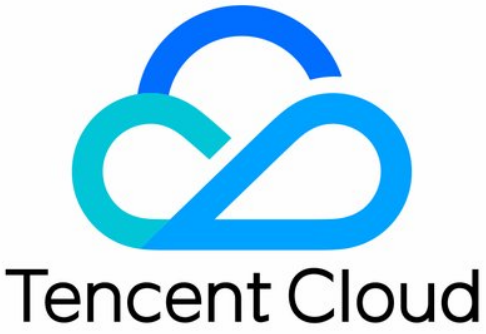
Internet of Things Hub FAQs



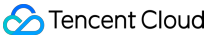
©2013–2025 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

The complete copyright of this document, including all text, data, images, and other content, is solely and exclusively owned by Tencent Cloud Computing (Beijing) Co., Ltd. ("Tencent Cloud"); Without prior explicit written permission from Tencent Cloud, no entity shall reproduce, modify, use, plagiarize, or disseminate the entire or partial content of this document in any form. Such actions constitute an infringement of Tencent Cloud's copyright, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Trademark Notice



This trademark and its related service trademarks are owned by Tencent Cloud Computing (Beijing) Co., Ltd. and its affiliated companies("Tencent Cloud"). The trademarks of third parties mentioned in this document are the property of their respective owners under the applicable laws. Without the written permission of Tencent Cloud and the relevant trademark rights owners, no entity shall use, reproduce, modify, disseminate, or copy the trademarks as mentioned above in any way. Any such actions will constitute an infringement of Tencent Cloud's and the relevant owners' trademark rights, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

©2013–2025 Tencent Cloud. All rights reserved.

FAQs
General

Last updated: 2025-03-19 15:05:33

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 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 Tencent Cloud IoT Explorer, which provides the application development capabilities.

©2013-2025 Tencent Cloud. All rights reserved.

Device Connection and Reporting

Last updated: 2025-03-19 15:05:53

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 wireless network environment, if the connection times out due to signal quality and environmental interference issues, 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 using the Android SDK for MQTT connection, the message "Incorrect username or password" is displayed.

©2013-2025 Tencent Cloud. All rights reserved.

If the device parameters (ProductId, DeviceName, DeviceSecret) are configured correctly, check the system time of the test device, such as using adb shell date to check the system time of the 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 can I get device status information notifications when the device's online or 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 written incorrectly. Ensure that the Topic in the code does not contain redundant characters or omissions, such as an extra space or '/', or a missing letter in a word.
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 using the Device side C-SDK for MQTT to subscribe to topics or publish messages at QoS1 level, if there is a Topic that does not exist, incorrect permissions, or network failure timeout, the SDK will indicate receipt of NACK or TIMEOUT events in the sample's callback function for investigation and locating the issue.

Will a device automatically reconnect after disconnection?

When establishing MQTT connections using the Device side SDK, if the initialization parameter enabled automatic reconnection (enabled by default), then automatic reconnection will be

©2013-2025 Tencent Cloud. All rights reserved.

performed. In the SDK's Yield function, the network connection status will be determined based on the behavior of packet transmission and heartbeat packet. If a connection disconnection occurs, it will automatically reconnect. To avoid frequent reconnections in case of network failure, the SDK's reconnection interval is dynamically adjusted, starting from the minimum value. If reconnection fails, the interval doubles. If the maximum value is reached and reconnection still fails, a connection timeout error will be returned.
If a user manually disconnects, such as by actively calling the Destroy function, automatic reconnection will not be performed.
In **qcloud_iot_export_variables.h**, the default maximum setting for reconnection interval is:

```
/* 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 a compilation error occurs using remote dependencies, it may be due to the remote Library not being updated in a timely manner. You can change the dependency method to local Library 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?

Here are some suggestions:

1. First, you can turn off unnecessary features. For example, set unused feature options to 'n' in make.setting and set BUILD_TYPE to release.
2. Check the memory usage of the HAL system call functions. For instance, on some systems, the getaddrinfo function allocates a lot of memory for IPv6. If the SDK only uses IPv4, consider optimizing the memory allocation operation in getaddrinfo to save RAM.
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 using the TLS library, you can crop the required encryption and key exchange algorithms based on the usage scenario. For example, the mbedtls library allows

©2013-2025 Tencent Cloud. All rights reserved.

customization of the feature macros in its config.h file.

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

MQTT uses a TCP persistent connection, requires a heartbeat mechanism to ensure the connection is active. The device side C-SDK follows the MQTT specification's Keep Alive mechanism. In **qcloud_iot_export_variables.h**, there is a default setting for the heartbeat sending interval:

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

Within a heartbeat sending cycle, if the device C-SDK does not successfully send MQTT control messages (including SUB/UNSUB/QoS1 PUB messages and receive the corresponding ACK), it will send an MQTT PINGREQ to the cloud and wait for the cloud to reply with a PINGRESP message. If the PINGRESP message is not received within a certain period, the device will consider the connection to be disconnected and will perform automatic reconnection.

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

Currently, the Internet of Things Hub platform supports MQTT QoS0 and QoS1 but does not support QoS2. For QoS0 messages, after the Publish function call returns success on the device side, the TCP/IP protocol stack ensures message delivery. The SDK does not perform further processing. For QoS1 messages, the SDK maintains a message status queue, tracks feedback based on MQTT PUBACK messages, and notifies the user through the corresponding event callback whether the QoS1 message was successfully delivered or timed out, allowing the user to decide whether to resend the message.

What is the purpose of the device C-SDK's Yield function?

The purpose of the Yield function is to perform tasks such as MQTT message reading, message processing, timeout requests, heartbeat packets, and reconnection state management within the current thread context. This is an important step for the device to carry out MQTT Internet of Things Hub operations. In a single-thread single-task scenario, the function needs to be called and executed within the user's logical code loop. In a multi-thread multi-task scenario, a separate thread task can be used to execute the function, and a certain thread priority should be set to avoid the thread being suspended for a long time. For specific usage, refer to the corresponding sample code.

©2013-2025 Tencent Cloud. All rights reserved.

Does the device C-SDK support multi-thread?

The device C-SDK supports multi-threading. To use MQTT interfaces in a multi-threaded environment, you need to pay attention to the following precautions. For detailed code examples, refer to `samples/mqtt/multi_thread_mqtt_sample.c`

1. Do not call IOT_MQTT_Yield, IOT_MQTT_Construct, and IOT_MQTT_Destroy in multiple threads.
2. You can call IOT_MQTT_Publish, IOT_MQTT_Subscribe, and IOT_MQTT_Unsubscribe in multiple threads.
3. As IOT_MQTT_Yield is a function for reading and processing MQTT messages and managing connection states, it should ensure certain execution time to avoid being suspended for a long time or preempted.

Does the device C-SDK support remote diagnosis?

Starting from version v2.3.1, the device C-SDK adds the device log reporting feature, which can report the device operation logs to the cloud over HTTP and display the logs in the console, enabling users to remotely diagnose and monitor 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.

©2013-2025 Tencent Cloud. All rights reserved.

Rule Engine

Last updated: 2025-03-19 15:06:07

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?

When using the rule engine to forward messages to other cloud products, the payload message reported by the device is JSON-encoded by the console. The payload field in the encoded message represents the payload message reported by the device. The console will handle it differently based on the forwarding scenario:

- When forwarding to CMQ/Ckafka, the payload field will be Base64-encoded. To extract the correct data, this part needs to be Base64-decoded.
- 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.

©2013-2025 Tencent Cloud. All rights reserved.

- The forwarding information entered in the rule is incorrect and causes the rule engine to fail to forward.

©2013–2025 Tencent Cloud. All rights reserved.

Console

Last updated: 2025-03-19 15:06:21

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

In the Product List, select the target product to enter the Product Management Page. Then click **Device List** to find the corresponding Device Name. In the "Status" column, you can view the device's Online Status.

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

In the Product List, select the target product to enter the product management page. Then click **Cloud Logs** to enter the log query page. By filtering the time period and device name, you can find the logs of the corresponding device. Cloud Logs provides logs for all key nodes in the message link.

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

In the Product List, select the target product to enter the product management page. Then click **Topic Management**, click **Add Custom Topic** to add publish permissions for the corresponding topic. If the device needs to receive messages from the topic as well, you can choose **Publish and Subscribe** for operational permissions.

©2013-2025 Tencent Cloud. All rights reserved.

How to migrate the message queue to the Rule Engine

Last updated: 2025-03-19 15:06:33

Operation Background

Due to the overlap in features between the message queue and the Rule Engine, the platform will discontinue the message queue feature. This document guides existing users on migrating from the message queue to the Rule Engine.

Operations

1. Log in to the IoT Hub console and click View details on the upper right of the Overview module. You will be directed to the Product List homepage by default.
2. On the Product List page, click the message queue to view the configured information.
   - Message Queue CMQ: Refer to the Rule Engine configuration rules and modify them to use Data Forwarding to TDMQ in the Rule Engine.
   - Refer to the rule engine configuration rules, modify to use the rule engine for Data Forwarding to CKafka.
3. Due to the configuration of the message queue features CMQ and CKafka, and the configuration of rule engine forwarding, the data format received by the customer's business platform is different, requiring the customer to make the following modifications to the server-side program.
4. After the server-side verification is successful, delete the configuration information in the message queue feature.

Modifications

Due to the different data formats received by the customer's business platform when using message queue feature forwarding and rule engine feature forwarding, follow the instructions below for modifications.

Device-reported message

- Data format for message queue feature forwarding:

```
{
    MsgType:    "Publish",
    Topic:      topic,
    Seq:        Equivalent to the reported mid,
```

©2013-2025 Tencent Cloud. All rights reserved.

```
    PayloadLen: payload length,
    Payload:    payload content,
    ProductId:  product id,
    DeviceName: Device name
}
```

- The data format forwarded by the rule engine function, refer to the rule engine forwarding to CKafka.

Modification point: Since the values of the MsgType field in the data received by the customer business platform forwarded by the two methods are different, if the customer business platform uses the MsgType field, it needs to be modified.

Device status change notification

- Data format for message queue feature forwarding:

```
{
    MsgType:    "StatusChange",
    Topic:      topic,
    Seq:        Equivalent to the reported mid,
    PayloadLen: 0,
    Payload:    "",
    ProductId:  product id,
    DeviceName: Device name
    Event: Device online status, values are Online and Offline
}
```

- The data format forwarded by the rule engine function, refer to the rule engine forwarding to CKafka.

Modification points:

- The value of the MsgType field is different. If the customer's business platform uses the MsgType field, it needs to be modified.
- The values of the device online status are different. The message queue feature obtains the device status from the Event field, but the rule engine feature needs to obtain the device status from the Payload. For data parsing in the Payload, refer to the rule engine forward to CKafka.

©2013–2025 Tencent Cloud. All rights reserved.