IoT Hub

FAQs

Product Documentation

Contents

FAQs

General

Last updated：2021-08-19 17:53:06

What features and services does IoT Hub provide?

IoT Hub provides features such as device management, device shadow, messaging, firmware update, and connection with Tencent Cloud services. As a message channel, it can be flexibly accessed by devices. Plus, it connects a wide variety of Tencent Cloud components to create full-stack services for message acquisition, storage, and computation.

What are the use limits of IoT Hub?

For more information on the use limits of IoT Hub, please see Use Limits.

What are the differences between the message queue and rule engine forwarding in IoT Hub?

Both message queue and rule engine forwarding can forward messages to the specified Tencent Cloud components, but they differ in the dimensions of forwarding, message filtering, and message format requirements.

- Message queue: it forwards messages in the product dimension. As long as a message queue is configured for a product, the messages of all devices under the product will be forwarded to the message queue unconditionally. There are no message filters or requirements for the message format.
- Rule engine forwarding: currently, messages should be forwarded by topic, and filters can be configured based on topic to specify which topic's messages can be forwarded. You can also set filters for message fields to specify that only the messages containing fields that meet certain conditions can be forwarded. For more information, please see Overview.

Does IoT Hub provide mobile application development?

No. Currently, IoT Hub focuses on feature components at the IoT platform layer but does not provide application capabilities. You need to build an application server by yourself and develop the corresponding mobile application or directly use IoT Explorer, which provides the application development capabilities.

Device Connection and Reporting

Last updated：2021-08-19 17:53:06

What are the causes of failures in device connection to IoT Hub?

There are many causes of device connection failures; for example, the connection between the device and the cloud network is unavailable, device authentication fails, or the connection times out due to poor Wi-Fi signal. You can troubleshoot accordingly based on the type of the error log in the SDK during device connection. Generally, you can troubleshoot in the following steps:

1. Check the connection between the local network of the device and IoT Hub first; for example, for MQTT connections, you can check the network connectivity in the following steps:
   - ping iotcloud-mqtt.gz.tencentdevices.com This is to check whether the server is reachable.
   - telnet iotcloud-mqtt.gz.tencentdevices.com 8883 (for TLS) or 1883 (for non-TLS protocols) This is to check the port connectivity.
   - If the execution results of the above commands are normal, you may also need to check the local firewall policy.
2. In a Wi-Fi environment, if connection times out due to poor signal or environmental interference, you can modify the timeout settings in the variable parameters of the SDK. The following is the default configuration in C-SDK code **qcloud_iot_export_variables.h**:

   ```
   /* default MQTT/CoAP timeout value when connect/pub/sub (unit: ms) */
   #define QCLOUD_IOT_MQTT_COMMAND_TIMEOUT (5 * 1000)
   ```
3. If the network connection is normal, the device connection failure may be caused by a device authentication error. Therefore, you need to check the following settings:
   - Check whether the used device information parameters are correct. Some common errors are extra spaces in the device information or key, mismatch between the device information and key information, or mismatch between the certificate filename and the filename written in the code.
   - For certificate-based connections, if the local time is incorrect, TLS connection will also fail. You need to install the NTP client locally to calibrate the time.
4. When you use the SDK for Android for MQTT connection, the message "Incorrect username or password" is prompted.
   If the device parameters (`ProductId` , `DeviceName` , and `DeviceSecret`) are all correctly configured, you can check whether the system time of the device is correct; for example, you can run `adb shell date` to view the system time of an Android device.

Why does a device keep going online and offline?

The IoT access layer has the logic of exclusive device login. If the same device ID is logged in from another place, the existing login will be kicked offline by the new login. Therefore, if the device keeps going online and offline, you can check whether there are different users or threads performing login operations by using the same device ID.

How do I get device status information notifications when a device's offline status changes?

You can configure device status change notifications in the message queue of the product information. Then, device status change notifications will be pushed to the corresponding message queue.

Why does a device fail to receive or send messages? How do I fix it?

Generally, there are the following causes:

1. The topic for sending messages does not exist or has no publishing permission, or the topic for receiving messages does not exist or has no subscribing permission. You need to create an IoT Hub topic in the console first, set its permissions, and then view the device permission list in the console to confirm that the corresponding topic has the message sending or receiving permission.
2. The topic is incorrectly written. Please make sure that the topic written in the code have no extra or missing characters such as spaces, "/", or omitted letters.
3. The connection is interrupted or network communication fails. MQTT messages are transferred over TCP. If the network fails, or the router is disconnected, the connection exception will only be detected after a long period of time due to the TCP retransmission mechanism.
4. When you use the device C-SDK for MQTT topic subscribing or QoS 1 message publishing, if the topic does not exist or does not have the required permission, or the network fails and connection times out, the SDK will prompt that a `NACK` or `TIMEOUT` event is received in the `sample` event callback function for you to troubleshoot the problem.

Will a device automatically reconnect after disconnection?

When you use the device SDK to establish an MQTT connection, if automatic reconnection is enabled (which is enabled by default) in the initialization parameters, the device will automatically reconnect. The `Yield` function in the SDK will determine the network connection status based on whether the message sending/receiving and heartbeat packet behavior are normal. If the connection is interrupted, automatic reconnection will be performed. In addition, to avoid frequent reconnection when the network fails, the SDK reconnection interval changes exponentially from the minimum value; that is, if reconnection fails again, the interval will be doubled. If the reconnection interval reaches the maximum value but reconnection still fails, a reconnection timeout error will be returned.
If the connection is manually interrupted in ways such as actively calling the `Destroy` function, automatic reconnection will not be performed.
The default setting of the maximum reconnection interval is in **qcloud_iot_export_variables.h**:

```
/* MAX MQTT reconnect interval (unit: ms) */
#define MAX_RECONNECT_WAIT_INTERVAL (60 * 1000)
```

What should I do if a compilation error occurs during integration of the SDK for Android into a project?

If an error occurs during compilation with remote dependencies, it is probably because that the remote library has not been updated in time. You can modify the dependency method to local dependency in the `gradle` file:

- compile project(':iot_core')
- compile project(':iot_service')

As embedded devices have limited resources, how do I reduce the C-SDK RAM usage and library size?

We recommend you do the following:

1. Disable unnecessary features first; for example, you can set unnecessary feature options to `n` and `BUILD_TYPE` to `release` in `make.setting` .
2. Check the memory usage of system functions at the HAL layer. For example, in some systems, the `getaddrinfo` system function assigns a large amount of memory for IPv6, and if the SDK only uses IPv4, you can consider optimizing the memory assignment operation by `getaddrinfo` , which reduces the RAM usage.
3. Among all authentication methods for device connection, TLS certificate requires the most storage and RAM resources and is the most secure. TLS key uses fewer resources while ensuring the security. NOTLS key uses the fewest resources and does not require the TLS library, but it is the least secure, as its data is transferred in plaintext, which may be stolen or tampered with. You need to choose an appropriate method based on your device resources.
4. When you use the TLS library, you can tailor the encryption algorithm and key exchange algorithm based on the use case; for example, you can customize the feature macros in `config.h` in the `mbedtls` library.

What is the heartbeat packet mechanism of the device C-SDK for MQTT connection?

MQTT uses persistent TCP connections and needs the heartbeat packet mechanism to ensure that the connection is active. The device C-SDK follows the keep-alive mechanism in the MQTT specifications.
qcloud_iot_export_variables.h contains the default setting of the heartbeat packet sending cycle:

```
/* default MQTT keep alive interval (unit: ms) */
#define QCLOUD_IOT_MQTT_KEEP_ALIVE_INTERNAL (240 * 1000)
```

In a heartbeat sending cycle, if the device does not successfully send the MQTT control messages (including `SUB` , `UNSUB` , and `QoS 1 PUB` messages) and receive the corresponding ACK messages, it will send `MQTT PINGREQ` to the cloud and wait for the cloud to return a `PINGRESP` message. If the device does not receive the `PINGRESP` message in a certain period of time, it will deem the connection interrupted and perform automatic reconnection.

How is MQTT QoS supported by the device C-SDK?

Currently, IoT Hub supports MQTT QoS 0 and QoS 1 but not QoS 2. For a QoS 0 message, after the `Publish` function is called and returns a success, the device will use the TCP/IP protocol stack to ensure the message delivery, and the SDK will not implement subsequent processing. However, for a QoS 1 message, the SDK will maintain a message status queue, further track and give a response based on the MQTT PUBACK message, and inform you of whether the QoS 1 message has been successfully delivered or failed due to timeout in the corresponding event callback. Then, you can determine whether to send the message again.

What does the `Yield` function in the device C-SDK do?

The `Yield` function is used to perform tasks such as reading MQTT messages, processing messages, timeout requests, and heartbeat packets, and managing reconnection status in the current thread context. It is an important step for devices to perform IoT communications over MQTT. In a single-thread single-task scenario, this function must be called and executed in your logic code loop. In a multi-thread multi-task scenario, you can use an independent thread task to execute this function and set a certain thread priority to prevent the thread from being suspended for a long time. For directions on how to use this function, please see the corresponding code sample.

Does the device C-SDK support multi-thread?

Yes. To use MQTT APIs in a multi-thread environment, you need to pay attention to the following. For more code samples, please see `samples/mqtt/multi_thread_mqtt_sample.c` .

1. You cannot use multi-thread to call `IOT_MQTT_Yield` , `IOT_MQTT_Construct` , or `IOT_MQTT_Destroy` .
2. You can use multi-thread to call `IOT_MQTT_Publish` , `IOT_MQTT_Subscribe` , and `IOT_MQTT_Unsubscribe` .
3. As the function to read and process MQTT messages and connection status, `IOT_MQTT_Yield` should have a certain execution time to prevent it from being suspended or preempted for a long time.

Does the device C-SDK support remote diagnosis?

Starting from v2.3.1, the device log reporting feature is available in the device C-SDK, which can report the device operation logs to the cloud over HTTP and display the logs in the console for you to remotely diagnose and monitor the device status. As log reporting uses an independent communication channel, remote diagnosis can be performed even when the network communication is normal but the MQTT connection is abnormal.

Rule Engine

Last updated：2021-08-19 17:53:06

What is the rule engine and what does it do?

The rule engine is a backend module that can process the messages reported by devices and forward them to other Tencent Cloud components. It can filter messages by topic or message content and extract the specified fields into new messages for forwarding to Tencent Cloud components responsible for tasks such as message storage and computation.

What are the requirements for the formats of the messages to be forwarded?

Currently, messages forwarded by the rule engine can be in JSON or binary format. JSON messages can be filtered, while binary messages can only be passed through for forwarding.

What are the formats of messages forwarded by the rule engine to other Tencent Cloud services in the console?

After a device reports a payload message, it will be encapsulated in JSON format in the console before being forwarded to other Tencent Cloud services by the rule engine. The `Payload` field in the encapsulated message indicates the original payload message reported by the device, and the console will use different methods to process it based on the forwarding scenario:

- If the message is forwarded to CMQ or CKafka, the encapsulated `Payload` field will be Base64-encoded and needs to be Base64-decoded when the correct data is extracted.
- If the message is forwarded to a third-party service (HTTP forward), the original payload message reported by the device will be checked. If it is in JSON format, it will be passed through; if it is in binary format, it will be Base64-encoded.

The rule engine was configured in the console to forward messages to other Tencent Cloud services, but the forwarding did not work. What should I do?

You can check the cloud logs of message forwarding in the IoT Hub console to confirm the forwarding situation. Common causes of message forwarding failures include:

- The format of the message body does not match the data format defined during product creation.
- The topic of the message is written incorrectly and does not match the topic configured in the rule.
- The forwarding information entered in the rule is incorrect and causes the rule engine to fail to forward.

Console

Last updated：2021-08-31 12:20:53

How do I view the online status of a device in the console?

Click the product name of a device in the product list to enter the product management page. Then, click **Device List**, find the corresponding device name, and you can see the online status of the device in the **Status** column.

How do I view device logs to determine whether messages are successfully sent or forwarded?

Click the product name of a device in the product list to enter the product management page. Then, click **Cloud Log** to enter the log query page and filter the time period and device name to find the logs of the device. The **Cloud Log** page provides logs for all the key nodes in the message link.

How do I grant topic permissions to a device in the console?

Click the product name of a device in the product list to enter the product management page. Then, click **Permission List** to enter the permission management page. Click **Add Topic Permission** to grant the publishing permission to the corresponding topic. If the device also needs to receive messages from the topic, select **Publish and Subscribe** as the permission.