IoT Device Management

FAQs

Issue 01

Date 2019-06-25

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks 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 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 FAQs

Specifications

- Which Access Modes and Protocols Does IoT Device Management Support?

Product Use

- Is Authentication Required from Huawei for Subscribing to IoT Device Management?
- Can Huawei Provide Modules, Hardware, and Application Software?
- How Do I Control Devices with IoT Device Management?

Secondary Development

- FAQs About Product Development
  - Profile Definition
    - How to Develop a Profile
    - Upload Is Unavailable When Uploading a Profile to the Developer Center
    - Incorrect File Format Is Displayed When Uploading a Profile to the Developer Center
    - Message Is Displayed Indicating That the Manufacturer ID and Device Model Exist When Uploading a Profile to the Developer Center
    - How to Select a Data Type When Developing a Profile Online
    - Profile Cannot Be Edited
  - Plug-In Development
    - How to Develop Codecs
    - Failed to Deploy a Plug-in That Is Developed Online
    - What Is the Code Rule for Fields of the string and varstring Types
    - What Is the Code Rule for Fields of the array and variant Types
    - Whether Plug-Ins That Are Developed Online Support Transparent Transmission
    - How to Use messageId During Online Codec Development
    - How to Configure a Command Delivery Response During Online Codec Development

2 Specifications

2.1 Which Access Modes and Protocols Does IoT Device Management Support?

NB-IoT, 2G/3G/4G, and gateway access modes are supported, and the MQTT(S), CoAP(S), and LWM2M protocols are supported.

3 Product Use

3.1 Is Authentication Required from Huawei for Subscribing to IoT Device Management?

Individual or enterprise real-name authentication is required.

3.2 Can Huawei Provide Modules, Hardware, and Application Software?

Huawei provides a device management platform, but not devices and applications. However, multiple partners have completed integration certification with the Huawei IoT platform. You are recommended to seek suitable suppliers in the Huawei cloud market.

3.3 How Do I Control Devices with IoT Device Management?

For details, see **Device Management** in the *User Guide*.

4 Secondary Development

4.1 FAQs About Product Development

4.1.1 Profile Definition

4.1.1.1 How to Develop a Profile

The IoT platform supports online and offline profile development. For details, see **online development guide** and **offline development guide**.

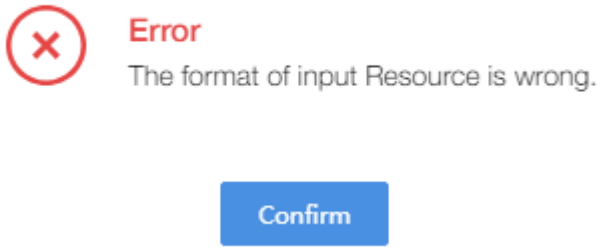
4.1.1.2 Upload Is Unavailable When Uploading a Profile to the Developer Center

Step 1 Check whether the name of the profile file is in the format of **deviceType_manufacturerId_model.zip** and whether the values of **deviceType**, **manufacturerId**, and **model** are the same as those defined in the **devicetype-capability.json** file.

Step 2 Check whether the profile file is compressed to a .zip file. If no, decompress the file, compress it to a .zip file, and upload it again.

----End

4.1.1.3 Incorrect File Format Is Displayed When Uploading a Profile to the Developer Center



Step 1 Check whether the format of each JSON file is correct. You can use the online JSON verification format tool.

Step 2 Check whether the values of **commands** and **properties** in the **servicetype-capability.json** file are in the array format (whether the value is enclosed in the brackets).

Step 3 Check whether the profile package contains hidden files. If yes, delete the files and upload the profile package again. The Mac is used as an example to describe how to view hidden files:

1. Start the **Terminal** program.
2. Enter the following command: **defaults write com.apple.finder AppleShowAllFiles -boolean true ; killall Finder**.
3. Press **Return**. The hidden files and folders are displayed in the **Finder** window.

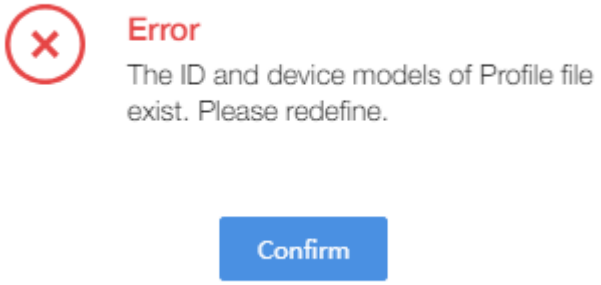
   NOTE

   To restore the hidden settings, run the following command:

   defaults write com.apple.finder AppleShowAllFiles -boolean false ; killall Finder

----End

4.1.1.4 Message Is Displayed Indicating That the Manufacturer ID and Device Model Exist When Uploading a Profile to the Developer Center



When this message is displayed, the profile and plug-in of the same device model and manufacturer ID exist on the IoT platform. You can perform either of the following operations:

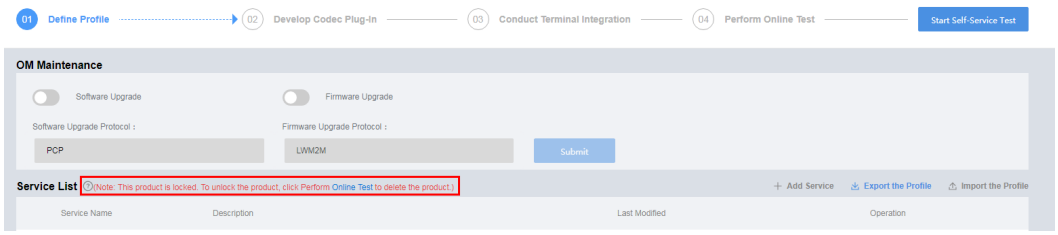
- Delete the plug-in and profile of another application or account (delete the plug-in first), and then import the profile.
- Modify the device model and manufacturer ID, and then import the profile.

4.1.1.5 How to Select a Data Type When Developing a Profile Online

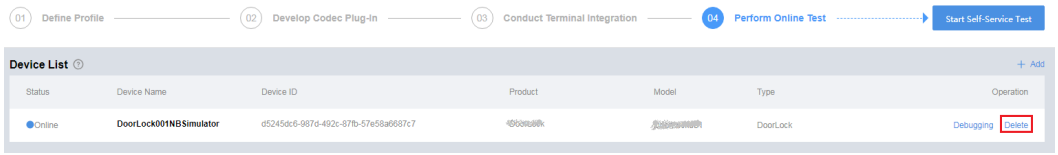
The rules for configuring **Data Type** are as follows:

- **int**: If the reported data is an integer or Boolean value, set the data type to this value. The plug-in can be of the int or array type.
- **decimal**: If the reported data is a decimal, set the data type to this value. The plug-in can be of the string, int, or array type.
- **string**: If the reported data is a string, enumerated value, or Boolean value, set the data type to this value. If enumerated or Boolean values are reported, use commas (,) to separate the values. The plug-in can be of the string or array type.
- **DateTime**: If the reported data is a date, set the data type to this value. The plug-in can be of the string or array type.
- **jsonObject**: If the reported data is in JSON structure, set the data type to this value. The plug-in can be of the string or array type.

4.1.1.6 Profile Cannot Be Edited



If devices using the product file have been registered, the profile cannot be edited. In this case, delete the registered devices, and then edit it.

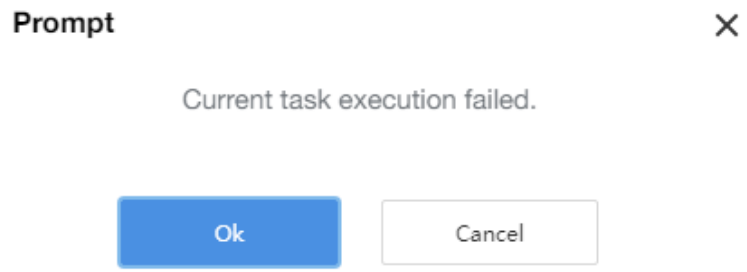


4.1.2 Plug-In Development

4.1.2.1 How to Develop Codecs

The IoT platform supports online and offline codec development. For details, see **online development guide** and **offline development guide**.

4.1.2.2 Failed to Deploy a Plug-in That Is Developed Online



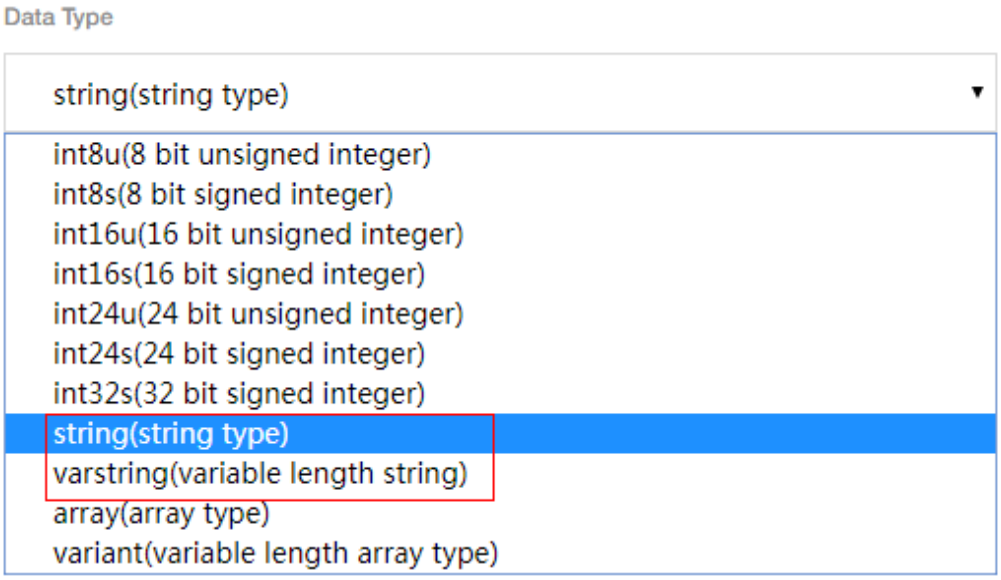
On the plug-in development page, download the codec plug-in.

- If the plug-in fails to be downloaded, the possible causes are as follows:
  - The value of **messageId** in the messages of the same type (for example, messages reported for two types of data) is duplicate or their positions in the messages are different.
  - **Data Type** is not set to **int** for **messageId**.
  - The default value of a field is not a hexadecimal number.
  - A field name contains a Java keyword, for example, **type** or **int**.
- If the plug-in is successfully downloaded but the plug-in fails to be deployed, the network may be faulty. In this case, check the network status.

4.1.2.3 What Is the Code Rule for Fields of the string and varstring Types

If **Data Type** of a field is **string(string type)** or **varstring(variable length string)**, the codec performs encoding and decoding using ACSII. For details on the codec online development, see **Codec for Strings and Variable-Length Strings**.

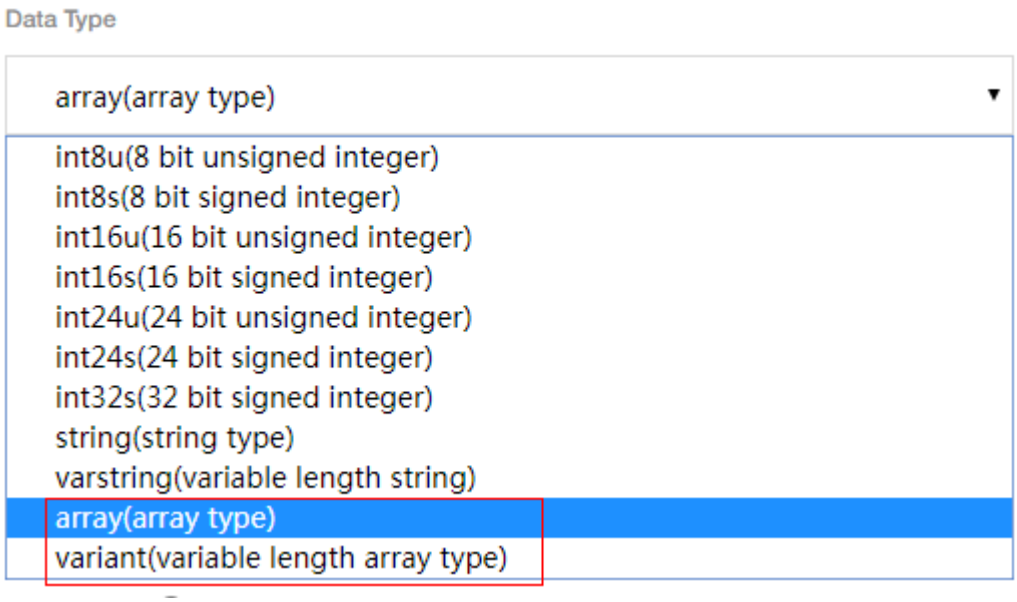
Figure 4-1 String and variable-length string



4.1.2.4 What Is the Code Rule for Fields of the array and variant Types

If **Data Type** is **array(array type)** or **variant(variable length array type)**, the codec performs encoding and decoding using Base64. For details on the codec online development, see **Codec for Arrays and Variable-Length Arrays**.

Figure 4-2 Array and variable-length array



4.1.2.5 Whether Plug-Ins That Are Developed Online Support Transparent Transmission

Plug-ins that are developed online do not support transparent transmission.

4.1.2.6 How to Use messageId During Online Codec Development

When messages of the same type are created, such as two data reporting messages, **messageId** must be configured to distinguish these messages and this parameter in each message must be in the same place on the list. The **messageId** field applies to the following scenarios:

- There are two or more data reporting messages or command delivery messages.
- A command response can be regarded as a type of data reporting message. Therefore, if a command response exists, **messageId** must be added to the data reporting message.
- A data reporting response can be regarded as a type of command delivery message. Therefore, if a data reporting response exists, **messageId** must be added to the command delivery message.

For details on the codec online development, see **Codec for Multiple Data Reporting Messages**.

4.1.2.7 How to Configure a Command Delivery Response During Online Codec Development

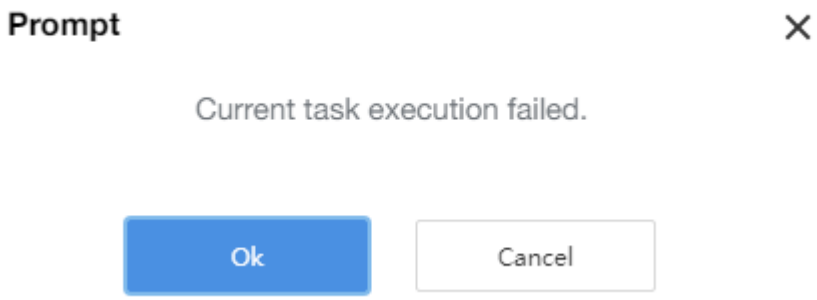
After receiving a command, a device sends an ACK message or a command delivery response. The ACK message indicates that the command has reached the device, and the

command delivery response indicates the command execution result. If the device needs to return a command delivery response after receiving the command, configure the following data:

- The **messageId** field must be configured in both the data reporting message and the command response, and this field in the two messages must be in the same place on the list, so that the codec can distinguish the data reporting message from the command response.
- The **mid** field must be configured in the command delivery message and the command response, and this field in the two messages must be in the same place on the list, so that the codec can associate the command delivery message with the corresponding command response.

For details on the codec online development, see **Codec for Containing Command Execution Results**.

4.1.2.8 Failed to Deploy a Plug-in That Is Developed Offline

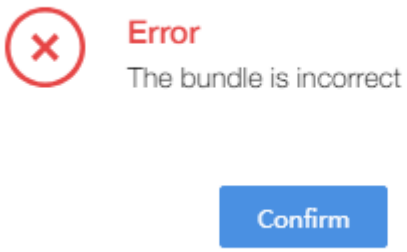


Step 1 Use the **NB-IoT codec plug-in check tool** to check the plug-in.

Step 2 Rectify the fault based on the error code returned by the tool. For details about how to handle the error codes, see the *NB-IoT Codec Plug-in Check Tool Instructions* in the tool package.

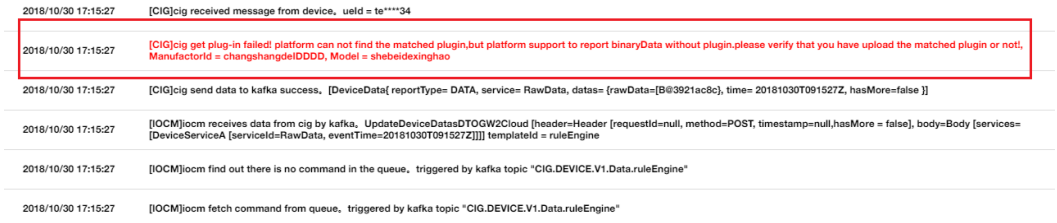
----End

4.1.2.9 Failed to Upload a Plug-in That Is Developed Offline



Plug-ins that are developed offline must be checked by using the codec plug-in check tool before being uploaded to the IoT platform. Use the codec plug-in check tool to check the plug-in package and rectify the fault based on the error message.

4.1.2.10 Plug-In Package Developed Offline Has Been Checked But Cannot Be Found When Uploading to the IoT Platform



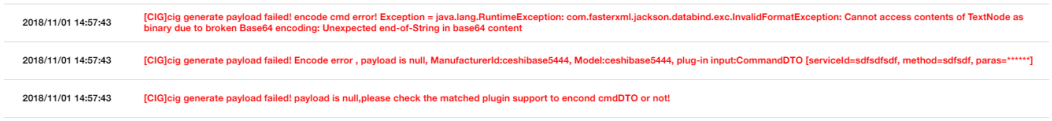
Step 1 Use the decompilation tool to open the .jar package in the **preload** folder and check whether the **OSGI-INF->CodecProvideHandler.xml** file is in the XML format and contains no garbled characters.

Figure 4-3 Example CodecProvideHandler.xml

Step 2 Check whether the paths of **name** and **implementation class** are the same as those of the plug-in code.

----End

4.1.2.11 Codec Exception Occurs After a Plug-In Package Developed Offline That Has Been Checked Is Uploaded to the IoT Platform



Step 1 This fault is caused by the plug-in code. Generally, the dependency is not introduced or the code logic is incorrect. You can rectify the fault based on the Java exception prompt in the log.

Step 2 Print logs at the key code of the offline plug-in (for example, at the entrance and exit of the decode function), and contact IoT platform personnel to obtain logs in the background to locate the fault.

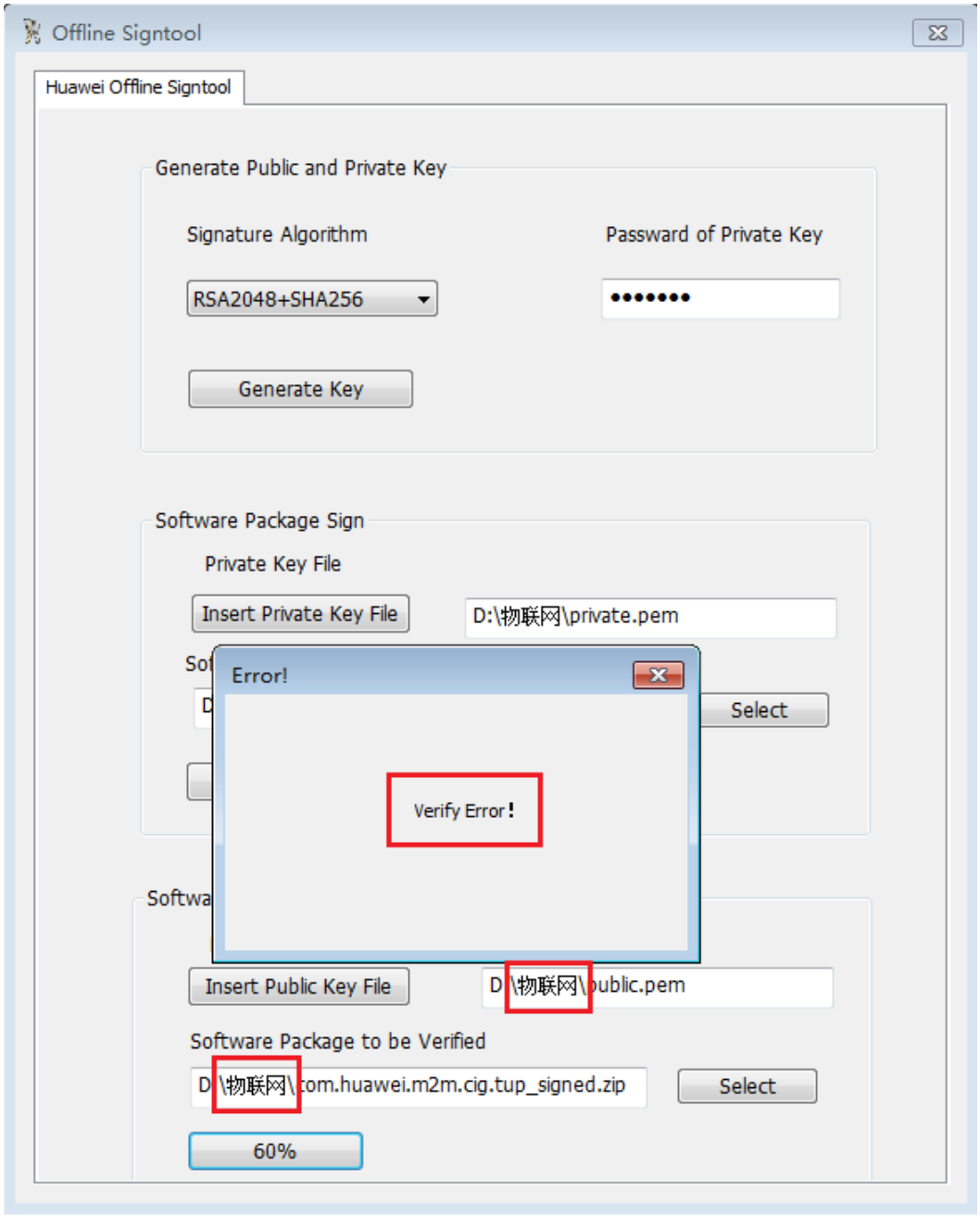
Figure 4-4 Example logs

```
public void activate() {
    logger.info("Codec demo HttpMessageHander activated.");
}

public void deactivate() {
    logger.info("Codec demo HttpMessageHander deactivated.");
}
```

----End

4.1.2.12 Offline Signature Fails



If the path of the offline signature tool or plug-in contains Chinese characters, the verification fails. Save the offline signature tool and plug-in to a path that does not contain Chinese characters.

4.1.3 Device Development

4.1.3.1 Registered Devices Cannot Connect to the IoT Platform

Step 1 Run the **AT+CGATT=1** command on the module to check whether an error is reported. If yes, perform the following operations:

- Confirm the status of the NB-IoT card with the NB-IoT network carrier.
- Confirm the module status with the module manufacturer.

Step 2 If the device is successfully connected to the network after you run the **AT+CGATT=1** command, check whether the IP address and port number of the IoT platform used by the device are correct.

NOTE

The IP address and port number are obtained from the IoT platform service provider. Port 5683 is used for non-encrypted access, and port 5684 is used for encrypted access.

----End

4.1.3.2 Message 513 Is Reported When Devices Are Connected to the IoT Platform

After being powered on, devices initiate a TUP registration flow to the IoT platform. TUP is a Huawei proprietary protocol over CoAP. TUP is similar to LWM2M. HiSilicon chipsets set that the TUP registration cannot exceed 4s. If the TUP registration is not completed within 4 seconds, a 513 message is reported.

When a 513 message is reported, perform the following operations:

Step 1 When a 513 message is reported due to poor network performance, contact the NB-IoT network carrier to check the network status.

Step 2 When service data is sent by running **AT+NMGS**, registration is triggered. If the t/d resources (receiving and sending service data resources) are not received within 4 seconds, an error is returned, but the registration is continued based on the CoAP-layer retransmission. If subscription to the t/d resources is not received after 160 seconds, the IoT platform determines that the registration fails. The 160-second duration is sufficient for device registration. If an error message is returned after 4 seconds, only the data of the first packet is discarded. You are advised to restart the device and run **AT+NMGS** to trigger the registration.

You can run **AT+NMSTATUS?** to query registration status. If +NMSTATUS:MO_DATA_ENABLED is returned, the registration is successful.

----End

4.1.3.3 DTLS Encryption Algorithms Supported by the IoT Platform

The IoT platform supports the security mode of the pre-shared key. The following encryption suites are supported:

- TLS_PSK_WITH_AES_128_CCM_8, as defined in [RFC6655]
- TLS_PSK_WITH_AES_128_CBC_SHA256, as defined in [RFC5487]

4.1.3.4 What Does CTNB Mean

CTNB indicates the APN of the NB-IoT card on the device. Currently, the NB-IoT card does not support customized APNs. The APN must be obtained from the NB-IoT network service provider.

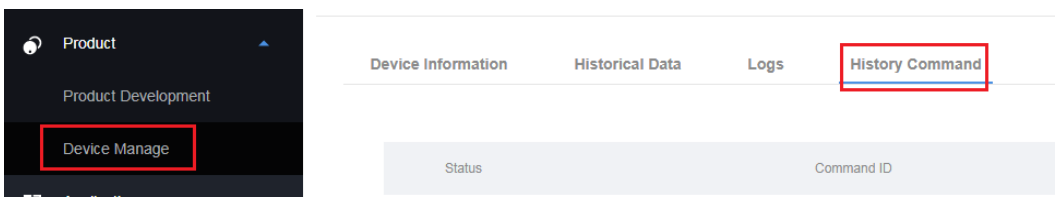
4.1.3.5 How to Select PSM, DRX, and eDRX

NB-IoT supports three power-saving modes: power saving mode (PSM), discontinuous reception (DRX), and extended discontinuous reception (eDRX).

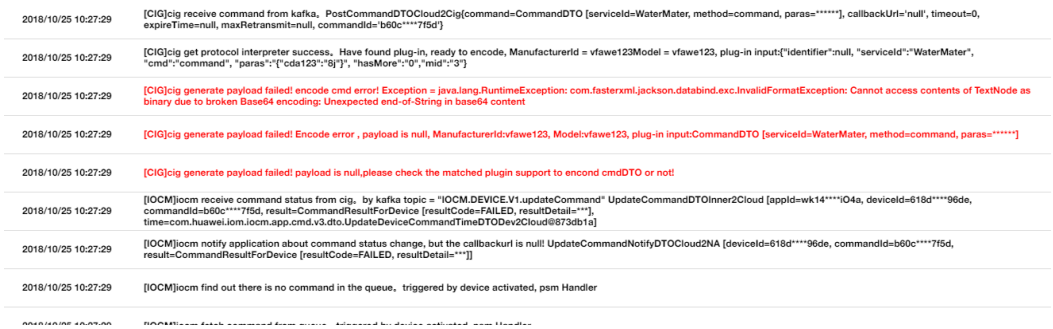
Mode	Description	Command Delivery Mechanism
PSM	In PSM mode, the delay of downstream services is not restricted. Downstream service messages can be sent after a device sends upstream data and enters the connected state. This decreases device power consumption.	**Immediate delivery (expireTime=0)**: The IoT platform delivers commands immediately. If a device is in PSM mode, the device cannot receive the commands. Only devices in the active or idle state can receive the commands. **Cache delivery (expireTime>0)**: The IoT platform caches the received commands. When a device reports data, the IoT platform delivers all cached commands in sequence.
DRX	In DRX mode, the delay of downstream services is strictly restricted. To use this mode, a device must be always online so that messages can be delivered immediately.	**Immediate delivery (expireTime=0)**: The IoT platform delivers received commands immediately without waiting. **Cache delivery (expireTime>0)**: The IoT platform caches received commands and delivers the command in sequence based on the sequence in which the commands are placed in the queue. After the previous command is sent to the device or times out (the timeout interval is fixed), the IoT platform delivers the next command.
eDRX	In eDRX mode, the delay of downstream services is strictly restricted. In this mode, messages can be sent immediately or buffered based on system configuration.	**Immediate delivery (expireTime=0)**: The IoT platform delivers received commands immediately, and devices can receive the commands. **Cache delivery (expireTime>0)**: The IoT platform caches received commands and delivers the command in sequence based on the sequence in which the commands are placed in the queue. After the previous command is sent to the device or times out (the timeout interval is configurable), the IoT platform delivers the next command.

4.1.3.6 Devices Cannot Receive Commands from the IoT Platform

Choose **Device Management** > **History Command** on the Developer Center.

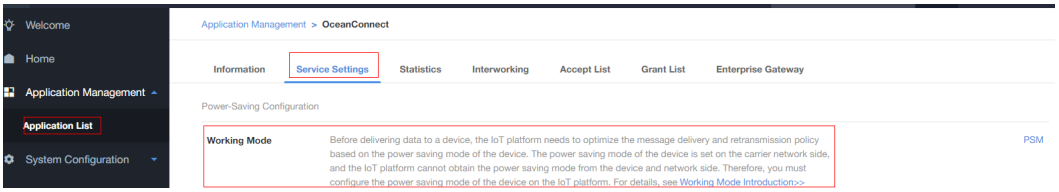


- If the command status is **Failed**, perform the following operations:
  - Check the device logs. If an encode error occurs, the codec plug-in is faulty. In this case, you need to use the codec plug-in check tool to check whether the codec plug-in is correct.



  - Choose **Device Management** to check the device status. If the device is abnormal or offline, connect the device to the network and enable it to send a message to the IoT platform.
- If the command status is **Sent** or **Timeout**, the command has been sent by the IoT platform. Contact the carrier to check the network and contact the module manufacturer to check the device.
- If the command status is **Delivered**, the IoT platform has received the ACK message sent from the module. The communication between the IoT platform and the module is normal. Obtain the module logs for analysis.
- If the command status is **Pending** and the power-saving mode is PSM, this is a pending command. The command can be received only after the device reports data.

  You need to contact the carrier to check the power-saving mode of the device.



4.1.3.7 IoT Platform Does Not Receive Data Reported by Devices

On the Developer Center, choose **Device Management**, select a device, and check whether logs are displayed on the **Logs** page.

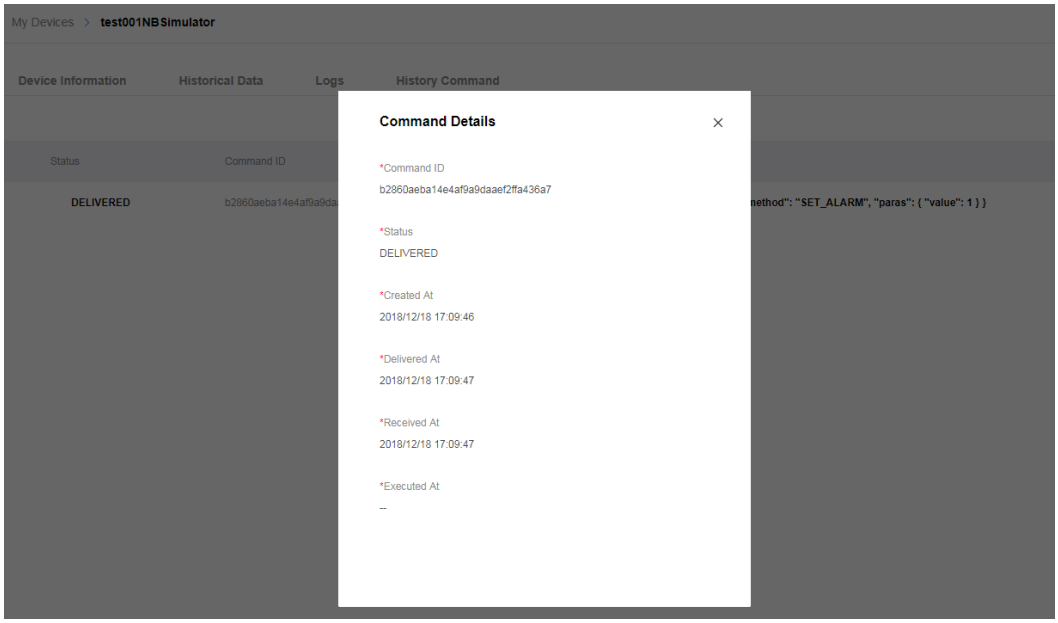
- If the log does not contain the reported data, the message does not reach the IoT platform. In this case, check the network connection and device.
- If the log contains the reported data, check whether a decoding failure occurs. If yes, check whether the codec plug-in is correct.

4.1.3.8 Historical Data Does Not Contain Data Reported By Devices While No Error Occurs

- Log in to the Developer Center, choose **Application** > **Interconnection**. Under **Industry Information**, set **Application Ability** to **Storage Mode**.
- Log in to the management portal, and choose **Application Management** > **Application List**. On the page displayed, select an application, and click the **Information** tab.

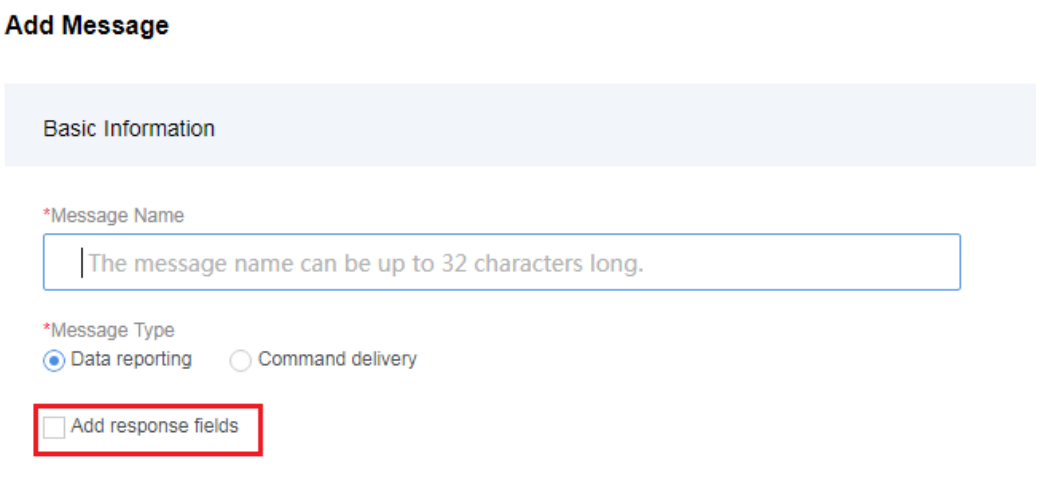
4.1.3.9 Duration for Devices to Receive Commands

On the Developer Center, choose **Device Management** > **History Command** to view the command details. For sent commands, the value is the time the command is sent from the IoT platform. For delivered commands, the value is the time when the IoT platform receives the ACK message from the device. You can calculate the time required for delivering the command to the device. The value is affected by the NB-IoT network status.

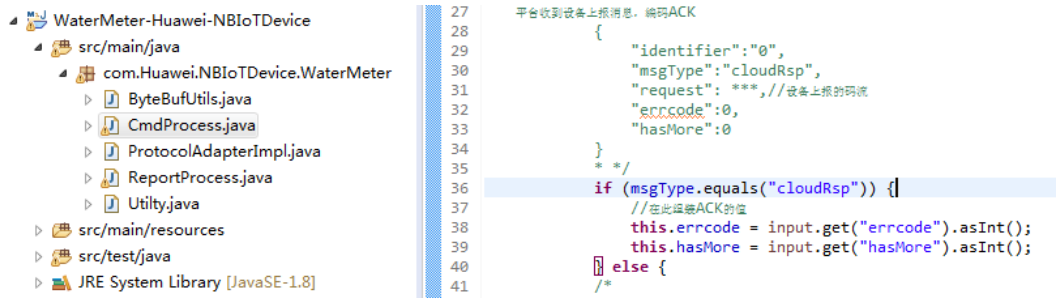


4.1.3.10 Devices Fail to Receive Data Reporting Responses

- If the codec is developed online, **Add response fields** must be selected under **Data reporting**.



- If the codec is developed offline, the cloudRsp logic must be defined in the codec code.



4.1.3.11 Data Reporting Is Successful at One Position But Fails at Another Position

Contact the NB-IoT network carrier to check whether the NB-IoT card has geographical restrictions and the local NB-IoT network status.

4.1.3.12 Command Status Is Not Changed to Successful After a Device Reports the Command Execution Result

On the Developer Center, choose **Device Management** > **Logs**, and enable the log function. After the device reports the command execution result, check whether logs are recorded.

- If no log is recorded, the message does not reach the IoT platform. In this case, check the network and device.
- If logs are recorded and no error message is displayed, the mid field in the codec may be incorrectly configured. For details on how to use mid, see **Codec for Containing Command Execution Results**.

4.1.3.13 Whether the IoT Platform Can Switch Between the Big-Endian and Little-Endian

The IoT platform supports only the Big-Endian.

4.1.3.14 Commands Fail to Be Delivered When a Device in DRX Mode Does Not Report Data in One to Two Days

In DRX mode, if a device does not report data within 25 hours, the device status changes to abnormal. If a device does not report data within 49 hours, the device status changes to offline. Therefore, devices in DRX mode must report heartbeat messages at least once every 25 hours to ensure that they are online.

4.2 FAQs About Application Development

4.2.1 Calling APIs

4.2.1.1 Authentication API of the IoT Platform Is Successfully Called Locally But Fails to Be Called on Application Servers

Step 1 You are advised to use the Demo or SDK provided by the IoT platform to call APIs.

Step 2 Check whether the JDK version is 1.8 or later. (The TLS version is 1.2.) The Linux OS is used as an example to describe how to check the JDK version.

1. Search for the directory where the JDK is installed.

   echo $JAVA_HOME;

   In this example, the directory is **/opt/soft/java**.

2. Open the **bin** folder of the **/opt/soft/java** directory.

   cd /opt/soft/java/bin

3. Check the JDK version.

   ./java -version

Step 3 Check whether the application server is integrated with the IoT platform certificate. The certificate is provided by the IoT platform service provider.

Step 4 Check whether the **url**, **body**, and **header** parameters are correct based on the Authentication API.

----End

4.2.1.2 Error Occurs When an NA Calls an API

View the error codes of the API in the API reference based on the original error code returned by the IoT platform. If an application has encapsulated error codes, you can use the postman to call the same APIs to obtain the original error code and description returned by the IoT platform, and then rectify the fault based on the recommendation.

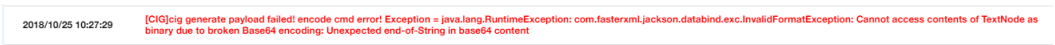
NOTE

The NA may encapsulate errors reported by the IoT platform. The original error codes reported by the IoT platform fail to be obtained if the postman is not used.

For details on how to use Postman to call an IoT platform API, see **Using Postman to Test IoT Platform APIs**.

4.2.1.3 Online Application Simulator Fails to Deliver Commands

Step 1 On the Developer Center, choose **Device Management** > **Logs** to view the logs. Check whether the IoT platform finds the corresponding codec plug-in and encodes the command successfully. If there is an encoding error, a codec plug-in error occurs. In this case, use the codec tool to detect the error and rectify the fault based on the error message.



Step 2 Check the command delivery mode and device work mode. If the command is delivered immediately, the device must work in DRX mode. To obtain the work mode of the device, contact the carrier.

Step 3 Change the command delivery mode to cache delivery (that is, set **expiretime** to a value other than **0** when calling the Creating Device Commands API). Then, check whether data is successfully delivered after a device reports data.

If a cached command is successfully delivered, the failure is caused by the aging of device links. If the device works in DRX mode and a heartbeat message is reported every 25 hours,

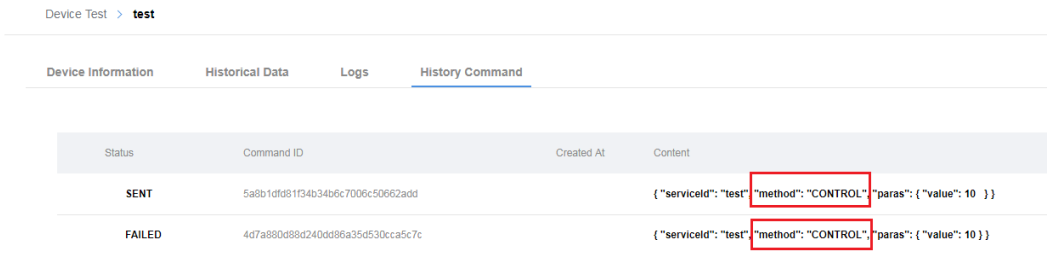
but the device link is still aged, the IoT platform service provider must check the parameter configuration of the TUP package.

----End

4.2.1.4 Command Is Successfully Delivered Using the Simulator But Fails to Be Delivered by Calling the API

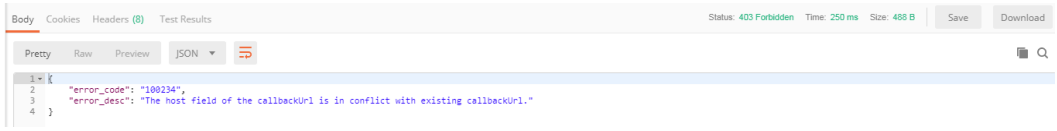
Step 1 Check whether the **url**, **body**, and **header** parameters are correct based on the API reference.

Step 2 Use the simulator and call the API to deliver the same command separately. Compare the JSON message bodies of the two commands.



----End

4.2.1.5 Application Server Fails to Deliver Commands and Receives Error Code 403



Step 1 View the subscription address by choosing **Subscription Test** on the Developer Center or calling the Querying Subscription in Batches API.

NOTE

The IP address and port number of the callback address of each subscription type under the same application must be the same.

Step 2 Check whether the **callbackurl** parameter carried when calling the Creating Device Commands API is the same as the IP address and port number of the subscription callback address. If no, change the values to the same.

----End

4.2.1.6 How Does an Application Server Deliver a Cache Command

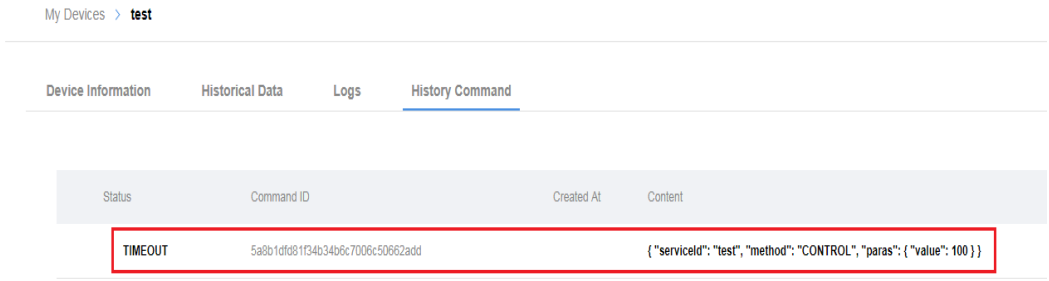
Set **expireTime** to a value greater than **0** when calling the Creating Device Commands API.

expireTime indicates the command expiration time, which is measured in units of seconds. Specifically, the validity period of the command is **expireTime** seconds. The command will not be delivered after the specified time elapses. If this parameter is not specified, the default validity period is 48 hours (86400 seconds x 2).

4.2.1.7 What Does an Expired Command Mean

If the application server delivers a cache command and the device is not connected to the network within the time specified by **expireTime** (or the device in PSM mode does not report data), the command status changes to **Expired**.

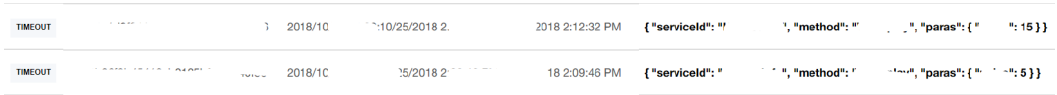
Figure 4-5 Command delivery expired



4.2.1.8 Does the IoT Platform Support Command Resending

The IoT platform supports command resending. After sending a command to a device, if the IoT platform does not receive an ACK message from the device, the IoT platform resends the command after 10s to 15s. You can view the command sending time on the Developer Center. If the IoT platform still does not receive an ACK message from the device, the IoT platform resends the command after 20s to 30s. If the IoT platform still does not receive an ACK message from the device, the IoT platform resends the command after 40s to 60s. If the IoT platform does not receive the ACK message from the device after 80s to 120s, the command status changes to **Timeout**.

4.2.1.9 What Does a Timeout Command Mean



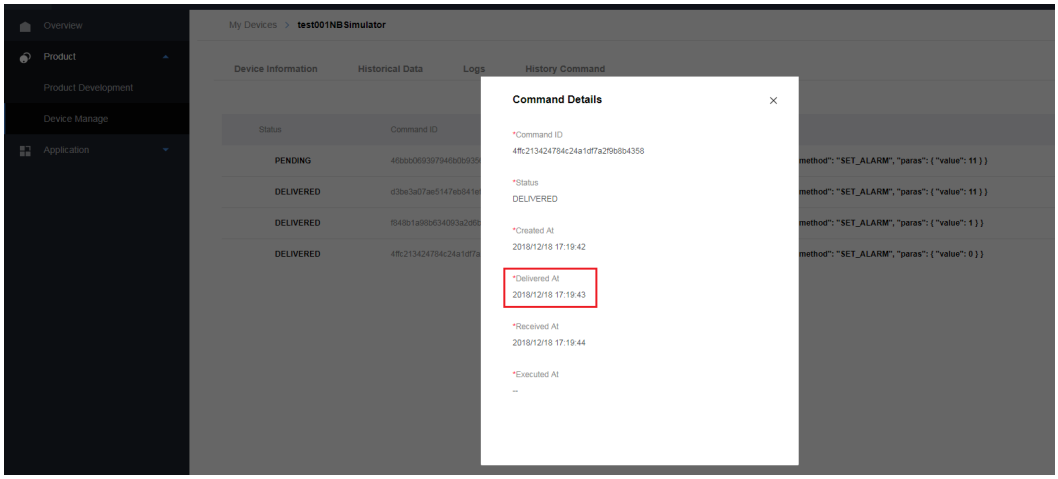
After sending a command to the IoT platform, if the IoT platform does not receive an ACK message from the device within 225s, the command status changes to **Timeout**. To rectify the fault, perform the following operations:

Step 1 Check the work mode of the device. If the work mode is DRX, the device must report a heartbeat message every 25 hours.

To obtain the work mode of the device, contact the carrier.

Step 2 On the Developer Center, choose **Product** > **Device Management** > **History Command** > **Command Details** to check the time when the IoT platform sends the command. Then, contact the network carrier and module manufacturer to check whether an ACK message is returned.

Figure 4-6 Viewing the time when the command is sent



----End

4.2.1.10 Can the IoT Platform Deliver Commands in Batches

Commands can be delivered by calling the Creating a Batch Task API of the IoT platform.

NOTE

Commands delivered in batches are cache commands.

4.2.1.11 Are the APIs of the IoT Platform Called Only Using Java

The APIs of the IoT Platform are standard RESTful APIs that support multiple languages, such as Java, PHP, and Python.

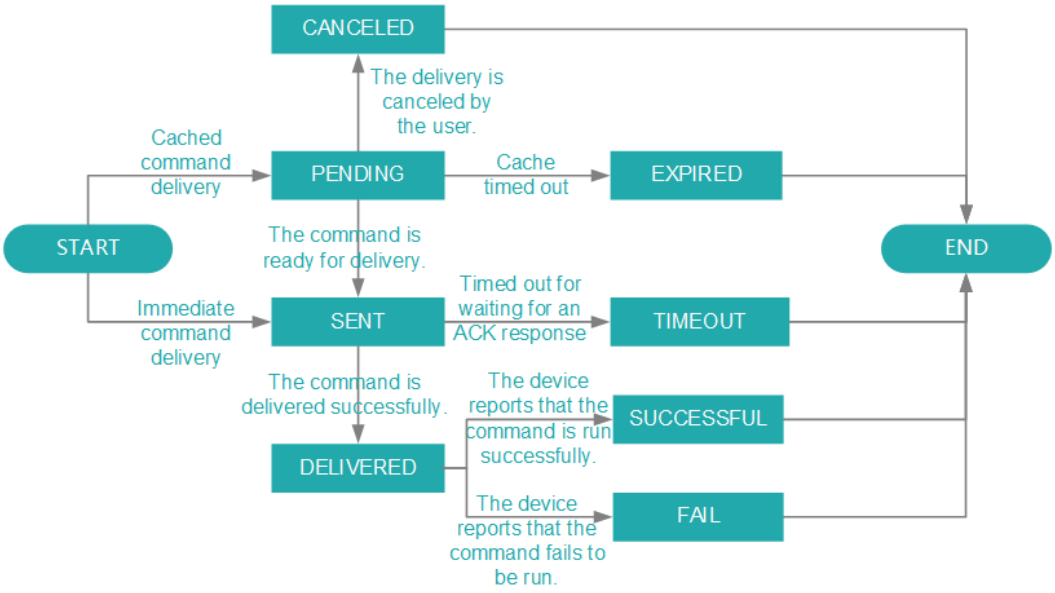
4.2.1.12 What Are the Command States on the IoT Platform

Commands sent by the IoT platform are in the following states:

- **Expired**: The command cache duration has expired on the IoT platform and is not delivered to the device.
- **Success**: The IoT platform has delivered the command to the device and received an execution result from the device.
- **Failed**: No result is displayed after the command is parsed by the codec plug-in, or the execution result contains **ERROR CODE**.
- **Timeout**: The IoT platform fails to receive an ACK message from the device within a specified period.
- **Canceled**: The command has been canceled on the application side.
- **Pending**: The IoT platform has buffered the command and has not delivered it to the device.
- **Sent**: The IoT platform has delivered the command to the device.
- **Delivered**: The IoT platform has delivered the command to the device and received an ACK message from the device.

Figure 4-7 shows the conversion between command states.

Figure 4-7 Relationship between the command status



4.2.1.13 Failed to Call the Calling the Registering Directly Connected Devices API

Rectify the fault based on the error information returned when the API is called. For details, see the Registering Directly Connected Devices API in the API reference.

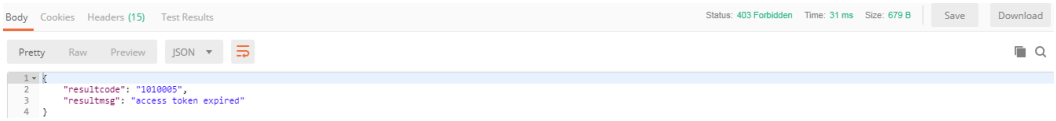
- If a message is displayed indicating that the device has been bound, check whether the device with the same verification code (IMEI or MAC address) exists under your account.

  NOTE

  If no device with the same verification code (IMEI or MAC address) is found under your account, contact the IoT platform service provider to check whether the verification code (IMEI or MAC address) has been used on the IoT platform.

- If a message is displayed indicating that the number of devices reaches the upper limit, contact the IoT platform service provider to increase the maximum number of devices supported by the IoT platform.

4.2.1.14 APIs Fail To Be Called After a Period of Time While Parameters Remain Unchanged

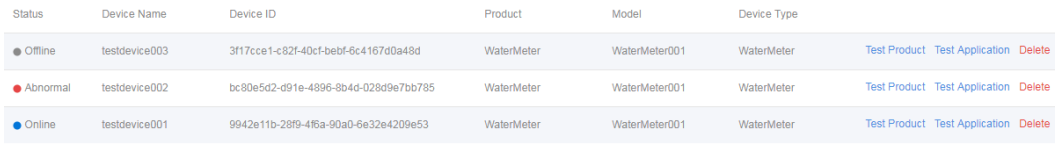


This issue occurs when the accessToken expires. In this case, call the Authentication API to obtain the new accessToken and then call a required API again.

4.2.1.15 Commands Fail to Be Sent After a Period of Time While the Previous Commands Are Successfully Sent

Check the device state on the Developer Center. If the device state is offline or abnormal, enable the device to report data to change the device status to online and then deliver the command.

Figure 4-8 Device states



4.2.1.16 Historical Data Is Not Returned When the Querying Historical Device Data API Is Called

The **pageNo** parameter in the Querying Historical Device Data API indicates the page number to be queried.

- If the value is **0**, the first page is queried.
- If the value is **null**, pagination query is not performed.
- If the value is an integer greater than or equal to **0**, pagination query is performed.

When this issue occurs, set **pageNo** to **0** to query the content on the first page.

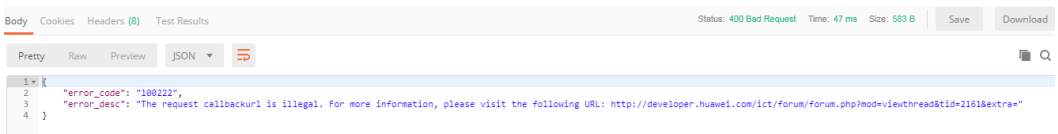
4.2.1.17 Devices Registered with the IoT Platform Are Deleted After a Period of Time

When a device is registered by calling an API, the **timeout** parameter must be configured. After the registration is complete, if the device is not bound to the IoT platform within the time specified by **timeout**, the IoT platform will delete the device.

The value range of **timeout** is 0-2147483647, which is measured in units of seconds. When this parameter is set to **0**, the device is permanently valid and will not be deleted.

4.2.2 Subscription and Push

4.2.2.1 Invalid Callback Address Occurs When Calling the Subscription API



A callback address must contain the public IP address (or domain name), port number, and file path.

- A correct example is as follows: http://IP:PORT/filePath or http://IP:PORT/.
- An incorrect example is as follows: http://IP:PORT or http://IP/filePath.

4.2.2.2 How to Obtain the subscriptionId When Calling the Querying Subscription in Batches API

The IoT platform returns the subscriptionId when the Subscription API is called. To query the subscriptionId, call the Querying Subscription in Batches API.

4.2.2.3 Application Server Fails to Receive Data Pushed by the IoT Platform

Step 1 View historical data on the management portal or Developer Center to check whether device data is reported.

Step 2 Check whether the callback address is correct. You can enter the callback address during application subscription on the Developer Center to check whether the subscription address is verified.

Step 3 Check whether the uploaded certificate is correct (only HTTPS certificates are supported). If not, create a certificate by following the instructions provided in **How Do I Export the HTTPS Push Certificate** and upload it to the IoT platform.

----End

4.2.2.4 How Do I Export the HTTPS Push Certificate

Step 1 Start a browser, and type the callback address in the address box. Internet Explorer is used as an example.

Step 2 Check the certificate. The methods for checking a self-signed certificate and non-self-signed certificate are different.

- If the callback address uses a self-signed certificate, the message "There is a problem with this website's security certificate" is displayed. Choose **Continue to this website (not recommended).** > **Certificate Error** > **View certificates**.

Figure 4-9 Self-signed certificate callback address prompt

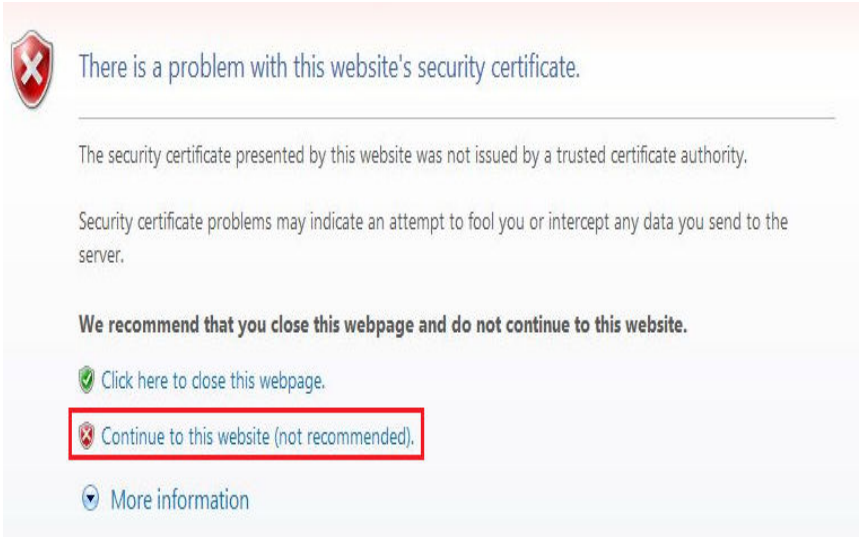
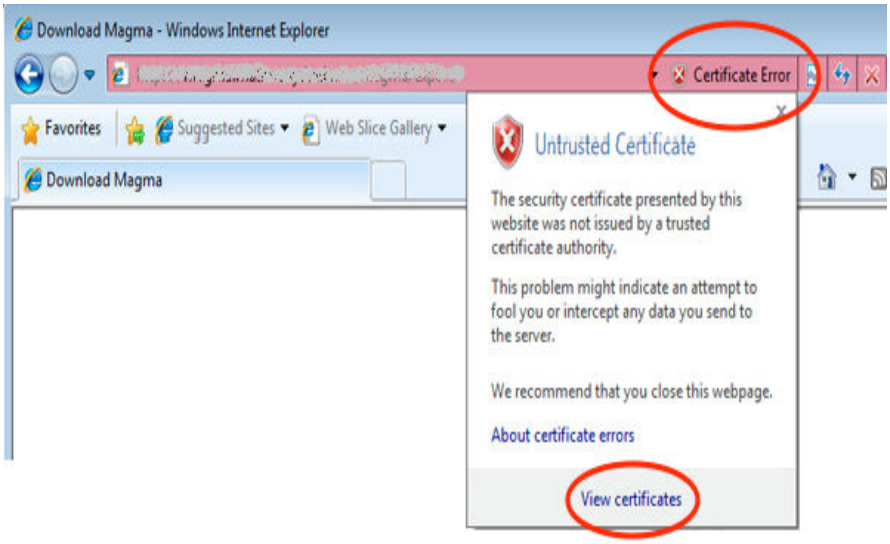
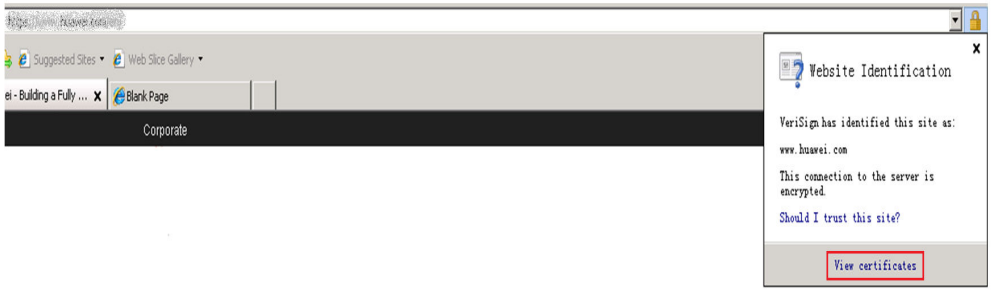


Figure 4-10 Checking the self-signed certificate



- If the callback address is not a self-signed certificate, choose **Security Report** > **View certificates**.

Figure 4-11 Checking the non-self-signed certificate



Step 3 Check the certificate level on the **Certification Path** tab page. The current certificate level is the last level of the certificate.

Figure 4-12 Certificate path

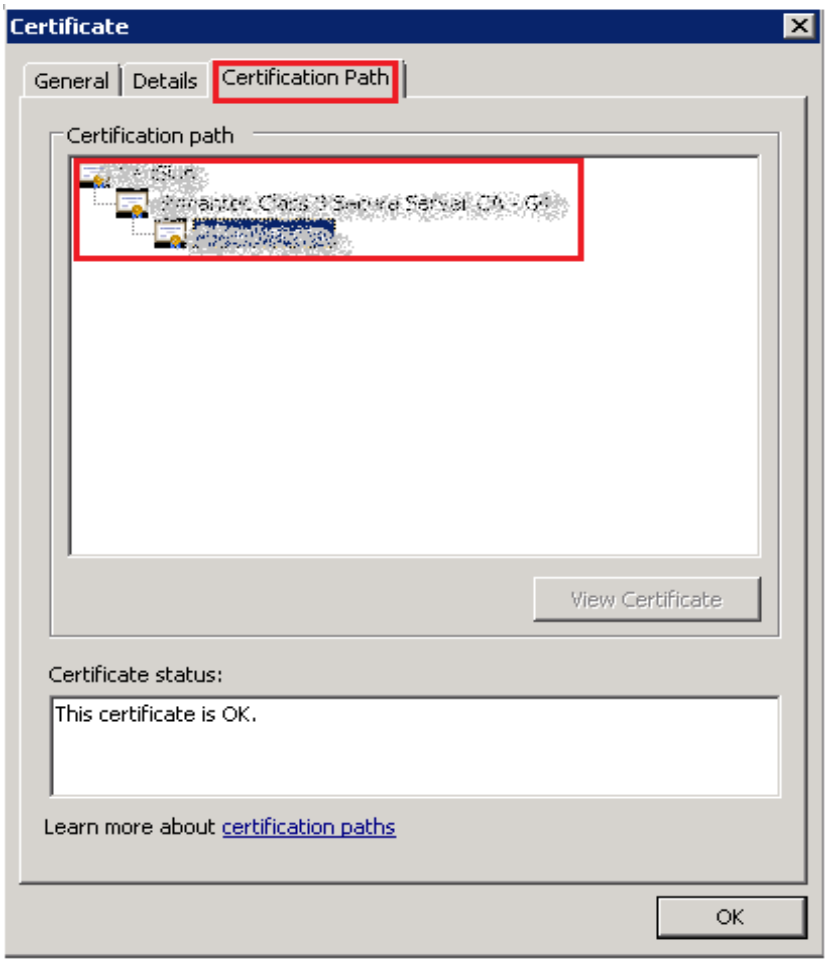
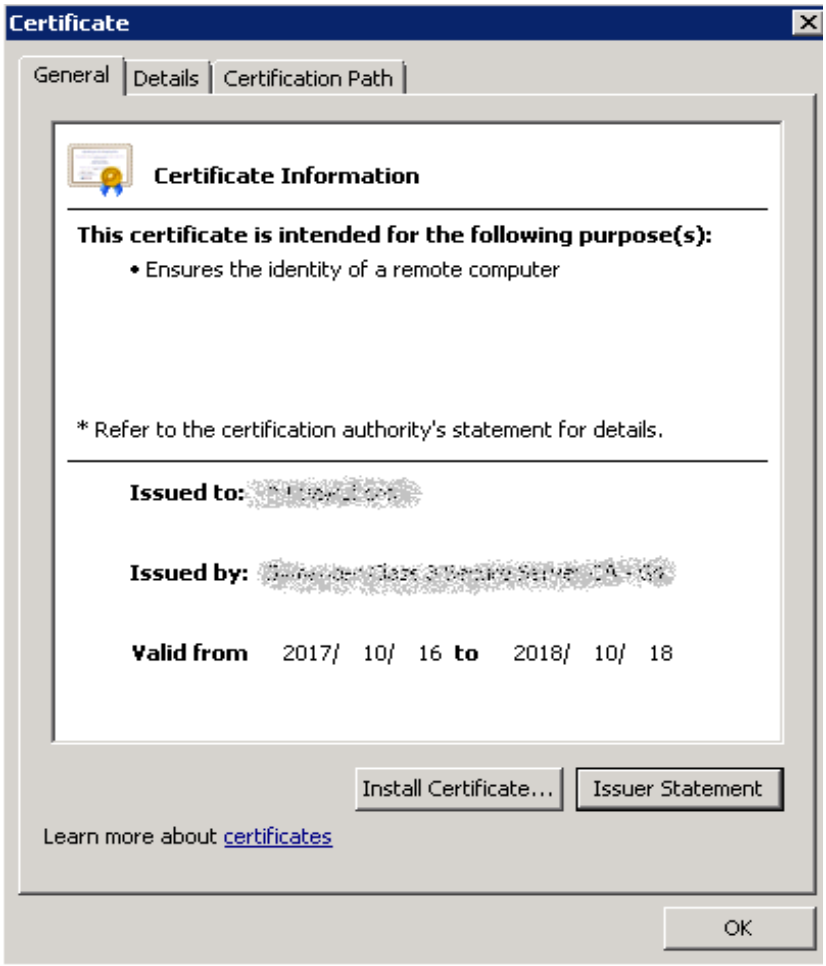


Figure 4-13 Common certificate information



Step 4 On the **Details** tab page, click **Copy to File**, and select **Base-64 encoded X.509 (.CER)** to export the certificate of the current level based on the certificate export wizard.

Figure 4-14 Detailed certificate information

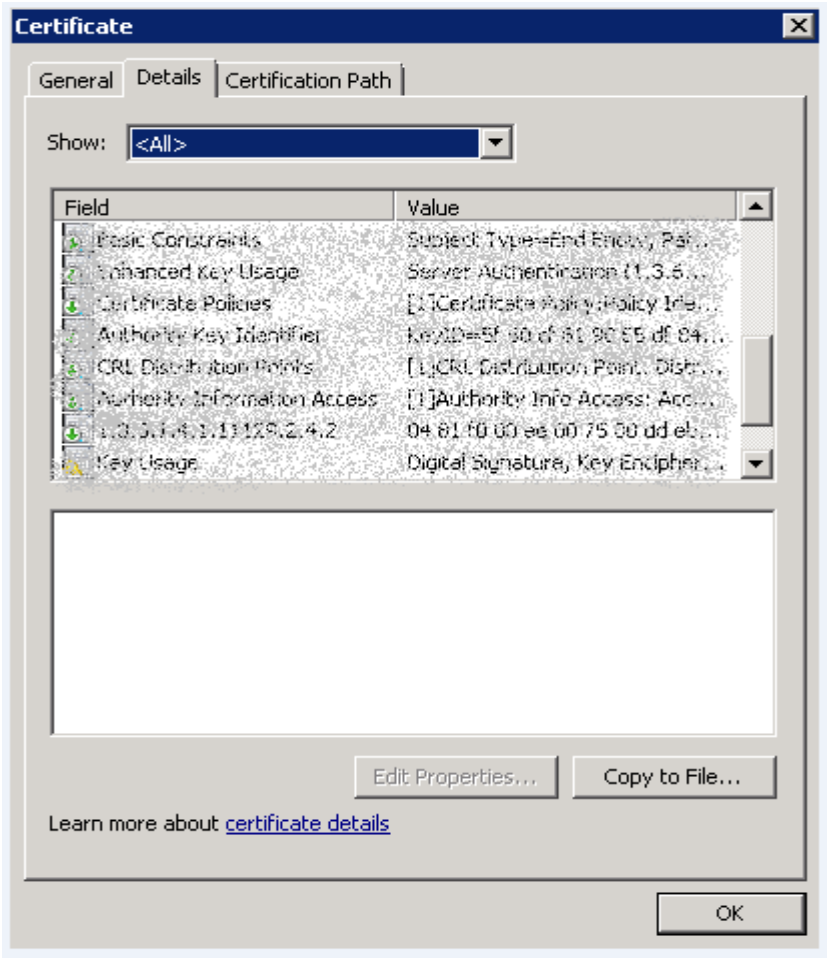
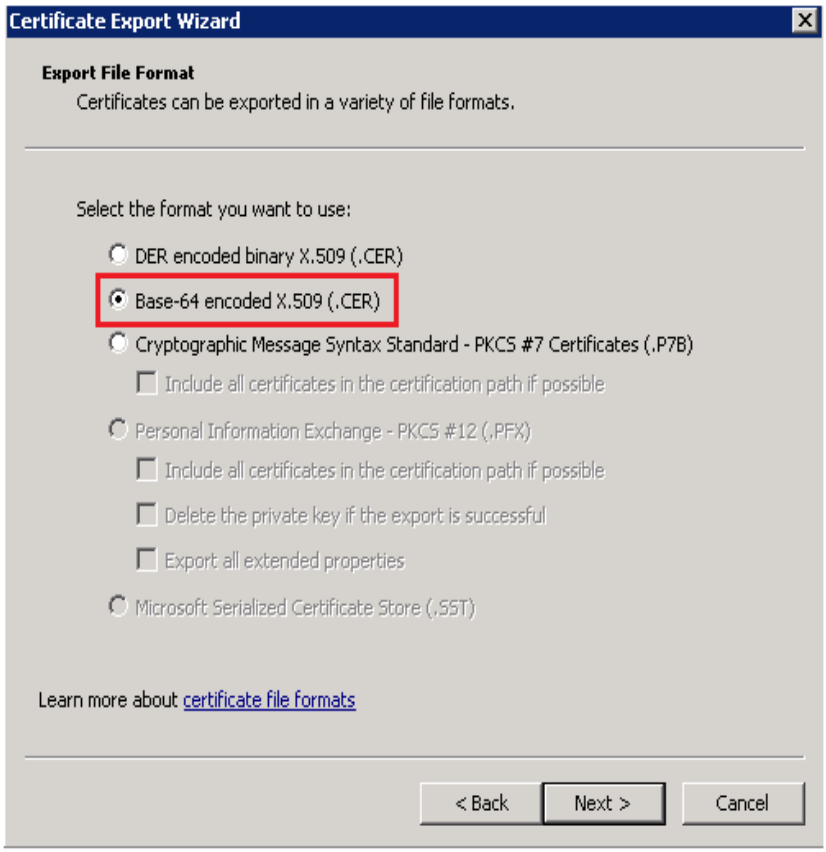


Figure 4-15 Selecting the certificate export format



Step 5 Double-click the upper-level certificate. In the displayed dialog box, choose **Details** > **Copy to File** > **Base-64 encoded X.509 (.CER)** to export the upper-layer certificate by following instructions provided by the certificate export wizard.

Step 6 Repeat **Step 5** until all levels of certificates are exported.

Step 7 Use the text editor to combine all the exported certificates into a file.

NOTE

No newline character exists between the combined files.

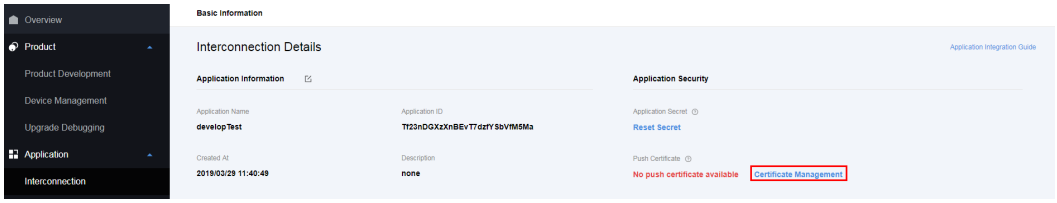
Figure 4-16 Combining certificates

```
pLkJkLQYWBSHuxOizGdwCjw1mAT5G9+443fNDsgN3BAAAAFc8uXxRAAABAMARjBE
AiA+PkSvZOvIBmxaBLXLceHH1NqW+Mcrz+xtXNmJOl2ElQIgfVeG4xeuWrb7bpoU
smR+LStIG1TPCwimn3JRE3we/fYwDQYJKoZIhvcNAQELBQADggEBADjrCz8a7cax
h7vpyuUFZ/fiKBHE7VLqfppgf3XYNBoqh21qM6gTGzdiSeZj+vx+KOUn38f080Rg
N2aEkag3n03cufIXR8Yn8haXcusz5PONSlMQnN5rZBwpZ8obItiO8KGOh5lgHQ+s
SloX/j8nDDCQgrNkcG2A78nUT+VxGGENxnPmqajP/O2h/kg02qjcnPoj6Elmm/At
5dWWANX374yS7c0fgLZZ1mfZoIqooaRxsSJl5RzyRNU3Bzv5CZCJCGYFqC3RS28Q
vTCjde7TMsAQiWkZ97IKlUMXdbHManm7K85aWcG4Wg8isr9d2GPUZYgcUSc8KfWY
aP5MzoeU6ug=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFODCCBCCgAwIBAgIQUT+5dDhwtzRAQY0wkwaZ/zANBgkqhkiG9w0BAQsFADCB
yjELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL
ExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTowOAYDVQQLEzEoYykgMjAwNiBWZXJp
U2lnbiwgSW5jLiAtIEZvciBhdXRob3JpemVkIHVzZSBvbmx5MUUwQwYDVQQDEzxW
ZXJpU2lnbiBDbGFzcyAzIFB1YmxpYyBQcmltYXJ5IENlcnRpZmljYXRpb24gQXV0
```

Step 8 Change the file name suffix to **pem**.

Step 9 On the Developer Center, choose **Interconnection** > **Application Security** > **Push Certificate** to upload the certificate to the IoT platform.

Figure 4-17 Uploading the certificate



----End

4.2.2.5 Difference Between deviceDatachanged and deviceDataschanged

Device change data is subscribed to over the Device Data Change and Batch Device Data Change APIs, and the format for encapsulating data is different.

For example: A water meter has two service types: Battery and Connectivity. The data of the two services is reported each time.

- If deviceDatachanged is subscribed to, the IoT platform pushes the data to the application server twice, Battery service data for the first time, and Connectivity service data for the second time. Examples are as follows:

```
{ "notifyType": "deviceDataChanged", "deviceId": "70a8d7cd-5ecd-4bda-a87c-
afc16bd31bda", "gatewayId": "70a8d7cd-5ecd-4bda-a87c-afc16bd31bda",
"requestId": null, "service": { "serviceId": "battery", "serviceType":
"battery", "data": {"batteryLevel": 66}, "eventTime": "20170211T034003Z" }}
{"notifyType": "deviceDataChanged","deviceId": "70a8d7cd-5ecd-4bda-a87c-
afc16bd31bda", "gatewayId": "70a8d7cd-5ecd-4bda-a87c-
afc16bd31bda","requestId": null, "service": { "serviceId": "Connectivity",
"serviceType": "Connectivity", "data": { "signalStrength": 72, "cellId":
4022250974, "tac": 61374, "mnc": 91, "mcc": 235 }, "eventTime":
"20170211T092317Z" }}
```

- If deviceDataschanged is subscribed to, the IoT platform encapsulates the data of the two services together and sends the data to the application server.

```
{ "notifyType": "deviceDatasChanged", "requestId": null, "deviceId":
"70a8d7cd-5ecd-4bda-a87c-afc16bd31bda",<br> "gatewayId": "70a8d7cd-5ecd-4bda-
```

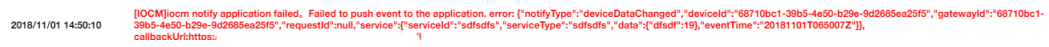


```
a87c-afc16bd31bda",<br> "services": [{ "serviceId": "battery", "serviceType":
"battery", "data": { "batteryLevel": 66 }, "eventTime": "20170211T034003Z" },
{ "serviceId": "Connectivity", "serviceType": "Connectivity","data":
{"signalStrength": 72, "cellId": 4022250974, "tac": 61374, "mnc": 91, "mcc":
235 }, "eventTime": "20170211T034003Z" } ]}
```

4.2.2.6 How Does an Application Server Obtain the IMEI of Devices

After the bindDevice notification is subscribed to, the IoT platform pushes the IMEI and deviceId to the application server when the device is connected to the network. For details, see the API reference.

4.2.2.7 Application Server Receives Data But an Error Is Displayed on the IoT Platform



After receiving data, the application server must return a 200 message to the IoT platform.

4.2.2.8 Push Messages May Fail to Be Sent to Application Servers

Step 1 Check whether the application server returns a 200 message to the IoT platform indicating that a push message is received. If no, configure it to return a 200 message.

Step 2 Check whether the network status is stable, for example, run the **ping** command to check the packet loss rate.

Step 3 Capture packets on the application server. If all push data can be obtained, check the service processing mechanism of the application server.

----End

4.2.2.9 How Does an Application Servers Obtain The Address Used by the IoT Platform to Push Messages

Obtain the information from the IoT platform support engineers.

4.2.2.10 Does the IoT Platform Supports Re-push

The IoT platform supports the re-push mechanism.

- If a message fails to be pushed, the IoT platform places the message in the retransmission queue and pushes it again.
- If the IoT platform fails to push the message for ten consecutive times, the callback address is added to the blacklist. After 3 minutes, the IoT platform tries again. After an application receives a push message, it returns a 200 message. If the message is successfully pushed, the IoT platform removes the callback address from the blacklist.

4.2.2.11 How Does an Application Server Receive a Command Status Change Notification

Configure the **callbackUrl** parameter when an application server calls the Creating Device Commands API. (The IP address and port number of this parameter must be the same as those of the subscription callback address.) When the command status changes, the IoT platform pushes a message to the address.

4.2.2.12 Application Servers That Have Subscribed to Confirmation Notifications and Command Response Notifications Fail to Receive Push Messages

The message confirmation and the command response notification are not applicable to the NB-IoT networks (CoAP and LWM2M). On NB-IoT networks, to enable application servers to receive response command notifications, configure the **callbackUrl** parameter when calling the Creating Device Commands API.

4.2.2.13 Does the IoT Platform Support Only HTTPS Callback Addresses

The IoT platform supports both HTTP and HTTPS. To configure the protocol, log in to the management portal, and choose **System Management** > **Application Management** > **Application List**. On the page displayed, select an application, click the **Information** tab, and set **Push Protocol** under **Message Push**.

4.2.2.14 Can the IoT Platform Push Data Reported by Different Devices Under the Same Application to Two Servers

The IoT platform cannot push data reported by different devices under the same application to two servers. Only one IP address and one port can be used to push data of the same application.

4.2.2.15 Can a Subscription Address Be a Domain Name

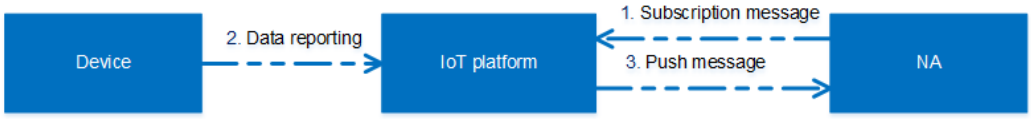
A subscription address can be an IP addresses or a domain name.

4.2.2.16 Can a Callback Address Be Changed

A callback address can be changed. To change the IP address and port number of the callback address, call the Deleting Subscriptions in Batches API to delete the previous callback address and then subscribe to it again.

4.2.2.17 How Do I Obtain the Callback URL When Calling the Subscription API

The following assumes that an NA subscribes to device data changes. The process of subscription and push is as follows:



1. The NA calls the subscription API of the IoT platform to subscribe to device data changes (the request carries the callback URL and notification type). The IoT platform stores the callback URL and the notification type in the subscription list.
2. The device reports data to the IoT platform.
3. Based on the callback URL stored during the subscription, the IoT platform automatically pushes the device data to the NA.

What Is a Callback URL?

The callback URL is the RESTful API address defined by the NA for external access. When the IoT platform pushes message to the NA, it calls the RESTful API of the NA to send data to the NA.

NOTE

The request method of the callback URL must be POST.

How Do I Obtain a Callback URL?

The callback URL consists of the communication mode, access address of the NA, and URL of the RESTful API. An example value is **https://server:port/url**.

HTTPS is recommended to secure the communication between the NA and IoT platform. When HTTPS is used for communication, the IoT platform must load a certificate and export the HTTPS push certificate. For details, see **How Do I Export the HTTPS Push Certificate**.

The access address of the NA varies according to the network where the NA is located.

- If the NA is deployed on the public network, the access address of the NA is the public IP address of the NA:port (or domain name:port).
- If the NA is deployed on the LAN, you must configure network penetration on the NA to generate the public network access address of the NA. For details, obtain the configuration procedure of the network penetration tool from the Internet.

NOTE

The callback URL can be the same or different for subscription of different notification types. It can be self-defined according to service requirements.

4.3 FAQs About Software/Firmware Upgrade

4.3.1 What Is Software/Firmware Upgrade

Software upgrade refers to the upgrade of the MCU, and the firmware upgrade refers to the upgrade of the module firmware.

To upgrade the software or firmware, upload the software or firmware package to the IoT platform. Devices obtain the software or firmware package from the IoT platform for remote upgrade.

4.3.2 Can IoT Platform Download Software/Firmware Packages From Third-party Servers

The IoT platform cannot download software/firmware packages from third-party servers. Currently, software/firmware packages must be uploaded to the IoT platform by choosing **Device Manage** > **Repository** > **Firmware** or **Device Manage** > **Repository** > **Software** on the management portal.

4.3.3 Can the Target Version Be Earlier Than the Source Version?

Software/Firmware can be rolled back from a later version to an earlier version. The IoT platform checks whether the target version is the same as the source version. If they are consistent, the message "The current version is the same as the target version" is displayed, and the IoT platform does not initiate a rollback. If they are inconsistent, the IoT platform starts a rollback.

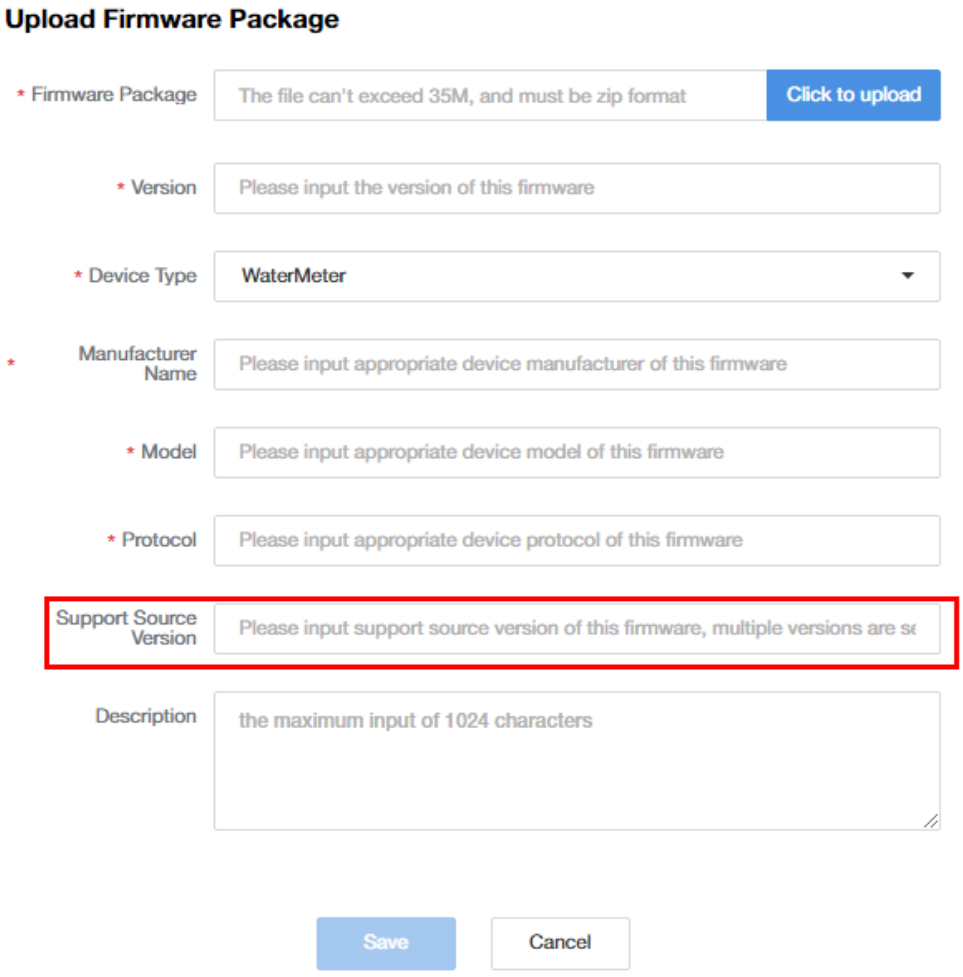
4.3.4 How Do I Obtain Software/Firmware Packages and Their Version Numbers

Obtain the software package and its version number from the device manufacturer. Obtain the firmware package and its version from the module manufacturer.

4.3.5 Software/Firmware Upgrade Task Is Ended Immediately After Being Created

Choose **Task Detail** > **Execute Detail**, and check the error information. For details, see **What Are Common Software/Firmware Upgrade Errors**.

4.3.6 What Is the Source Version Required for Uploading a Firmware Package on the Management Portal



An upgraded is allowed only when the current version of the device is included in the source version. To obtain the current version, choose **Device Manage** > **Device** > **All Devices** > **Device Details** > **Information** on the management portal.

4.3.7 When Reporting Data, Devices Receive the Software/Firmware Version Query Command from the IoT Platform

This issue can be resolved in either of the following ways:

- If the upgrade function is required, devices report the version number when receiving the version query command.
- If the upgrade function is not required, disable the software/firmware upgrade function by choosing **Device Manage** > **Product Model** > **Detail** > **Maintenance Capability Configuration** on the management portal. When the upgrade function is required, enable it as required.

4.3.8 Are Services Interrupted During the Software/Firmware Upgrade

Services are interrupted during the software/firmware upgrade. Services cannot be processed during module or chipset upgrade.

4.3.9 What Are Common Software/Firmware Upgrade Errors

During the software/firmware upgrade, you can view the error description in the task list on the **Execute Detail** tab. Table 4-1 describes common error codes.

Table 4-1 Error codes

Failure Cause	Description	Recommendation
Device Abnormal is not online	The device is not online.	Check whether the device is online.
Task Conflict	A task conflict occurs.	Check whether a software/firmware upgrade is being performed for the current device.
The task failed to start, unable to find protocol service based on device information	Failed to start the task.	The basic information such as the device manufacturer ID, device type, device model, and device protocol does not match those in the profile.
Waiting for the device online timeout	Waiting for the device to go online times out.	Check whether the device is connected to the network.
Waiting for report cellId timeout	Waiting for the device to report cellId times out.	View the module log to check whether the device has reported the cellId.
Waiting for report device firmware version timeout	Waiting for the device to report the firmware version times out.	View the module log to check whether the device has reported the cellId.

Failure Cause	Description	Recommendation
Wait for the device to report upgrade result timeout	Waiting for the device to report the upgrade result times out.	View the module log to check whether the device has reported the upgrade result.
Updating timeout and query device version for check timeout	Waiting for the upgrade result times out and waiting for the device version times out.	View the module log to check whether the device has reported the upgrade result and device version.
Waiting for device downloaded package timeout	Waiting for the device to download the firmware package times out.	View the module log to check whether the device has downloaded the firmware package.
Integrity check failure for new downloaded package	The integrity check of the downloaded firmware package fails.	View the module log to check whether the firmware package downloaded by the device is complete.
Unsupported package type	The firmware package type is not supported.	View the module log to check whether the device status and firmware package provided by the manufacturer are correct.
Not enough storage for the new firmware package	The storage space is insufficient for the firmware package.	Check the storage space on the device.
Out of memory during downloading process	The memory is insufficient during the download process.	Check the device memory.
Connection lost during downloading process	The connection is interrupted during the download.	Check the device connection status.
Invalid URI	The URI is unavailable.	Check whether the download address of the firmware package on the device is correct.
Firmware update failed	The firmware fails to be updated.	View the module log to check the device.

4.3.10 Retry When Some Devices in the Group Failed to Be Upgraded

If the entire task fails, only the failed subtasks are retried. The devices that have been successfully upgraded will not be upgraded again.

4.3.11 Is Resumable Transmission Supported When the IoT Platform Sends Software/Firmware Packages to Devices

Resumable transmission is supported when the IoT platform sends software/firmware packages to devices. Devices must record the fragment location of the software/firmware packet when the transmission is interrupted, so that software/firmware packet fragments that are not transmitted can be sent again after the transmission is restored.

4.4 Others

4.4.1 Does the IoT Platform Support Device Registration in Batches

Currently, northbound APIs are not available. Devices can be registered in batches on the management portal.

Step 1 Click **Device Manage** on the upper navigation bar.

Step 2 In the navigation tree on the left, choose **Device** > **Registration** > **Batch Registration**.

Step 3 Click **Create** in the upper right corner. In the dialog box displayed, set the parameters, and click **Submit**.

----End

4.4.2 Does the IoT Platform Perform Flow Control on Applications and Devices

Yes. The IoT platform performs flow control for each application and device.

- Application side: The flow control threshold for an application is 100 messages per minute.
- Device side: The flow control threshold for a single device is 1200 messages per 20 minutes. The flow control threshold for all devices of an application is 120 messages per second.

4.4.3 How Does an Application Server Obtain Data Reported by Devices to the IoT Platform

An application server obtains data reported by devices to the IoT platform using either of the following ways:

- The application server calls the Querying Historical Device Data API to obtain data reported by devices from the IoT platform.
- The application server calls the Subscribing to Service Data of the IoT Platform API so that the IoT platform pushes data to the application server when devices report data.

4.4.4 Is the Maximum Number of Applications and Devices Restricted on the IoT Platform

The maximum number of applications and devices is restricted on the IoT platform. For details about the specifications, see **Restrictions**.

4.4.5 Why Is an Online Device Changed to an Abnormal or Offline State After a Period of Time

If a device does not report data within 25 hours, its status is changed to abnormal. If a device does not report data within 49 hours, its status is changed to offline.

4.4.6 Are Command Delivery and Data Reporting Successful After a Device Becomes Abnormal or Offline

After a device reports data, its status is changed to online. If a device is abnormal or offline, the IoT platform fails to send commands to the device.

4.4.7 How Long Does a Device Receive an Immediately-Delivered Command in eDRX Mode

In eDRX mode, when the IoT platform sends a command to a device, the command is cached on the core network for a period of time. The command is sent to the device only after the core network pages the device. You can obtain the cache duration from the carrier.

4.4.8 How Long Is Data Stored on the IoT Platform

Data is stored on the IoT platform for 7 days by default.

4.4.9 Can Multiple Devices Use One IMEI

The IoT platform does not allow multiple devices to use one IMEI. One IMEI corresponds to only one device.

