Upgrades	Online	-	Mattrial/1010000000000000000000000000000000000	Detection, Millions	Strates	Drafty consolid	View Debug More ~



Reporting Properties

A device reports its properties to IoTDA. The sample code implements scheduled reporting. You can view the data reported by the device in IoTDA. For details, see **Device Reporting Properties**.

//publish data
char *payload = "{\"services\":[{\"service_id\":\"BasicData\",\"properties\":{\"luminance\":32},\"eventTime
\":NULL}]}";

- The message body payload is assembled in JSON format, and service_id must be the same as that defined in the product model. properties indicates a device property.
- **luminance** indicates the street light brightness.
- **eventTime** indicates the UTC time when the device reports data. If this parameter is not specified, the system time is used by default.

If the property reporting is successful, the message "publish success" is displayed in the demo.

The reported properties are displayed on the device details page.

Figure 1-55 Viewing reported data - luminance

ToTDA Instances	JTDA Instances / All Devices / Device Details							
< 200406	In Table 21.2 Service Terrareller Online (0					Cuick Lini	ks
Device Info	Cloud Run Logs Cloud Delivery	Device Shadow Message Tra	ace Device Monitori	ng Child Devices Tags				
Device Name Device ID Node Type Description Last Online	Directly connected - 2	Reso Node Firma Roge MGT Paran	Nurce Space B ID ware Version – stored T Connection View meter	λομουρ.με απ. δ	Product Authentication Type Software Version Activated	Secret Read Secret		
Product Mo Property data Note: If the re	Product Model Data Property data reported to the product model definition. Note: If the reported property mane is not contained dots (), defar symbols (3), or empty durit (the hexadecimal ASCII code is 00), the property data cannot be updated.)	
Enter the se	nkice name. Q	Latest Reported Time:	() >				Enter the property name. Q	

Receiving a Command

After subscribing to a command topic, you can deliver a synchronous command on the console. For details, see **Synchronous Command Delivery to an Individual MQTT Device**.

If the command delivery is successful, the command received is displayed in the demo:

Obtaining Data Reported by a Device from the Cloud

After the platform receives, an application can receive push messages using AMQP. For details, see **Obtaining Data Reported by a Device from the Cloud**.

Additional Information

For more development guides, see **Using IoT Device SDKs for Access** and **Using MQTT Demos for Access**.

2 Quick Device Access - Message Sending and Receiving

2.1 Overview

IoTDA allows devices to access the IoT platform using MQTTS or MQTTT and send and receive messages. This section uses MQTT.fx and Java SDKs as examples to describe how a device sends and receives messages using MQTTS or MQTTT.

Procedure



Figure 2-1 Procedure

Perform the following steps.

- 1. Subscribe to IoTDA. You can select the instances specifications and number of units as required.
- 2. Register a device. Create a product and register a device on the **IoTDA** console.
 - A product is a collection of devices with the same capabilities or features.
 - Device: A device is a physical entity under a product and is used for identity authentication (necessary) when the device connects to the platform. You can use an application to call the API for creating a device, register a device on the console, or use the self-registration capability to automatically register the device when it is connected to the platform.
- 3. Send and receive messages.
 - Use MQTT.fx to send and receive messages.
 - Use device SDKs to send and receive messages.

2.2 Subscribing to IoTDA

This section describes how to subscribe to a free IoTDA instance of the standard edition in the CN-Hong Kong region.

- Step 1 Access the IoTDA service page and click Access Console.
- **Step 2** In the navigation pane, choose **IoTDA Instances** and click the button for subscribing to a free instance.

Figure 2-2 Standard edition - Enabling free instances

IoT Device Access IoTDA Instances Decumentation	IoT Device Access (IoTDA) Haavel Cloud IoT Device Access (IoTDA) Haavel Cloud IoT Device Access (IoTDA) allows you to connect your physical devices to the cloud, where you can collect device state and devices that remote control. It can also work with other Haavel Cloud services to help you guickly develop IoT solutions (Integrating the Standard Estion) Environment of the Standard Estion)	
	C, Barch ty Indaco Name ty debut.	Privilege for enterprise $\[mathbb{Q}_0\]$ Contact Contact our professionals for specific needs within IoT industry.
		Shortcuts
	No data available. No instance data available Buy instance fint. Buy instance	FAQs Learn more In What Scenarios Can the IoT Platform Be Applied?

Step 3 Configure as required (recommended: default value).

Figure 2-3 Instance - Configuring a free instance

Specification	s			
Unit Type		Messages Per Day per Unit ③	Size of Each Message	
SUF		10,000	4 KB	
Tags T7	MS's predefined tags are recommended for adding the same tag to different cloud resources.	Create predefined tags 🗹 📿		
Tags TI	MS's predefined tags are recommended for adding the same tag to different cloud resources	Create predefined taps 🗹 🔾		
+	+ Add Tag			
Description	Ordional			
Description	opoteni			
	0/256.4			

Step 4 Click **Create**, and the instance page is displayed. Refresh the page and wait until the instance status changes to **Running**, indicating that the free instance is successfully created.

Figure 2-4 Instance - Free instance created

IoT Device Access	IoTDA Instances Huave Cloud IoT Device Access (IoTDA) allows you to connect your physic can collect device data and deliver commands to devices for remote control. Cloud services to help soy quickfl devices (IoT access)	al devices to the cloud, where you It can also work with other Huawei			€	Old Version 👌 Highlights Buy Instance
IoTDA Instances						
Documentation (2)	Search by Instance Name by default. freeStandardInstance O Running	10	10,000	1,000	ୟ R Details & Modify ନି Unsubscribe	Privilege for enterprise Q_{0} Contact Contact our professionals for specific needs within IoT industry.
	Bandard SUF * 1 Total Records: 1 5 v < 1 >	Messages TPS	Max Messages	Max Register Devices	Pay-per-use Created on	Shortcuts

----End

2.3 Registering a Device

Before using IoTDA to send and receive messages, you need to create a product and device.

- **Step 1** Access the **IoTDA** service page and click **Access Console**. In the navigation pane, choose **IoTDA Instances** and click the target instance card.
- **Step 2** Create a product. Choose **Products** in the navigation pane and click **Create Product** on the left. Set the parameters as prompted and click **OK**.

Parameter	Description
Product Name	Set it to MyProduct .
Data Type	Select JSON .
Industry	Set this parameter as required.
Device Type	Set this parameter as required.

Х

Figure 2-5 Creating a product - MQTT

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 ⑦	JSON ~
Device Type Selection	Standard profile Custom
* Device Type ᠀	
Advanced Settings \land	Custom Product ID Description
Product ID 🕜	
Description	
	0/128 //
	Cancel OK

Step 3 Register a device. In the navigation pane, choose Devices > All Devices, click Register Device, set parameters based on the following table, and click OK. After the device is registered, keep the device ID and secret properly for authentication when the device accesses the IoT platform.

Parameter	Description
Product	Select the product to which the device belongs.
Node ID	Set it to test001 .
Device Name	Set it to test001 .
Authenticatio n Type	Select Secret .

Parameter	Description
Secret	Customize the secret used for device access. If the secret is left blank, the platform automatically generates one.

Figure 2-6 Device - Registering a secret device

Register Device			
★ Resource Space ⑦			~
* Product			~
* Node ID 💿			
Device ID 💿			
Device Name			
Description			
			0/2,048 %
Authentication Type (Secret	X.509 certificate	
Secret			\$
Confirm Secret			۵
			Cancel OK

----End

2.4 Using MQTT.fx to Send and Receive Messages

Introduction

MQTT.fx is an MQTT client written in Java based on Eclipse Paho. It supports Windows, macOS, and Linux OSs. It can be used to simulate the connection of devices to Huawei Cloud IoTDA through MQTTS/MQTT, and the publishing and subscription of messages through topics. This section uses Windows as an example to describe how to use MQTT.fx to access Huawei Cloud IoTDA and send and receive messages.



Figure 2-7 Message exchange process between MQTT.fx and IoTDA

Using MQTT.fx to Connect to IoTDA

- 1. Download MQTT.fx (64-bit OS) or MQTT.fx (32-bit OS) and install it.
- 2. Open the MQTT.fx client and choose **Extras** > **Edit Connection Profiles** from the menu bar.
- 3. On the Edit Connection Profiles page, set related parameters and click OK.

Figure 2-8 MQTT.fx connection parameters

Profile Name MQTTTest	
Profile Type MQTT Brok	er 🗸
MQTT Broker Profile Settings	
Broker Address	st1.iotda-device.cn-north-4.myhuawe
Broker Port 8883	
Client ID	_0_1_2024 Generate
General User Credentials SSL/TLS	Proxy LWT
User Name	
Password	
Revert	Cancel OK Apply

Parame ter	Descriptio n	Example Value
Profile Name	Name of the configurati on file.	Enter MQTT Test .
Profile Type	Type of the connection to be configured.	The value is fixed at MQTT Broker , indicating that the MQTT server is connected.
Broker Address	Access address of the MQTT server.	Access your instance, choose Overview , click Access Details , and obtain the MQTTS access address. For details, see MQTTS access address .
Broker Port	Access port of the MQTT server.	Enter 8883 .
Client ID User Name Passwo rd	loTDA can send and receive messages only after device access authenticat ion is successful. For details about device authenticat ion parameters, see Device Connectio n Authentica	Go to the device details page, find MQTT Connection Parameter , and click View to check the clientId, username, and password.
SSL/TLS		
Enable SSL/TLS	Whether to use the SSL or TLS encryption protocol.	Yes

Parame ter	Descriptio n	Example Value
Protoco l	Protocol version.	Select TLSv1.2.
CA certifica te file	CA certificate file.	Obtain the CA certificate of the corresponding region from the certificate resource page.

4. After setting the parameters, click **Connect**. If the icon in the upper right corner turns green, MQTT.fx has been connected to Huawei Cloud IoTDA. If the icon in the upper right corner turns red, the connection fails. Click the **Log** tab to check logs, modify the configuration based on the log information, and try again.

Figure 2-9 MQTT.fx connection

MQTT.fx - 1.7.1				 1		\times
File Extras Help						
MQTT Test	- K Connect	Disconnect				-
Publish Subscribe Scripts Broker Status	Log					
>> home/garden/fountain	• Publish	1	Q050 Q	051 Qo52	Retained	

5. Access the **IoTDA** service page and click **Access Console**. In the navigation pane, choose **IoTDA Instances** and click the target instance card. In the navigation pane, choose **Devices** > **All Devices** to check the device status. The device status is expected to be online.

Figure 2-10 Device list - Device online status

< 🖸 usemessage	👩 ann marginal and a local v O Running 🗵 Debis & Mody									
Overview	All Devices Total devices 2 • Act	tivated devices 1 • Online	devices 1					Cuick Links		
Products										
Devices ^	Devices A Device Lat Batch Registration Batch Update Batch Deletion Batch Add Devices To Group File Uploads									
All Devices	Register Device Delete	Untreeze	oeze							
Groups	Q. Search by node ID by default.							() (Advanced Search ~) ()		
Policies	Status 0	Device Name 😣	Node ID 😣	Device ID 😔	Resource Space 😔	tesource Space 🖯 Product 🖯		Operation		
Software/Firmware Upgrades	Online	well .		840716071020ee07011	Delasting, Hillings	Strains	Dracky contracted	View Debug More ~		

Using MQTT.fx to Send Messages

- 1. Click the **Publish** tab on the MQTT.fx client.
- 2. On the displayed tab page, enter the topic name in the **Topic** text box on the left, for example, **/test/deviceToCloud**. Enter the message content in the **Message** text box, for example, **hello**. Click **Publish** to send the message.

Figure 2-11 MQTT.fx message sending

@ MQTT.fx - 1.7.1						-		×
File Extras Help								
MQTTTest	· •	Connect Disconnect						a 🔴
Publish Subscribe Scripts	Broker Status Log							
> /test/deviceToCloud		Publish	Qos 0	Qo51	Qo5 2	Retaine	d)	(0;▼)
hello								

 Access the IoTDA service page and click Access Console. In the navigation pane, choose IoTDA Instances and click the target instance card. In the navigation pane, choose Devices > All Devices. On the displayed page, click View. On the Message Trace tab page, check the messages sent by MQTT.fx.

Figure 2-12 Message tracing - Viewing results

< 2024075471241262Device	1 Dear Dear Dear Dear Dear Dear Dear Dear									
Device Info Cloud Run Logs	Cloud Delivery E	Device Shadow Message Trace	Device Monitoring	Child Devices	Tags	Groups				
Traced messages help you quickly loca To ensure data validity and prevent the	Tacks messages help yor splot/s todate and devely faller cuases. Law more To ensure data waldy and prevent the platform from coupling to many read and write compute and thronge resources, the platform can only face messages for to to 10 devices at a first or any splate set, and for no more than three days.									
Implementation [Running] Start 6	ne ar 1. 200 1. 0. 0. 0.	End time: Jul 10, 3	147-01.01					Stop Trace	Clear Data	
Edit Configuration Export I	Data									
Q Search by service details by defa	ut								0	
Service Type	Service Step 😑	Service Details (e)				Recorded (e)	Message Status \ominus	Operation		

4. After MQTT.fx sends messages to the platform, configure **data forwarding rules** to forward the messages to message middleware, storage, data analysis, or service applications.

Using MQTT.fx to Receive Messages

- 1. Click the **Subscribe** tab on the MQTT.fx client.
- 2. On the displayed tab page, enter the topic name in the **Topic** text box on the left and click **Subscribe**. **/test/cloudToDevice** is used as an example. After the topic is subscribed to, check the topic in the subscription list.

-						_	272.5
WQTT.fx - 1.7.1					-		×
File Extras Help							
MQTTTest	• Ø Connec	Disconnect					a 🔴
Publish Subscribe Scripts Broker Status	Log						
/test/cloudToDevice	 Subscribe 		(0050) Q	oS 1 QoS 2	Autoscr	oll	0,*
/test/cloudToDevice Dump Messages Mute Unsubscrit							

Figure 2-13 MQTT.fx subscription topic

3. Access the **IoTDA** service page and click **Access Console**. In the navigation pane, choose **IoTDA Instances** and click the target instance card. In the navigation pane, choose **Devices** > **All Devices**. On the device list, click a device to access its details page.

4. Click the **Cloud Delivery** tab. On the **Message Delivery** tab page, click **Deliver Message**. In the displayed dialog box, configure the content and the parameters for the message to deliver.

Deliver Message			×
★ Topic Type	System topic	Custom topic	
* Topic	Custom p V	/test/cloudToDevice	
Message Format	 Message content 	only OSystem forma	at
Base64 Encoding	● No		<i>⊳</i>
★ Message Content	String Js	on	
	hello		
			4
Aging Time	- 1,440 +	minutes	
Property Parameter 🗸 🗸			
			Cancel OK

Figure 2-14 Delivering a message - Custom topic

5. On the MQTT.fx client, click the **Subscribe** tab. The message received from the subscribed topic is displayed.

	enting mes.	ages as		~					
MQTT.fx - 1.7.1							-		×
File Extras Help									
MQTTTest		Connect	Disconnect					•	•
Publish Subscribe Script	s Broker Status I	og							
/test/cloudToDevice		Subscribe			(0050) Q	S 1 Qo5 2	Autoscro		•
/test/cloudToDevice Dump Messages	Mute Unsubscribe	/test/cloudToDev	ice					Q	1 oS 0
Topics Collector (0)	Scan Stop OG*	/test/cloudToI 21-08-2024 17: hello	Device 40.58.63658683					Q	1 oS 0
				Pay	/load decoded b	y Plain Text I	Decoder		•

Figure 2-15 Checking messages using MQTT.fx

2.5 Using Device SDKs to Send and Receive Messages

Introduction

Devices can use **SDKs in multiple languages** to quickly connect to IoTDA for upstream and downstream message communications. This section uses Java sample code to demonstrate how a device accesses IoTDA using MQTTS/MQTT and publishes and subscribes to messages using topics.



Figure 2-16 Message exchange between the SDK and IoTDA

Device-side SDK Sending Messages

 Configure the Maven dependency of the SDK on the device. <dependency> <groupId>com.huaweicloud</groupId> <artifactId>iot-device-sdk-java</artifactId> <version>[1.2.1,)</version> </dependency>

- 2. Connect the device to Huawei Cloud IoTDA.
 - Modify the device connection parameters before establishing a a. connection by referring to the **sample code**. // Replace xxx.st1.iotda-device.cn-north-4.myhuaweicloud.com with the actual access address. // To obtain the domain name, log in to the Huawei Cloud IoTDA console. In the navigation pane, choose Overview and click Access Details in the Instance Information area. Select the access address corresponding to port 8883. IoTDevice device = new IoTDevice("ssl://xxx.st1.iotda-device.cnnorth-4.myhuaweicloud.com:8883", "your device id", "your device secret", tmpCAFile); device.getClient().setConnectListener(new MessageSample(device)); if (device.init() != 0) { return; ļ Connect the device to the platform, and use it to subscribe to messages b. delivered by the platform. @Override public void connectComplete(boolean reconnect, String serverURI) { // Subscribe to downstream messages. device.getClient().subscribeTopic("/test/cloudToDevice", new ActionListener() { @Override public void onSuccess(Object context) { System.out.println("subscribeTopic success"); } @Override public void onFailure(Object context, Throwable var2) { System.out.println("subscribeTopic failure"); }, rawMessage -> { System.out.println(" on receive message topic : " + rawMessage.getTopic() + ", payload : " + new String(rawMessage.getPayload(), StandardCharsets.UTF_8)); }, 1); }
 - c. Run the program. The device is displayed as online on the platform.

Figure 2-17 Device list - Device online status

< 👩 usernessage	R Details & Moody ***									
Overview	All Devices Total devices 2 Act	ivated devices 1 • Online	devices 1					Culck Links		
Products Devices All Devices	Device List Batch Registra	Devec Lat Bach Registration Batch Update Batch Detelson Batch Add Devices To Group File Uploads								
Groups Policies	 Q. Search by node ID by default. □ Status ⊕ 	Device Name 🖯	Node ID 😣	Device ID 😔	Resource Space 🖯	Product (e)	Node Type 🛞	(2) (Advanced Search >) (3) Operation		
Upgrades	Online	teril?		Mall/1448/1921/Market/752.1	Deladings_Hillions	Strates	Drach; consolid	View Debug More ~		

3. Specify a topic to report device messages.

device.getClient().publishRawMessage(new RawMessage("/test/deviceToCloud", "hello", 1), new
ActionListener() {
 @Override
 public void onSuccess(Object context) {
 System.out.println("reportDeviceMessage success: ");
 }
 @Override
 public void onFailure(Object context, Throwable var2) {
 System.out.println("reportDeviceMessage fail: " + var2);
 }
});

4. Access the **IoTDA** service page and click **Access Console**. In the navigation pane, choose **IoTDA Instances** and click the target instance card. In the navigation pane, choose **Devices** > **All Devices** and click **View**. On the

Message Trace tab page, check whether the IoT platform receives the message.

Figure 2-18 Message tracing - Viewing results

ITDA Insta	nces / All Devices /	Device Detail	\$											
< 1 100	4075471241262	DeviceSin	Online	0										🖸 Quick Links
Device Inf	Cloud Run	Logs C	loud Delivery	Device Shadow	Message Trace	Device Monitoring	Child Devices	Tags	Groups					
Traced To ensi	messages help you o re data validity and pr	uickly locate an revent the platfo	d identify failure caus orm from occupying b	ees. Learn more oo marry read and write o	ompute and storage reso	urces, the platform can only	race messages for up to	o 10 devices a	t a time for a single user, an	d for no more than three day	s.			
Impi	amentation (Running) Start time:	w 10.2004 10.02.20	End time	c Jul 10, 2	10.00							Stop Trace	Clear Data
Ed	Configuration	Export Data)											
Q	earch by service deta	ils by default.												0
Servi	ce Type		Service Step 🖯		Service Details ()					Recorded O		Message Status 🖯	Operation	

Application-side SDK Receiving Messages

After a device sends messages to the platform through the SDK, you can configure **data forwarding rules** to forward the messages to the message middleware, storage, data analysis, or service applications. This section uses **Java SDK** as an example to describe how to receive messages reported by devices and process services.

Application-side SDK Delivering Messages

Configure the SDK on the application side to deliver messages.



```
//.withRegion(IoTDARegion.CN_NORTH_4)
        // For the standard or enterprise edition, create a region object.
       .withRegion(REGION_CN_NORTH_4).build();
     // Instantiate a request object.
     CreateMessageRequest request = new CreateMessageRequest();
     request.withDeviceId("<YOUR DEVICE_ID>");
     DeviceMessageRequest body = new DeviceMessageRequest();
     body.withPayloadFormat("raw");
     body.withTopicFullName("/test/cloudToDevice");
     body.withMessage("hello");
     request.withBody(body);
     try {
        CreateMessageResponse response = client.createMessage(request);
        System.out.println(response.toString());
     } catch (ConnectionException e) {
       e.printStackTrace();
     } catch (RequestTimeoutException e) {
       e.printStackTrace();
     } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(e.getHttpStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
     }
  }
}
```

 Check the device logs. The message sent by the application to the device is displayed as follows: on receive message topic : /test/cloudToDevice , payload : hello

Issue 1.0 (2024-11-06) Copyright © Huawei Cloud Computing Technologies Co., Ltd.



The IoT platform provides various APIs to reduce the application development difficulty and improve the application development efficiency. This topic uses local debugging (Postman) as examples to describe how to connect an application to IoTDA using HTTPS.

Local Debugging

This section uses Postman to describe how to use IoTDA by calling APIs on the application side.

The procedure is as follows:

Step 1 Enable IoTDA. Access the **IoTDA** service page, and click **Access Console** to subscribe to the service.

Step 2 Create an MQTT product.

Step 3 Configure the environment. Download and install Postman 7.17.0.

Step 4 Call the service. Use Postman to call the API and check the returned result, status code, and error code.

• Step 1 Enable IoTDA.

Currently, IoTDA is available in AP-Bangkok, AP-Singapore, CN-Hong Kong, and AF-Johannesburg regions.

• Step 2 Create a product.

Create a product on IoTDA before calling APIs.

- a. Access the **IoTDA** service page and click **Access Console**. Click the target instance card.
- b. Choose Products in the navigation pane and click Create Product.
- c. Set the parameters to create a product that uses MQTT, and click OK.

Basic Information

Resource Space	The platform automatically allocates the created product to the default resource space. If you want to allocate the product to another resource space, select the resource space from the drop-down list. If a resource space does not exist, create one.
Product Name	Customize the product name. The value can contain up to 64 characters. Only letters, digits, and special characters (_?'#().,&%@!-) are allowed.
Protocol	MQTT is recommended.
Data Type	Select JSON .
Industry	Set this parameter as required.
Sub- industry	Set this parameter as required.
Device Type	Set this parameter as required.
Advanced Se	ettings
Product ID	Set a unique identifier for the product. If this parameter is specified, IoTDA uses the specified product ID. If this parameter is not specified, IoTDA allocates a product ID.
Description	Provide a description for the product. Set this parameter as required.

• Step 3 Configure the environment.

Download and install Postman. For details, see **Installing and Configuring Postman**.

• Step 4 Call the service.

After configuring Postman, debug the following APIs when the application simulator connects to IoTDA using HTTPS:

- Obtaining the Token for an IAM User
- Listing Projects Accessible to an IAM User
- Creating a Product
- Querying a Product
- Creating a Device
- Querying a Device

Advanced Experience

After using Postman to connect a simulated application to the platform, you may understand how the application 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**.